UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

Aplicación móvil para la difusión de oferta de empleos y gestión de información para el reclutamiento de personal de maquiladoras de ciudad Juárez

Reporte Técnico de Investigación presentado por:

Lucio Ortiz de la Torre 132027

Cristian Valadez Najera 132143

Requisito para la obtención del título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Maestro Hugo Brito Holguín

Ciudad Juárez, Chihuahua 22 de noviembre de 2019

Ciudad Juárez, Chihuahua, a 20 de Noviembre de 2019

Asunto: Liberación de Asesoría

Mtro. Ismael Canales Valdiviezo
Jefe del Departamento de Ingeniería
Eléctrica y Computación
Presente.-

Por medio de la presente me permito comunicarle que, después de haber realizado las asesorías correspondientes al reporte técnico Aplicación móvil para la difusión de oferta de empleos y gestión de información para el reclutamiento de personal de maquiladoras de ciudad Juárez., los alumnos Lucio Ortiz de la Torre , Cristian Valadez Najera, de la Licenciatura en Ingeniería en Sistemas Computacionales, considero que lo han concluido satisfactoriamente, por lo que pueden continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente

Hugo Brito Holguín

Profesor Investigador IIT

UACJ
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y COMPUTACIÓN

Ccp:
Coordinador del Programa de Sistemas Computacionales
Lucio Ortiz de la Torre
Archivo

Ciudad Juárez, Chihuahua, a 20 de Noviembre de 2019

Asunto: Autorización de Publicación

C. Lucio Ortiz de la Torre

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del documento de Aplicación móvil para la difusión de oferta de empleos y gestión de información para el reclutamiento de personal de maquiladoras de Ciudad Juárez, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

Dr. Vianey Guadalupe Cruz Sánchez

Profesor Titular de Seminario de Titulación II

Ciudad Juárez, Chihuahua, a 20 de Noviembre de 2019

Asunto: Autorización de Publicación

C. Cristian Valadez Najera

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del documento de Aplicación móvil para la difusión de oferta de empleos y gestión de información para el reclutamiento de personal de maquiladoras de Ciudad Juárez, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

Dr. Vianey Guadalupe Cruz Sánchez

Profesor Titular de Seminario de Titulación II

Declaración de Originalidad

Nosotros, Lucio Ortiz de la Torre , Cristian Valadez Najera, declaramos que el material contenido en esta publicación fue elaborado con la revisión de los documentos que se mencionan en el capítulo de Bibliografía, y que la solución obtenida es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra institución de educación superior.

Lucio Ortiz de la Torre Cristian Valadez Najera

Agradecimientos

Cristian Valadez Najera

El desarrollo de esta tesis fue un escalón más para poder concluir con mis estudios y a pesar de no haber sido nada fácil quiero agradecer tanto a mi asesor como a mi maestra de materia por haberme guiado y orientarme durante el desarrollo de mi proyecto, además de los maestros que impactaron en mi para mejorar mi educación y obtención de los conocimientos necesarios para llegar hasta aquí. También quiero agradecer a mis padres quienes desde el inicio de mi carrera estuvieron apoyándome y confiando en mí. Agradezco a mi novia quien me dio el apoyo moral cuando las cosas se complicaban y Gracias a mi compañero de carrera quien realizó junto conmigo el proyecto.

Lucio Ortiz

Durante el proceso y conclusión de mi carrera universitaria estuvieron muchas personas que contribuyeron a alcanzar mi meta. En primer lugar quiero agradecer a mis maestros quienes me ayudaron guiándome, enseñándome, retroalimentándome, aportando lo mejor de ellos para mí formación. También quiero agradecer a mis padres los cimientos fuertes que me dieron siempre ganas de salir adelante a pesar de todas las adversidades que tuve en mi camino, siempre confiando en mi y en cada paso que doy en mi vida. Gracias a mi compañero de carrera quien realizó junto conmigo el proyecto.

Gracias a todas y cada una de las personas que me ayudaron, no fue sencillo pero gracias a que siempre tuve mi convicción clara el amor y apoyo incondicional de toda mi familia,

incluyendo: amigos, compañeros, maestros, padres y esposa. Hoy en dia puedo decir que soy una mejor persona más capaz y mejor preparada.

Dedicatoria

Cristian Valadez Najera

Dedico esta tesis a mis padres quienes me apoyaron durante mis estudios y mis hermanos que a pesar de todo nunca dejaron de confiar en mí. A mi novia quien siempre estuvo apoyándome cuando más lo necesite.
A todos los que me apoyaron para escribir y concluir esta tesis. Para ellos es esta dedicatoria de tesis, pues es a ellos a quienes les debo haber concluido mis estudios.

Lucio Ortiz

Este logro se lo dedico a mis padres por siempre confiar en mi y apoyarme en los momentos más difíciles que llegue a pasar, se lo dedico a mis hermanos que también fueron un apoyo incondicional. A mi esposa que siempre esta apoyándome y me ayuda a esforzarme más para conseguir mis metas. Se lo dedico a la persona más especial para mí, la cual es mi hijo que siempre esta conmigo, el cual es mi motor para seguir adelante y seguir mejorando cada día para cuando nos volvamos a reencontrar vea lo mucho que hemos avanzado.

Índice general

Índice de figuras

Índice de tablas

Resumen

El presente proyecto de titulación hace un análisis de los métodos actuales de la difusión de las vacantes de empleo de las empresas maquiladoras, así mismo se hace referencia a la necesidad de incluir dentro de las formas de contratación de personal a las tecnologías. Aprovechando el auge que tienen los dispositivos móviles en la población de Ciudad Juárez se plantea contar con un prototipo de aplicación orientada para las plataformas de Android e IOS, esto para abarcar un mayor número de participación ciudadana al cual se le pueda ofrecer un medio alternativo digital para la búsqueda de empleo.

Se utilizo la metodología de prototipo de alta fidelidad, el cual es muy parecido a la aplicación final pero, sin algunas funcionalidades no tan criticas. La base de datos utilizada es un servicio de google el cual lleva por nombre Firebase, con esto se conectan las dos plataformas(iOS y Android) para así mantener la misma información.

Los resultados obtenidos en las pruebas realizadas, arrojaron que el proyecto fue satisfactorio ya que la retroalimentación obtenida por los usuarios es buena, la información mostrada en la aplicación resultó bien para ellos además del agrado de la empresa en tener una aplicación dedicada a la difusión de vacantes de empleo.

Algunas recomendaciones para mejorar la aplicación fueron obtenidas por los usuarios de la aplicación, los cuales recomiendan más información acerca del transporte de la empresa, también recomiendan comunicarse abiertamente con la empresa con un chat en la misma aplicación para agilizar el proceso y resolver las dudas que lleguen a tener. En conclusión el

proyecto cumplió con los objetivos planteados al inicio del mismo.

Introducción

Gracias a los avances tecnológicos, actualmente se dispone de una gama de aplicaciones para smartphones y tablets, que están ayudando a empresas y profesionales en sus procesos. Las empresas requieren de sistemas de control implementados a través de procesos internos y externos, uno de ellos el proceso logístico, que interviene en las áreas de compras, suministros, inventarios, transporte, servicio al cliente, contratación y algunos otros, todos ellos requieren de tecnologías actualizadas y eficientes para poder estar a la vanguardia, en el mercado nacional como internacional.

El desarrollo de aplicaciones tecnológicas requiere de adquisición de conocimientos específicos, para apoyar en la resolución de los problemas, e ir minimizando las dificultades a las que se enfrentan las organizaciones a través de sistemas actualizados, que sean adaptables a las necesidades de cada una [1].

En la región de Ciudad Juárez, la asociación de maquiladoras Índex Juárez, en su reporte mensual infograma No 3, publicado el 13 de marzo del 2018 [2], en una de las estadísticas, identifica a esta ciudad, como la principal generadora de empleos de la Industria manufacturera de exportación en el país siendo esta ciudad una gran generadora de empleos.

El área de interés de este proyecto se delimita a la fase de convocatoria y reclutamiento de personal en una empresa tipo maquila, para agilizar estos procesos con la finalidad de ahorrar tiempo y recursos. Partiendo desde este enfoque, se propone el desarrollo de una aplicación móvil, donde se pueda realizar la difusión de ofertas de empleos y la gestión de información

requerida, dirigida hacia la población que haga uso de dispositivos móviles, y eventualmente se obtenga la respuesta esperada de personas idóneas para cumplir con los requerimientos de reclutamiento de personal de las empresas en Ciudad Juárez.

En el capítulo uno se describe el problema, los objetivos que se quieren alcanzar, y la justificación del desarrollo del proyecto. El segundo capítulo abarca el marco teórico, la problemática que se tiene en la ciudad al momento de buscar empleo y cómo las empresas utilizan medios anticuados o de bajo rango de difusión que limita el alcance de la vacante. También se menciona la programación y cómo se llevará acabo en conjunto con la base de datos.

En el tercer capítulo se aborda el desarrollo de la aplicación, describiendo cada fase de la metodología y dividiéndose en la parte del desarrollo del producto ya que son diferentes plataformas, sin embargo ambas siguen una misma metodología.

En el cuarto capítulo se discuten los resultados obtenidos con las encuestas realizadas a los usuarios y empleados de una empresa con la aplicación, en los apartados de rendimiento, diseño y aceptación de la misma. El capítulo final redacta las conclusiones a las que se llegaron, con las encuestas se responde el objetivo general y los objetivos específicos, por último se ofrecen algunas recomendaciones para futuras investigaciones de este campo.

Capítulo 1
Planteamiento del Problema

El sector industrial es uno de los medios más utilizados para la contratación de personal que aún se manejan mediante un sistema obsoleto. La difusión de las vacantes por medio de módulos es el método de contratación de muchas empresas. Una de las maneras de aprovechar mejor los procesos de contratación es por medio de las tecnologías actuales. Estas tecnologías permiten utilizar herramientas para elaborar una aplicación que dé solución alternativa a algunas de las problemáticas que se puedan presentar en el actual sistema de difusión de empleos. Esta aplicación tecnológica debe tener la característica de usar herramientas de publicación para proporcionar información que ayude al usuario en su búsqueda de empleo. Usar servicios como sistemas de base de datos en la nube y aprovechar los dispositivos en una aplicación que se ajuste al sistema de difusión de empleos que permita tener una alternativa a los métodos actuales de contratación de personal para maquiladoras.

1.1. Antecedentes

La evolución de las apps se dio rápidamente gracias a las innovaciones en tecnología WAP y la transmisión de data (EDGE) esto vino acompañado de un desarrollo muy fuerte de los celulares, cuando Apple lanza el iPhone y por otro lado se lanza las propuesta de smartphones android, es aquí que empieza un aumento de aplicaciones móviles como: juegos,

noticias, diseño, arte, fotografía, medicina, etc. por lo que se puede decir que la fabricación de los dispositivos así como el software se han convertido factores muy importantes para incrementar los servicios y el impacto en la sociedad en general.

Una aplicación que permite encontrar empleo o crear currículum desde la aplicación es InfoJobs [3]. Que cuenta con un portal web con el cual inició, pero al pasar los años y con el avance de los celulares lanzaron la aplicación móvil InfoJobsApp[3]. Otra aplicación es JobandTalent[4] que a diferencia de InfoJobs no cuenta con un portal web, con lo que si cuenta esta aplicación es con la función de servicios administrativos como lo es gestión de nóminas, los cálculos de horas trabajadas y hasta la firma del contrato, todo ello desde la misma aplicación.

La aplicación JobToday[5] está especializada en los sectores de la hostelería, comercio y servicios, su mayor ventaja es que simplifica los procesos de selección y aseguran que el candidato recibirá una respuesta por parte de la empresa empleadora en menos de 24 horas[6]. La aplicación Indeed[7] también cuenta con portal web y a diferencia de las otras se encuentra disponible en más de 60 países con 28 idiomas diferentes, desde Indeed tienen la oportunidad de recibir ofertas de empleo sin la necesidad de haber rellenado un formulario o haber enviado un currículum, sino que solamente adjuntando uno propio en las ofertas[6]. WorkToday[8] es una aplicación que a diferencia de las otras se especializa en trabajos de un día, 48 horas o empleos temporales.

La herramienta inteligente para una propuesta de empleo basado en minería de datos[9] utiliza la minería de datos para la obtención de anuncios de empleos y Os-Commerce (una aplicación de software abierto para crear tiendas virtuales)[10]. A diferencia de esta aplicación, el sistema desarrollado es diferente ya que es una aplicación móvil y además sólo se centra en Juárez.

1.2. Definición del problema

En Ciudad Juárez la mayor oferta de empleo es proveniente de la industria maquiladora, el problema es que la difusión de las vacantes se limita a publicidad impresa (volantes, periódico) además, tiene un límite hacia el personal requerido lo que podría ser una vacante en especifico y no llega personal con las características solicitadas por la empresa y se ve obligada a contratar personal no calificado para completar su plantilla laboral[11].

1.3. Objetivos

Objetivo general

Desarrollar una aplicación móvil, bajo las plataformas Android e IOS, que ofrezca una solución alternativa para la difusión de información de oferta de empleo de empresas maquiladoras de Ciudad Juárez.

Objetivos específicos

•Establecer la base de datos y configurarla para un buen desempeño de la aplicación.

•Conseguir empresas que publiquen sus vacantes.

•Lograr buena interacción entre el usuario y la aplicación.

1.4. Justificación

Debido a la gran competencia y requerimientos de contratación de personal entre las empresas en Ciudad Juárez, es indispensable que se integren soluciones de propuestas tecnológicas para enfrentar los desafíos en términos de rapidez de respuesta y requerimiento, en sus procesos logísticos de divulgación y contratación, ya que este tipo de soluciones puede tener impacto en la reducción de costos y el aumento la productividad.

La aplicación móvil desarrollada, bajo dos de las plataformas más populares en la actualidad Android e iOS, ofrece una alternativa de solución tecnológica que apoya a la difusión de información de oferta de empleo para empresas maquiladoras y que beneficia tanto a las empresas que requieren personal como a las personas interesadas en incursionar en el ámbito laboral.

Se visualiza que la aplicación repercutirá en factores como: el acceso fácil y actualizado de oferta en el mercado laboral de maquiladoras, aumentar la difusión la información de oferta de empleos al público en general, incrementar el reclutamiento y selección de personal, mantener una relación entre el solicitante y la empresa y por supuesto ofrecer una imagen de innovación de la empresa.

Capítulo 2
Marco teórico

En esta sección se expone la situación con base a la resolución del proyecto en donde se explicará la secuencia de solución de la aplicación. Este proyecto se basa en utilizar las tecnologías disponibles, que se puedan aplicar a la demanda de empleo en específico de una ciudad en donde una problemática a resolver, es definir una solución alternativa a la difusión de la información de las vacantes de la industria manufacturera. Para definir el proyecto se dividirá en dos partes fundamentales como conceptos en donde se buscar relacionar los detalles de las ofertas de empleo tanto en su esquema de funcionamiento en la ciudad como sus limitaciones para poder entran en otro concepto que es la programación de la aplicación donde se mostraran los conceptos base para el desarrollo y las tecnologías que integraremos para la aplicación de un primer prototipo que demuestre la solución al problema.

Las vacantes de trabajo de la ciudad

En Ciudad Juárez la oferta de empleo se maneja bajo un sistema obsoleto en donde aplicar un medio tecnológico para su uso representa un obstáculo, por lo cual, se debe buscar una solución que implique utilizar un desarrollo tecnológico aplicable al sistema propio de la difusión de las vacantes. Para eso se determinaron las variables posibles a utilizar como conceptos para el desarrollo.

Como variable principal fue la determinación de las empresas que se encuentran contratando y conocer las formas en las que lo hacen. La implicación en esto fue procesar toda esta información a una base de datos de modo que pudiera ser escalable y flexible. Escalable debido al crecimiento de la ciudad y la demanda de empleo, flexible por el hecho de no tener que rescribir toda la información de nuevo ante posibles cambios.
Otro concepto fue determinar la manera en que las aplicaciones en sus distintas plataformas (IOS y Android) se comunicarían con la base de datos. Utilizando una plataforma como servicio proporcionada por Google se creó una única base de datos donde las aplicaciones de ambas plataformas se conectaran obteniendo la misma información. Google en su plataforma de nombre Firebase ubicada en la nube, permitió utilizar su herramienta de base de datos para alojar la información que se utilizó para acceder en las distintas plataformas móviles casi al instante de ser publicada.

Programación

La aplicación fue desarrollada para los sistemas operativos Android e IOS utilizando la herramienta AndroidStudio para Android y XCode para IOS, empleando estas herramientas y las API de Google se obtuvieron los servicios de la plataforma Firebase.

Firestore es una base de datos creada por Google ubicada en la nube lo cual nos permite 1: la fácil integración con el proyecto, además esta base de datos permite 2: almacenar información, lo cual nos permite 3: estar accediendo a dicha información desde cualquier plataforma ya sea Android o IOS. Con estas herramientas se desarrollaron dos perfiles dentro de la aplicación una para las empresas y otra para el usuario, el perfil de la empresa es la encargada de enviar la información a la base de datos, lo anterior se consiguió utilizando el servicio Firestore de la plataforma Firebase, publicando las vacantes requeridas de la empresa además de recibir el currículum del postulante.

El perfil del usuario permite al momento del registro ingresar la información relevante para su contratación el cual puede editar dependiendo de sus experiencias laborales, el usuario puede estar revisando las vacantes de las empresas y aplicar a la que más posibilidades tenga.

Capítulo 3
Desarrollo del Proyecto

La metodología empleada en el desarrollo del proyecto es utilizando un prototipado de alta fidelidad la cual consiste en llevar a cabo lo más real posible la implementación de la aplicación con el producto esperado.

El desarrollo del proyecto se divide en tres fases, la primera muestra la base de datos ubicada en la nube, su proceso de realización y de configuración. La segunda fase indica la metodología del desarrollo del producto en la plataforma en Android y la fase tres demuestra los pasos para el desarrollo de la aplicación en su versión para iOS. Los procesos de desarrollo de las fases 2 y 3 se realizaron en paralelo sólo cambia los tiempos de duración de cada actividad acorde a la plataforma.

3.1. Producto propuesto

Se desarrolló una aplicación como alternativa de difusión y gestión de vacantes para el sector maquilador que proporciona al usuario sugerencias de empleos con base a su historial de trabajo, además muestra una ayuda visual para poder localizar a las empresas y con las que podrá tener un contacto más directo antes de su contratación.

Las empresas tienen las funciones tales como publicar nuevas vacantes y determinar si los

postulantes son contratables.

Figura 3.1: Arquitectura de la aplicación.

Como se muestra en la Figura 3.1 las empresas interactúan con la aplicación subiendo la información de las vacantes actuales, la cual se guarda en la base de datos Firebase, y el usuario por su parte puede acceder a esta información de manera inmediata (dependiendo de su conexión a internet) y así tener la información lo más rápido posible. Además, el usuario puede enviar su solicitud de la vacante si es de su interés para que la empresa reciba de manera inmediata y darle así una pronta respuesta de su solicitud.

Los dispositivos IOS y Android pueden publicar y acceder a la misma información gracias a la conexión que existe entre la aplicación y la base de datos, no teniendo que modificar información para cada uno de ellos por separado.

3.2. Metodología de prototipo (Alta fidelidad)

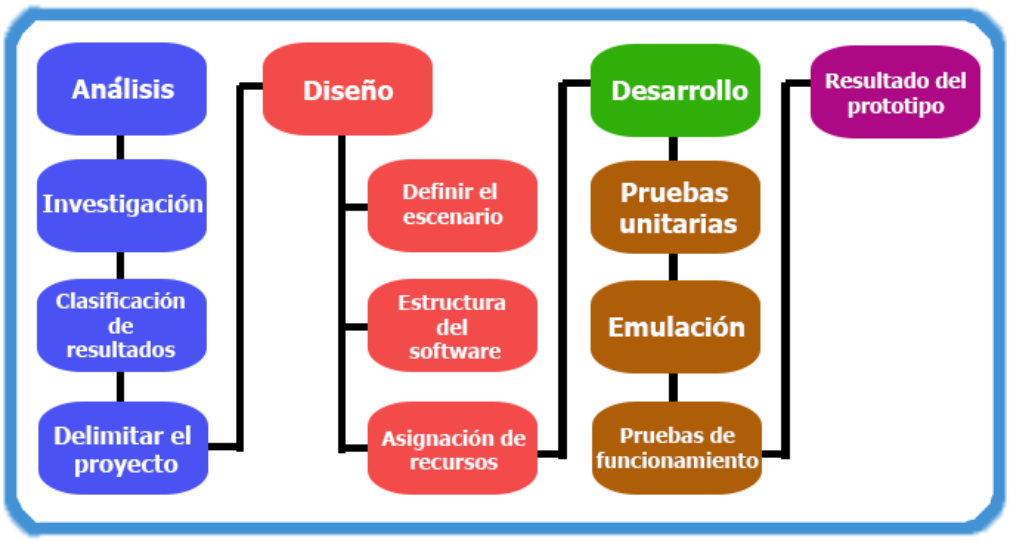


Figura 3.2: Prototipo de alta fidelidad.

La metodología de prototipado de alta fidelidad es una forma de realizar el prototipo que se utilizará para las pruebas. El cual corresponde a un producto con una interfaz diseñada lo más real posible. Éste prototipado es una herramienta para realizar proyectos de software con funcionalidad, donde acciones que se solicitan en la interfaz podrán ser eventos respondidos tal y como fuera el producto real.

Este tipo de metodología se utiliza cuando el equipo de desarrollo no cuenta con el capital suficiente y a la vez carece de tiempo para terminar el proyecto. Si la fidelidad del prototipo es alta, este puede ser utilizado para pruebas finales con base a lo establecido en los requerimientos iniciales.

3.3. Análisis

Como inicio del desarrollo se llevó a cabo un análisis de la problemática y cómo poder solucionarla aplicando las tecnologías actuales, en este caso el desarrollo de una aplicación móvil que permite tanto a los usuarios y empresas mejorar su forma de contratación utilizando una alternativa a las ya usadas hoy en día.

Llevando un análisis de requerimientos conforme a los especificados por el cliente, el prototipo final se espera que cumpla con algunas de las siguientes funcionalidades para poder ser revisada y valorada por la empresa interesada.

Requerimientos del cliente para el producto:

-Implementar un método de autenticación fácil donde se cuente con las funcionalidades de inicio de sesión, cierre de sesión y revocación de autenticado.

-Orientar la aplicación a dos vistas que permita diferenciar su uso ya sea el usuario, una empresa o un postulante.

Vistas orientadas al usuario

-Conformar un apartado para la integración del historial del postulante que pueda ser editable y legible en su momento de aplicar.

-Mostrar las vacantes disponibles en la vista principal de la aplicación.

-Contar con una sección de vacantes por empresa al igual de poder ver la descripción de cada vacante.

-Indicar al postulante en cuál fase del proceso se encuentra su solicitud.

-Que el usuario administre sus datos a través de un perfil.

Vistas orientadas a la empresa

-Tener como vista principal las solicitudes de los postulantes a las vacantes.

-Una sección para administrar las vacantes publicadas.

-Formato para poder publicar nuevas vacantes.

-Un perfil para la empresa.

3.4. Investigación

En primera instancia, después de realizar el análisis con base a la problemática presentada, se identificaron las herramientas necesarias que se utilizan para el desarrollo en las plataformas de Android e IOS.

El desarrollo de aplicaciones móviles ha crecido los últimos años debido al gran augue que han tenido los teléfonos inteligentes en la población.

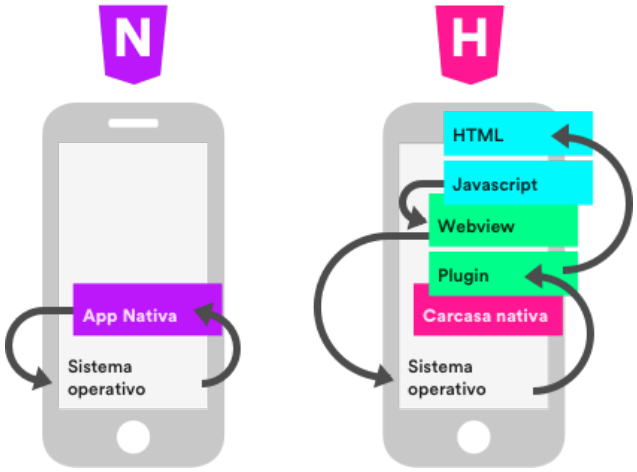


Figura 3.3: Distintos tipos de aplicaciones móviles.

Actualmente, se ofrecen diferentes maneras de desarrollar aplicaciones entre las que se encuentran las siguientes:

Aplicaciones Web Este tipo de desarrollo se utiliza cuando es necesario adaptar la aplicación a diferentes tipos de dispositivos ya sea computadoras o dispositivos móviles. Debido a que es una Web App se puede programar en los lenguajes html y javascript aumentando

las posibilidades de utilizar cualquier plataforma que acepte estos lenguajes y reduciendo el tiempo de desarrollo, ya que no es necesario implementar código para cada uno de los sistemas operativos móviles.

Ventajas

- Menos tiempo de desarrollo.
- Compatibilidad con navegadores de internet.
- No es necesario escribir código para cada sistema operativo.
- Más plataformas de desarrollo.

Desventajas

- Problemas de adaptabilidad.
- No se utilizan los recursos del dispositivo al cien porciento.
- Nuevas actualizaciones no disponibles.
- Menor rendimiento.
- Depende de más uso de red.
- Novedades para el desarrollo móvil no implementadas.

Aplicaciones híbridas

Para realizar este tipo de aplicaciones, actualmente se cuenta con diferentes entornos de desarrollo entre los más conocidos Ionic 3 y Flutter, estos entornos permiten realizar aplicaciones al estilo de las web apps a diferencia de que al compilar el código se genera el proyecto para distintos sistemas operativos resultando una aplicación al estilo de una nativo por su tipo de diseño y programación.

Ventajas

- Menos tiempo de desarrollo.
- Más uso de los recursos del dispositivo.

- Posibilidad de desarrollo para diferentes plataformas.
- Diseño de interfaces más fácil.
- Menos programación

Desventajas

- No se utiliza por completo los recursos de los dispositivos.
- El rendimiento es menor.
- No compatible con diferentes librerías.
- Las novedades de desarrollo móvil no implementadas.
- Se utiliza para desarrollos no tan complejos.

Aplicaciones Nativas

Las aplicaciones nativas son el producto de realizar a través de los compiladores oficiales el proyecto para cada uno de los sistemas operativos no permitiendo ser multiplataforma. El desarrollo nativo permite utilizar por completo las novedades de cada uno de los dispositivos e implementar las actualizaciones más recientes de cada uno de las plataformas y permite el desarrollar proyectos más complejos.

Ventajas

- Mejor rendimiento de aplicaciones.
- Uso de las novedades de cada dispositivo.
- Implementación de más herramientas externas.
- Más compatibilidad de librerías.

Desventajas

- No son multiplataforma.
- Más complejidad de desarrollo.

- El tiempo de desarrollo es mayor.

Para fines del presente proyecto, se seleccionó el tipo de desarrollo nativo debido a la amplia variedad de recursos que ofrece y para implementar una aplicación con las ultimas novedades.

Para desarrollar aplicaciones en Android, se investigaron varios entornos de programación en donde Android Studio[12] resultó el mejor ambiente de desarrollo para este prototipo, por ser el oficial establecido por Google según se expone en[13].

Este entorno ofrece la posibilidad de desarrollar aplicaciones para el sistema Android utilizando Kotlin un lenguaje de programación lanzado por JetBrains o el ya conocido Java[14] que por su actual documentación y ser uno de los lenguajes primitivos es más completo para el desarrollo.

Entre los requerimientos para correr aplicaciones en distintos CPUs es necesario utilizar una máquina virtual que sea compatible con estos procesadores. El Java Develovment Kit[11] proporciona la máquina virtual Java Runtime Environment[15] que en conjunto con el entorno de desarrollo hace posible la compilación de código Java para Android.

Para desarrollar la aplicación en IOS, se investigó el entorno de programación Xcode[17], por ser el oficial y el que más se adecua a la aplicación. Este entorno da la posibilidad de desarrollar aplicaciones para el sistema IOS, utilizando el lenguaje Swift[17] que se identifica como uno de los más populares más accesibles y fácil de utilizar debido a su sintaxis simplificada, además de proporcionar un simulador para hacer las pruebas durante el desarrollo[18].

Además, se encontró un gestor de librerías como CocoaPods[10] que brinda las librerías necesarias para poder gestionar la base de datos FireBase. Entre los requerimientos para correr aplicaciones IOS, se requiere de arquitecturas iPhone 5S hasta el más nuevo a la fecha el iPhone 8, debido que Apple limita su portabilidad a los modelos anteriores al iPhone 5S.

En ambas plataformas seleccionadas Android e IOS, se requiere guardar información de

una manera externa por lo que se necesita utilizar un servicio que ofrezca la posibilidad de realizar tareas de almacenamiento y base de datos[20] donde se pueda acceder desde cualquier lugar y momento, para ello se utiliza la plataforma ofrecida por Google. Firebase[21] ofrece distintos tipos de servicios, entre ellos una base de datos NoSQL[22] flexible y escalable que utiliza servicios en la nube además permite la programación del lado del cliente y servidor. Firebase también ofrece el servicio Storage[23] un sistema de almacenamiento que permite guardar distintos tipos de archivos.

Firebase a comparación de otros servicios brinda un plan gratuito para utilizar sus componentes completamente para aplicaciones prototipo, el cual da una ventaja sobre otros en gastos previos al desarrollo que ofrece más atributos, además, de la base de datos y almacenamiento pudiendo ampliar las capacidades del proyecto[24].

3.5. Clasificación de resultados

En este apartado se especifican los requerimientos funcionales que debe realizar la aplicación con la finalidad que cumpla con lo especificado al realizar la entrega final, al igual que los requerimientos no funcionales que son las que definen al proyecto en cuanto a portabilidad, rendimiento, tiempo y costos.

Tabla 3.1: Funcionalidades de usuario.

Funcionalidad	Descripción
Registro	Permitir al usuario registrarse antes de usar la aplicación
Currículo	El usuario puede ingresar su historial laboral
Vacantes	Posibilidad de ver las últimas vacantes dentro de la aplicación
Empresas	Mostrar las empresas disponibles para contratación
Detalles de vacante	Acceder a la información detallada de cada vacante
Aplicación	Permitir al usuario poder enviar su currículo a alguna empresa
Estatus	Indicar al postulante en qué fase se encuentra su solicitud

Los requerimientos funcionales de la aplicación se divide en tres partes, la primera es con base a las tareas por parte del usuario. La tabla 3.1 muestra las partes que debe contener las vistas del lado de los usuarios interesados en adquirir una vacante.

Tabla 3.2: Funcionalidades de empresa.

Funcionalidad	Descripción
Registro	Permitir a la empresa registrarse antes de usar la aplicación
Identificación	La empresa se debe verificar antes de usar la aplicación
Perfil	Mostrar los datos básicos de la empresa
Publicar	Publicar nuevas vacantes
Administrar	Posibilidad de administrar cada vacante creada
Gestión	Navegar entre cada uno de los postulantes y determinar su estatus
Pre-contratación	Poder seleccionar postulantes para su llamado a contratación

La segunda parte determina las tareas de lado de la empresa, permitiendo tener un perfil diferente al del usuario postulante con las funcionalidades especificadas en la tabla 3.2.

Tabla 3.3: Funcionalidades externas.

Funcionalidad	Descripción
BD en la nube	Leer y escribir datos en la nube
Autenticación	Mantener una sesión activa durante el uso de la aplicación
Registros	Tabla de registros de usuario
Almacenamiento	Enviar y recibir imágenes desde el servidor
Back-end	El servidor debe de escribir datos en la nube

Como se muestra en la Tabla 3.3, la aplicación cuenta con funcionalidades externas las cuales especifican las tareas implementadas para hacer funcionar los requerimientos de la aplicación.

Los requerimientos no funcionales indican las partes del proyecto que no son suministradas por la aplicación, determina las propiedades del mismo como el rendimiento, disponibilidad, seguridad entre otras funciones como se observa en la Tabla 3.4.

Tabla 3.4: Requerimientos no-funcionales.

Requisito	Descripción
Versión mínima	Específica la versión de Andorid o iOS mínima de la aplicación
Densidad de pantalla	Determina el tamaño mínimo de la pantalla
Rendimiento	Recursos mínimos del dispositivo para una buena ejecución
Fiabilidad	El sistema debe tener el menor número de fallos posibles
Usabilidad	El uso de la aplicación debe ser amigable con el usuario
Seguridad	Evitar la pérdida o divulgación de la información

3.6. Diseño

Con base al análisis obtenido referente a los requerimientos y la investigación para el desarrollo del prototipo se diseñaron los diagramas conforme a las especificaciones mencionadas. Los diseños se utilizan para tener un punto de partida antes del desarrollo y definir las vistas que se realizaran dentro de la aplicación.

Los diseños corresponden a dos tipos de Usuarios, una para definir las interfaces y funcionalidades orientadas al usuario, y otras para definir las funcionalidades de la empresa, se especifican sus interfaces desde el inicio de la aplicación pasando por el registro hasta desplazarse por cada uno de los componentes según la acción realizada.

Diseño de acciones del usuario

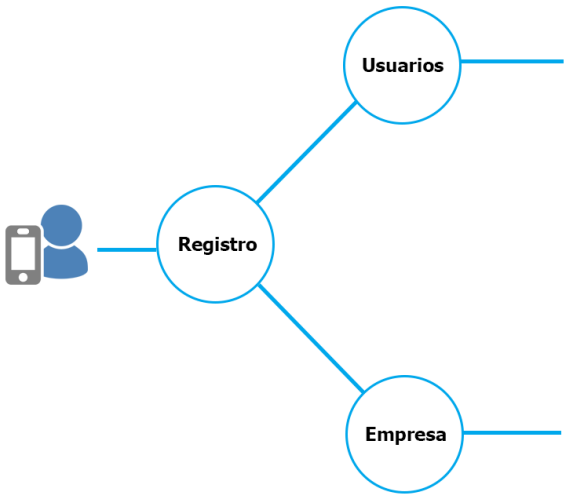


Figura 3.4: Diseño registrar empresa o usuario.

Al inicio de la aplicación el usuario puede ver una pantalla que corresponde al registro,

como se manejan dos tipos de usuario, uno que es el interesado en buscar empleo y otro que es el reclutador de la empresa. En este caso el usuario uno tendrá que registrarse para buscar vacantes, para lo cual sólo es necesario una cuenta de Gmail establecida en su dispositivo, ver Figura 3.4.

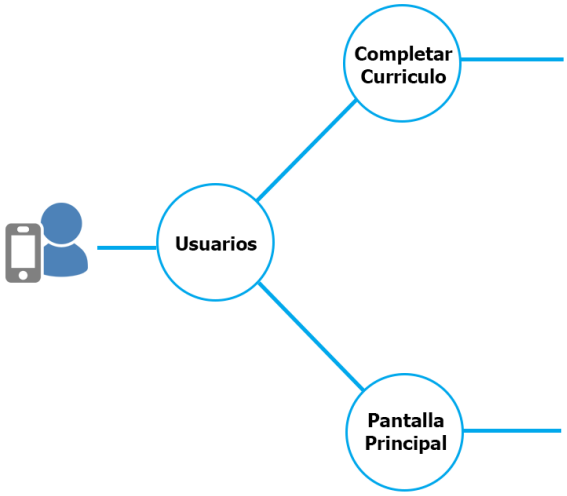


Figura 3.5: Diseño registro de usuario.

El usuario al crear una sesión con su cuenta de Google cambia de vista hacia la pantalla de completar su currículum u omitir esta parte y pasarse directamente al menú principal de la aplicación. En caso de omitir el llenado del currículum el usuario podrá volver a acceder a él desde su perfil, ver Figura 3.5.

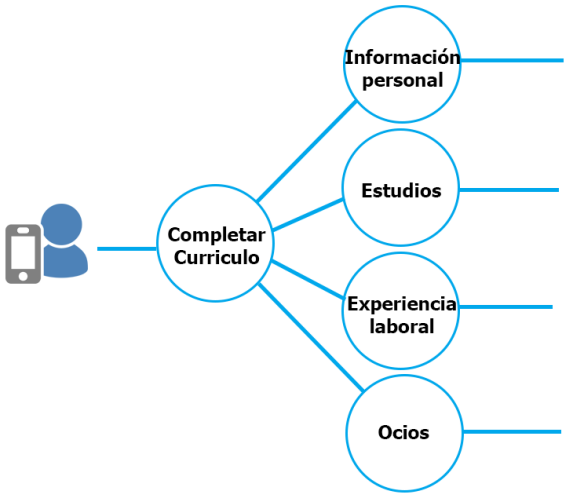


Figura 3.6: Diseño currículum del usuario.

La pantalla del currículum muestra al usuario un formato para completar su experiencia dividido en cuatro partes, la primera consta de completar los datos relacionados con su información personal y domicilio. El formato de estudios está compuesto por el grado académico del usuario comprendido con los datos de la escuela, duración, si concluyó la escuela y en caso de seguir estudiando especificar para su uso en las vacantes con condiciones.

En la experiencia laboral el usuario puede ingresar hasta sus últimos tres trabajos con la información del nombre de la empresa y su duración. Por último se agrega una sección para que el usuario especifique algunos ocios que realiza en su vida diaria, ver Figura 3.6.

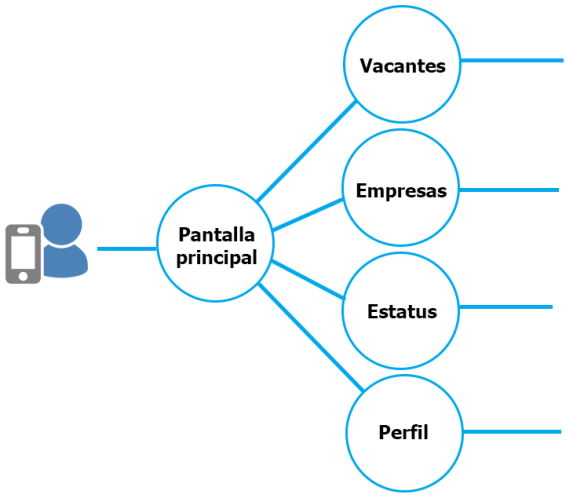


Figura 3.7: Tabla requerimientos no-funcionales.

Al acceder a la pantalla principal, el usuario puede observar diferentes opciones para navegar, la primera es la importante, se refiere a las vacantes disponibles, en esta vista el usuario puede ver y aplicar para las ofertas de empleo que seleccione. La segunda vista es para mostrar qué empresas se encuentran registradas en la aplicación.

El estatus muestra el proceso de selección y es donde el usuario observa si fue aceptado o rechazado de la vacante. La vista de perfil muestra información básica del usuario, ver Figura 3.7.

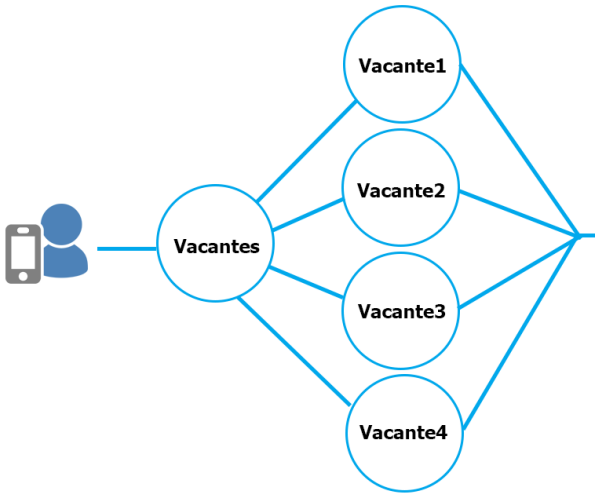


Figura 3.8: Diseño lista de vacantes en general.

La vista de vacantes muestra las diferentes ofertas de empleo disponibles, el usuario puede desplazarse por ellas y elegir una que le convenga, ver Figura 3.8.

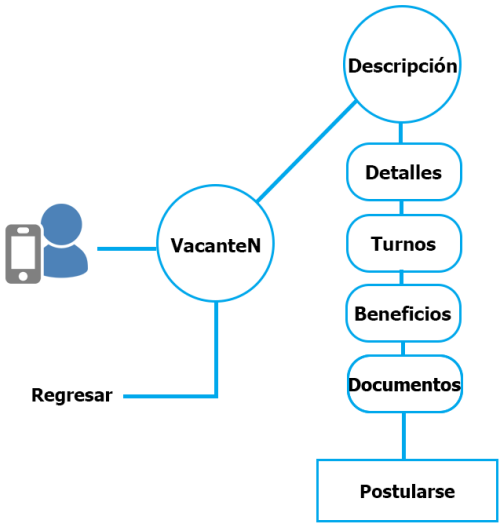


Figura 3.9: Diseño de la información de la vacante.

Al seleccionar una de las vacantes, se le muestra al usuario la descripción detallada, puede ver del empleo sus horarios, turnos, requisitos y si cuenta con beneficios adicionales. Si el usuario es convencido por la vacantes puede postularse desde esta misma sección por medio de un botón enviando su solicitud al responsable de la vacante, ver Figura 3.9.

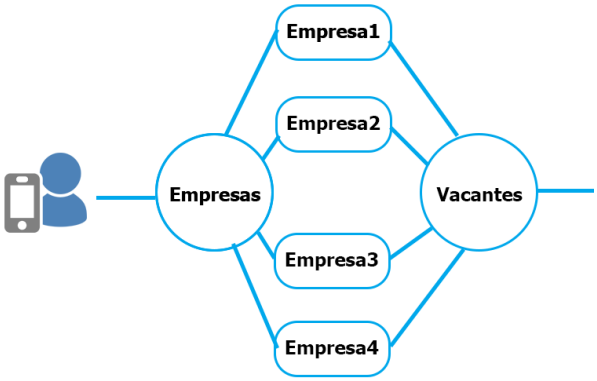


Figura 3.10: Diseño lista de vacantes por empresas.

La segunda vista del menú principal corresponde a la interfaz donde se muestran todas las empresas registradas, el usuario puede seleccionar entre las disponibles y ver información básica e incluso puede calificar a la empresa basándose en su experiencia trabajando en ese lugar o sólo opinar. Esta calificación no representa la calidad de la empresa, sólo se utiliza para hacer valoraciones internas de los usuarios, ver Figura 3.10.

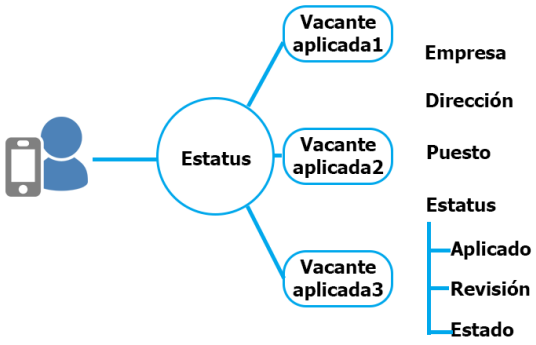


Figura 3.11: Diseño lista de solicitudes del usuario.

En la vista de estatus, el usuario puede visualizar todas sus solicitudes aplicadas, en cada solicitud puede ver el estatus de su proceso de selección, este consta de tres fases:

La primera fase es la comprobación de que la solicitud del postulante fue enviada correctamente.

La segunda fase indica que la solicitud esta en revisión. y la última es el estado del postulando, donde se indica si fue aceptado o rechazado, ver Figura 3.11.

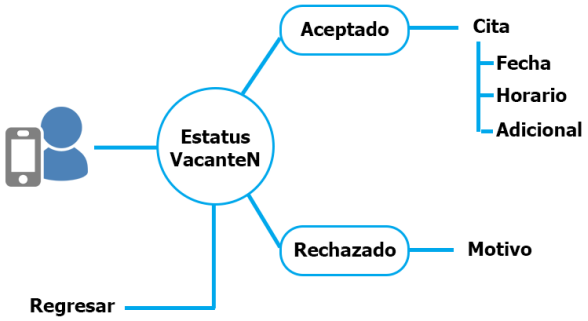


Figura 3.12: Diseño estado de la solicitud del usuario.

Después de que la solicitud de un postulante fue aceptada por la empresa, se le indica al usuario datos de la cita para acudir personalmente, los datos de ejemplo son: la fecha, hora e información adicional que la empresa indique, ver Figura 3.12.

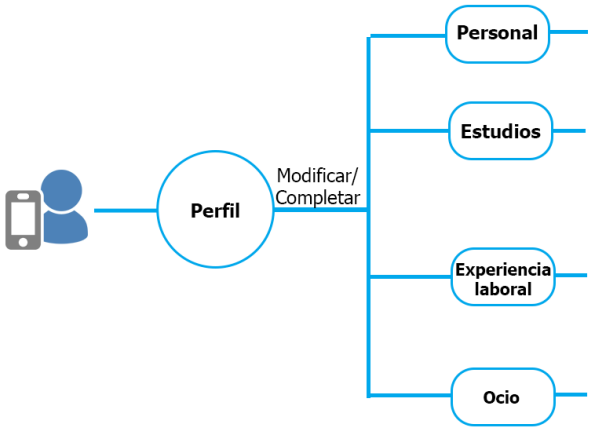


Figura 3.13: Diseño perfil usuario.

En el diseño del perfil, el usuario puede visualizar un resumen de su currículum virtual y además tiene la posibilidad de poder editar estos datos y en caso de no haber completado su información puede hacerlo en esta misma vista, ver Figura 3.13.

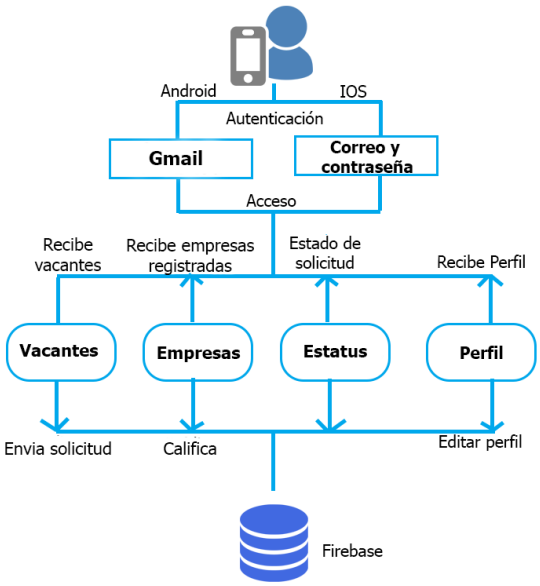


Figura 3.14: Diagrama de acciones del usuario.

La Figura 3.14 muestra la relación de las vistas con otras y sus referencias hacia la base de datos, en donde algunas vistas envían y reciben datos al servidor como lo es la de vacantes, empresas y perfil, en cambio cuenta con una vista donde sólo recibe datos desde la nube.

Diseño de acciones de la empresa.

Las empresas cuentan con un perfil, esto permite ver sólo información relevante para las mismas, una empresa no puede acceder a los dos enfoques de la aplicación. El perfil de la empresa le permite hacer las publicaciones de las vacantes y la gestión de los postulantes.

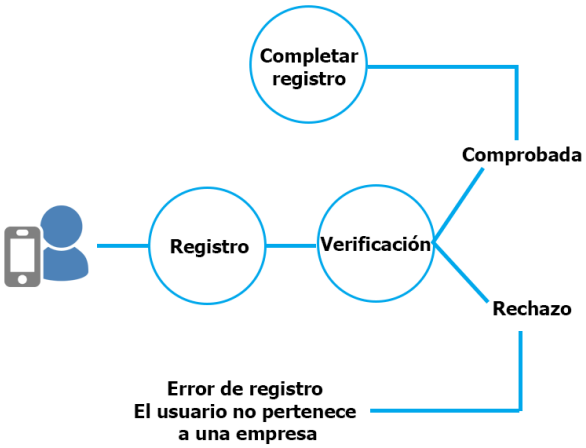


Figura 3.15: Diseño registro de empresa.

El registro de la empresa es diferente al del usuario, el reclutador o encargado de crear el perfil de la empresa necesita completar un formulario con información básica respecto a su compañía. Al completar el proceso de registro los datos son recibidos por los administradores que se encargaran de verificar la veracidad de la empresa, este proceso es manual, lo que quiere decir que no implica alguna acción dentro de la aplicación, ver Figura 3.15.

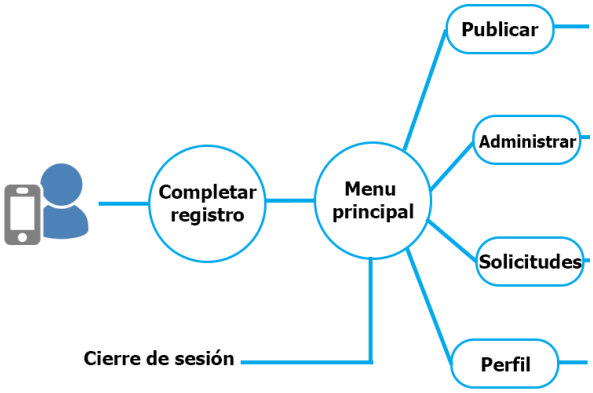


Figura 3.16: Diseño completar registro de la empresa.

Si la empresa ha sido verificada puede tener acceso al menú principal de la aplicación, al igual que los usuarios que buscan trabajo, las empresas tienen diferentes opciones para publicar y administrar sus vacantes, además de contar con la posibilidad de gestionar a los postulantes de sus ofertas de empleo, ver Figura 3.16.

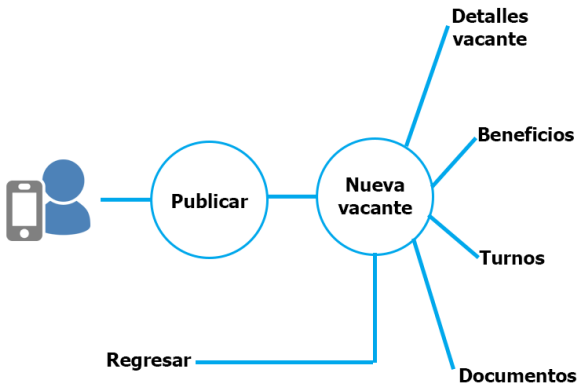


Figura 3.17: Diseño empresa publicar vacantes.

La empresa necesita publicar vacantes y para ello se le proporciona una interfaz donde pueda realizar esta acción, mediante un formulario compuesto de: detalles de la vacante, beneficios, turnos, requerimientos, documentos, entre otras cosas. Al completar el formulario puede publicar la vacante y estar disponible de manera inmediata para los usuarios en busca de un empleo, ver Figura 3.17.

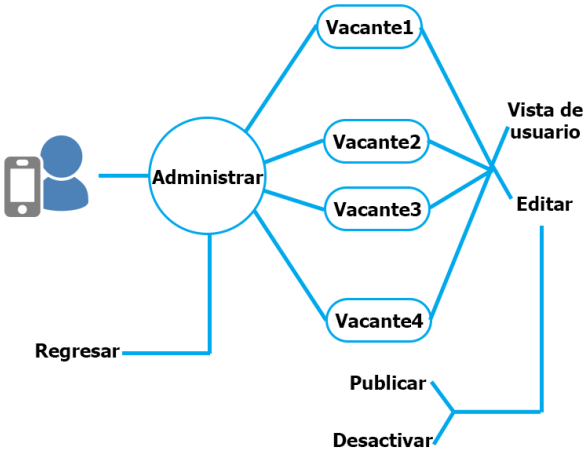


Figura 3.18: Diseño administrar vacantes publicadas.

La empresa después de publicar sus vacantes puede administrarlas desde la segunda interfaz del menú, donde visualiza cada una de sus vacantes y al acceder a una en primera instancia se le muestra los detalles de su empleo en formato como lo ve un usuario que busca trabajo, ver Figura 3.8, ahí mismo cuenta con la opción de poder editar su vacante donde es enviado al diseño ver Figura 3.17 para hacer las modificaciones o incluso puede poner inactiva la vacante.

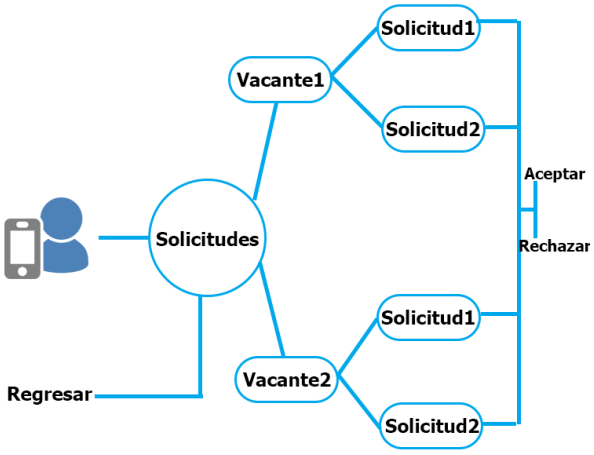


Figura 3.19: Diseño administrar solicitudes enviadas a las vacantes publicadas.

Al publicar una vacante, la empresa recibe solicitudes de muchas personas y necesita gestionar cada una de ellas, para esto se le muestra una interfaz llamada solicitudes donde accede a cada una de las ofertas de empleo publicadas y se le indica cuántas personas están interesadas en el trabajo, la empresa accede a cada una de las postulaciones y decide si es el indicado según sus necesidades, ver Figura 3.19.

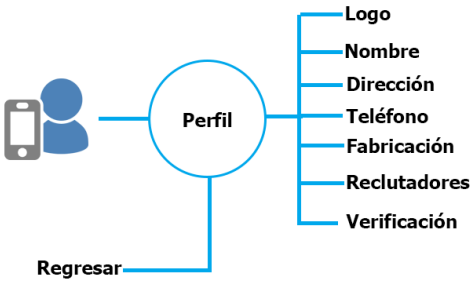


Figura 3.20: Diseño perfil de la empresa.

La empresa maneja un perfil donde puede registrar sus datos de dirección y contacto, información útil para que al momento de publicar una vacante no tenga que ingresar esta información tantas vacantes publique, ver Figura 3.20.

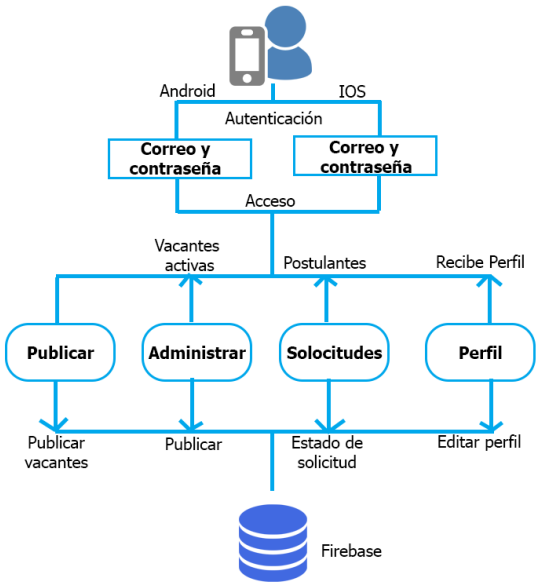


Figura 3.21: Diseño de las acciones generales que puede realizar la empresa.

Algunas de las vistas necesitan estar conectadas a un servidor de base de datos para poder almacenar y acceder a la información desde cualquier lugar y momento, por lo que estas vistas hacen referencias a la nube, como se observa en la Figura 3.21.

3.6.1. Estructura del Software

Las principales acciones del usuario están resumidas en el caso de uso de la Figura 3.22 las cuales son necesarias para el correcto funcionamiento de la base de datos y de las aplicaciones.

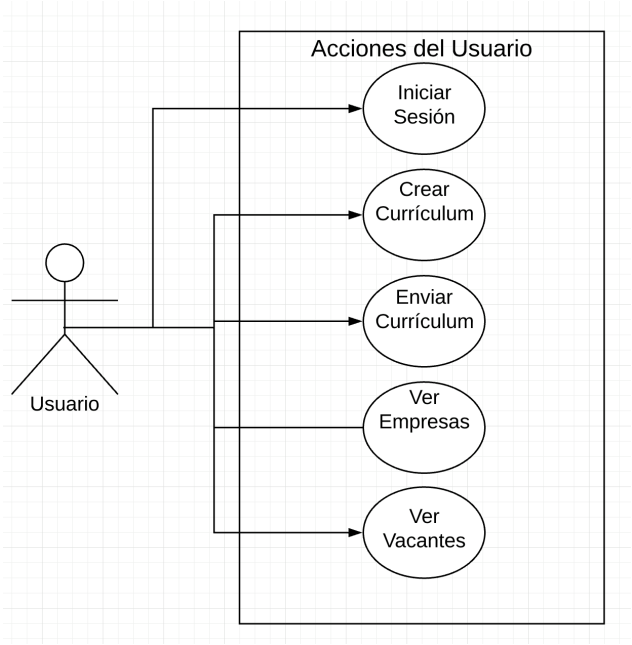


Figura 3.22: Caso de uso Usuario.

El primer paso es autenticar al usuario para obtener su identificador único y así poder registrar sus datos, ver las empresas y enviar su currículum. Una vez que el usuario se ha registrado, lo siguiente es ingresar los datos de su currículum con los datos más importantes, esto es necesario para poder enviarlo a las empresas ya que de lo contrario no podrá hacerlo. También podrá navegar en todas las vacantes publicadas ya sea por empresas o navegar por todas las empresas en general, siendo estas las acciones más importantes para difundir de manera inmediata las vacantes de las empresas.

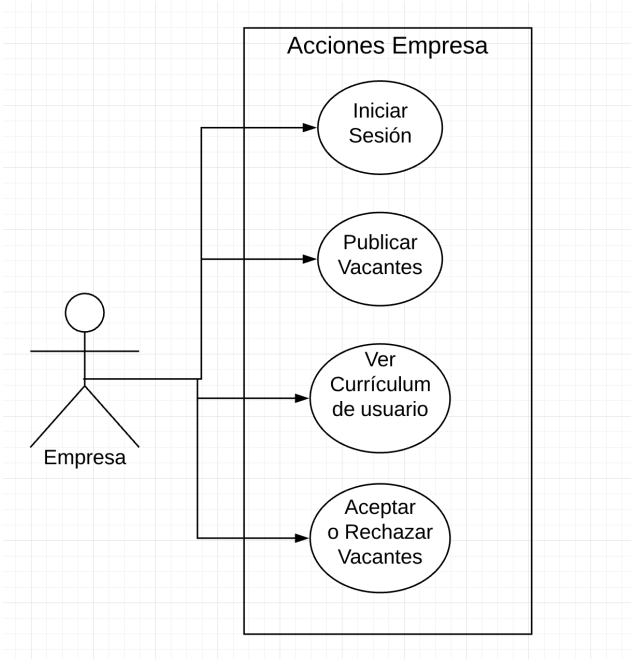


Figura 3.23: Caso de uso Empresa.

Las acciones principales de la empresa están en el caso de uso de la Figura 3.23. Estas acciones son el complemento de las acciones del usuario por así decirlo. Las empresas al registrarse tendrán que esperar una verificación la cual confirma si es o no una empresa, una vez echo lo anterior si la empresa se verifica correctamente entonces, podrá entrar a la vista principal y desde ahí publicar vacantes. (Este paso se lleva acabo para evitar que los usuarios pierdan el tiempo con vacantes que no son reales). Una vez verificada entonces podrá publicar vacantes disponibles en la empresa, posteriormente podrá editar la vacante si lo desea. Por último, también podrá administrar las solicitudes de las vacantes decidiendo si acepta o rechaza la solicitud de los usuarios.

3.6.2. Asignar Recursos

Los recursos que se utilizarán para la elaboración de la aplicación no sólo involucran herramientas de desarrollo si no también recursos humanos, financieros y tecnológicos.

Recursos humanos requeridos para el desarrollo de la aplicación:
• Una persona para desarrollar la aplicación en Android
• Una persona para desarrollar la aplicación en IOS
Recursos tecnológicos:
• Dos computadoras portátiles ambas con accesos a internet, una con sistema operativo Mac OS y otra con Windows.
• Ambientes de desarrollo: Android Studio, Xcode y Firebase como plataforma de servicio externo.
• Dos dispositivos móviles, ambos con accesos a internet, uno con sistema operativo IOS y otro con Android.
Recursos financieros:
•Plataforma de servicios. El plan spark de Firebase permite usar las herramientas sin la necesidad de hacer un pago inicial.

3.7. 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. Se verifica el funcionamiento del prototipo. 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.

3.7.1. Herramientas externas

Para complementar el desarrollo de la aplicación y que cumpla con lo establecido se seleccionó una plataforma ofrecida por Google; Firebase[25], el cual ofrece distintos tipos de servicios para el desarrollo de aplicaciones móviles. Firebase a comparación de otros servicios, brinda un plan gratuito para utilizar sus componentes completamente para aplicaciones prototipo el cual da una ventaja sobre otros en gastos previos al desarrollo y ofrece más posibilidades de poder ampliar las capacidades del proyecto[26].

Se enlistan algunos de los servicios que ofrece la plataforma de Firebase y que se utilizan para el desarrollo del proyecto.

Cloud Firestore[27] es el servicio de base de datos NoSQL ubicada en la nube ofrecido por Firebase que ofrece sincronización en tiempo real para apps clientes. Esta base de datos se maneja a través de colecciones, documentos y datos, ver Figura 3.24.

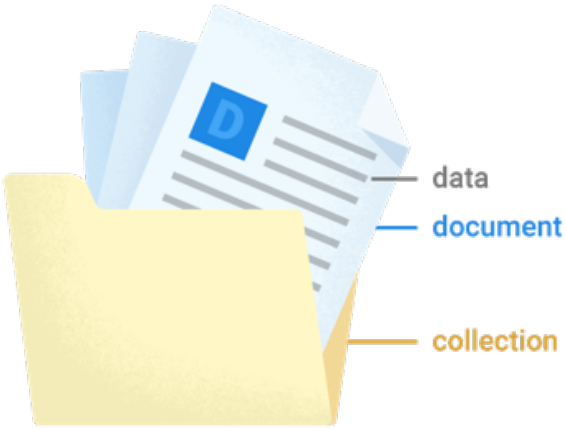


Figura 3.24: Representación de la base de datos Cloud Firestore.

La colección en la nube permite hacer apartados de secciones para separar los tipos de información, estas secciones son llamados documentos dentro del ámbito de Cloud Firestore el cual permite almacenar diferentes tipos de datos según lo requerido. Por último están los tipos de datos, Firestore permite usar datos desde un String hasta marcas de tiempo.

En la Figura 3.24. se muestra un ejemplo a grandes rasgos de cómo utilizar el servicio en la nube de Firestore aplicado al prototipo.

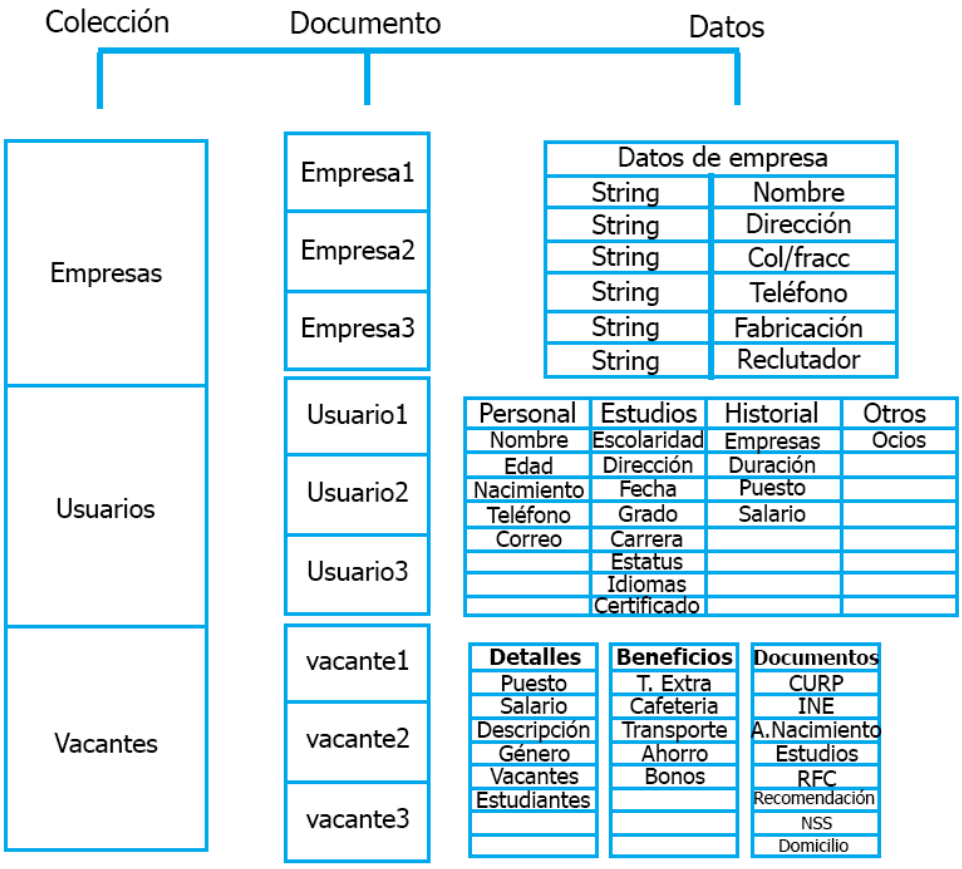


Figura 3.25: Representación de datos en Cloud Firestore.

Firebase Auth[26]: Un sistema de autenticación para el control de usuarios y estadísticas, esta herramienta da la oportunidad de iniciar sesión desde diferentes proveedores lo que permite expandir el modo de registro y mantener la seguridad de datos.

Figura 3.26: Cloud Messaging ofrece la autenticación de diferentes proveedores.

Firebase Cloud Messaging[26]: Permite integrar las notificaciones para informar al usuario de posibles actualizaciones de la aplicación o publicaciones de nuevas vacantes.

Firebase Cloud Functions[26]: Es la parte del Back-End para el desarrollo de aplicaciones, Functions ofrece distintas herramientas para poder gestionar la aplicación desde el servidor, ayuda durante el desarrollo a que los dispositivos no realicen las ejecuciones pesadas sino ejecutar una función en el back-end y que el dispositivo sea sólo un disparador.

Firebase Storage[29]: Es un servicio de almacenamiento que permite la subida y descargas de diferentes tipos de archivos de imágenes, videos, audio y otros tipos de contenido. Al ser un servicio integrado de Firebase permite la interacción directa de contenido entre el servidor y apps clientes.

3.7.2. Creación del proyecto en Firebase

Para obtener los permisos de uso de la plataforma de Firebase sólo es necesario acceder al navegador con una cuenta de Google. Dentro de la plataforma se encuentra una consola que es el manejador de los diferentes proyectos de Firebase desde el cual se crea uno para configurar el proyecto.

Figura 3.27: Creación del proyecto en Firebase.

Pasos para la creación de un proyecto en Firebase desde la consola de la plataforma

El proyecto se creó con el nombre de Vacantes Juárez

Figura 3.28: Creación del proyecto en Firebase paso 1.

En el segundo paso de la creación del proyecto, Firebase solicita si se desea usar los servicios de Google Analytics el cual se agregó al proyecto debido a que Google Analytics

ofrece diferentes herramientas para dar seguimiento a la aplicación ya sea en su escala o en los fallos presentados durante su producción.

Figura 3.29: Creación del proyecto en Firebase paso 2.

Como parte final se asocia una cuenta de Google Analytics al proyecto para la compartición de datos de estadística entre Firebase y Analytics.

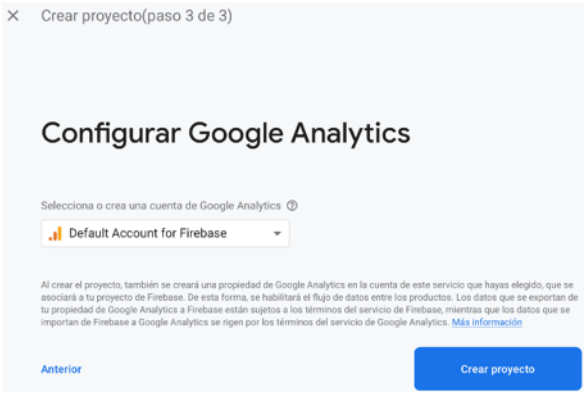


Figura 3.30: Creación del proyecto en Firebase paso 3.

Al terminar los pasos para crear un proyecto en Firebase se habilitan las herramientas de

base de datos en la nube, autenticación, almacenamiento y otras más con las que cuenta el servicio.

Herramientas de Analíticas de Google.

Google Analytics es una herramienta integrada dentro de la plataforma de Firebase, te permite revisar el escalamiento de la aplicación conforme a los nuevos usuarios registrados según tiempos determinados, proporciona estadísticas de comportamiento de los usuarios mientras interactúan con las funciones de la aplicación.

Las herramientas para usar dentro de analíticas son las siguientes:

• Firebase Crashlytics: Es una herramienta que permite de obtener informes de los errores de la aplicación mientras interactúa con los servicios de Firebase. La obtención de fallas por medio de esta herramienta se da en un tiempo real dentro de la plataforma y proporciona datos como:

• La cantidad de usuarios afectados

• Actividad afectada dentro de la aplicación

• La gravedad de la falla.

• El código que provoca el error.

Al usar esta herramienta se espera diagnosticar con tiempo las fallas en la aplicación en caso de presentarse y poder solucionarlas lo antes posible.

3.7.3. Desarrollo para Android

Como inicio del proceso de desarrollo para la plataforma Android se desglosan los pasos a realizar con base a las especificaciones y diseño implementadas.

En esta etapa se realizan las actividades que especifican los pasos a seguir durante el desarrollo del proyecto para la plataforma de Android. Por medio de varios subtemas se

muestran las acciones que permiten empezar el producto específico para concluir con el prototipo final.

Inicio de proyecto en Android Studio

Al abrir el compilador de Android Studio se selecciona la opción de crear un nuevo proyecto, esta acción manda a una vista donde se muestran varios diseños iniciales con los que puede empezar el proyecto, estos diseños son opcionales a elegir, pero permiten el ahorrar tiempo en la programación de varios aspectos.

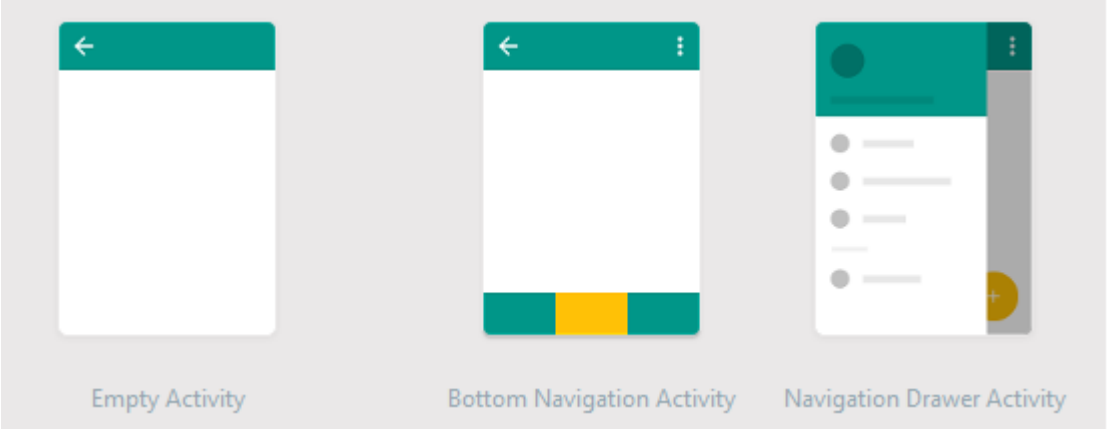


Figura 3.31: Proyectos preestablecidos.

Diseños prestablecidos en Android Studio

Empty Activity: Como su nombre lo indica es un proyecto con una actividad vacía el cual es utilizado para ir creando desde cero todo el desarrollo.

Bottom Navigation Activity: Este tipo de actividad ya viene con una configuración para utilizar un menú ubicado en la parte inferior de la pantalla, cuenta con código inicial que permite ir partiendo desde dicho menú.

Navigation Drawer Activity: A comparación del Bottom Navigation esta actividad cuenta con un menú lateral izquierdo, es de los más utilizados en las aplicaciones y viene ya configurado con vistas y código inicial.

Para fines del proyecto se seleccionó partir con el Bottom Navigation Activity para ahorra tiempo en el desarrollo de un menú y que además se adapta a las necesidades del desarrollo del producto.

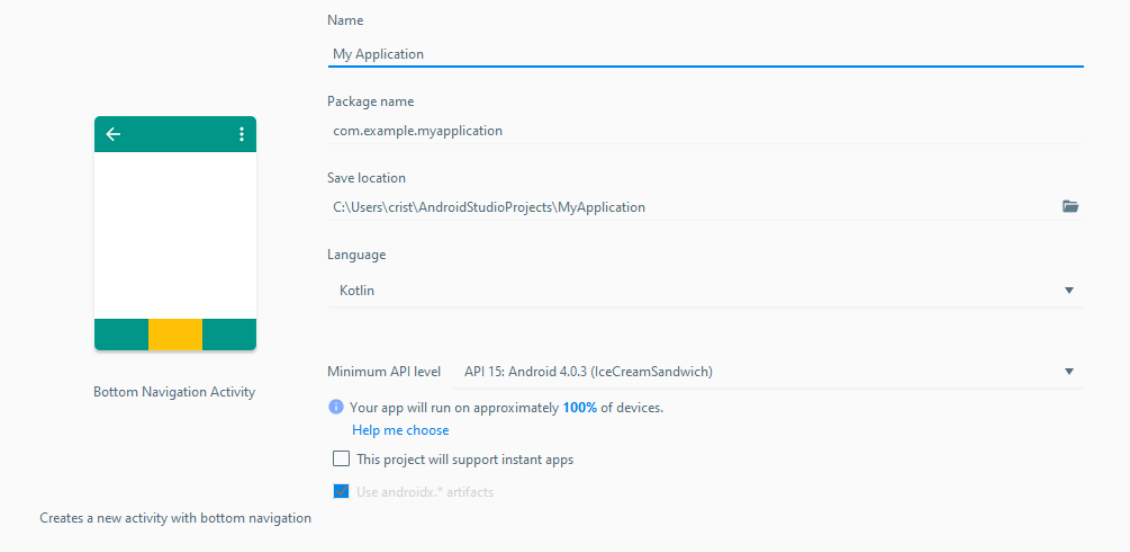


Figura 3.32: Configuración del proyecto.

Como segundo y último paso fue definir el entorno de desarrollo, el compilador pide seleccionar varias opciones entre ellas el nombre del proyecto, un identificador e incluso el lenguaje de programación con el que se desea trabajar. ver Figura 3.31.

Configuración del proyecto:

Nombre: Se ingresa el nombre con el que será llamado todo el proyecto creado, se puede modificar posteriormente.

Nombre de paquete: Es el identificador del proyecto, es único para cada aplicación de Android y no se puede modificar. Este identificador tiene una estructura, Android Studio pide como parámetro inicial indicar un dominio, en general es usado 'com' para después seguir con el nombre de la empresa y por último un nombre de proyecto, todo esto separados por puntos. **com.valtiz.vacantesjuarez**

Lenguaje: Android Studio permite elegir entre los lenguajes de Java y Kotlin. Debido a que Kotlin es un lenguaje joven y no cuenta con suficiente documentación y foros de ayuda se elige el lenguaje Java que como se sabe es un lenguaje de mucho uso y cuenta con la documentación necesaria para trabajar en aplicaciones de Android.

Nivel Mínimo de Api: Se indica la versión mínima a trabajar para el desarrollo de la aplicación el cual se especifica el Api 17 con versión de Android 4.2, quiere decir que un dispositivo con una versión anterior ya no será compatible con la aplicación.

Con estos pasos se finaliza la creación del proyecto configurado con una vista personalizada.

Agregar Firebase al proyecto

Se inicia la integración de nuestro servidor al proyecto, se especifican los pasos para agregar las dependencias y hacer la conexión.

Para agregar un proyecto dentro de la consola de Firebase se debe especificar un nombre de aplicación con el cual se identifica al producto dentro de la plataforma además de su

identificador único que es el nombre de paquete específico para cada aplicación, ver Figura 3.32.

Al final del registro se requiere ingresar una clave SHA1 para poder utilizar el servicio de autenticación de Google.

Figura 3.33: Registro de una aplicación Android en Firebase.

Después de registrar la aplicación en Firebase se genera un archivo json el cual se agrega al proyecto, sirve para identificar la aplicación dentro de la plataforma a través de datos de configuración.

```
{
  "project_info": {
    "project_number": "645938247507",
    "firebase_url": "https://empleos-juarez-maquilas.firebaseio.com",
    "project_id": "empleos-juarez-maquilas",
    "storage_bucket": "empleos-juarez-maquilas.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:645938247507:android:c31528101b97bd76",
        "android_client_info": {
          "package_name": "com.valtiz.empleosjuarez"
        }
```


Figura 3.34: Archivo Json generado por Firebase.

El archivo Json, ver Figura 3.34. Fue que ser agregado a la subcarpeta 'app' dentro del proyecto de Android.

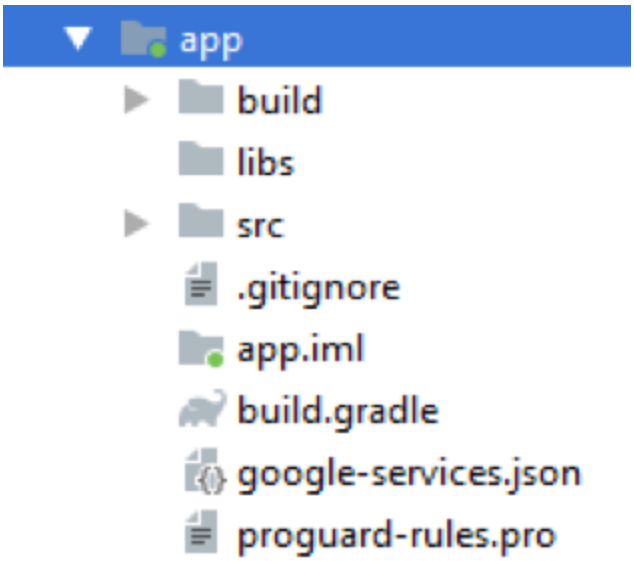


Figura 3.35: Ubicación del archivo Json.

Al agregar el archivo de configuración google-services se modifica el build.gradle de Android para utilizarlo el complemento. Para esto es necesario agregar un un path dentro de las dependencias del proyecto como se muestra en la Figura 3.35.

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.2'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

Figura 3.36: Integración de los servicios de Google al path.

Por consiguiente, se agregan las librerías correspondientes al uso de Google Analitycs junto al indicador para aplicar el plugin de Firebase.

```
apply plugin: 'com.android.application'

dependencies {
  // Add the Firebase SDK for Google Analytics
  implementation 'com.google.firebase:firebase-analytics:17.2.0'
  // Add SDKs for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-lib
}
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

Figura 3.37: Implementación de la librería Analitycs y el plugin de los servicios de Google.

La aplicación quedo configurada dentro de la plataforma de Firebase por lo que la conexión dentro entre el servidor y la aplicación se concluyó con éxito.

Propiedades de XML

Un archivo xml en Android tienes diferentes tipos de uso el cual el más importante

corresponde a que es el generador de las interfaces de usuario. Se desglosan algunas etiquetas más utilizadas dentro del desarrollo, ver Figura 3.38.

Estas etiquetas se utilizan exclusivamente para el diseño de las interfaces o vistas de cada parte de la aplicación.

Recurso	Etiqueta
Texto	<TextView/>
Botón	<Button/>
Imagen	<ImageView/>
Texto plano	<EditText/>
RadioGroup	<RadioGroup/>
Botón de radio	<RadioButton/>
Líneas de guía	<GuideLine/>
Diseño Linear	<LinearLayout/>
Diseño Constraint	<ConstraintLayout/>

Figura 3.38: Etiquetas de recursos en XML.

Con los recursos de xml se crea las vistas de usuario y los elementos con los que se interactua, como se ve en la Figura 3.37, cada etiqueta contiene una serie de propiedades dependiendo del tipo de elemento que ayuda a ir generando componentes completos.

Vista de Autenticación de usuario

Como primera parte de diseño de la aplicación móvil se desarrolla una interfaz que permita al usuario registrarse y poder hacer uso del sistema. Debido a que en los dispositivos de Android se encuentra una cuenta de Google el cual es necesaria para poder usar el móvil con sus servicios se asume que utilizar Gmail como método de inicio de sesión es el adecuado.

A comparación de iOS en la autenticación de Android no es necesario crear interfaz de

ingresos de correo y contraseña por lo que sólo es necesario agregar un botón el cual se encargará de realizar todo el proceso de registro, ver Figura 3.39.

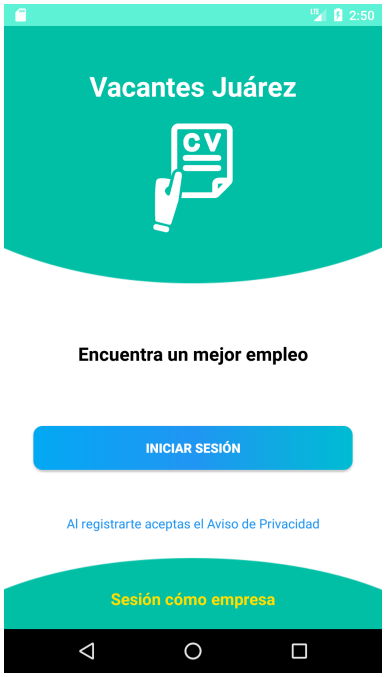


Figura 3.39: Interfaz de autenticación del usuario.

La interfaz es creada desde un archivo XML el cual usa etiquetas para cada parte que corresponda a una vista ya sea un botón, imágenes o texto. Para definir un botón usando las etiquetas de XML se necesita el código de la Figura 3.39.

```
<Button
    android:id="@+id/btn_registro"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:background="@drawable/button_login"
    android:drawableStart="@mipmap/google_icon"
    android:paddingStart="20sp"
    android:paddingEnd="20sp"
    android:text="Iniciar Sesión"
    android:textColor="#fff"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/guideline34"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline35" />
```

Figura 3.40: Botón en XML.

Como XML se maneja con marcas o etiquetas, para hacer uso de un botón es necesario crear una etiqueta con su correspondiente tipo, dentro de la marca se agregan diferentes tipos de propiedades correspondientes a un botón entre ellas las más usadas son:

android:id. Lo que indica esta propiedad es asignar un identificador para poder ser localizado dentro del código en Java y para diferenciar entre diferentes elementos con la misma marca.
android:background. Esta opción permite modificar el color del botón ya sea agregando un recurso como imagen o un color sólido.
android:text. Sirve para ingresar un nombre visible al frente del botón su parámetro es un string.

Estas propiedades están presentes en diferentes tipos de marcas como imágenes y texto, no se pueden repetir dentro de un mismo elemento y deben estar completamente llenos según sea su parámetro.

Las propiedades dentro de las etiquetas en el XML provienen de los recursos de Android por lo que es necesario tener dentro del archivo XML un llamado a estos recursos, ver Figura 3.41.

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

Figura 3.41: Recursos de Android en XML.

Como se muestra en la Figura 3.39, la interfaz de registro cuenta con una imagen prediseñada en Photoshop junto con un texto superior indicando el nombre de la aplicación y en la parte inferior el botón que será el encargado de hacer el registro del usuario a través del método por Google.

Funcionalidad de la Autenticación

En el tema anterior se hace referencia a la creación de la interfaz utilizando recursos de Android, por consiguiente es implementar la funcionalidad para que el botón de inicio de sesión de la Figura 3.41 realice la acción de registrar a un usuario a la plataforma de Firebase con la autenticación de Google.

Para hacer uso del servicio de la autenticación por medio del sistema de Google es necesario habilitar el método en la consola de Firebase como se muestra en la Figura 3.39 el cual permite a la aplicación acceder al servicio.

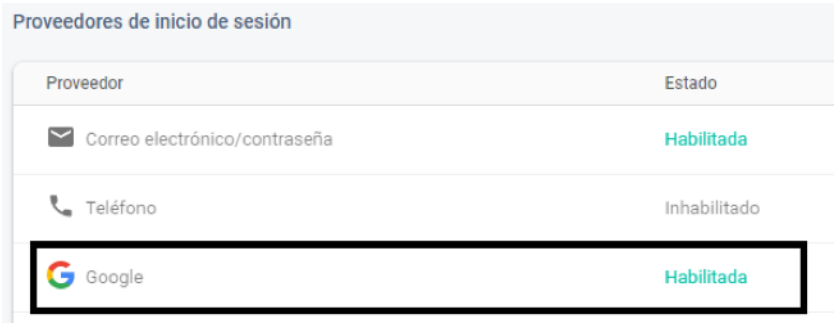


Figura 3.42: Proveedores de inicio de sesión.

Para poder implementar el método de autenticación por Google es necesario obtener las credenciales de la Api de servicios del proyecto, para ello es necesario acceder a Google Cloud Platform, este servicio se habilita con una cuenta de Gmail. Dentro de la plataforma de Google en el apartado de credenciales se obtiene un Id de cliente y secreto de cliente ver Figura 3.42 el cual sirven para poder habilitar el método de autenticación.

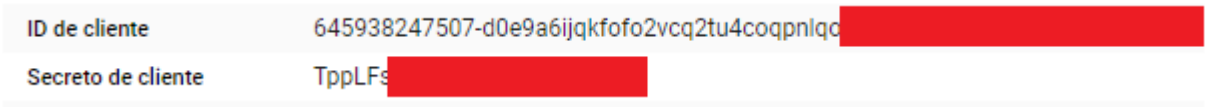


Figura 3.43: Id de cliente y secreto de cliente.

Cada servicio de Google se maneja por medio de dependencias para su implementación en el proyecto de Android, la dependencia correspondiente al servicio de autenticación se agrega al archivo build.gradle donde se encuentran las demás librerías, ver Figura 3.44 parte A.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'androidx.vectordrawable:vectordrawable:1.1.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
A)  implementation 'com.google.firebase:firebase-auth:19.0.0'
    implementation 'com.google.firebase:firebase-analytics:17.2.0'
    testImplementation 'junit:junit:4.12'
B)  implementation 'com.google.android.gms:play-services-auth:17.0.0'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
}
```

Figura 3.44: Dependencias del proyecto.

Debido a que hay una conexión entre el servidor de Firebase con el método de acceso de Google es necesario agregar la dependencia correspondiente a los servicios de Google enfocado a la autenticación, ver Figura 3.44 parte B.

Iniciando la programación de la autenticación como primera parte es necesario crear un archivo Java donde se hará el proceso de registro.

A continuación, se enlista el proceso de desarrollo de la autenticación para Android utilizando el método de acceso por Google:

•Se implementa el evento onClickListener que permita llamar al método encargado de procesar la sesión con Google todo esto utilizando el botón inicio de sesión creado en la interfaz, ver Figura 3.45.

```
@Override
public void onClick(View v) {
    int i = v.getId();
    if (i == R.id.btn_registro) {
        signIn();
    }
}
```

Figura 3.45: Método onClick.

Después se conFigura la autenticación de Google (ver Figura 3.46), por lo que es necesario obtener un token, que es un id de cliente de servidor el cual permite a la aplicación obtener la información de contacto de la cuenta de Google del usuario.

```
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken("645938247507-d0e9a6ijqkfofo2vcq2tu4coqpnlqoe0.apps.goog...")
        .requestEmail()
        .build();
```

Figura 3.46: Acceso al token de cliente.

El token de cliente se obtiene de las credenciales proporcionadas por la plataforma en la nube de Google (GCP) ver Figura 3.42 estas credenciales se crean al momento de configurar un nuevo proyecto en la plataforma Firebase.

• Al completarse la conexión con el servidor de Google, se le pasa la acción al botón de inicio de sesión para que obtenga el token y pueda continuar con la sesión. Se crea un método llamado SignIn el cual será llamado al momento de presionar el botón, este obtendrá una instancia para después pasarlo al método onActivityResult, ver Figura 3.47.

```
private void signIn() {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            // Google Sign In was successful, authenticate with Firebase
            GoogleSignInAccount account = task.getResult(ApiException.class);
            assert account != null;
            firebaseAuthWithGoogle(account);
        } catch (ApiException e) {
            // Google Sign In failed, update UI appropriately
            Log.w(TAG,  msg: "Google sign in failed", e);
            // ...
        }
    }
}
```

Figura 3.47: Métodos para la obtención de la sesión con Google.

Se crean las variables a utilizar durante el llamado de sesión (ver Figura 3.48) el cual son colocadas como globales para acceder a ellas dentro de los métodos.

```
private static final String TAG = "GoogleActivity";
private static final int RC_SIGN_IN = 9001;
private FirebaseAuth mAuth;
private GoogleSignInClient mGoogleSignInClient;
```

Figura 3.48: Variables de inicio de sesión.

La variable TAG se utiliza sólo para nombrar a la actividad lanzada por Google al momento de solicitar el permiso para tomar acceso a la cuenta del usuario.

RC-SIGN-IN es una llave que utiliza Google para saber que la solicitud del código del método SignIn mostrado en la Figura 3.49 es para obtener un token resultante que de permiso

a la aplicación de acceder a las cuentas del usuario.

La variable mAuth de FirebaseAuth es creada para obtener una instancia y saber si el usuario ya cuenta con una sesión activa.

MGoogleSingInClient obtiene el token de acceso del servidor de Google para acceder y obtener una sesión con la plataforma.

Al concluir de realizar este proceso, es necesario saber si la conexión con el servidor de Google fue exitosa para la obtención de la cuenta del usuario y poderlo registrar dentro del proyecto en la plataforma de Firebase.

```
private void firebaseAuthWithGoogle(GoogleSignInAccount acct) {
    Log.d(TAG,  msg: "firebaseAuthWithGoogle:" + acct.getId());

    AuthCredential credential = GoogleAuthProvider.getCredential(acct.getIdToken(),  s1: null);
    mAuth.signInWithCredential(credential)
            .addOnCompleteListener( activity: this, (task) → {
                    if (task.isSuccessful()) {
                        // Sign in success, update UI with the signed-in user's information
                        Log.d(TAG,  msg: "signInWithCredential:success");
                        FirebaseUser user = mAuth.getCurrentUser();
                        updateUI(user);
                    } else {
                        // If sign in fails, display a message to the user.
                        Log.w(TAG,  msg: "signInWithCredential:failure", task.getException());
                       // Snackbar.make(findViewById(R.id.main_layout), "Authentication Failed.", Snackbar.LENGTH_SHORT).show();
                        updateUI( user: null);
                    }

                    // ...
            });
}
```

Figura 3.49: Método de registro en la plataforma de Firebase.

El método FirebaseAuthWithGoogle define si la sesión creada anteriormente fue exitosa y con ello poder registrar al usuario en la plataforma de Firebase mandando el correo de Google del usuario al sistema de autenticación y generando un id único para el usuario registrado.

Currículum virtual de usuario

El usuario al estar registrado correctamente se le posiciona en la vista de completar su currículum vitae donde la primera interfaz representa los datos personales del usuario como se ve en la Figura 3.50A.

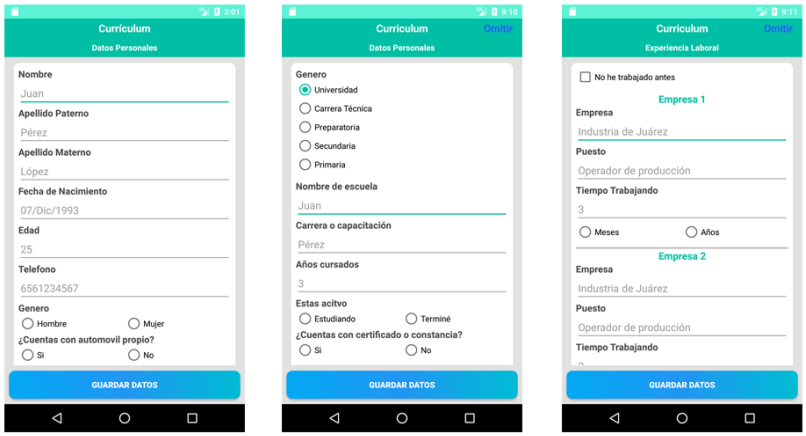


Figura 3.50: Vistas del currículum virtual.

Los datos personales a completar el usuario en la primera interfaz del currículum se muestran en la Figura 3.50

Personal
Nombre
Apellido paterno
Apellido Materno
Fecha de nacimiento
Edad
Teléfono
Genero
Automóvil
Diseño Constraint

Figura 3.51: Datos personales del currículo.

Por último se especifica los datos de domicilio para concluir con la primera parte de la información del currículo, ver Figura 3.51.

Domicilio
Calle
Numero
Colonia o Fraccionamiento
Código postal

Figura 3.52: Datos de domicilio del currículo.

Para continuar con la siguiente parte del currículum virtual el usuario debe completar toda la información mencionada anteriormente. Esta parte se puede omitir y continuar directamente al menú principal de la aplicación dejando pendiente el completado del currículum.

Currículum virtual de usuario parte 2.

La siguiente parte del currículum virtual es el diseño de la interfaz donde el usuario ingresará los datos correspondientes a sus estudios.
Utilizando el componente RadioGroup, se almacena dentro las opciones de estudios del usuario utilizando los RadioButton, donde el usuario sólo podrá indicar una sola opción según su último grado académico. Utilizando TextViews y EditText se terminan el formato para ingresar los datos de su grado académico. Ver Figura 3.52.

Estudios
Nombre
Carrera
Duración
Activo
Certificación

A)

Grado académico
Universidad
Carrera técnica
Preparatoria
Secundaria
Primaria

B)

Otros estudios
Idiomas
Certificaciones

C)

Figura 3.53: Datos del formato de estudios.

Currículum virtual de usuario parte 3

La última parte del currículum es el diseño de la interfaz orientada al formato de experiencia laboral, donde el usuario pueda ingresar algunos datos importantes que denote sus anteriores trabajos, ver Figura 3.49C. En este formato se utilizan TextViews y EditTexts para que el usuario pueda ingresar los datos de sus antiguos trabajos.

El formato del currículum virtual es necesario que el usuario lo complete en su totalidad a excepción de aquellos datos que el usuario carece. En caso de que esta parte sea omitida en el registro se podrá volver acceder a ella desde el apartado de perfil del usuario.

Configuración del menú principal

Definir un estilo del menú de una aplicación es importante por el echo de que es la manera en que el usuario se moverá de entre las diferentes vistas. En el capítulo inicio de proyecto en Android Studio se menciona el formato de la aplicación. El estilo Bottom Navigation permite navegar desde sus opciones ubicadas en la parte inferior.

Se conFigura las opciones del Bottom Navigation para ajustar las vistas de la aplicación en el siguiente orden de izquierda a derecha: Vacantes, Empresas registradas, Estatus, Perfil y ayuda, ver Figura 3.54.

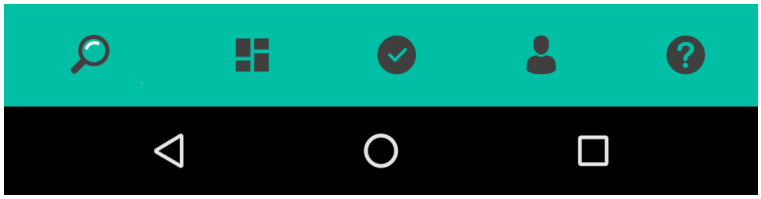


Figura 3.54: Bottom Navigation configurado.

Cada opción del Botton Navigation funciona como un botón por lo que se debe crear las instancias para que cada uno pueda llamar a su interfaz correspondiente. Un FragmentActivity es una clase con soporte para fragmentos por medio de transiciones, lo que quiere decir que es posible mantener dentro de una sola clase diferentes fragmentos sin necesidad de crear

diferentes clases para cada interfaz.

El Botton Navigation es un diseño perteneciente a la clase MainActivity, el cual es la encargada de llamar a cada FragmentActivity según la opción seleccionada del menú.

```
private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
    = new BottomNavigationView.OnNavigationItemSelectedListener() {

  @Override
  public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    Fragment fragment = null;
    switch (item.getItemId()) {
      case R.id.navigation_vacantes:
        fragment = new UsuarioVacantes();
        break;
      case R.id.navigation_empresas:
        fragment = new UsuarioEmpresasRegistradas();
        break;
      case R.id.navigation_estatus:
        fragment = new UsuarioEstatus();
        break;
      case R.id.navigation_perfil:
        fragment = new UsuarioPerfil();
        break;
    }
    return loadFragment(fragment);
  }
};
```

Figura 3.55: Fragments Activitys en el método Bottom Navigation.

Como se muestra en la Figura 3.55 dentro del método que acciona el Botton Navigation se hace el llamado de cada uno de los Fragment Acitvity para sobreponer sobre la vista principal la interfaz que fue seleccionada.

El utilizar estos Fragments permite seguir manteniendo el Botton Navigation aún cuando la vista principal ha cambiado.

Interfaz Vacantes.

La vista donde el usuario podrá ver todas las vacantes disponibles es la primera en el Botton Navigation. Se desarrolló con un formato de tipo lista para desplegar hacia abajo todas las ofertas de empleo publicadas. Para implementar la lista se utiliza un clase llamada RecyclerView para cual es necesario agregar una dependencia al proyecto para poder llamarla. Ver Figura 3.56.

```
implementation 'androidx.recyclerview:recyclerview:1.0.0'
```

Figura 3.56: Dependencia del RecyclerView.

Después de que se implementó la dependencia, en el xml archivo correspondiente a la interfaz se agregó el componente para hacer el llamado al RecyclerView, ver Figura 3.57.

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline9" />
```

Figura 3.57: Componente RecyclerView en xml.

Con el componente agregado al archivo xml se hace la referencia desde la clase Usuarios-Vacantes para poder llamarlo. Para instanciar un componente del archivo xml desde la clase se utiliza un método llamado findViewById que como su nombre lo indica busca el elemento por medio de su id, en este caso el elemento RecyclerView tiene un identificador de nombre recyclerView.

Para trabajar con la lista es necesario crear una nueva clase de tipo adapter de nombre UsuarioVacantesAdapter el cual sirve para manejar los elementos que serán mostrados en el RecyclerView. Código en el anexo A.

Desde la clase UsuarioVacantes se crearon siete variables de tipo ArrayList<String>en el cual su entrada es de sólo cadenas que se utilizan para pasar al adapter los elementos a mostrar en la interfaz.

Para llenar los ArrayList se obtiene la información desde la base de datos Firebase para esto se agrega el método addOnCompleteListener que lee los datos una sola vez y se asignan a las variables, ver Figura 3.57.

```
DocumentReference docRef = db.collection("cities").document("SF");
docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
  @Override
  public void onComplete(@NonNull Task<DocumentSnapshot> task) {
    if (task.isSuccessful()) {
      DocumentSnapshot document = task.getResult();
      if (document.exists()) {
        Log.d(TAG, "DocumentSnapshot data: " + document.getData());
      } else {
        Log.d(TAG, "No such document");
      }
    } else {
      Log.d(TAG, "get failed with ", task.getException());
    }
  }
});
```

Figura 3.58: Leer desde la base de datos.

Después de haber obtenido la información desde la base de datos esta se pasa al adapter para que puedan ser mostrados en la interfaz, ver Figura 3.58.

```
UsuarioVacantesAdapter mAdapter = new UsuarioVacantesAdapter(getActivity(),
nombreDeEmpresa, direccionDeEmpresa, vacantesDeEmpresa, puestoDeEmpleo,
descripcionVacante, postulantesDeEmpresa, fechaPublicacion);
mAdapter.setClickListener(this);
recyclerView.setAdapter(mAdapter);
```

Figura 3.59: Paso de variables al adapter.

Las variables son pasadas al adapter por lo que al cargar la información este genera la vista donde se mostraran todas las vacantes que haya obtenido desde la base datos, ver Figura 3.59.

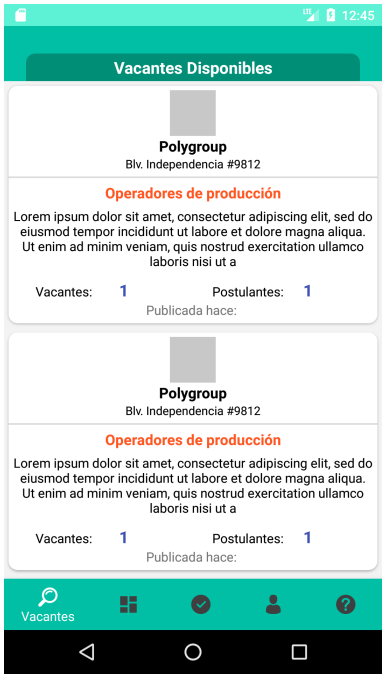


Figura 3.60: Interfaz Vacantes.

La interfaz muestra información relevante respecto a la vacante, el usuario puede observar datos como el tipo de puesto, la descripción y las vacantes disponibles y puede desplazar

hacia abajo para pasar por los diferentes empleos ofertados.

Al hacer presionar una vacante se llama a un Fragment Activity que contiene la descripción completa de la vacante y de donde el usuario puede postularse. Esta vista es de sólo lectura para el usuario por lo que se utilizó para su desarrollo de gran parte de TextViews dentro del contenedor de diseño LineraLayout, ver Figura 3.60

Figura 3.61: Interfaz descripción de vacante.

La información de la vacante es obtenida con la llamada del método addOnCompleteListener el cual es almacenada en distintas variables para luego ser mostradas a la interfaz.

Vista Empresas Registradas

La vista donde se muestran las empresas que están registradas en la aplicación esta basada en gran parte de la vista de vacantes por lo que se omite la explicación de los componentes.

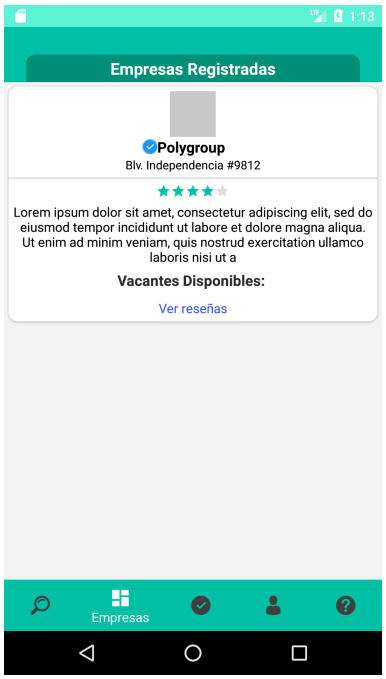


Figura 3.62: Interfaz Empresas registradas.

Utilizando un RecyclerView para implementar una lista se despliega en la interfaz las empresas que están registradas y verificadas dentro de la aplicación. La información son obtenidos desde el colección Empresas de la base de datos en Firebase. Se creó el adapter UsuarioEmpresasAdapter para pasar los datos y estos sean mostrados al usuario.

Al ser presionado una de las empresas mostradas en la interfaz, la clase UsuarioEmpresaRegistradas llama al Fragment Activity UsuarioDescripciónEmpresas el cual sustituye al interfaz de las empresas para mostrar la descripción completa de la empresa, ver Figura 3.62.

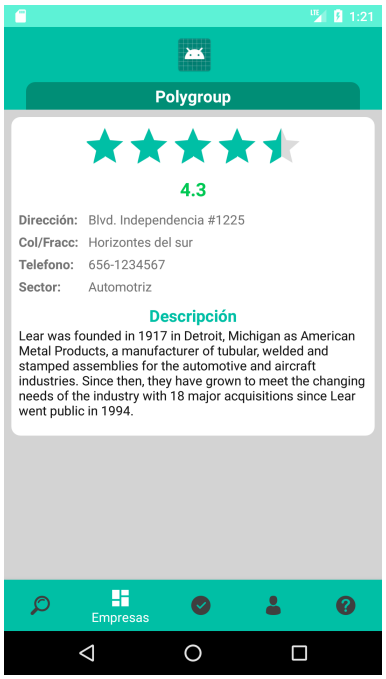


Figura 3.63: Interfaz Descripción de la empresa.

La descripción de la empresa está compuesta por un rating donde el usuario puede observar la calificación que entre todos los usuarios han elegido, los datos de domicilio y contacto de la empresa y una descripción sobre la historia o el sector de la empresa según corresponda. La vista de empresas registradas que es la segunda del Botton Navigation es sólo para informar a los usuarios sobre las empresas que han sido registradas en la aplicación.

Interfaz Estatus

La vista de estatus se refiere a que es la encargada de mostrar a los usuarios de cómo va el proceso de su solicitud respecto a la vacante donde se postuló. La interfaz utiliza en gran parte del mismo formato que la vista de vacantes por lo que no se explican algunos componentes.

El usuario al postularse en una vacante necesita llevar un seguimiento y como puede

aplicar para diferentes empleos es necesario mostrar todas sus solicitudes, para esto en la interfaz de estatus se utiliza el ya mencionado RecyclerView el cual ayuda a mostrar junto con su adapter todas las postulaciones del usuario, ver Figura 3.63.

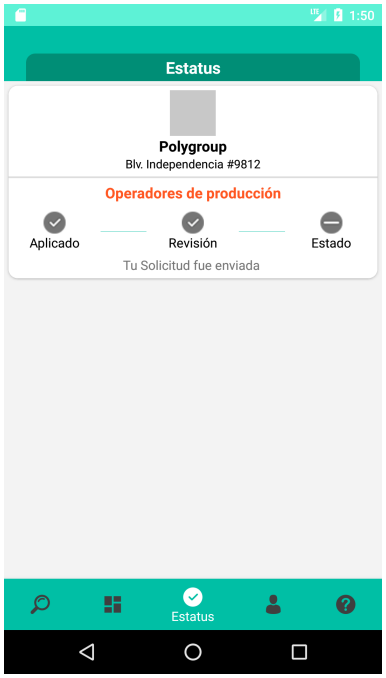


Figura 3.64: Interfaz Estatus.

La vista es la tercera del Botton Navigation y al estar implementada con un RecyclerView el usuario puede deplazar hacia abajo con respecto a la cantidad de postulacione que tenga. El estatus muestra tres fases:

•Aplicado. Indica que su solicitud fue enviada correctamente y que esta pendiente a ser revisada.

•Revisión. La solicitud del usuario se encuentra en revisión por uno de los encargados de la empresa.

•Estado. Esta fase le muestra al usuario si su solicitud fue aceptada o rechazada. Si el usuario es aceptado podrá ver información proporcionada por la empresa para presentarse y continuar

con su proceso de contratación.

Si es rechazado podrá ver el motivo por el cual no fue seleccionado en caso de que la empresa así lo decida.

Interfaz Perfil

La cuarta interfaz del Botton Navigation es el perfil del usuario, esta vista sólo muestra información básica del currículum, ver Figura 3.65

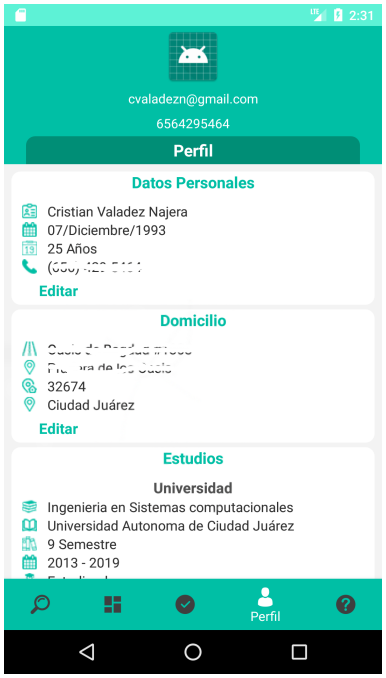


Figura 3.65: Interfaz Perfil.

Estos datos son traídos desde la colección Usuarios, obteniendo el documento del usuario con base a su identificador. La información se obtiene siempre y cuando el usuario haya completado su currículum, en caso contrario, se le muestra un botón el cual tiene como acción el enviar al usuario al Fragment Activity para completar su resumen.

En el capítulo de Currículum virtual puede observar más detalles sobre esta interfaz.

Interfaz ayuda

Es la última opción del Botton Navigation, la vista de ayuda muestra al usuario de que se aborda en cada interfaz de la aplicación, no hace conexión con la base de datos por lo que sólo es de aspecto visual su uso, ver Figura 3.65.

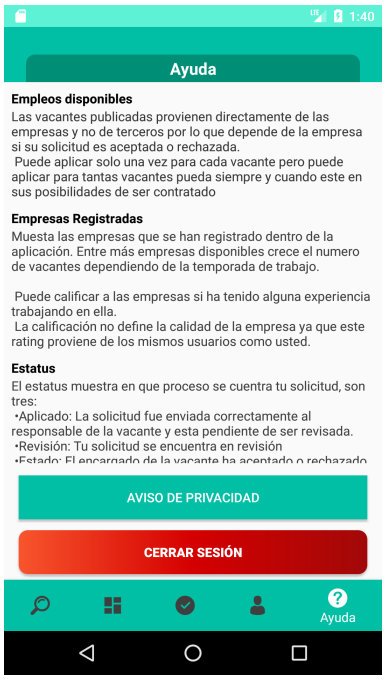


Figura 3.66: Interfaz Ayuda.

La interfaz cuenta con dos botones, uno hace referencia al aviso de privacidad del usuario y el otro cierra la sesión y es redirigido hacia la pantalla de inicio de sesión.

Vistas con enfoque a la empresa

El desarrollo de la aplicación implica trabajar con dos enfoques distintos, uno dirigido hacia los usuarios en busca de un trabajo y otro para las empresas encargadas de publicar las vacantes. Estos enfoques se definen como mantener un rol para cada tipo de usuario dentro de la aplicación sin la necesidad de tener que diseñar una aplicación diferente para cada caso.

En gran parte de los diseños para el desarrollo de las vistas de empresa son similares al que utiliza el usuario que busca un empleo, sólo se ajusta a las necesidades de la empresa para un mejor manejo de sus ofertas de empleo y la gestión de sus postulantes.

Sesión de empresas

Para poder publicar vacantes provenientes de una empresa es necesario que se registren como tal, para esto se crea sólo un perfil por maquiladora, los reclutadores utilizan este perfil único para poder crear y administrar las ofertas de empleo.

La interfaz se diseñó de una manera en que el reclutador inicie sesión después de haber sido verificada la empresa que registró. A diferencia de la vista del usuario esta si necesita un login por medio de correo y contraseña tomando en cuenta que estos datos pertenecen a la empresa y no son del reclutador, esto con la finalidad de que entre las personas encargadas del perfil puedan compartir la cuenta entre ellos.

Figura 3.67: Interfaz de Inicio de Sesión.

Como se ve en la Figura 3.66 la empresa inicia sesión con correo y contraseña esto siempre y cuando haya sido verificada por los administrados de la aplicación. Si la empresa aún no cuenta con su verificación el inicio de sesión no será permitido, esto con la intención de evitar la publicación de vacantes que probablemente no existan.

En la vista se de sesión se agregó un botón de registro para que empresas que son nuevas dentro de la aplicación puedan registrar sus datos y ser enviados al proceso de verificación.

Nota: El proceso de verificar las empresas se hace de manera manual, acudiendo con el reclutador directamente, como es externo al desarrollo de la aplicación no se agrega al contenido.

Registro de empresas

La aplicación esta orientada a trabajar con diferentes empresas por lo que se creó una interfaz que permita a nuevos reclutadores completar un formulario con datos básicos que permita a la aplicación crear un perfil para que pueda publicar vacantes.

La vista de registro se crear por medio de un formulario utilizando en su mayoría Text-Views y EdidText, ver Figura 3.67.

Figura 3.68: Interfaz de Registro.

El reclutador encargado de registrar la empresa debe completar en su totalidad el formulario que está compuesto por datos como: nombre de la empresa, dirección, nombre del reclutador, teléfono de contacto, además de ingresar un correo y contraseña para el inicio de sesión.

Menú Principal de empresas

Las empresas cuentan con un menú donde tienen la posibilidad de navegar entre las posibles opciones dentro de la aplicación. Se utiliza al igual que en la interfaz del usuario un Botton Navigation, el cual es un menú de opciones ubicadas en la parte inferior de la interfaz, ver Figura 3.68.

Figura 3.69: Interfaz de Registro.

El Botton Navigation esta compuesto por las siguientes opciones: Publicar, Administrar, Solicitudes, Perfil y Ayuda. Las opciones del menú permiten al reclutador navegar entre las diferentes partes de la aplicación ya que estan diseñadas por medio de Fragment Activity el cual permite sin cambiar de Activity ir por cada una de ellas evitando que la aplicación consuma más memoria.

Interfaz de Publicar vacante

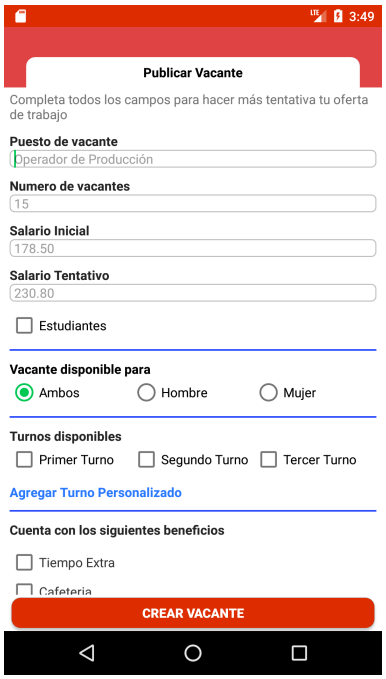


Figura 3.70: Interfaz Publicar vacante

La vista de publicar una vacante, ver Figura 3.68 esta diseñada con estilo de un formulario, el reclutador ingresa la información de su oferta de empleo como lo es el turno, puesto, horario, requisitos y habilidades.

El formulario está compuesto por Textviews para indicar al reclutador el tipo de campo que debe completar, los EditText que son para ingresar texto desde el teclado capturan los datos del reclutador y estos mantiene esa información mientras el usuario no salga de la vista debido a que es un Activity y se crea sobre el menú principal.

Interfaz Administrar vacantes

Es la vista encargada de mostrar al reclutador todas las vacantes que ha publicado, esta creada con un RecyclerView para aprovechar el formato de tipo lista que proporciona y

mostrar de forma vertical todas las ofertas de empleos creadas anteriormente.

El RecyclerView recibe datos basicos de las vacantes publicadas que se encuentran almacenadas en la base de datos ubicadas en la colección Vacantes. Para distinguir las vacantes de cada empresa, se le agrego a cada una el identificador único de cada empresa.

Al acceder a las vacantes las variables de tipo ArrayList ya comentadas anteriormente guardan los datos y se pasan al RecyclerView, este, instancia los elementos del xml y les pasa la información para ser mostrados en pantalla, ver Figura 3.69.

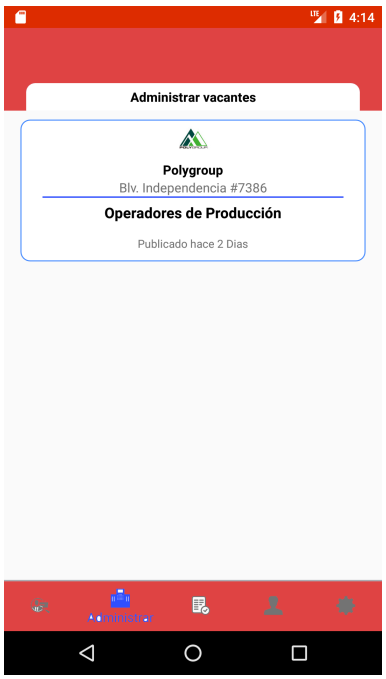


Figura 3.71: Interfaz Administrar vacantes.

Al presionar en una vacante se envía al usuario a la vista de administración que esta compuesta de dos partes, la primera hace referencia a como se ve la vacante en modo usuario ver Figura 3.60 y la segunda muestra la posibilidad de poder editar la vacante o darla de baja para que no se muestre más para los usuarios. Ver Figura 3.70

Interfaz Solicitudes

La empresa necesita gestionar las solicitudes recibidas por cada una de sus vacantes, la interfaz de solicitudes muestra al reclutador todas las solicitudes que ha recibido. Esta interfaz utiliza el mismo diseño de un RecyclerView el cual permite crear una lista vetical dependiendo del tamaño en este caso de solicitudes recibidas, ver Figura 3.72

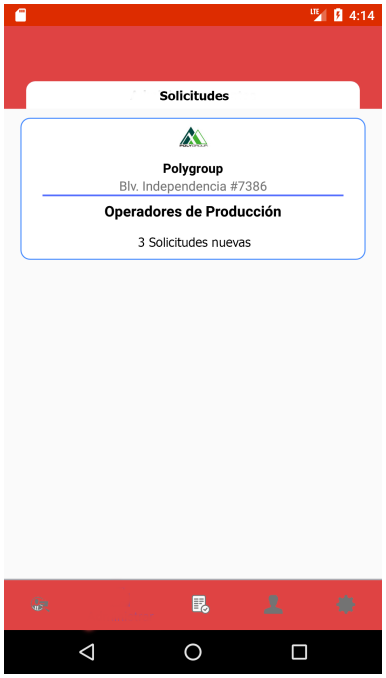


Figura 3.72: Interfaz Solicitudes.

En caso de haber solicitudes nuevas serán mostradas en la vacante disponible.

Interfaz Perfil de la empresa

La vista perfil de la empresa esta relacionada con la vista de usuario de empresa registradas ver Figura 3.62 donde la empresa puede ver su información registrada además de su valoración asignada por los usuarios.

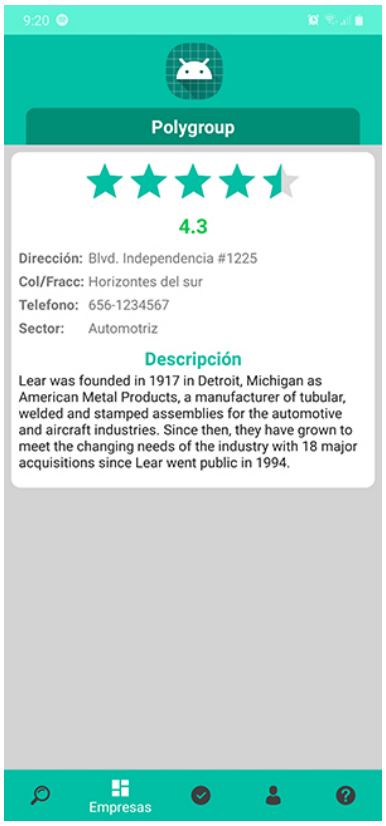


Figura 3.73: Interfaz Perfil de empresa.

Interfaz Ayuda

La última vista fue diseñada para ayudar a las empresas a manejar las diferentes opciones del menú de la aplicación, esta compuesta sólo de textos informativos utilizando los Textviews, además de tener el botón para cerrar la sesión en el dispositivo, ver Figura 3.74.

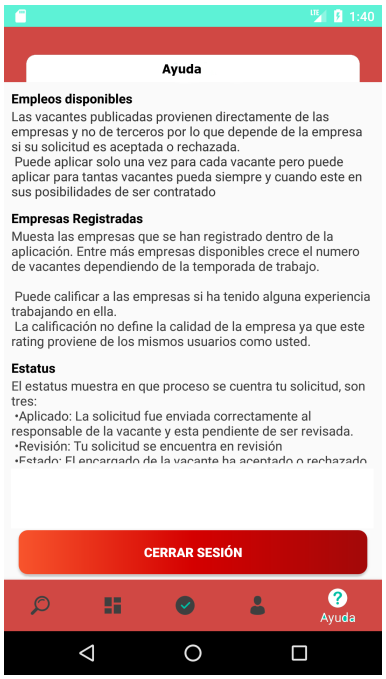


Figura 3.74: Interfaz Ayuda.

El código de todas las interfaces se puede consultar en el anexo A dirigido al desarrollo para Android.

3.7.4. Desarrollo para IOS

Siguiendo la segunda etapa del desarrollo del proyecto siendo desarrollada en paralelo con la aplicación de Android, abarca los pasos de la metodología de prototipos-alta fidelidad aplicados al proceso en IOS dentro del entorno de desarrollo Xcode.

Creación del proyecto

En esta etapa se realizan las actividades que especifican los pasos seguidos durante el desarrollo del proyecto para la plataforma IOS. Por medio de varios subtemas se desglosan las acciones que permiten empezar el producto específico para concluir con el prototipo final.

Figura 3.75: Creación del proyecto en Xcode.

Se creó un nuevo proyecto de Xcode, ver Figura 3.75, se creó con el nombre Empleos Juárez para tenerlo identificado.

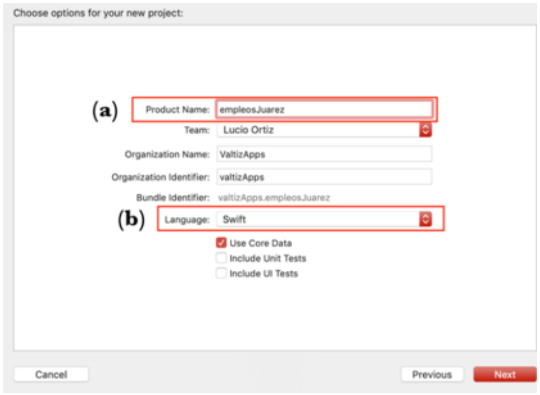


Figura 3.76: Creación del proyecto en Xcode paso 1.

- El "Product Name:" es el que indica el nombre del proyecto como se muestra en la Figura 3.76(a), en esta parte de la creación del proyecto, también se elige el lenguaje de

programación del mismo, en este caso es swift Figura 3.76(b) debido a las facilidades que tiene el lenguaje en la creación de aplicaciones para IOS.

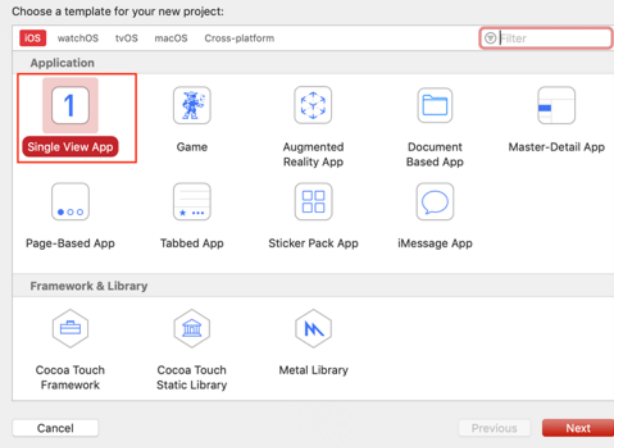


Figura 3.77: Creación del proyecto en Xcode paso 2.

Se creó una aplicación de una sola vista, ver Figura 3.77, ya que es más fácil el diseño de la aplicación si se empieza con una sola vista y ayuda a tener mejor organizado el espacio de trabajo conforme se avanza en el proyecto.

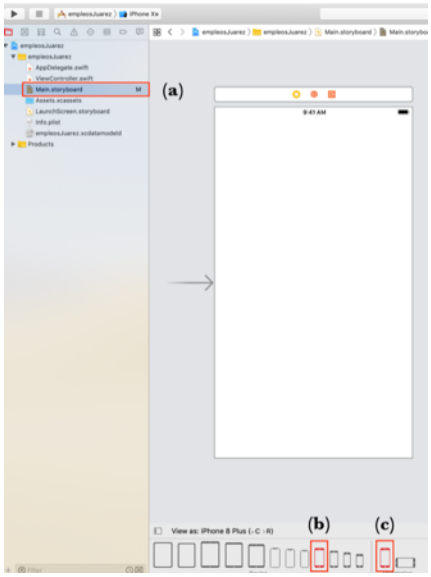


Figura 3.78: Orientación y tamaño de pantalla.

Desde el "Main StoryBoard" de la Figura 3.78(a). Se cambia el dispositivo por una pantalla más intermedia como se muestra en la Figura 3.78(b), debido al manejo de los constrains que genera Xcode para los diferentes tamaños de Iphone y manejar una pantalla que este en el medio de todos los dispositivos de la familia Iphone, también se selecionó el modo Portrait el cual indica la orientación de la pantalla en vertical, no es necesario que la aplicación se vea en horizontal por el formato de las vistas.

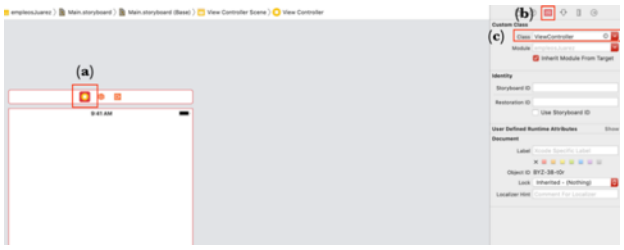


Figura 3.79: Conectar ViewController con la vista.

Lo siguiente es verificar la conexión de la vista con el archivo ViewController el cual controla desde código la interacción del usuario con la vista, lo cual, significa que para cada vista se debe tener un ViewController el cual es el encargado de identificar y hacer las acciones necesarias para el funcionamiento de la vista, para poder verificar esto se selecciona la vista como se muestra en la Figura 3.79(a), y enseguida se seleccionan las clases con las que se relaciona está vista Figura 3.79(b), y en el apartado Class se muestra la clase con la cual está enlazado.

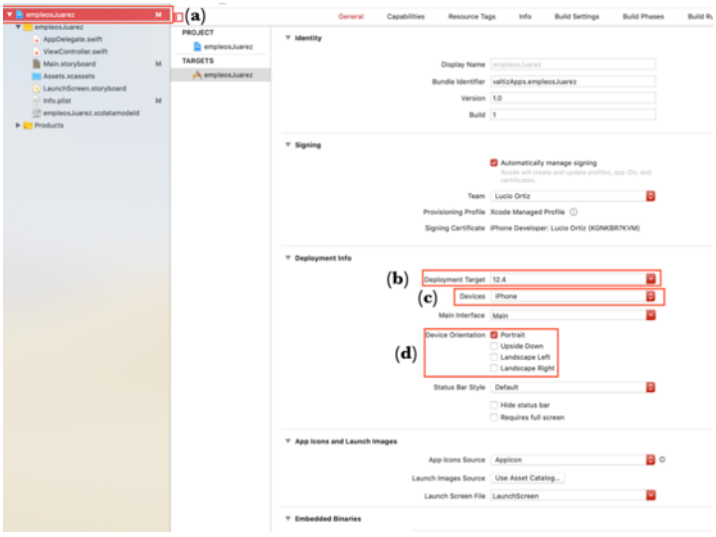


Figura 3.80: Configuración final del proyecto en Xcode.

Una vez terminado de configurar la parte inicial del proyecto, lo siguiente es especificar el objetivo del desarrollo para el IOS actual el cual se muestra en la Figura 3.80(b), indica el sistema operativo mínimo del dispositivo Iphone para poder descargarlo e instalarlo. Además se seleccionó de dispositivo único el Iphone ya que la aplicación no se desarrollará para el Ipad, si se hiciera esto retrasaría el desarrollo, Figura 3.80(c). En esta sección también se limita la orientación del dispositivo siendo la única orientación marcada la vertical, ya que las vistas no se programaron para la orientación en horizontal, Figura 3.80(d).

Agregar Firebase al proyecto

Lo primero para agregar firebase al proyecto de Xcode es, ingresar a la configuración de la base de datos y ahí añadir una nueva aplicación, la cual en este caso es IOS.

Figura 3.81: Agregar Firebase al proyecto de Xcode paso 1.

Lo siguiente fue ingresar el identificador de compilación de la aplicación a la BD, este debe de ser el mismo que el de la aplicación de Xcode ya que si no la integración de Firebase con la aplicación no sucedería, ver Figura 3.81.

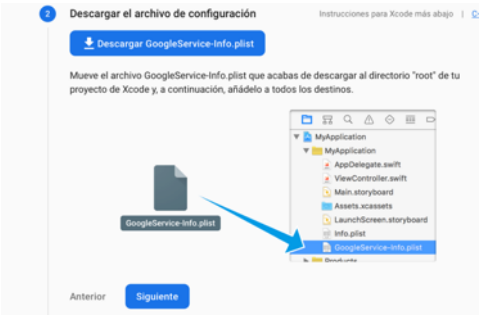


Figura 3.82: Agregar Firebase al proyecto de Xcode paso 2.

La base de datos genera un archivo ".plist" el cual se debe agregar al documento, este archivo debe agregarse a la carpeta del proyecto de Xcode, luego agregar la referencia en

Xcode ya que si se hace como se muestra en el ejemplo de la Figura 3.82, es probable que hubiera error en el proyecto.

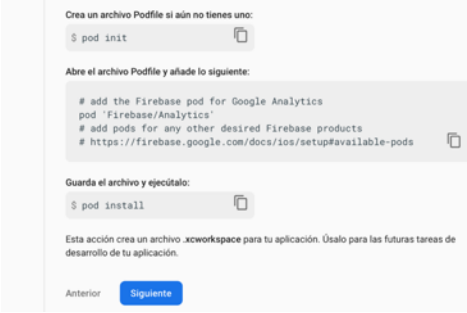


Figura 3.83: Agregar Firebase al proyecto de Xcode paso 3.

Por último se inicia el pod en la carpeta del proyecto desde la terminal con CocoaPods de la Mac, como marca la Figura 3.83, una vez inicializado se crea un archivo pod en el cual se ingresan las librerías necesarias para trabajar en Firebase. Se guardan los cambios y se da el comando de instalarlas al proyecto desde la terminal de la Mac. Después que las librerías han sido integradas en el proyecto ya sólo falta importar Firebase y configurar en el appdelegate para su enlace con la aplicación. Si todo salió bien la base de datos detecta la comunicación con la aplicación y la integración de Firebase con la aplicación está terminada.

Registrar - Iniciar sesión.

Figura 3.84: Interfaz de autenticación Ios.

Lo siguiente fue crear las interfaz de inicio de sesión, de la cual, se crearon otras a partir de la interfaz de la Figura 3.84(d), está interfaz es la principal siendo la única con un archivo ViewController, es la unica que controla la funcionalidad de los componentes de las interfaces, a partir de esta se mandan a llamar a las demás las cuales aparecerán encima de esta y desaparecerá presionando el botón regresar de las Interfaces de la Figura 3.84(a,b,c). La Figura 3.84(a) autentica al usuario si ya está registrado en la base de datos, caso contrario le informa al usuario que aún no se encuentra registrado.

La vista de la Figura 3.84(d), registra al usuario en la base de datos, con el nombre, apellido, correo y contraseña que el proporcione o con una cuenta de Google que tenga vinculada al dispositivo para agilizar el proceso, si el usuario prefiere el registro con correo

y contraseña tendrá que poner su nombre y apellido para poderse registrar, la base de datos automáticamente crea los demás campos que son necesarios para una mejor consistencia de los datos y evitar problemas si el usuario no termina con su registro. Si el usuario se registra con la cuenta de Google, automáticamente toma el nombre y apellido de la cuenta.

La vista de la Figura 3.84(c) recupera la contraseña del usuario en caso de que la olvide, sólo tiene que ingresar el correo con el que se registro y se le enviará un correo para restablecer su contraseña.

Una vez terminado el inicio de sesión lo siguiente que se realizó fue el registro de su currículum virtual, se decidió hacerlo en pasos para que sea más fácil para el usuario registrarse y que no sea tan tedioso el registro.

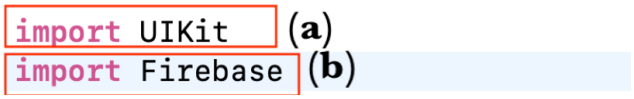


Figura 3.85: Librerías principales de IOS.

Se importaron las librerías UIKit y Firebase, la librería, de la Figura 3.85(a), UIKit incluye todo lo necesario para el desarrollo de aplicaciones en IOS, está librería incluye la arquitectura de ventana y vista para implementar las vistas que utilizáremos, también, incluye lo necesario para implementar las interacciones entre el usuario, el sistema y la aplicación, es decir, todo lo necesario para la interfaz y funcionalidad de la aplicación por lo mismo la librería es muy importante.

La siguiente librería es Firebase Figura 3.85(b), la cual se menciona en el apartado de base de datos de este mismo documento, en la vista es necesaria la librería para poder autenticar, registrar o recuperar la contraseña.

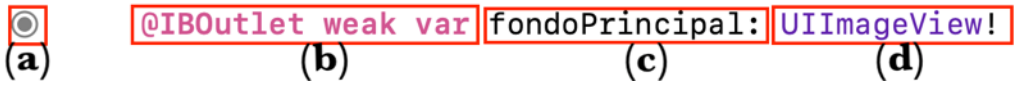


Figura 3.86: Declaración de variables con la interfaz en IOS.

El funcionamiento de las variables que se comunican con la vista, se enlazan desde el documento con la aplicación, para así identificarlo y darle la funcionalidad deseada. La forma en la que se declara es la siguiente: Figura 3.86(b), declara el tipo de objeto con el que se va a interactuar, la parte de la Figura 3.86(c), es el nombre que tendrá la variable y tiene que ser único para el tipo de variable que se declara en la parte de la Figura 3.86(c). La figura 3.86(a) indica si esta se encuentra conectada la variable al objeto de la vista en el Main StoryBoard, para así controlar las conexiones de las variables con los objetos de la vista.

```
46      //Acción del boton inicar sesion de la vista principal
47      @IBAction func boton(_ sender: Any) {        (a)
48
49          self.view.addSubview(popOver1)          (b)
50          popOver1.center = self.view.center
51  (c)     UIView.animate(withDuration: 0.5, delay:0, options: [], animations: {
52              UIView.setAnimationRepeatCount(0)
53              self.popOver1.alpha = 1
54          }, completion: nil)
55          UIView.animate(withDuration: 0.1, delay:0, options: [], animations: {
56              UIView.setAnimationRepeatCount(0)
```

Figura 3.87: Mostrar View de iniciar sesión.

La Figura 3.87 muestra la acción que realiza el botón de iniciar sesión de la vista principal de la Figura 3.84(d), la Figura 3.87(a), indica que es una acción del tipo Button la cual se encuentra conectado a la vista , La figura 3.84(b) agrega a la vista principal la subvista iniciar sesión para poder interactuar con ella. La última parte, Figura 3.87(c), es una animación pequeña para poder aparecer y desaparecer las vistas.

```
68      //accion de botton regresar del view iniciar sesión
        @IBAction func regresarAction(_ sender: Any) {
70          UIView.animate(withDuration: 0.1, delay:0, options: [], animations: {
71              UIView.setAnimationRepeatCount(0)
72              self.popOver1.alpha = 0                                    (a)
73          }, completion: nil)
74          delayWithSeconds(0.1){
75              self.popOver1.removeFromSuperview()                        (b)
```

Figura 3.88: Acción del botón regresar.

La Figura 3.88 muestra la acción que realiza el botón regresar de la Figura 3.88(a), lo que hace es remover la vista con una animación que la hace transparente, esto se hace para no consumir recursos del dispositivo.

```
182 // MARK: - Registrarse
    @IBAction func createAccount(_ sender: AnyObject){                    (a)
184
185     if apellidopopOver2textField.text == "" ||
186         emailRegistrarsepopOver2texField.text == "" || contraRegistrarsepopOver2TexFiexl.text == "" || nombrepopOver2textField.text == ""{
187         let alertController = UIAlertController(title: "Error", message: "Por favor ingresa tu Nombre, Apellido, Correo y Contraseña", preferredStyle: .alert)
188
189         let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
190         alertController.addAction(defaultAction)                                                                          (b)
191
192         present(alertController, animated: true, completion: nil)
193     }else {
194         Auth.auth().createUser(withEmail: emailRegistrarsepopOver2texField.text!, password: contraRegistrarsepopOver2TexFiexl.text!) { (user, error) in   (c)
195             if error == nil{
196                 print("Te has registrado correctamente")
197                 let userID = Auth.auth().currentUser!.uid
198                 let db = Firestore.firestore()
199                 db.collection("Usuarios").document(userID).setData([
200
201                     "Nombre": self.nombrepopOver2textField.text!,
202                     "Apellido": self.apellidopopOver2textField.text!,                                                     (d)
203                     "Correo": self.emailRegistrarsepopOver2texField.text!,
204                     "UID": "\(userID)"
205
206                 ]) { err in
207                     if let err = err {                                                                                    (e)
208                         print("Error writing document: \(err)")
209                     } else {
210                         print("Document successfully written!")
211                         print("UID \(userID)")
212                         let vc = self.storyboard?.instantiateViewController(withIdentifier: "Home")                      (f)
213                         self.present(vc!, animated: true, completion: nil)
214                     }
215                 }
216             }else {
217                 let alertController = UIAlertController(title: "Error", message: error?.localizedDescription, preferredStyle: .alert)
218                 let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
219                 alertController.addAction(defaultAction)                                                                  (g)
220                 self.present(alertController, animated: true, completion: nil)
221             }
222         }
223     }
224 }
```

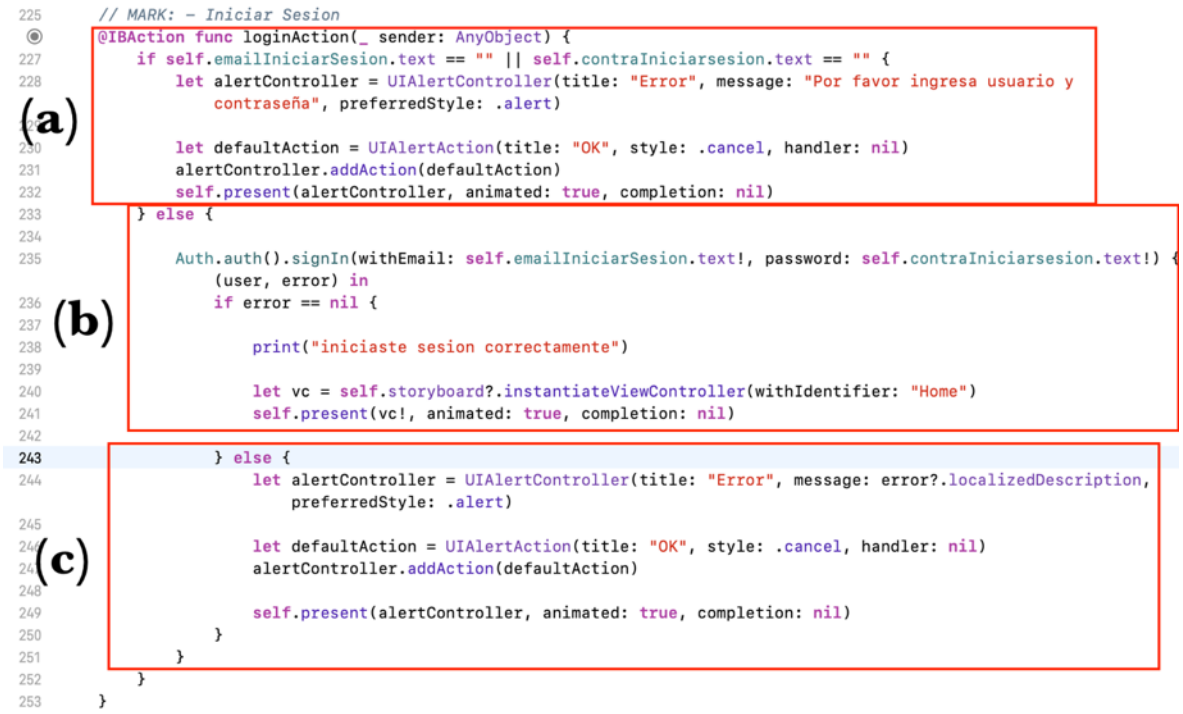
Figura 3.89: Acción para registrar usuarios IOS.

La Figura 3.89 muestra el método de registrar al usuario en la base de datos Firebase, la

Figura 3.89(a), muestra la conexión con el botón de la vista, mientras que la Figura 3.89(b), muestra la condición necesaria para poder intentar registrar al usuario en la base de datos, esta condición pide al usuario haber ingresado los campos de Nombre, Apellido, Correo y Contraseña, si no es así, le muestra un mensaje al usuario avisando que no se puede registrar sin llenar los campos ya que es un requisito para dar de alta al usuario. Una vez el usuario a ingresado los campos necesarios, se intenta registrar en la base de datos, la encargada de registrar al usuario con el correo y contraseña ingresados es la función de la Figura 3.89(c).

Si la condición del inciso (d) de la Figura 3.89 se cumple, entonces registra al usuario con los datos que ingresó y adicionalmente agrega el campo UID el cual es un identificador único que crea la base de datos para la consistencia de los datos, este identificador nos ayuda a leer y escribir en el documento del usuario siempre que el usuario lo quiera. Si la condición no se cumple, entonces, imprime en consola el error del porque no se pudo registrar.

La Figura 3.89(f), muestra la parte donde todas las condiciones se cumplieron para el registro, la cual imprime en consola el UID del usuario para poder seguirlo en la base de datos y así agilizar las pruebas y envía al usuario al viewController con el identificador Registro el cual lo lleva a la primer vista del registro del usuario. Por último si la primera condición no se cumple, entonces se muestra un mensaje al usuario especificando por qué no se puede registrar en la base de datos.




```
225     // MARK: - Iniciar Sesion
        @IBAction func loginAction(_ sender: AnyObject) {
227         if self.emailIniciarSesion.text == "" || self.contraIniciarsesion.text == "" {
228             let alertController = UIAlertController(title: "Error", message: "Por favor ingresa usuario y
                    contraseña", preferredStyle: .alert)

230             let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
231             alertController.addAction(defaultAction)
232             self.present(alertController, animated: true, completion: nil)
233         } else {
234
235             Auth.auth().signIn(withEmail: self.emailIniciarSesion.text!, password: self.contraIniciarsesion.text!) {
                    (user, error) in
236                 if error == nil {
237
238                     print("iniciaste sesion correctamente")
239
240                     let vc = self.storyboard?.instantiateViewController(withIdentifier: "Home")
241                     self.present(vc!, animated: true, completion: nil)
242
243                 } else {
244                     let alertController = UIAlertController(title: "Error", message: error?.localizedDescription,
                            preferredStyle: .alert)
245
246                     let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
247                     alertController.addAction(defaultAction)
248
249                     self.present(alertController, animated: true, completion: nil)
250                 }
251             }
252         }
253     }
```

Figura 3.90: Acción para iniciar sesión IOS.

La Figura 3.90 muestra el método de iniciar sesión en la base de datos, lo primero que hace la función es verificar que el campo correo y contraseña no estén vacíos para evitar peticiones de inicio sin datos a la BD, si el usuario ingresa sus datos correctamente, entonces intenta iniciar sesión con el correo y contraseña ingresados como se muestra en (a) de la Figura 3.90 si no se ingresarón correctamente, muestra un mensaje en el cual se alerta al usuario que no ingreso información en alguno de los campos, si los datos son correctos, la BD verifica la información y si es correcta lo envía a la vista con el identificador Home la cual es la vista principal de toda la aplicación, como se muestra en el inciso (b) de la Figura 3.90. Si por el contrario no se puede iniciar sesión, muestra un mensaje al usuario describiendo el error del intento de inicio de sesión, los cuales pueden ser por falta de conexión a internet o porque no existen sus datos en la BD.

```
@IBAction func submitAction(_ sender: AnyObject) {

    if self.correoRecuperarContraPopOverTextField.text == "" {
        let alertController = UIAlertController(title: "Oops!", message: "Porfavor Ingresa un Email.",
            preferredStyle: .alert)

        let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
        alertController.addAction(defaultAction)

        present(alertController, animated: true, completion: nil)

    } else {
        Auth.auth().sendPasswordReset(withEmail: self.correoRecuperarContraPopOverTextField.text!, completion: {
            (error) in

            var title = ""
            var message = ""

            if error != nil {
                title = "Error!"
                message = (error?.localizedDescription)!
            } else {
                title = "Success!"
                message = "Se envio un correo al email."
                self.correoRecuperarContraPopOverTextField.text = ""
            }

            let alertController = UIAlertController(title: title, message: message, preferredStyle: .alert)

            let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
            alertController.addAction(defaultAction)

            self.present(alertController, animated: true, completion: nil)
        })
    }
}
```

(a) (b) (c)

Figura 3.91: Acción para recuperar contraseña IOS

Para recuperar la contraseña lo primero que se hace es verificar que el campo del correo electrónico no este vacío como se muestra en la Figura 3.91 (a), si el campo por el contrario esta vacío, avisa al usuario. Si el usuario ingresa el correo y tiene conexión a internet, el usuario recibirá un correo electrónico para restablecer la contraseña como se muestra en la Figura 3.91 (b), si por alguna razón el usuario no tiene conexión a internet o no se encuentra registrado se mostrará un mensaje en la aplicación advirtiendo de ello como se muestra en la Figura 3.91 (c). Con todas estas funciones finaliza la codificación de la vista autenticación, la cual es el inicio del proyecto y parte esencial para poder identificar a los usuarios, y llevar un mejor control en la base de datos entre las empresas y el usuario.

Creación del currículum virtual Interfaz IOS

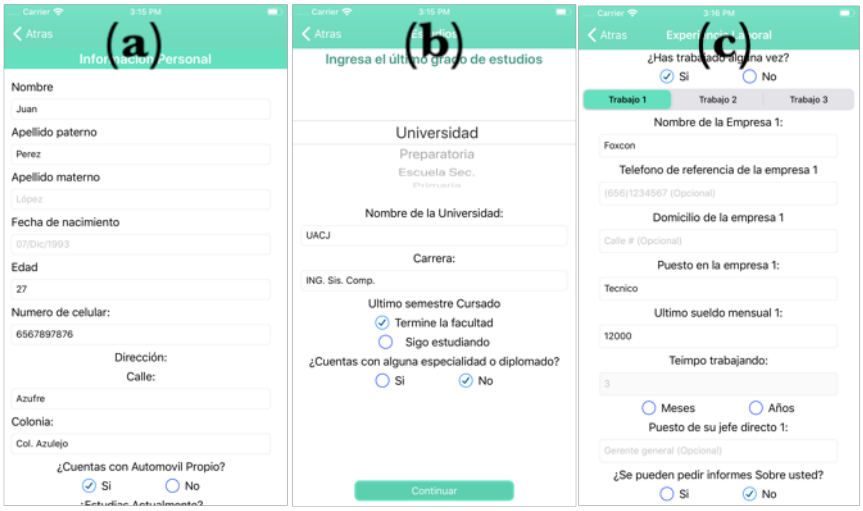


Figura 3.92: Vistas del currículum virtual

La primer vista del registro del CV (Currículum Virtual) de información personal (a) de la Figura 3.92 la cual incluye su nombre y apellido aunque estos se llenan automáticamente con la información registrada previamente, incluye el número de celular para que las empresas puedan comunicarse directamente con él, si él, así lo desea. También tiene que registrar la dirección para saber si la empresa cuenta con transporte en esa dirección, las demás opciones son información adicional que puede ser interesante para algunas empresas.

El segundo paso para el registro, son los estudios del usuario, el cual sólo exige el último grado de estudios para agilizar el proceso de registro. En la vista de la Figura 3.92(b), el usuario selecciona su último grado de estudios el cual puede ser la universidad, la preparatoria, la escuela secundaria y la escuela primaria. En este caso el usuario selecciona la universidad en la cual sólo tiene que ingresar el nombre de la universidad y de la carrera, también dependiendo de si ya terminó la facultad o si todavía no termina aparecerán diferentes opciones, como

si tiene algún diplomado o algo de interés para mostrarlo en el currículum, o por ejemplo, si todavía no termina la facultad tiene que poner el semestre actual que cursa.

La tercer vista (c) de la Figura 3.92 es la información laboral o experiencia laboral. La cual especifica si el usuario ha trabajado antes o no. Sí el usuario a trabajado antes entonces tendrá que llenar la información de la empresa como la dirección y teléfono de esta, así como el salario y el puesto que desempeñaba. Siendo los campos de la dirección, el télefono, puesto del jefe directo y último sueldo mensual opcionales ya que tal vez sea información que no quiere o no puede compartir.

Interfaz principal de la aplicación TabBar.

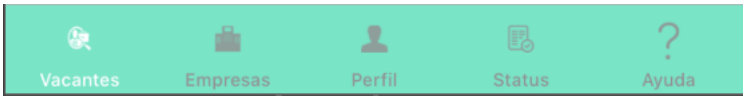


Figura 3.93: Vista principal TabBar.

La Figura 3.93 muestra el ”Tab Bar Controller” el cual es la vista principal y es el que controla la interacción con los diferentes views, ya que es el encargado de mostrarlos según el apartado que el usuario seleccione de la parte (a) de la Figura 3.93, desde esta barra controladora podremos movernos entre las siguientes vistas: Vacantes, Empresas, Perfil, Status y Configuración. Esto funciona para agilizar la navegación entre estas vistas ya que son las mas importantes de la aplicación. Esta vista es mostrada una vez que el usuario a concluido su registro.

Interfaz que muestra todas las vacantes.

Figura 3.94: Muestra todas las vacantes activas.

La vista de vacantes, muestra todas las vacantes que se encuentran disponibles en la aplicación, las cuales pueden ser diferentes empresas o de la misma empresa con diferentes puestos o requisitos como se muestra en la Figura 3.94

Interfaz que muestra las vacantes por empresas.

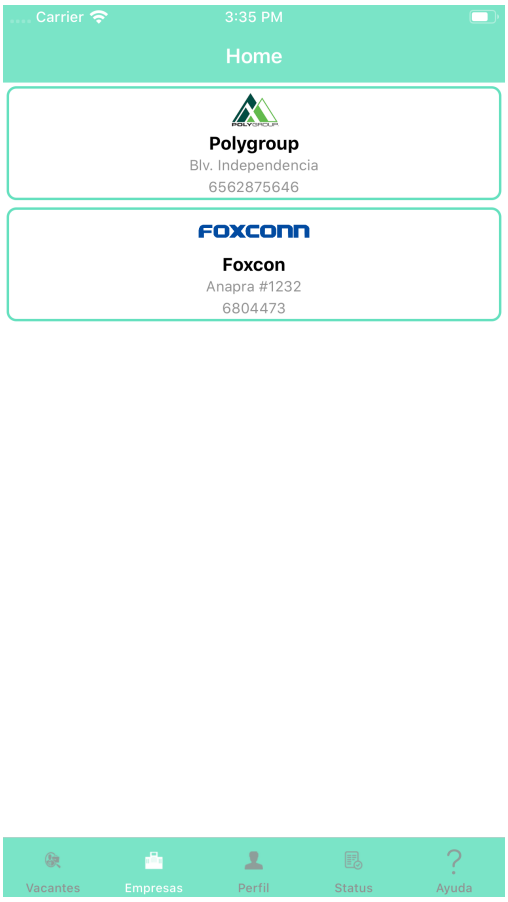


Figura 3.95: Muestra todas las vacantes activas por empresas.

La Figura 3.95 muestra la vista de las Empresas, la cual muestra las vacantes ordenadas por empresa, una vez seleccionada la empresa muestra todas las vacantes publicadas y disponibles de la empresa seleccionada, esto para ordenar las vacantes y el usuario pueda seleccionar mejor la vacante.

Interfaz de la información de la vacante.

Figura 3.96: Información de la vacante.

La Figura 3.96 muestra la vista de la descripción de la vacante la cual se muestra si presionas en la vacante de la vista de las vacantes por empresas Figura 3.95 o en vacantes 3.94. La vista de la descripción de la vacante, como su nombre lo dice muestra toda la información de la vacante. Aquí se muestra las vacantes disponibles así como los usuarios que han aplicado a ella, muestra una breve información de la empresa. También especifica si se necesita experiencia para el trabajo, los turnos en horario y días, descripción breve de la vacante o información relevante que considere la empresa, muestra los requerimientos para poder aplicar, especifica si buscan sólo hombres, mujeres o ambos, por último se muestra la

documentación necesaria para poder obtener la vacante. Con toda esta información el usuario decide si puede o no aplicar a dicha vacante ya que con la información publicada se sabe que tan cerca le queda de su domicilio asi como con los servicios que cuenta la empresa los cuales podrian ser tranporte o cafeteria y el sueldo que ofrecen. Estos factores importantes ayudaran a los postulantes (usuarios) a tomar la decisión de postularse o no a dicha vacante.

Interfaz perfil usuarios.

Figura 3.97: Interfaz Perfil de usuario IOS.

En la Figura 3.97 se puede observar la vista de Perfil, la cual muestra la información más relevante del usuario, como lo puede ser su nombre, el correo de contacto, su numero de

teléfono así como su dirección actual. También se muestra la información del último grado de estudios obtenido actualmente por el usuaro así como algunos datos de sus experiencias laborales pasadas e idiomas si es que el postulante es bilingüe.

Interfaz de estatus de la solicitud.

Figura 3.98: Interfaz de estatus de la solicitud.

La Figura 3.98 muestra la vista de status la cual da una vista previa del estado actual de la solicitud de la vacante, la cual se sabe el estado gracias a las tres imágenes que estan debajo, la primera imagen avisa si el curriculum fue enviado, la segunda si el curruculum fue leido por la empresa, y la ultima imagen muestra si la solicitud fue aprobada o rechazada.

Esto significa que se pueden tener varias solicitudes a las vez para aumentar las posibilidades de obtener alguna vacante.

Interfaz de ayuda usuarios.

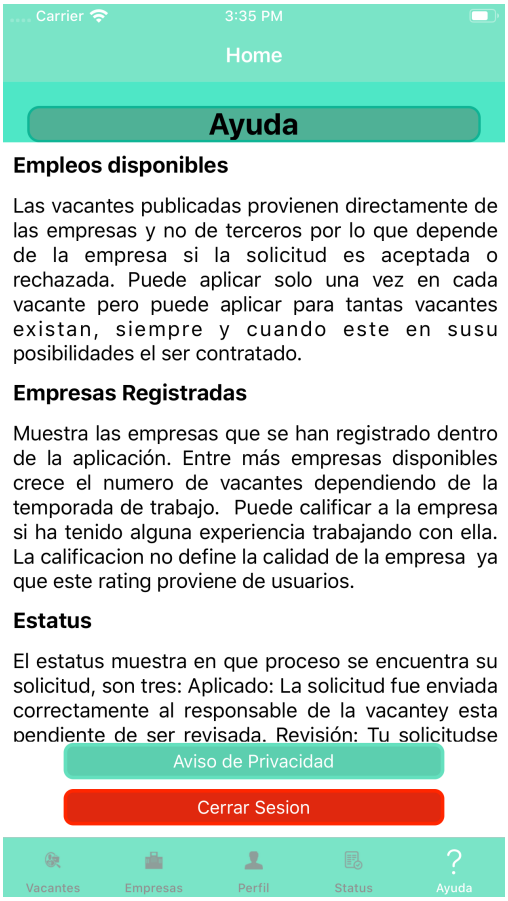


Figura 3.99: Interfaz de estatus de la solicitud.

La última vista del "Tab Bar Controller" es la de ayuda que se puede observar en la Figura 3.99, en la cual el usuario puede cerrar la sesión, además de ayudar al usuario describiendo un poco la funcionalidad de cada vista de la aplicación y orientarlo para que obtenga mejores resultados a la hora de enviar su solicitud.

Vistas con enfoque a empresas IOS.

En la siguiente sección se abordan las vistas en las cuales la empresa puede navegar para publicar las vacantes así como administrar las solicitudes de las mismas que envían los usuarios.

Registrar empresa.

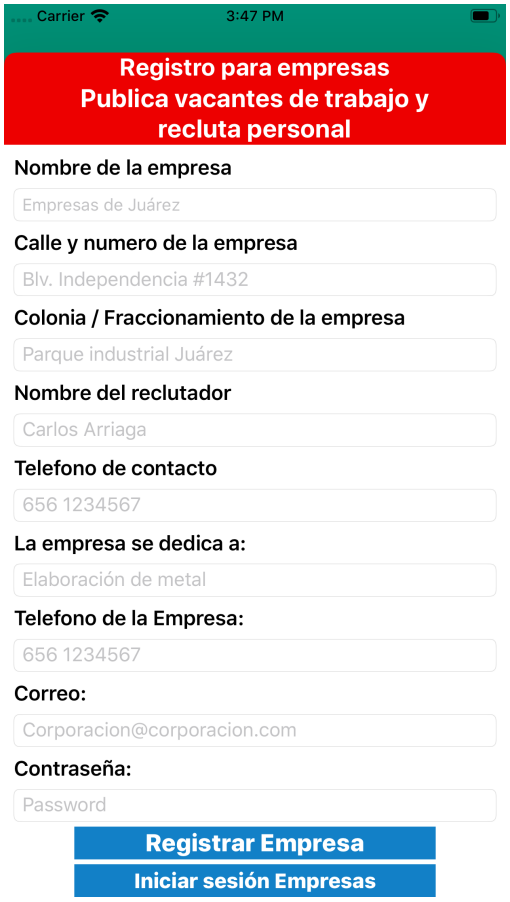


Figura 3.100: Registro para empresas.

Cuando una empresa busca publicar una vacante en la aplicación, lo primero que necesita es registrarse para poder publicar vacantes. La empresa debe enviar sus datos como se muestra

en la 3.100, en los cuales se incluye nombre de la empresa, dirección de la empresa, número de la empresa y un número de contacto para poder comunicarse con ellos y verfificar que efectivamente se trata de una empresa. Si la empresa aun no es verificada es enviada a una vista en la cual no hace nada mas que esperar hasta que sea verificada. Una vez verificada se permite acceder a la aplicación para poder publicar y recibir solicitudes.

Iniciar sesión empresa.

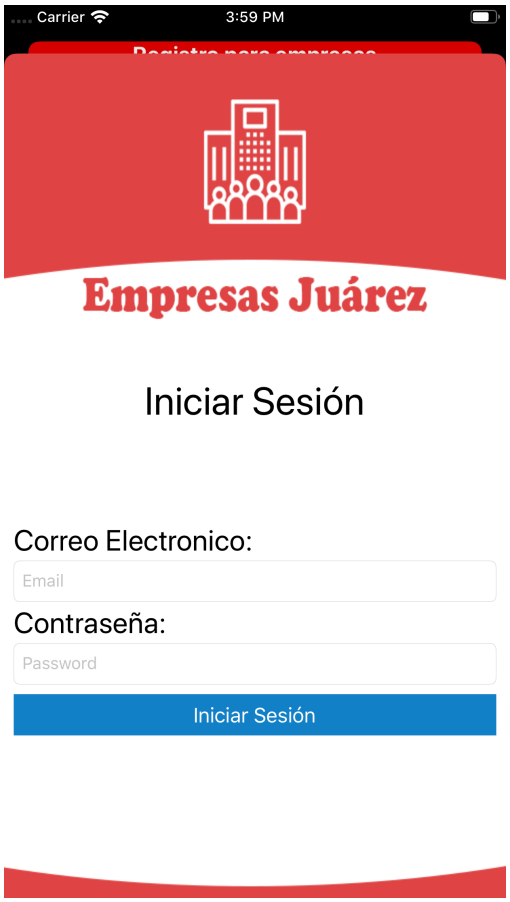


Figura 3.101: Iniciar sesión empresas.

Cuando el usuario empresa pierda las credenciales de autenticación, podrá iniciar sesión como se muestra en la Figura 3.101 una vez se autentica se verifica si esta validado para

decidir a cual vista es enviado.

Una vez el usuario es verificado y pueda entrar a la vista principal, entrara a un "TabBar.[el] cual es similar al de usuarios de la Figura 3.93 ya que es una manera muy cómoda de navegar entre las distintas vistas de la aplicación. Sin embargo no son las mismas vistas que tiene el usuario normal, las empresas tiene las vistas de: publicar vacante, editar vacante, perfil, administrar solicitudes y configuración las cuales se describen a continuación.

Crear vacante.

Figura 3.102: Boton crear vacante.

Cuando el usuario empresa busque crear una vacante se le mostrara esta vista la cual

funciona para no mostrar de una sola vez todo lo que puede ingresar el usuario para crear la vacante. Si no para que se mire un poco más estetica la interfaz del usuario empresa. El usuario presiona el boton crear vacante de la Figura 3.102 y entonces es enviado a la vista de la Figura 3.103

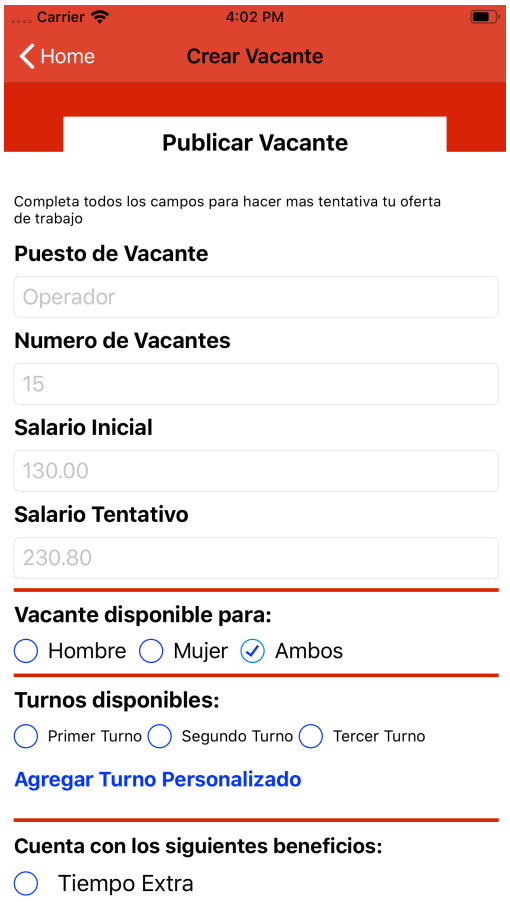


Figura 3.103: Interfaz crear vacante.

Para poder controlar los datos que las empresas publiquen de sus vacantes, creamos la interfaz de la Figura 3.103 la cual pide el nombre de la vacante para poder identificar las diferentes vacantes que creemos en la aplicación, el numero de vacantes disponibles para ese puesto, el salario inicial y el salario maximo que se puede obtener en esa vacante. Los campos

de: numero de vacantes, salario inicial, salario máximo, están restringidos a valores numéricos ya que así se manejan en la Base de Datos. lo siguiente que puede elegir la empresa es el género de los empleados que busca. Los turnos también los puede crear de lunes a viernes siendo asi el primer, segundo y tercer turno de los cuales sólo puede elegir los horarios en los que entra y sale el personal de los turnos. También puede agregar dos turnos especiales de los cuales puede seleccionar los días labórales, el horario de cada día y agragarlo a los tres turnos anteriores, esto se implementó por las empresas que tienen turnos especiales y necesitan configurarlo a su manera. La empresa también puede elegir la documentación que se necesita entregar si es que es contratado, también puede agragar aptitudes que busque en el postulante asi cómo una descripción de la vacante para hacerla mas llamativa para los postulantes.

Editar vacante.

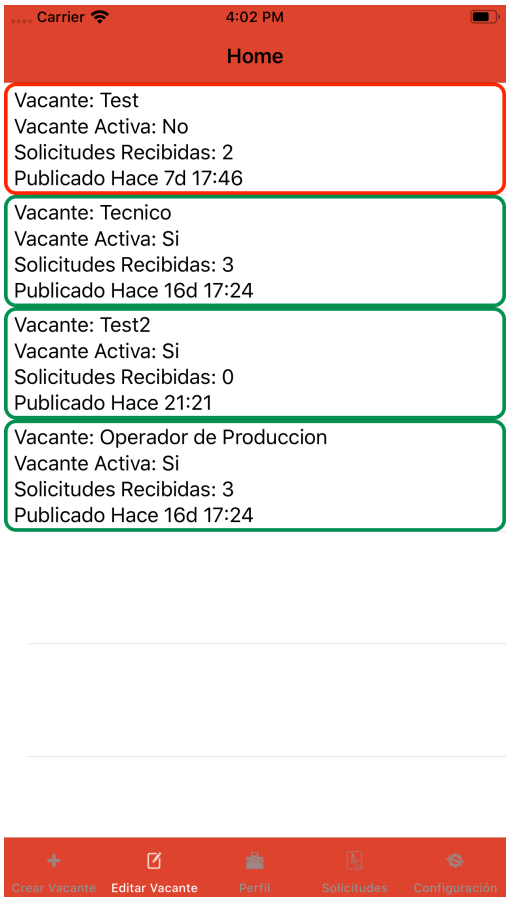


Figura 3.104: Lista de vacantes creadas.

Una vez publicada la vacante aparecera en la vista de editar vacante como se muestra en la Figura 3.104. Aquí aparecen las vacantes creadas por las empresas para poder seleccionar una y así poder editarlas.

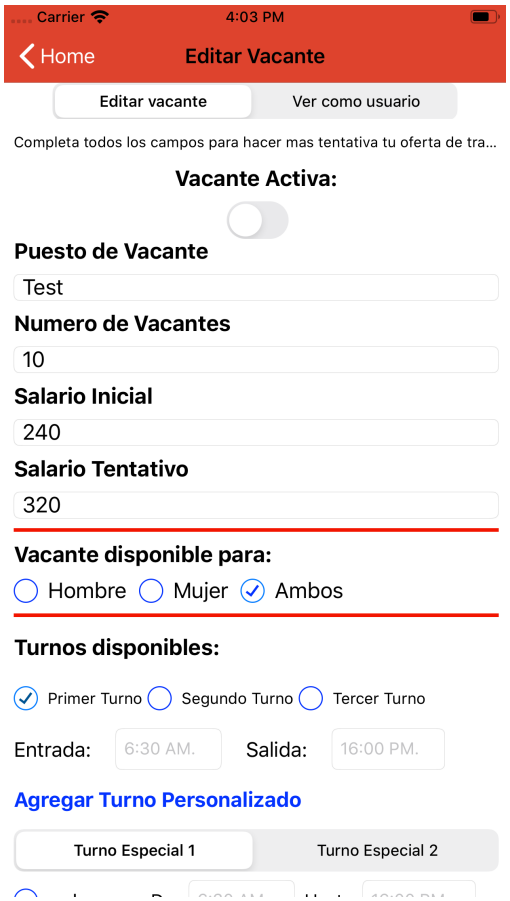


Figura 3.105: Editar vacante

Cuando el usuario empresa quiere editar la vacante la selecciona y se muestra la misma vista de crear la vacante pero con los datos ingresados cuando la creo, ademas de poder visualizar la vacante como se le mostraria al usuario esto se puede observar en la Figura 3.96. Con esto el usuario empresa puede modificar la vacante para mostrar la información deseada, ya que se puede volver a modificar toda la información de cuando la creo inicialmente.

Perfil empresa.

Figura 3.106: Perfil empresa.

En la parte del perfil de la empresa puede editar todos sus datos y agregar el logotipo de la empresa para que se muestre en las vacantes que publique. El nombre del reclutador y el teléfono del mismo nunca se mostraran en las vacantes la única información que se mostrara si el usuario así lo conFigura sera el teléfono y dirección de la empresa

Solicitudes recibidas.

Figura 3.107: Solicitudes de usuarios.

En la sección de solicitudes de la empresa 3.107, se gestionan las solicitudes de las vacantes activas actualmente, se muestra información relevante de la vacante como lo son las solicitudes recibidas y el nombre de la vacante.

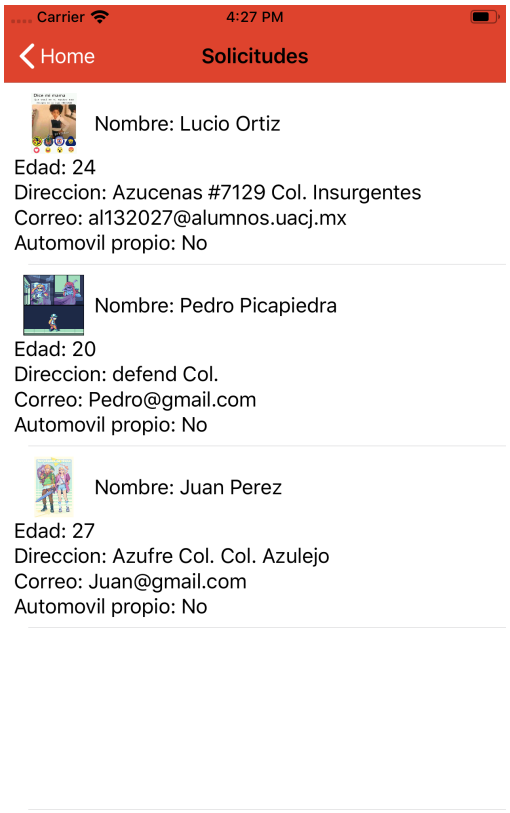


Figura 3.108: Usuarios que enviaron solicitud.

Los usuarios que se muestran en esta parte de la aplicación Figura 3.108 son los que enviaron las solicitud a la vacante, Se da un vistazo de datos personales del usuario como lo son el nombre, edad, correo electrónico y si el usuario cuenta con automóvil propio, los usuarios empresa pueden acceder a cualquier solicitud recibida para ver el currículum completo del usuario, antes de poder ver el currículum del usuario se le preguntara si acepta la solicitud, si no la acepta entonces no podrá ver el currículum completo ya que si no es de su interés en un primer vistazo no es necesario que mire todo el currículum.

En la vista de la solicitud completa, se muestra la información registrada por el usuario,

la empresa decide si lo contrata o no, si lo quiere contratar se le abrirá una interfaz para poder enviar los datos para que acuda a su contratación en la empresa, caso contrario se le avisa al usuario que su solicitud no fue aceptada.

Interfaz de ayuda y cerrar sesión.

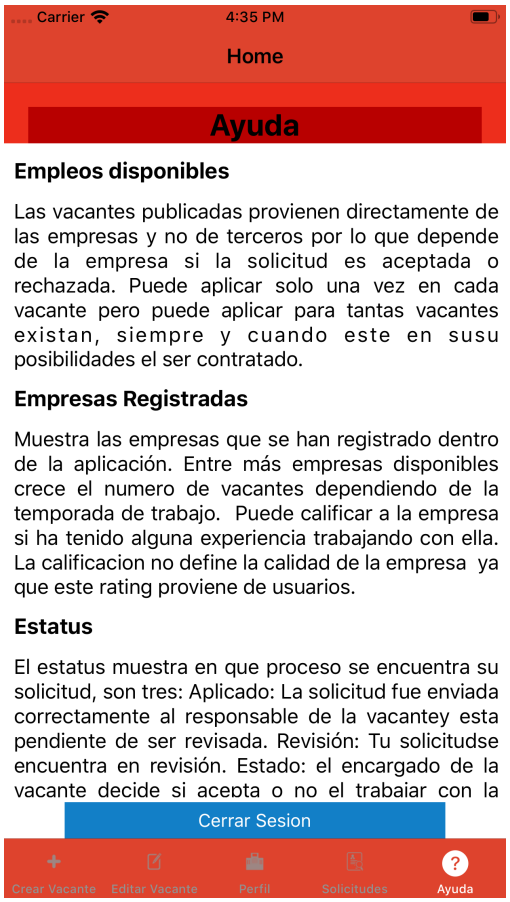


Figura 3.109: Solicitudes de usuarios.

En la vista de ayuda, Figura 3.109 para la empresa tiene el mismo funcionamiento que el de los usuarios esta también busca mejorar la manera en que publican las empresas sus vacantes para tener una mejor recepción con los usuarios que buscan empleo.

El código más importante de cada interfaz creada se encuentra en el Apéndice A al final del documento.

3.7.5. Pruebas unitarias

Android

Cada parte de la aplicación explicada en el desarrollo esta compuesta por diferentes módulos, esto permite hacer las pruebas sólo en partes especificas sin la necesidad de correr todo el proyecto completo. Tomando esta ventaja del código desacoplado, se hacen las pruebas en cada interfaz de la aplicación para comprobar que su funcionamiento y diseño esta trabajando de forma correcta.

Modulo	Funcionamiento	Prueba	Resultado
Registro	Crear una sesión dentro de la aplicación	✓	La sesión se crea correctamente y se registra al usuario en la base de datos
Currículum virtual	Formulario del historial del usuario	✓	Los datos se registraron correctamente y se agregaron nuevos campos
Menú Principal	Botton Navigation	✓	Hace correctamente el llamado de cada uno de los Fragment
Vacantes	Muestra las ofertas de empleo	✓	Carga correctamente cada una de las vacantes y su descripción
Empresas registradas	Muestra las empresas	✓	El rating de la empresa no cargaba correctamente y fue cambiado un valor de float a double para la calificación
Estatus	Muestra las solicitudes aplicadas por los postulantes	✓	Se enviaron 3 solicitudes y el Fragment las cargo correctamente mostrando el estatus de cada una.
Perfil	Muestra datos del usuario	✓	Carga correctamente el perfil del usuario usando el currículum y en caso de no esta completo se le muestra un botón para terminar de llenarlo realizando el llamado correcto de los Fragment
Ayuda	Muestra el cierre de sesión e información de las vistas	✓	Al presionar el cierre de sesión la aplicación manda al usuario a la pantalla de inicio para que vuelva a entrar con su usuario.

Figura 3.110: Tabla de pruebas de usuario

Como se ve en la tabla 3.110 las pruebas de cada modulo fueron realizadas con éxito, se

corrigieron algunos errores de diseño y se mejoro el código para algunos elementos.

Modulo	Funcionamiento	Prueba	Resultado
Sesión	Crear una sesión dentro de la aplicación para empresas	✓	La sesión se crea correctamente y se inicia la aplicación.
Registro	Formulario para registrar la empresa	✓	Los datos se registraron correctamente y se manda la información a la base de datos.
Menú Principal	Botton Navigation	✓	Hace correctamente el llamado de cada uno de los Fragment
Crear vacante	Opción para crear una nueva vacante	✓	Manda con éxito la nueva vacante publicada
Administrar vacantes	Muestra las vacantes publicadas	✓	Se publican 2 vacantes y estas se muestran correctamente en la sección al igual que carga su descripción y después se dan de baja correctamente.
Solicitudes	Muestra las solicitudes aplicadas por los postulantes	✓	Se enviaron solicitudes y el Fragment las cargo correctamente mostrando el perfil de los postulantes y su currículum completo.
Perfil	Muestra datos de la empresa	✓	La información de la empresa se muestra en la sección.
Ayuda	Muestra el cierre de sesión e información de las vistas	✓	Al presionar el cierre de sesión la aplicación manda al reclutador a la pantalla de inicio para que vuelva a entrar con su usuario y contraseña.

Figura 3.111: Tabla de pruebas de empresa

Los resultados de las pruebas para el perfil de la empresa concluyeron correctamente como se ve en la tabla 3.111.
Las pruebas de cada modulo fueron realizadas con información ficticia simulando a usuarios y a una empresa.

IOS

Las pruebas unitarias se realizaron en el emulador de Xcode con porciones pequeñas del código que podría llegar a fallar. Las pruebas se realizaron exitosamente y ayudaron a corregir errores al momento de registrarse en la aplicación ya que los datos se enviaban dos veces a la base de datos, también surgieron problemas a la hora de traer la información de las vacantes ya que se guardaban doblemente en el arreglo que las mostraba. También se redujo el código con la corrección y mejorando la eficiencia del mismo se pueden cargar los datos de una manera mejor.

3.7.6. Emulación

La emulación se llevo a cabo en Xcode Emulator el cual puede emular dispositivos con sistema operativo IOS desde iPhone 8, iPhone 8 plus, hasta el iPhone 11 pro, lo cual ayuda al desarrollo de las aplicaciones para poder comprobar como se muestras las vistas en los diferentes tamaños de pantallas además de mostrar el funcionamiento actual del desarrollo.

Las pruebas de emulación se llevaron acabo en todos los dispositivos que permite el Emulator, para comprobar el funcionamiento de toda la aplicación antes de poder publicarla antes de su distribución. Las pruebas fueron satisfactorias y ayudaron a resolver problemas de visualización de los labels y textfields de la aplicación ya que se agrandaban demasiado y no se miraban muy bien en pantallas mas grandes que el iPhone 8 plus, una vez corregido esto la aplicación queda prácticamente lista para las pruebas de funcionamiento.

3.7.7. Pruebas de funcionamiento

Las pruebas de funcionamiento se realizaron en dispositivos reales, en específico se utilizaron los dispositivos: iPhone 7, iPhone 7 plus, iPhone XR(Para realizar la pruebas en IOS), Samsung M20, Samsung M10 y Nokia N3(Para realizar las pruebas en Android). esto con el fin de validar los textfield que muestren sólo las numeraciones, numero de teléfono o texto según el tipo de dato que se maneja en la base de datos, también se registraron usuarios en cada dispositivo hasta enviar la solicitud y ser aceptados por la empresa para verificar el correcto funcionamiento de todas las interfaces disponibles de la aplicación.

Además de verificar también el apartado de los usuarios empresa haciendo las mismas pruebas. Las vistas cargan toda la información y se miran bien en todos los dispositivos probados lo cual asegura un buen funcionamiento del prototipo final.

Capítulo 4
Resultados y Discusiones

En el presente capítulo se muestran los resultados obtenidos en las encuestas. Se aplicaron dos tipos de encuesta, la primera encuesta está enfocada en los usuarios que buscan empleo, la cual consta de tres partes: la primera parte busca saber los conocimientos del entrevistado a la hora de pedir trabajo o cómo es que le resultan los métodos con los que reclutan las empresas actualmente. La segunda parte se enfoca en el diseño, rendimiento y funcionamiento de la aplicación y la última parte se enfoca en la experiencia del usuario con la aplicación.

Cabe mencionar que algunas personas si pudieron instalar la aplicación en su dispositivo móvil y otras que al no poder instalar la aplicación, Se optó por prestarles un celular con la aplicación instalada ya que era necesario para poder responder algunas preguntas de la encuesta.

Encuesta enfocada al usuario

Las encuestas del usuario tienen como objetivo el conocer si beneficia tener una aplicación dedicada a la gestión de información de vacantes, ofertadas por las maquiladoras de Ciudad Juárez y en qué podría mejorar para futuras versiones.

Conocimientos del usuario buscando empleo

La primera pregunta tuvo como objetivo conocer el medio de información que regularmente utiliza para encontrar empleo.

1.¿Habitualmente cómo busca empleo?. Las posibles respuestas fueron: periódico, facebook, volantes, módulo de contratación y otro(especifique).

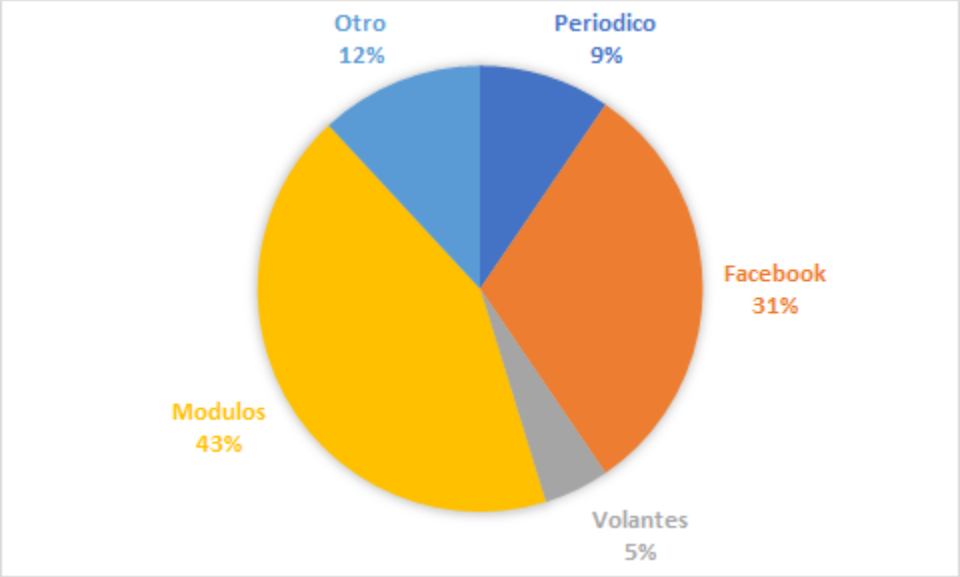


Figura 4.1: Resultados de la pregunta uno

Como se puede observar en la Figura 4.1. La manera más utilizada para obtener empleo es acudiendo a los módulos de contratación que colocan las empresas en zonas estratégicas de la Ciudad. Con esto se observa que los encuestados buscan la información de mano de la empresa o la información más detallada posible para saber si es una opción viable para ellos.

Para identificar qué tan buena es la información que se ofrece en los medios de comunicación mencionados en la primera pregunta, se utilizo la escala de Likert, siendo las posibles respuestas: muy buena, buena, regular, mala y muy mala. Lo anterior para conocer la satisfacción de los encuestados con la información que publican estos medios para encontrar trabajo, ver Figura 4.1

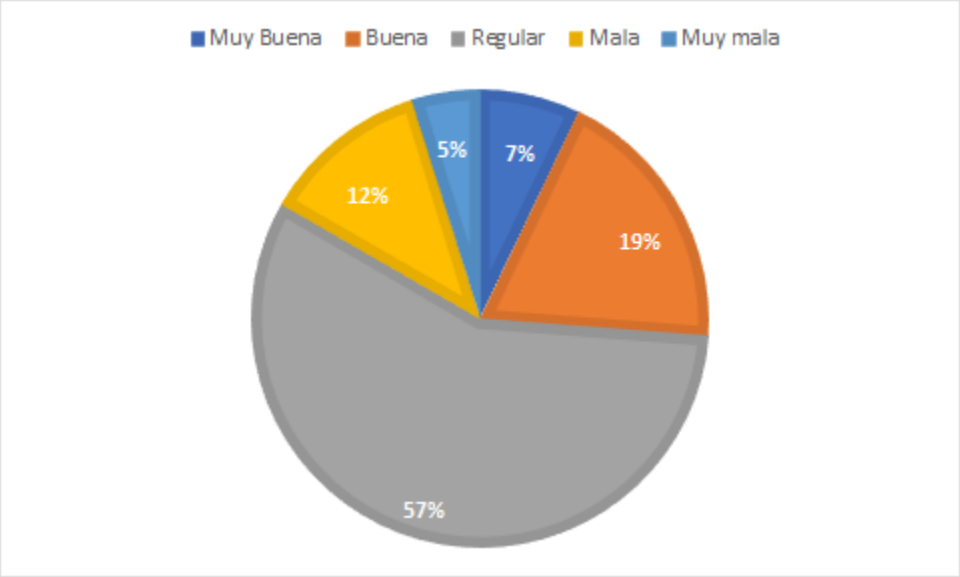


Figura 4.2: Resultados del medio de comunicación periódico

Los encuestados respondieron que la información de las vacantes mostradas en el periódico es regular. Lo anterior podría ser, debido a que la información de las vacantes en el periódico es muy pequeña en algunas ocasiones ya que el segundo porcentaje más alto demuestra que algunas personas están conformes con la información, ver Figura 4.2

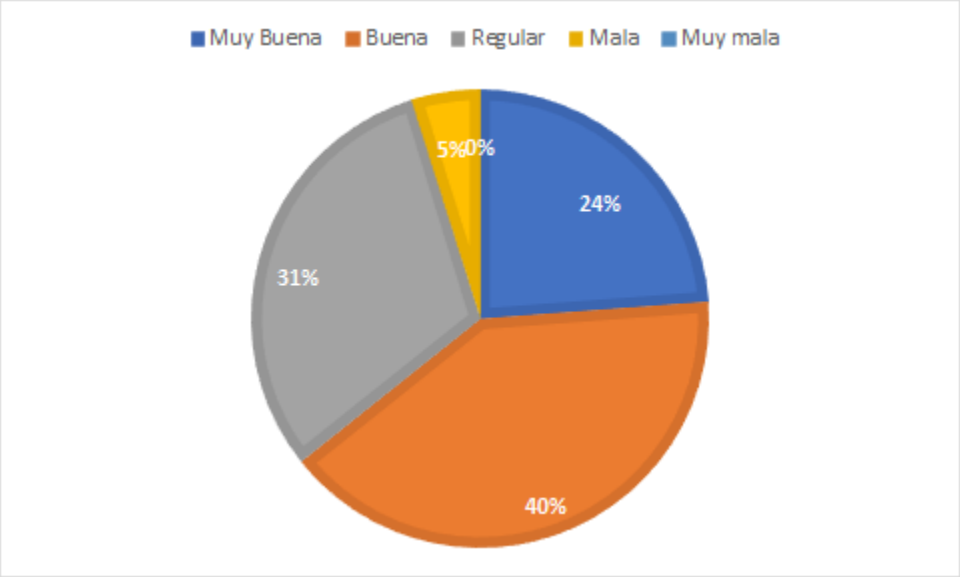


Figura 4.3: Resultados del medio de comunicación Facebook

La primera pregunta arroja que Facebook es la segunda forma en la que buscan empleo, por eso sorprende que sólo el 24 % cree que es muy buena ya que en algunas ocasiones la información publicada en Facebook es por parte del mismo personal de la empresa. Con todo esto sigue teniendo buena aceptación para los encuestados, ver Figura 4.3.

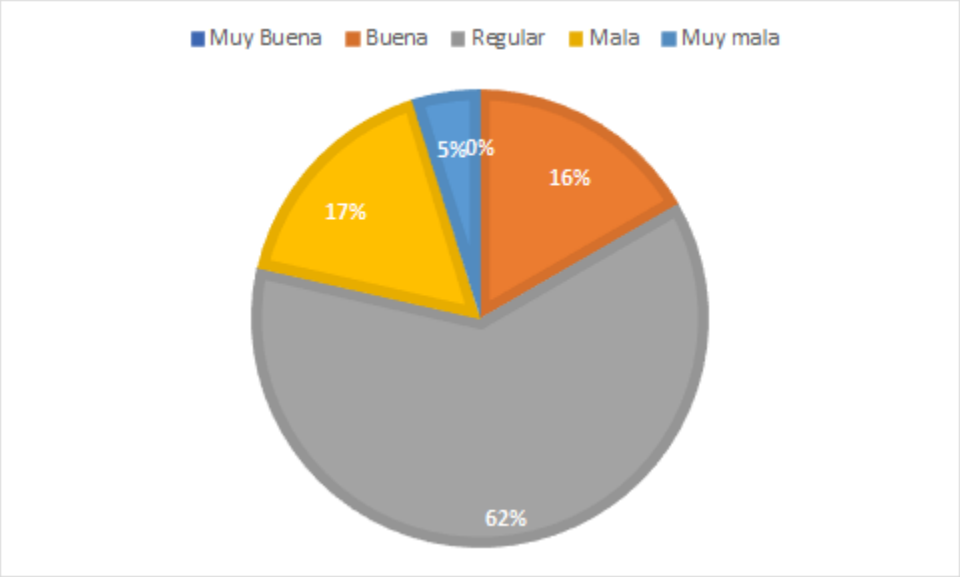


Figura 4.4: Resultados del medio de comunicación Volantes

Los volantes son los menos utilizados a la hora de buscar trabajo ya que según los entrevistados no son muy comunes y además no siempre son oportunos lo anterior se refiere, a que tal vez ya se cuenta con empleo y no siempre hay volantes de vacantes. Además la información que presentan de la vacante no es suficiente por lo que no son muy populares entre los encuestados, ver Figura 4.4.

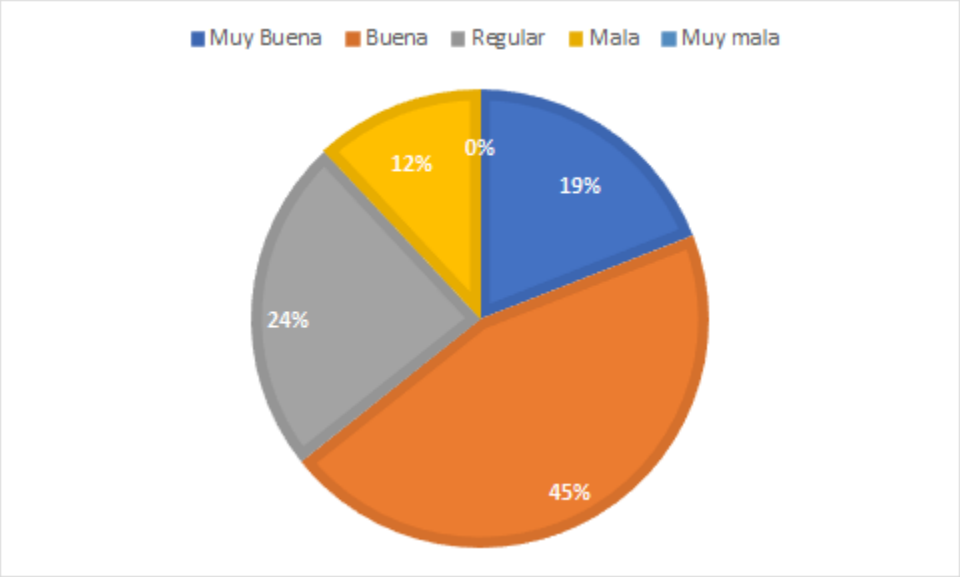


Figura 4.5: Resultados del medio de comunicación Módulos

Los resultados que arroja la encuesta acerca de los módulos de contratación que utilizan las empresas son buenos, ya que en varias ocasiones son contratados ahí mismo debido a que normalmente es personal de la empresa el que se encuentra a cargo de los módulos, sin embargo, existen casos donde las personas no forman parte de la empresa por lo que no se ofrece la información completa al usuario que busca empleo, ver Figura 4.5.

En la respuesta Otros no se realizó gráfica debido a que son diferentes las respuestas entre si, lo más destacable son los anuncios de televisión y acudir directamente a la empresa para ser contratados ya que algunas empresas acostumbran colocar mantas afuera de su ubicación para anunciar sus vacantes.

Con toda esta información se destaca la importancia de la información de la vacante, además que los encuestados buscan la información lo más veraz y cercana a la empresa para buscar empleo y así elegirlo.

La siguiente pregunta esta orientada a conocer el tiempo que consume de su día para

obtener empleo y con ello demostrar un beneficio extra si se llega a contratar por medio de la aplicación.

pregunta 2.¿Aproximadamente qué tiempo consume de su día para conseguir empleo?

Respuestas asignadas: Menos de una hora, De una a dos horas, Más de dos horas.

Figura 4.6: Resultados de la pregunta dos

Los resultados muestran que las personas en la mayoría de los casos tardan más de dos horas, esto se le puede atribuir a diferentes factores como el transporte público o las contrataciones de varias personas a la vez, ver Figura 4.6.

El objetivo de la pregunta tres es averiguar un poco más el tiempo en que tardan en conseguir empleo, saber si les toma más de un día conseguirlo.

Pregunta 3.¿Cuánto demora en conseguir empleo?

Respuestas asignadas: Un día, Más de un día, Una semana, Un mes.

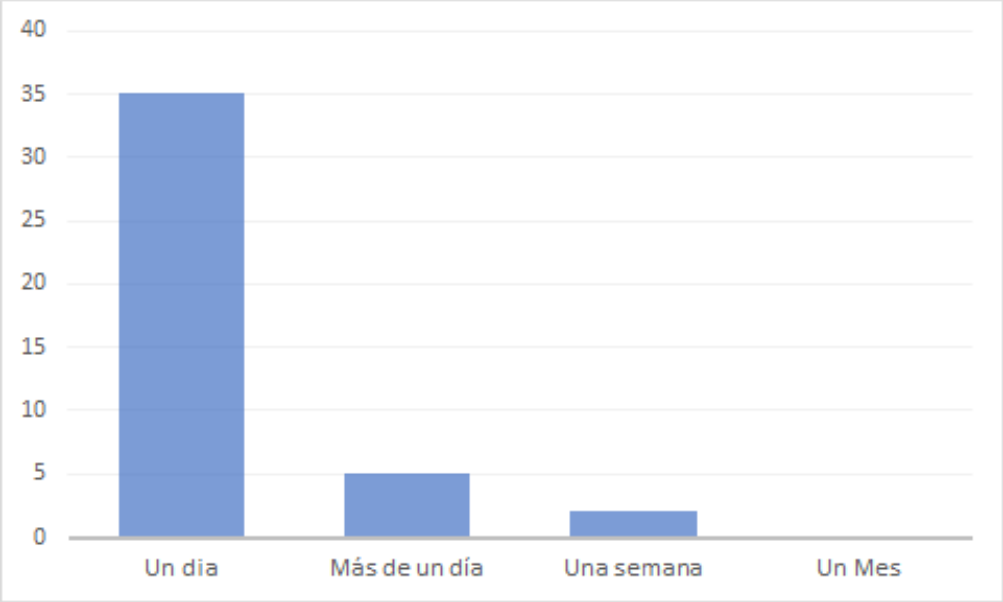


Figura 4.7: Resultados de la pregunta tres

Los resultados arrojaron que la mayoría de las veces se contratan en el primer día de buscar empleo lo cual también arroja porqué las personas que buscan empleo se quedan en el primer lugar donde los contraten y normalmente no buscan más opciones, ver Figura 4.7.

La pregunta cuatro se enfoca en obtener qué medios de difusión de vacantes conocen los entrevistados para conocer cuál es el más popular entre la población.

Pregunta 4.¿Cuáles de las siguientes páginas para conseguir empleo conoce o a utilizado?, puede seleccionar varias.

Respuestas asignadas: Index Juárez, Indeed, Empleos en Juárez, Paginas de agencias de empleo, Ninguna

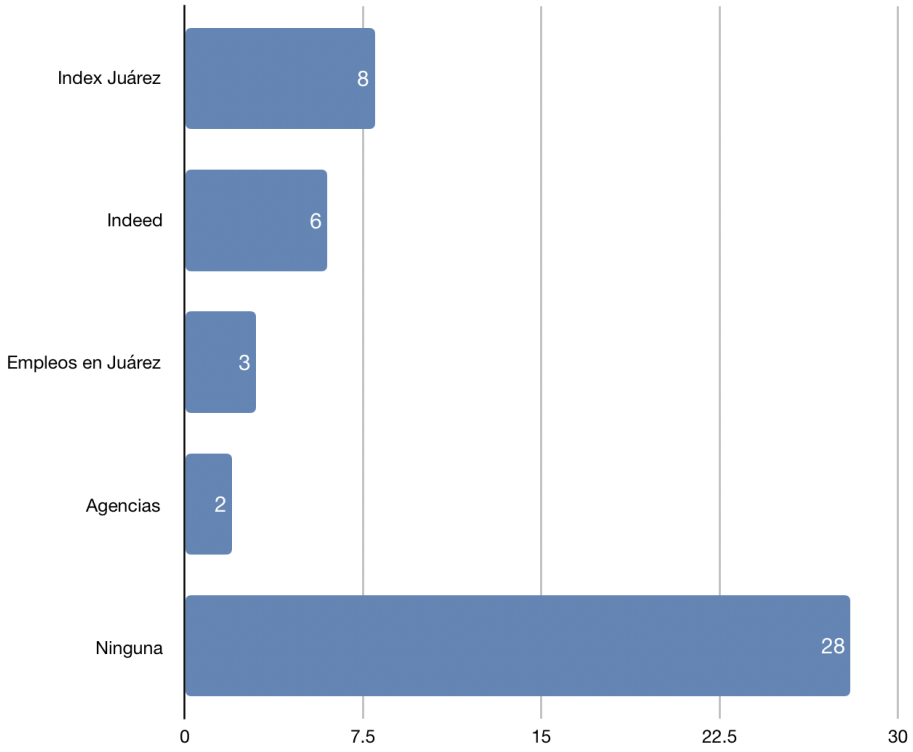


Figura 4.8: Resultados de la pregunta cuatro

Los resultados obtenidos demuestran que las personas no acostumbran buscar empleo por medios virtuales lo cual es una barrera que se tendrá que superar para lograr contrataciones por medio de la aplicación, ver Figura 4.8.

La siguiente pregunta busca conocer la información de la vacante más importante para el usuario, con el objetivo que la vacante cuente con esta información para hacerla más completa.

Pregunta 5.¿Cuál de las siguientes opciones considera que es la más importante en una vacante? Seleccione sólo una.

Respuestas asignadas: Salario, Transporte, Turnos(Horarios), Documentos solicitados,

Otro(Especifique).

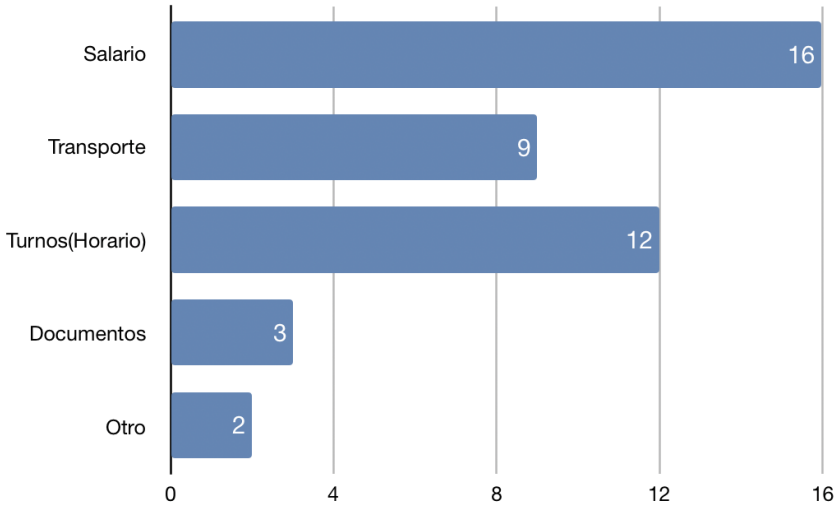


Figura 4.9: Resultados de la pregunta cinco

Los resultados como podemos observar, lo más importante para los usuarios que buscan empleo es saber el salario con el que serán contratados, seguido del horario para así obtener cual se ajusta más a sus necesidades, ver Figura 4.9.

Las siguiente pregunta es la última de esta primera parte de la encuesta la cual busca conocer que tan familiarizado esta el usuario con las aplicaciones móviles.

Pregunta 6.¿Cómo es su grado de conocimiento sobre aplicaciones móviles?

Respuestas asignadas: Muy bueno, Bueno, Regular, Malo y Muy malo

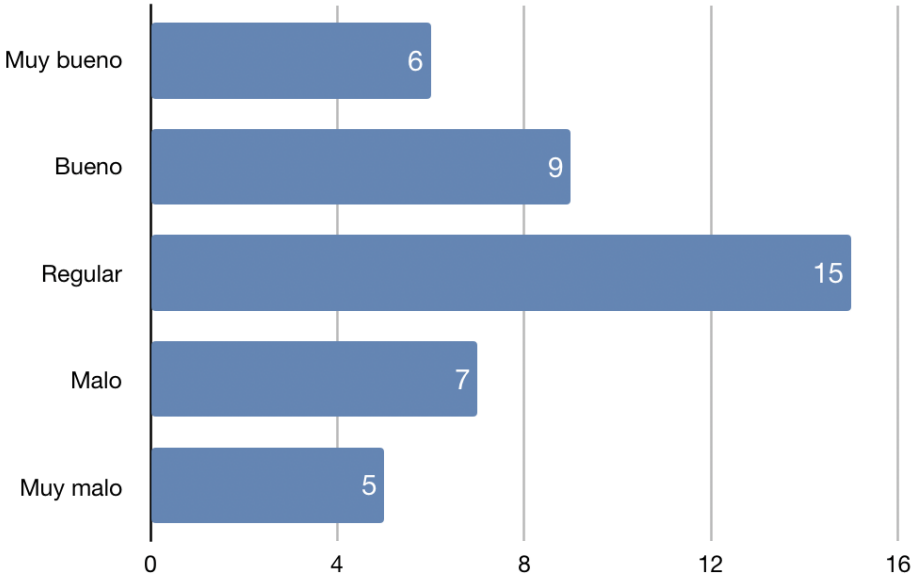


Figura 4.10: Resultados de la pregunta seis

Como se puede observar el conocimiento de los encuestados es regular en la mayoría de los casos por lo que no son muy hábiles al utilizar las aplicaciones móviles, lo cual puede ser una limitante a la hora de presentar la aplicación, ver Figura 4.10.

Rendimiento de la aplicación

En esta segunda parte de la encuesta no enfocamos en el rendimiento de la aplicación para comprobar el grado de compatibilidad con los dispositivos además de verificar el buen funcionamiento de la misma.

La primera pregunta tiene como objetivo conocer el grado de compatibilidad de la aplicación con los dispositivos de los usuarios encuestados.

Pregunta: 1.¿La aplicación es compatible con su dispositivo?

Respuestas asignadas: Si, cuento con una versión igual o mayor a la aplicación, No, mi

dispositivo tiene una versión menor, Mi dispositivo no es compatible

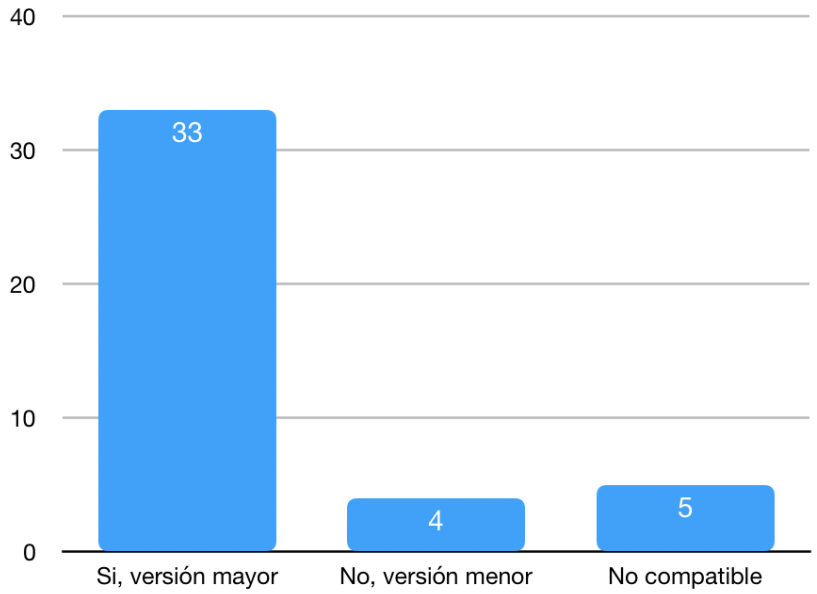


Figura 4.11: Resultados de rendimiento pregunta uno

Para la mayoría de los usuarios la aplicación es compatible con su dispositivo móvil aunque aun ay excepciones ya hay usuarios que su celular no soporta la aplicación debido a que son antiguos o son sólo para llamar, ver Figura 4.11.

La pregunta dos esta enfocada en conocer que bien funciona el registro de la aplicación Pregunta: 2.¿Logró registrarse correctamente?

Respuestas asignadas: Si, No(¿Por qué?)

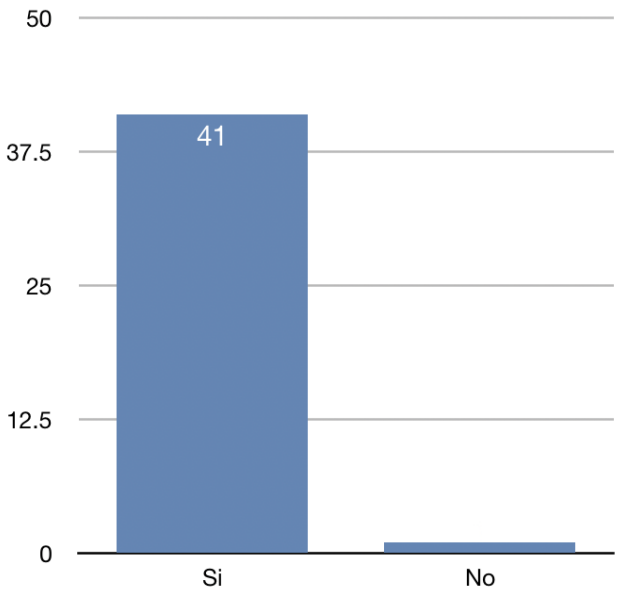


Figura 4.12: Resultados rendimiento pregunta dos

Se puede observar que la mayoría de los usuarios pudieron registrarse correctamente en la aplicación, sólo un usuario no pudo en un principio por causas de la conexión de internet que se tenía en ese momento, ver Figura 4.12.

La pregunta tres de este apartado esta enfocada en conocer si la aplicación tiene algún fallo a la hora de cargar la información o un error que detenga la aplicación y se cierre de repente.

Pregunta: 3.¿La aplicación se detuvo en algún momento?

Respuestas asignadas: Si, ¿En que parte de la aplicación?, No

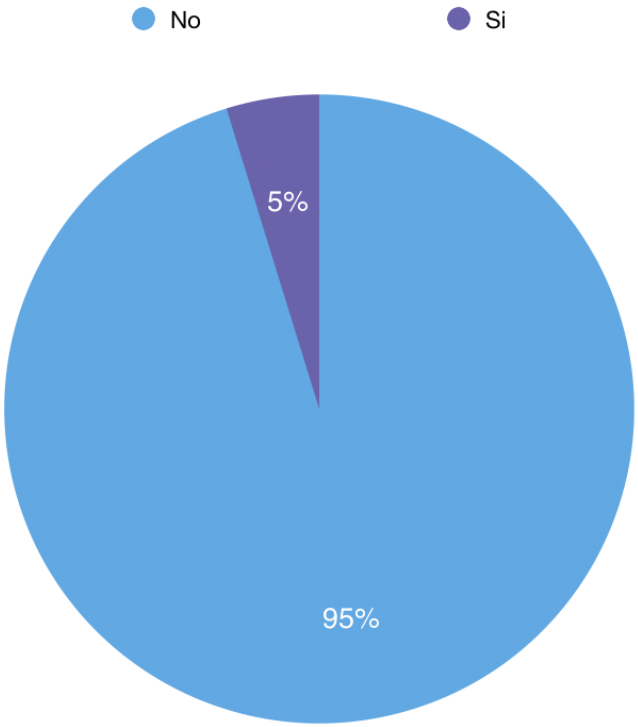


Figura 4.13: Resultados rendimiento pregunta tres

Los resultados son muy buenos debido a que solo se tuvieron dos fallas lo cual es el 5 % de la muestra a cual se le aplicó la encuesta. Además funcionó para corregir un error que se tenia a la hora de cargar las vacantes, la aplicación no cargaba nada y se cerraba, ver Figura 4.13.

La pregunta cuatro tiene como objetivo ver si la información que muestra la aplicación se carga de manera correcta y si la carga toda. Si lo hace, que este en el lugar que corresponde para verificar los constrains de las aplicaciones.

Pregunta: 4.¿La aplicación mostró la información correctamente?

Respuestas asignadas: Si, No, ¿Qué información no cargo?

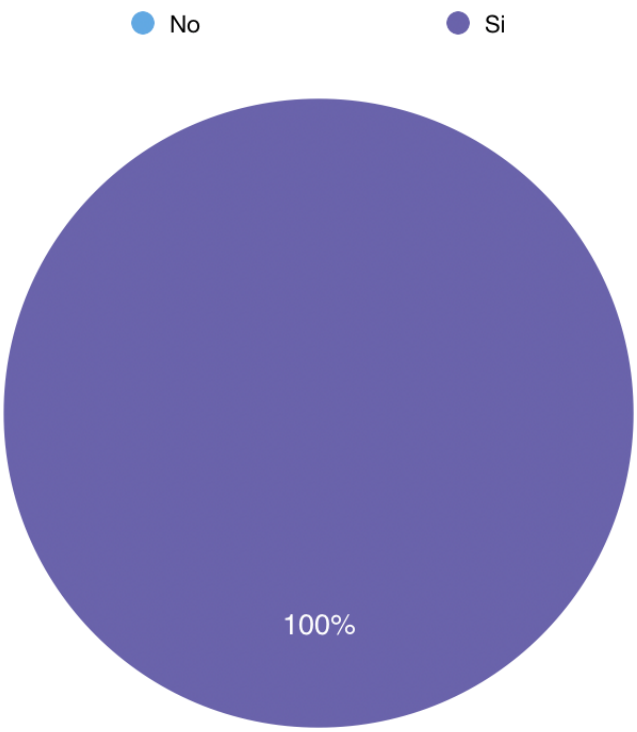


Figura 4.14: Resultados rendimiento pregunta cuatro

Toda la información mostrada por la aplicación se carga y se mira de manera correcta según las encuestas aplicadas, ver Figura 4.14.

La última pregunta es un apartado de observaciones en el cual los usuarios dan observaciones si hay algo que les gustaría cambiar o agregar a la aplicación. Las cuales sugieren un chat para comunicarse directamente con la empresa si es que aceptan la solicitud, otra petición destacable es que los usuarios puedan ver los recorridos del transporte de personal que tiene cada empresa.

Usabilidad y retroalimentación En esta ultima parte de la encuesta el enfoque esta dirigido hacia lo amigable que es la interfaz de la aplicación con los usuarios que la utilizan, ver si es de utilidad para los mismos y mejora o hace más sencillo la búsqueda de trabajo.

La pregunta número uno de esta parte busca saber que tan útil es la aplicación con la información que se maneja de cada vacante y si con esta se puede conseguir empleo, que el usuario sienta que esta más acorde con el.

Pregunta: 1.¿Cree que la aplicación le ayuda a buscar empleo mas acorde a sus habilidades?

Posibles respuestas: Si, me permite ver varias opciones, Si, la información es muy detalla, La información no es útil, No, ¿Por qué?

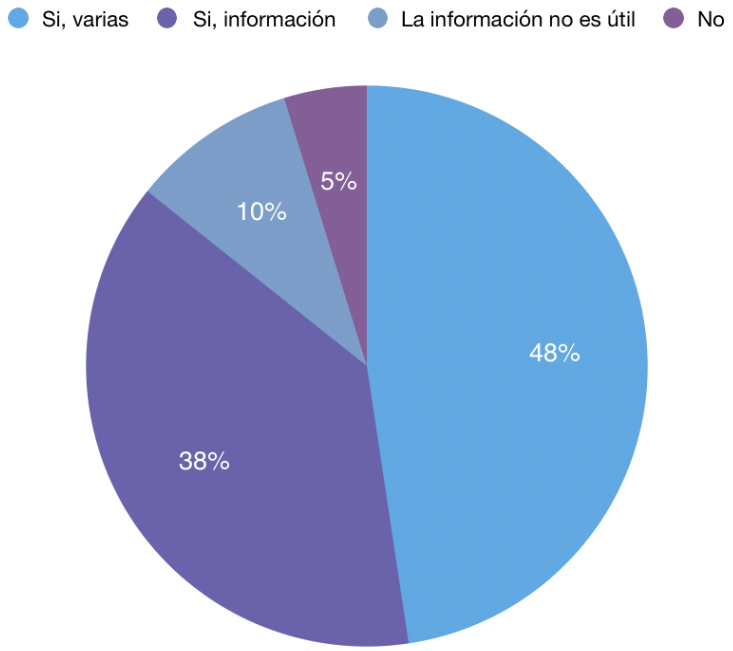


Figura 4.15: Resultados de usabilidad pregunta uno

Los resultados son muy satisfactorios ya que la mayoría de los usuarios opina que la información es muy buena y detallada con lo que el beneficio de buscar empleo se incrementa. Por otro lado los entrevistados también opinan que hace falta mas variedad de empresas lo

cual es correcto pero al estar en una fase temprana la aplicación. Se necesitaría contactar con empresas que estén interesadas en publicar sus vacantes para una mayor variedad, ver Figura 4.15.

La pregunta dos busca comparar la calidad de la información ofertada en la aplicación con la que se ofrece en un modulo de contratación.

Pregunta: 2.En comparación a los módulos de información, ¿Cree que la aplicación es inferior, igual o mejor?

Posibles respuestas: Es muy superior, Tiene mejor información, Es igual, Tiene menor información, Los módulos son mejores

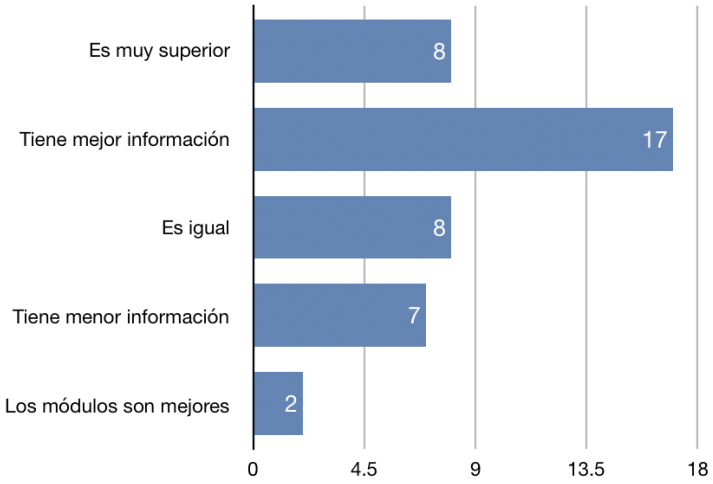


Figura 4.16: Resultados de usabilidad pregunta dos

La aceptación de las personas entrevistadas entre las aplicación y los módulos de información es buena si consideramos que normalmente son las empresas las encargadas de los módulos de información, tienen toda la información de la vacante y la pueden dar de persona a persona, además de resolver todas las dudas que se generen o tenga el usuario en el mismo

momento, así que son muy aceptables los resultados, ver Figura 4.16.

La siguiente pregunta nos ayuda con la interfaz de la aplicación, para saber que tan amigable e intuitiva es.

Pregunta: 3.¿Del 1 al 5 como considera la interfaz de la aplicación? Siendo 1 muy mala y 5 excelente.

Posibles respuestas: 1 Muy mala, 2 Mala, 3 Regular, 4 Buena, Excelente

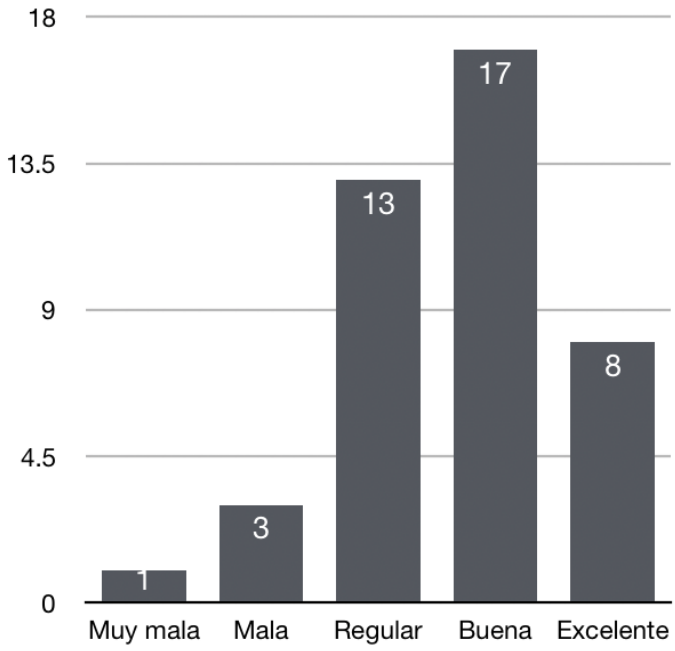


Figura 4.17: Resultados usabilidad pregunta tres

La interfaz de la aplicación tiene una buena aceptación entre los entrevistados ya que sus respuestas son muy positivas al diseño de la aplicación lo cual sugiere que la linea seguida es la correcta, ver Figura 4.17.

La pregunta cuatro es un poco diferente de las además ya que esta pregunta se deja abierta

a los usuarios para observaciones, por ejemplo que tanto les agrada la propuesta presentada para buscar empleo. También como retroalimentación de la aplicación mencionan aspectos para mejorarla, cosas que no les agradaron o que les gustaría ver integradas en una futura actualización.

Pregunta: 4.¿Qué es lo que más te llamó la atención positiva o negativamente de la utilidad que ofrece la aplicación?

Figura 4.18: Resultados positivos de usabilidad pregunta cuatro

Respuestas Positivas de los usuarios	Numero que repite
Ayuda a localizar empleo más rápido y cómodo	16
Se puede enviar solicitud a varias empresas	12
Ahorra tiempo	8
Me gusto la aplicación	4
Funciona con mi dispositivo	2
Puedes comparar salarios de empresas	7
Tiene información de la empresa	2

Los aspectos positivos de la aplicación son muy satisfactorios para conseguir la aceptación de la mayoría de la población de Ciudad Juárez para lograr convertir la aplicación en un medio apto para la difusión de las vacantes, ver Figura 4.

Figura 4.19: Resultados recomendaciones de usabilidad pregunta cuatro

Respuestas Positivas de los usuarios	Numero que repite
Hay muy pocas empresas registradas	11
Solo hay maquiladoras, debería haber mas diversidad, no lo maquiladoras	3
No muestra si el transporte pasa por donde vivo	7

Algunas recomendaciones dadas por los usuarios están se enlistan en la Figura 4 siendo las tres recomendaciones para mejorar la aplicación. La que más quieren los usuarios, es una

mayor diversidad de empresas ya que así les parece muy incompleta, otro apartado es que la aplicación no muestra si el transporte público pasa por donde viven, este sólo dice si la empresa cuenta con transporte. Por último a los usuarios también les gustaría que se ofertara todo tipo de empleo y no sólo empresas.

La última pregunta de la encuesta esta orientada a definir si ayuda o no la aplicación a difundir las vacantes de las empresas.

Pregunta: 5.- ¿Cree que la aplicación ayuda a difundir la información de la vacante? Posibles respuestas: Si, No, ¿Por qué?

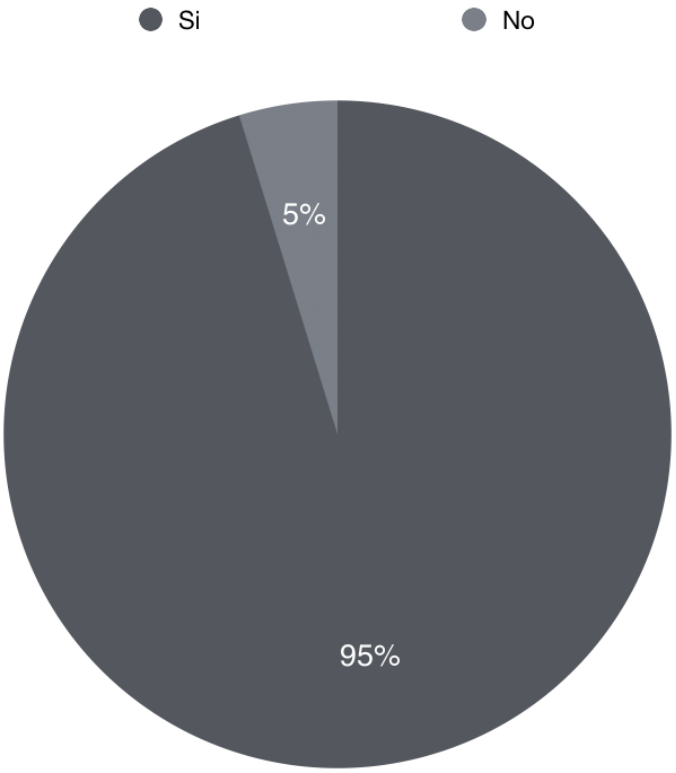


Figura 4.20: Resultados de usabilidad pregunta cinco

Los resultados arrojados por esta pregunta son muy buenos ya que la mayoría esta de

acuerdo con que la aplicación es un buen medio para ayudar a la difusión de la información de las vacantes. Por el contrario el porcentaje negativo cree el problema mas grande de la aplicación es que los usuarios la descarguen y la utilicen cuando deseen buscar empleo.Pero en general los resultados son muy satisfactorios, ver Figura 4.20.

Encuesta enfocada al personal de la empresa

La segunda encuesta se aplicó a las personas de recursos humanos encargados de publicar las vacantes, la cual también esta dividida en tres partes: la primera parte enfocada en la usabilidad, la segunda parte: se enfoca en la navegación dentro de la aplicación y el rendimiento para cada usuario, la última parte se dirige en la satisfacción del cliente conforme a los requerimientos solicitados.

El objetivo de la encuesta en la empresa, es conocer las opiniones del personal respecto al producto que se propuso, con base en los requerimientos de aspecto y funcionalidad.

Usabilidad

En esta parte se busca identificar el nivel de dificultad que le resulta al usuario utilizar la aplicación, por sus diferentes opciones y la experiencia que le dejo al interactuar con cada una de las funciones implementadas.

La pregunta uno, busca identificar el nivel de dificultad para poder registrarse como empresa.

Pregunta 1.¿Completo sin dificultad el registro para empresas?

Respuestas asignadas: Muy Fácil, Fácil, Regular, Difícil, Muy Difícil

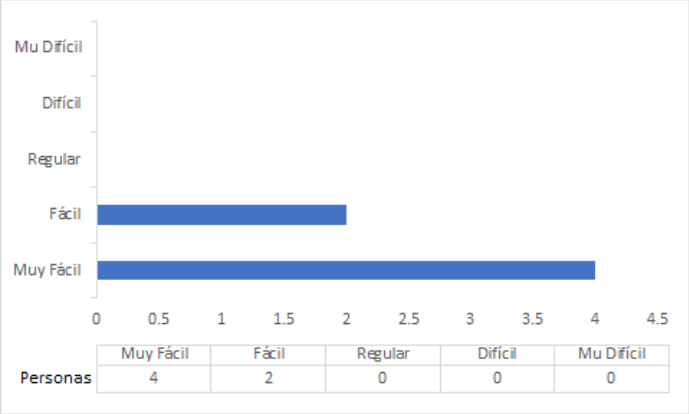


Figura 4.21: Resultados de la pregunta 1

Los resultados obtenidos de la pregunta uno hacen referencia a que los usuario no presentaron ningún problema para registrarse e iniciar sesión como empresa, ver Figura 4.21.

La pregunta dos hace referencia a la funcionalidad principal de la aplicación por lo que se busca obtener la opinión de los encargados de publicar nuevas vacantes.

Pregunta 2.¿Pudo publicar una vacante sin problemas?

Respuestas asignadas: No tuve ningún problema, Publique pero tuve dudas, No pude publicar una vacante.

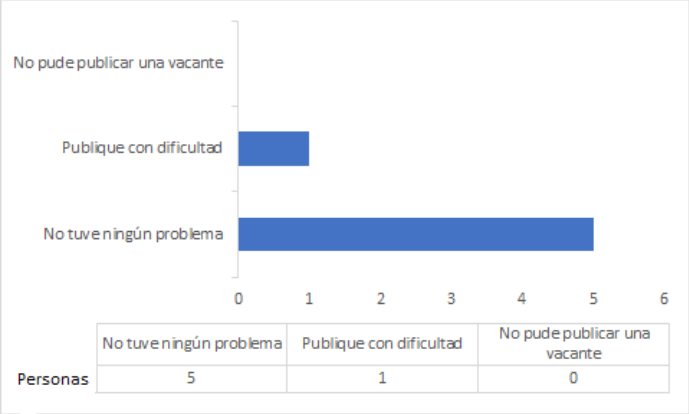


Figura 4.22: Resultados de la pregunta 2

Con base a los resultados los usuarios definieron que la forma en la que se publica la vacante es correcta y no hubo dificultada para realizar esta acción, ver Figura 4.22.

La pregunta tres, busca identificar la falta de algún componente visual o que no cuente con una descripción.

Pregunta 3.- ¿La estructura del formulario para crear una nueva vacante es la correcta?

Respuestas Asignadas: Esta completo, Bien, pero faltan componentes, No es lo que esperaba.

Figura 4.23: Resultados de la pregunta 3

Los resultados de la pregunta cuatro, indican que la estructura del formulario es la correcta para sus necesidades de publicar nuevas vacantes, ver Figura 4.23.

La cuarta pregunta de la encuesta se centra en la interfaz de solicitudes, se busca identificar algún problema al momento de revisar a los candidatos para sus ofertas de empleo.
Pregunta 4.- ¿Encontró la interfaz de solicitudes y pudo visualizar las disponibles? Respuestas asignadas: Encontré la interfaz y sus solicitudes, La encontré pero con dificultad, No encontré la interfaz.

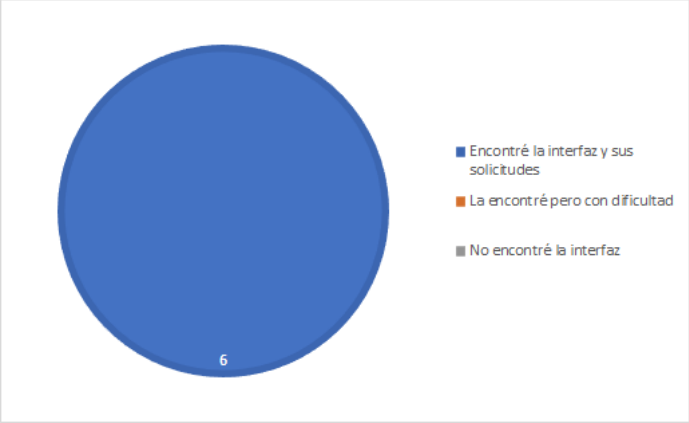


Figura 4.24: Resultados de la pregunta 4

Conforme a los resultados obtenidos, se observa que la mayoría del personal encontró de forma correcta la sección de solicitudes y pudieron ver las candidaturas disponibles en el momento, ver Figura 4.24.

Las postulaciones mostradas fueron diseñadas solo para el test.

La ultima pregunta de la encuesta de usabilidad, hace referencia a la gestión de los candidatos a una vacante. Se busca definir si es la manera correcta para poder reclutar personal.

Pregunta 5.¿La forma para aceptar o rechazar a un candidato es la correcta para usted?

Respuestas asignadas.- Cumple con lo esperado, Es correcta pero faltan detalles, No es la adecuada.

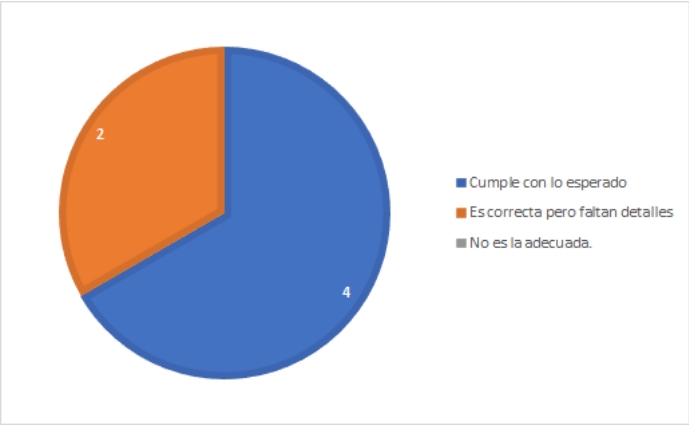


Figura 4.25: Resultados de la pregunta 5

Los resultados indican que la manera para aceptar o rechazar a un candidato al empleo, es la adecuada a sus necesidades pero mencionan que es necesario visualizar toda la información del currículum para tomar la decisión, esto refiere a que la aplicación al momento de realizar las pruebas con el personal, solo muestra un resumen del postulante. La opinión fue tomada en cuenta para modificaciones antes del prototipo final.

Como conclusión de la prueba de usabilidad, se observa que el personal encargado de usar la aplicación de lado de la empresa esta conforme con el diseño, ellos mencionan que resulto fácil el uso de las funciones diseñadas para los reclutadores, ver Figura 4.25.

Navegación y rendimiento

La segunda parte de las pruebas realizadas al personal de la empresa, es definir si la navegación dentro de la aplicación es correcta para moverse entre las diferentes vistas. Se incluyen las pruebas de rendimiento para detectar problemas en los componentes antes de terminar la versión final del prototipo.

La pregunta uno se relaciona con la navegación del usuario hacia una interfaz oculta al iniciar la aplicación, por lo que se busca detectar si fue posible llegar de forma correcta hasta la vista de registro para empresa.

Pregunta 1.¿Los botones para llegar al inicio de sesión funcionan correctamente? Respuestas asignadas: Si, No existe el botón, Un botón te manda a otra vista diferente.

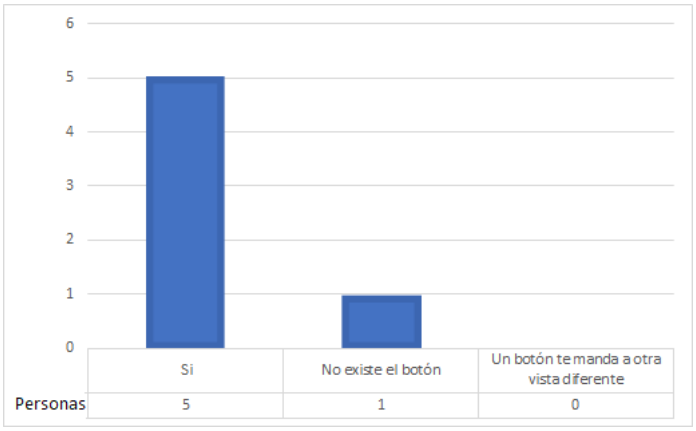


Figura 4.26: Resultados de la pregunta 1

Con los resultados obtenidos se concluye el correcto funcionamiento de los botones para registrar a una empresa, esto resulta satisfactorio debido a que es una de las funciones más importantes dentro de la aplicación, ver Figura 4.26.

La pregunta dos busca detectar algún fallo en el menú principal de la aplicación ya que este es el encargado de manejar la navegación entre las diferentes interfaces.

Pregunta 2.¿El menú principal lleva de forma correcta a cada una de las interfaces? Respuestas asignadas: Funciona correctamente, Una o más secciones fallan, El menú no funciona. Especifique la falla.

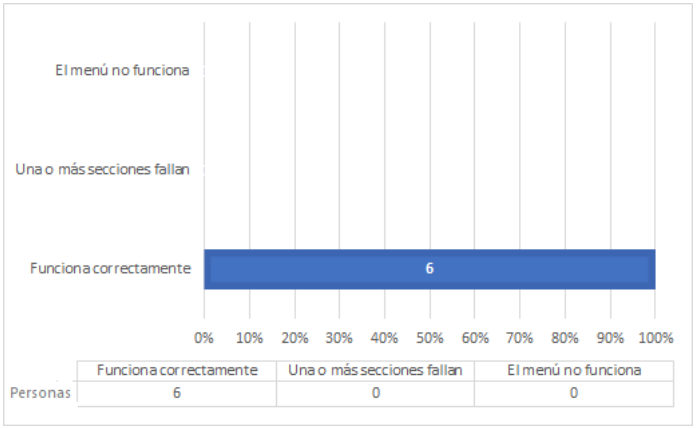


Figura 4.27: Resultados de la pregunta 2

El menú de navegación principal hace el llamado a cada una de las interfaces, los resultados así lo definen y se concreta que este componente de la aplicación terminó de forma correcta, ver Figura 4.27.

La pregunta tres hace mención durante la navegación en la aplicación es necesario retroceder a una vista anterior, para esto se asignaron botones de retroceso, por lo que se busca detectar si todos funcionan correctamente o si alguna vista carece de esta opción.

Pregunta 3.¿Los botones para regresar a una vista anterior funcionan correctamente? Respuestas asignadas: Todos funcionan bien, Uno o más no funcionan, funcionan pero algunas vistas no tienen el botón.

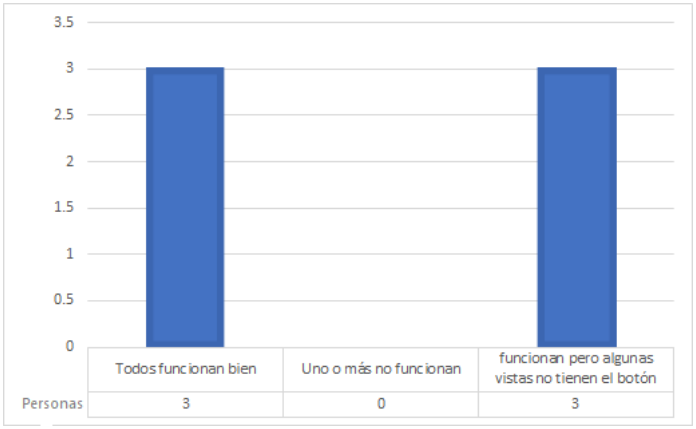


Figura 4.28: Resultados de la pregunta 3

Los resultados indican que los botones que si fueron asignados a las interfaces funcionan correctamente pero cabe destacar que algunos de los encuestados detectaron que había interfaces que no contaban con la opción, se tomo esto en cuenta y se resolvió el problema, ver Figura 4.28.

La pregunta cuatro se centra en el rendimiento de la aplicación, del dispositivo de la persona encuestada, para esto se le especifica al usuario que detecte algún problema de calentamiento o de lentitud.

Pregunta 4.¿Como sintió el desempeño de la aplicación en su dispositivo? Respuestas asignadas: Fluye correctamente, Se calienta mi dispositivo, se traba la aplicación.

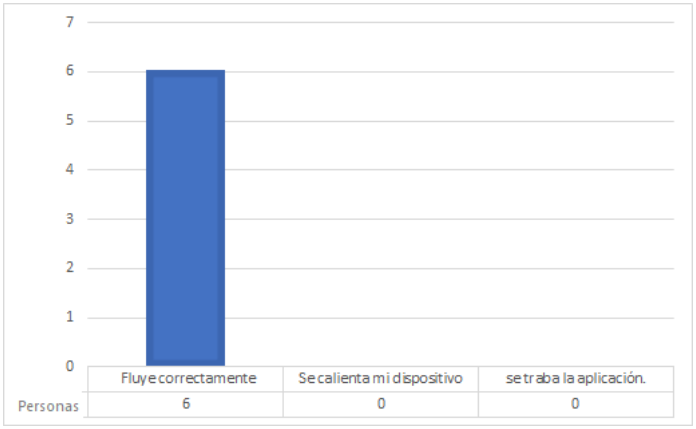


Figura 4.29: Resultados de la pregunta 4

Los resultados indicaron que la aplicación corre correctamente en los dispositivos de prueba. Debido a qué en Android existe una gran variedad de dispositivos, al inicio del desarrollo del proyecto se especifica los requerimientos mínimos para usar la aplicación, por lo que si el dispositivo cumple con estos no debe presentar ningún problema para hacer uso del prototipo, ver Figura 4.29.

La última pregunta busca identificar alguna falla en general de la aplicación, por lo que se busca que los encuestados estresen la aplicación, a esto se refiere de abrir todas las interfaces y presionar todos los botones, etc.

Pregunta 5.¿La aplicación falló al realizar alguna función?
Respuestas asignadas: No fallo, Si fallo. Especifique en cual función.

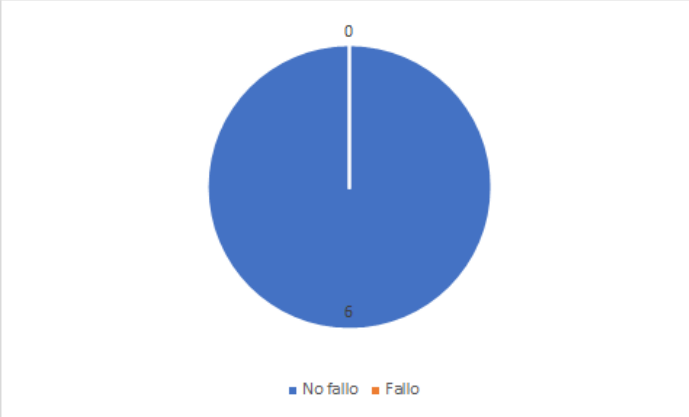


Figura 4.30: Resultados de la pregunta 5

Con los resultados obtenidos se concluyo que la aplicación funciona correctamente, el caso único de falla fue sobre problemas de conectividad en el dispositivo el cual influyo en la aplicación al cargar algunos datos, ver Figura 4.30.

Experiencia de usuario

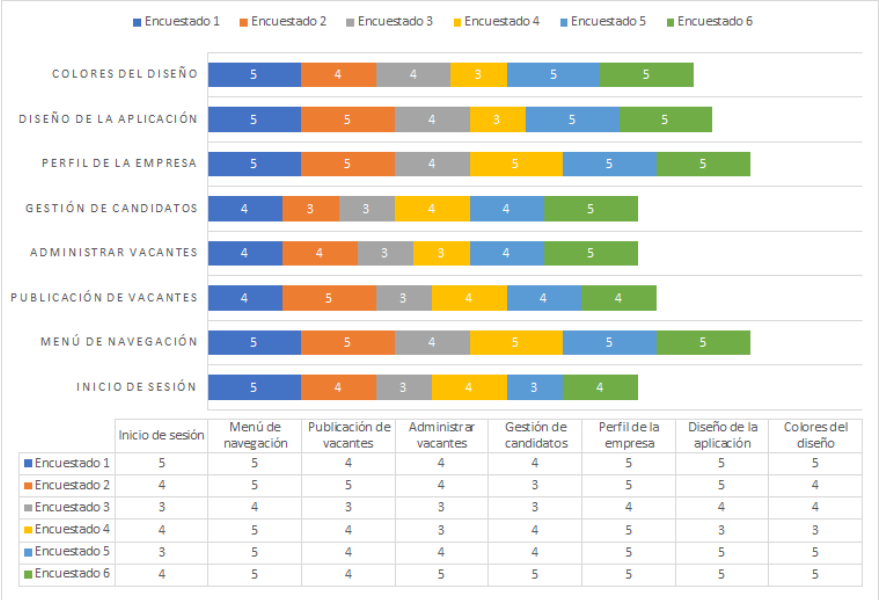
La última encuesta fue realizada utilizando la escala de Likert para conocer el nivel satisfacción del usuario con el prototipo. Esta encuesta busca detectar que funciones de la aplicación están cumpliendo correctamente con lo establecido en los análisis y requerimientos implementados al inicio del desarrollo.

Titulo de la encuesta de Likert: Indique que tan satisfecho esta con las funciones de la aplicación respecto a su experiencia de uso.

Los enunciados que el usuario debe de evaluar se compone de elementos directos de la aplicación como lo son: inicio de sesión, menú de navegación, publicación de vacantes, administrar vacantes, gestión de candidatos, perfil de la empresa, diseño de la aplicación y colores del diseño.

La escala de Likert se define con un nivel de satisfacción del usuario por cada uno de los

enunciados anteriores, esta compuesta por: nada satisfecho, poco satisfecho, satisfecho, muy satisfecho y totalmente satisfecho.



	Inicio de sesión	Menú de navegación	Publicación de vacantes	Administrar vacantes	Gestión de candidatos	Perfil de la empresa	Diseño de la aplicación	Colores del diseño
Encuestado 1	5	5	4	4	4	5	5	5
Encuestado 2	4	5	5	4	3	5	5	4
Encuestado 3	3	4	3	3	3	4	4	4
Encuestado 4	4	5	4	3	4	5	3	3
Encuestado 5	3	5	4	4	4	5	5	5
Encuestado 6	4	5	4	5	5	5	5	5

Figura 4.31: Encuesta de satisfacción con escala de Likert

Como se ve en la tabla de resultados ver Figura 4.31, los encuestados en su mayor parte, determinaron que están muy satisfechos con el prototipo en general. Como se observa en los resultados la calificación más baja resultó la funcionalidad de administrar vacantes y la gestión de candidatos, esto impacta en que estas funciones cumplen con lo esperado, pero que se esperaba más de ellas, por lo que se agrega a las observaciones para futuras versiones de la aplicación.

Los resultados de la encuesta ver Figura 4.31, muestran los niveles de satisfacción de nuestro cliente referente al prototipo entregado con los requerimientos especificados antes del desarrollo. Esta encuesta se toma como una opción para determinar si se cumplió con las expectativas pactadas y continuar con la entrega del prototipo.

Capítulo 5
Conclusiones

Con los resultados de las pruebas realizadas tanto a los candidatos en busca de un empleo y la empresa se concluye que utilizar una aplicación móvil para la difusión de vacantes de empleo es funcional, lo que quiere decir que las personas están dispuestas a tomar el reto de utilizar las tecnologías para conseguir un nuevo empleo y agilizar su proceso de contratación ahorrando tiempo y generando un menor gasto posible en cuanto a los traslados del postulante.

Por otro lado, el personal que realizó las pruebas con enfoque hacia las empresas se mostró dispuesta a utilizar la herramienta como alternativa para la difusión de las vacantes, mencionando que esto les ayuda a que la empresa participe con su área de recursos humanos en el ámbito tecnológico.

Debido a lo anterior, se puede concluir que una aplicación móvil como herramienta para buscar o difundir vacantes de empleo, delimitado a una ciudad, pero orientado a dos plataformas populares entre los dispositivos móviles puede ayudar a integrar a la población con las empresas en el sector tecnológico, facilitando entre ellos uno de los procesos de recursos humanos más dados en la ciudad como lo es la contratación de personal.

5.1. Con respecto al objetivo de la investigación

El propósito del desarrollo de este proyecto está dado por el echo de implementar las tecnologías para la difusión de las vacantes de empleo aprovechando el auge que tienen los dispositivos móviles en la ciudad y el acceso oportuno del internet.

El uso del dispositivo móvil viene acompañado de las aplicaciones. Realizar una aplicación para dispositivos móviles implica llegar a una mayor población a comparación de otros medios de difusión de información. Es aquí donde se toma esta ventaja para crear un proyecto que funcione como una alternativa para publicar información con respecto a una oferta de empleo, que las empresas cuenten con diferentes opciones para la obtención de nuevo personal, sin la necesidad de dejar de lado las actuales formas como lo son los módulos de contratación.

El haber desarrollado un prototipo de una aplicación móvil orientada a dos sistemas operativos de gran uso como lo son Android e IOS y enfocada a dos tipos de usuario que son los reclutadores de la empresa y los candidatos a un trabajo significa el aumentar las formas de difusión y las maneras para conseguir un empleo.

Debido a los resultados obtenidos con base al prototipo final se concluye que una aplicación móvil donde las personas puedan crear su propio currículum y mandarlo a diferentes vacantes sin la necesidad de acudir directamente a la empresa agiliza el método de contratación sin especificar si es mejor que otros.

Al igual que las empresas puedan difundir sus empleos por medio de una aplicación aumenta las posibilidades de llegar a los candidatos requeridos esto obtenido de los resultados de las pruebas del prototipo. Se hace mención que este es sólo un prototipo para la difusión de vacantes de empleo y las expectativas fueron tomadas con base a un cliente en específico y no aplicado en general a toda la ciudad y sus empresas.

5.2. Recomendaciones para futuras investigaciones

Las recomendaciones dadas en esta sección son las que surgieron durante las entrevistas, las cuales ayudarán hacer más atractiva la aplicación.

La primera de ellas es implementar un apartado de transporte de la empresa en el cual se muestre en el mapa las rutas que actualmente maneja la empresa, esto con la finalidad de apoyar al usuario de manera más exacta si se cuenta con transporte a su vivienda. La segunda recomendación es integrar un chat por así decirlo, en la cual la empresa y el usuario que solicita empleo puedan comunicarse entre si para resolver dudas o ya sea para pedir información que tal vez la empresa no publicó.

Se recomienda también utilizar en el caso de la aplicación de Android el lenguaje de programación Kotlin debido a que es el nuevo lenguaje de programación oficial y cuenta con nuevas herramientas de desarrollo que reduce el código, y mejora el rendimiento de las aplicaciones.

Bibliografía

[1] D. N. da Silva, R. K. Vieira, A. K. Vieira, and M. de Santiago, "Optimización del Proceso de Innovación para Proyectos Internos en las Empresas. (Spanish)," Optim. Innov. Process Intern. Proj. Companies., vol. 27, no. 3, pp. 119–129, May 2016.

[2] bruno.castro@indexjuarez.org, "Asociación de Maquiladoras -Index Juárez," 2018.

[3] S. L. Schibsted Classified Media Spain, "InfoJobs," AppStore. 2015.

[4] J. Ltd, "Jobandtalent," AppStore. 2018.

[5] J. S.A., "JOB TODAY," AppStore. 2015.

[6] X. MÓVIL, "Siete aplicaciones para encontrar trabajo desde el móvil en Android y iOS," Xataka Móvil. 2018.

[7] I. Inc., "Indeed," AppStore. 2017.

[8] W. S.L., "Worktoday," AppStore. 2017.

[9] M. U. de la Rosa and S. E. Ortiz Díaz, "Herramienta inteligente para una propuesta de empleo basado en minería de datos," Universidad Autónoma de Ciudad Juárez, 2017.

[10] Miguelo, "¿Qué es OsCommerce?," desarrolloWeb. 2005.

[11] bruno.castro@indexjuarez.org, "Asociación de maquiladoras -Index Juárez," Infograma No 3, 2018.

[12] B. C. Zapata, Android Studio Application Development: Create Visually Appealing Applications Using the New IntelliJ IDE Android Studio. Birmingham, UK: Packt Publishing, 2013.

[13] G. Intellij, "Android Studio," 2018. [Online]. Available: https://developer.android.com/studio/index.html?hl=es-419.

[14] J. C. Sheusi, Android Application Development for Java Programmers. Boston, MA: Course Technology PTR, 2013.

[15] I. Apogee Software, "Apogee Announces JavaTM Runtime Environments for AndroidTM," Business Wire (English). Feb-9AD.

[16] Apple Inc, "Xcode IDE," Apple Developer. pp. 1–86, 2017.

[17] Apple, "Swift," Apple Developer. 2018.

[18] Apple, "Programming with," Apple Developer. pp. 1–44, 2000.

[19] K. Dimitris, T. Danielle, T. Orta, P. Beusterien, and S. Giddins, "CocoaPods," 2018.

[20] I. E. S. L. Velez de Guevara, Gestión de Bases de Datos, 1st ed. 2018.

[21] Google, "Firebase," 2018. [Online]. Available: https://firebase.google.com/?hl=es-419.

[22] Google, "Cloud Firestore," 2018. [Online]. Available: https://firebase.google.com/docs/firestore/?hl=es-419.

[23] Google, "Cloud Storage," 2015. [Online]. Available: https://firebase.google.com/docs/storage/?hl=es-419.

[24] D. Engines, "System Properties Comparison Firebase Realtime Database vs. MySQL vs. SAP Advantage Database Server," 2016, 2016. [Online]. Available: https://db- engines.com/en/system/Firebase+Realtime+Database

[25] Google, "Firebase," 2018. [Online]. Available: https://firebase.google.com/?hl=es-419.

[26] C. Apress, Berkeley, Using Authentication in Firebase. In: The Definitive Guide to Firebase. CA: Apress, Berkeley, CA, 2017.

[27] Google, "Cloud Firestore," 2018. [Online]. Available: https://firebase.google.com/docs/firestore/?hl=es-419.

[28] Google, "Firebase Authentication." [Online]. Available: https://firebase.google.com/products/auth/?hl=es.

[29] Google, "Cloud Storage," 2015. [Online]. Available: https://firebase.google.com/docs/storage/?hl=es-419.

Capítulo 6
Apéndice A

6.0.1. Código Interfaz de usuario Android

Inicio de sesión

```
//Credenciales de sesión con Google
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAUL
            .requestIdToken(getString(R.string.default_web_client_id))
            .requestEmail()
            .build();

    mGoogleSignInClient = GoogleSignIn.getClient(this, gso);

    mAuth = FirebaseAuth.getInstance();

    // Revisar si el usuario ya inicio sesión
@Override
public void onStart() {
    super.onStart();
    FirebaseUser currentUser = mAuth.getCurrentUser();
```

```
        updateUI(currentUser);
    }

    //Acción del botón inicio de sesión
      @Override
    public void onClick(View v) {
        int i = v.getId();
        if (i == R.id.btn_registro) {
            signIn();
        } else if (i == R.id.btn_registro_empresa) {
            registroEmpresa();
        }
    }
```

6.0.2. Currículum virtual

```
    //Función que guarda los datos en Firebase
  private void guardarDatos() {

        Button enviar = findViewById(R.id.cvSiguiente1);
        enviar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View v) {
                if (traerDatos() != null) {
```

```
                db.collection("Usuarios").document(user.getUid())
                        .set(traerDatos())
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Intent myIntent = new Intent(UsuarioCurriculum.this,
                                startActivity(myIntent);
                            }
                        })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Snackbar.make(v,"¡Error de red!",Snackbar.LENGTH_SHO
                            }
                        });
            }
        }
    });
```

Vacantes

```
//Envió de datos hacia el adapter
  RecyclerView recyclerView = view.findViewById(R.id.rvVacantes);
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
```

```
        adapter = new VacantesAdapter(getActivity(), nombreDeEmpresa,direciconDeEmpresa,
        adapter.setClickListener(this);
        recyclerView.setAdapter(adapter);

        //Recibiendo datos en el adapter
          VacantesAdapter(Context context, ArrayList<String> eNombre, ArrayList<String>
        this.mInflater = LayoutInflater.from(context);
        this.empNombre = eNombre;
        this.empDireccion = eDireccion;
        this.empVacantes = eVacantes;
        this.empPostulantes = ePostulantes;
        this.empPuesto = ePuesto;
        this.empTurno = eTurno;
        this.fecha = efecha;
    }

    //Instancia de los componentes
      public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickLis
        TextView nombre;
        TextView direccion;
        TextView puesto;
        TextView postulantes;
        TextView vacantes;
        TextView fecha;
        TextView turno;
        ImageView logo;
```

```
        ViewHolder(View itemView) {
            super(itemView);
            nombre = itemView.findViewById(R.id.vacNombre);
            direccion = itemView.findViewById(R.id.vacDireccion);
            vacantes = itemView.findViewById(R.id.vacOfertas);
            puesto = itemView.findViewById(R.id.vacPuesto);
            postulantes = itemView.findViewById(R.id.vacPostulados);
            fecha = itemView.findViewById(R.id.vacFecha);
            turno = itemView.findViewById(R.id.vacTurno);
            logo = itemView.findViewById(R.id.vacImg);

            itemView.setOnClickListener(this);
        }
```

Empresas Registradas

```
    //Envió de datos al adapter
      RecyclerView recyclerView = view.findViewById(R.id.rvEmpresas);
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
        adapter = new EmpresasAdapter(getActivity(), nombreDeEmpresa,direciconDeEmpresa,
        adapter.setClickListener(this);
        recyclerView.setAdapter(adapter);


    //Recibiendo los datos
```

```
    EmpresasAdapter(Context context, ArrayList<String> eNombre,ArrayList<String> eDire
      this.mInflater = LayoutInflater.from(context);
      this.empNombre = eNombre;
      this.empDireccion = eDireccion;
      this.empVacante = eVacantes;
  }
```

Estatus Usuario

//Envió de datos al adapter RecyclerView recyclerView = view.findViewById(R.id.rvEstatus); recyclerView.setLayoutManager(new LinearLayoutManager(getContext())); adapter = new EstatusAdapter(getActivity(), nombreDeEmpresa,direciconDeEmpresa,puestoDeEmpresa, proceso1,proceso2,proceso3); adapter.setClickListener(this); recyclerView.setAdapter(adapter);

Menú principal

```
   //Función para manejar el menú de la aplicación
  private BottomNavigationView.OnNavigationItemSelectedListener mOnNavigationItemSelec
          = new BottomNavigationView.OnNavigationItemSelectedListener() {

      @Override
      public boolean onNavigationItemSelected(@NonNull MenuItem item) {
          Fragment fragment = null;
          switch (item.getItemId()) {
              case R.id.navigation_vacantes:
```

```
                fragment = new UsuarioVacantes();
                break;
            case R.id.navigation_empresas:
                fragment = new UsuarioEmpresasRegistradas();
                break;
            case R.id.navigation_estatus:
                fragment = new UsuarioEstatus();
                break;
            case R.id.navigation_perfil:
                fragment = new UsuarioPerfil();
                break;
            case R.id.navigation_ayuda:
                fragment = new UsuarioAyuda();
                break;
        }
        return loadFragment(fragment);
    }
};

//
  private boolean loadFragment(Fragment fragment) {
    //switching fragment
    if (fragment != null) {
        FragmentTransaction transaction = getSupportFragmentManager().beginTransacti
        transaction.setCustomAnimations(android.R.anim.fade_in, android.R.anim.fade_
        transaction.replace(R.id.fragment_container, fragment);
        transaction.addToBackStack("");
```

```
            transaction.commit();
            return true;
        }
        return false;
    }
```

6.0.3. Código Interfaz de usuario IOS

Iniciar sesión

```
    // funcion para retrasar acciones
    func delayWithSeconds(_ seconds: Double, completion: @escaping () -> ()) {
        DispatchQueue.main.asyncAfter(deadline: .now() + seconds) {
            completion()
        }
    }


    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
        self.view.endEditing(true)
    }

    // empieza a editar
    func textFieldDidBeginEditing(_ textField: UITextField) {
        moveTextField(textField, moveDistance: -50, up: true)
```

```
    }

    // termina de editar
    func textFieldDidEndEditing(_ textField: UITextField) {
        moveTextField(textField, moveDistance: -50, up: false)
    }

    // esconder el teclado cuando presiona la tecla return
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }

    // Mover textfield hacia arriba
    func moveTextField(_ textField: UITextField, moveDistance: Int, up: Bool) {
        let moveDuration = 0.3
        let movement: CGFloat = CGFloat(up ? moveDistance : -moveDistance)

        UIView.beginAnimations("animateTextField", context: nil)
        UIView.setAnimationBeginsFromCurrentState(true)
        UIView.setAnimationDuration(moveDuration)
        self.view.frame = self.view.frame.offsetBy(dx: 0, dy: movement)
        UIView.commitAnimations()
    }
//    Funcion para iniciar sesion automaticamente
    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
```

```
//comprueba si esta registrado
if Auth.auth().currentUser != nil {
    let userID = Auth.auth().currentUser!.uid
    let db = Firestore.firestore()
    print("el id del usuario es: \(userID)")
    // si esta registrado, comprueba si es empresa o usuario
    db.collection("Usuarios").document("\(userID)").getDocument { (document, err
        if let document = document, document.exists {
            // lo envia a la vista principal de usuario
           let vc = self.storyboard?.instantiateViewController(withIdentifier: "
           self.present(vc!, animated: true, completion: nil)
        } else {
            print("Document does not exist")
            db.collection("Empresas").document("\(userID)").getDocument { (docum
                if let document = document, document.exists {
                   let results = document.data()
                    // si es empresa, comprueba que este verificada antes de env
                    if let idData = results["DatosEmpresa"] as? [String: Any] {
                        let verificado = idData["Verificado"] as? Bool ?? false
                        if verificado == true{
                            // lo envia a la vista principal de empresa
                            let vc = self.storyboard?.instantiateViewController(
                            self.present(vc!, animated: true, completion: nil)
                        }else{
                            // si no esta verificado lo envia a las vista de esp
                            let vc = self.storyboard?.instantiateViewController(
```

```
                                self.present(vc!, animated: true, completion: nil)

                            }
                        }
                    }
                }
            }
        }

    }
    }
```

Currículum virtual

```
// guarda los datos que ya se subieron en la base de datos
func traerDatos(){
    let db = Firestore.firestore()
    let docRef = db.collection("Usuarios").document("\(userID)")

    docRef.getDocument { (document, error) in
        if let document = document {

            let myData2 = document.data()

            var myData = myData2["Perfil"] as? [String: Any]
```

```
//                  self.comida = myData["Publicidad"] as? String ?? ""
                let automovil1 = myData?["Automovil"] as? Bool ?? false
                if automovil1 == true{
                    self.automovilPropioSi.isChecked = true
                    self.automovil = true

                }else{
                    self.automovilPropioNo.isChecked = true
                }
                let estudiasActualmente2 = myData?["Estudias Actualmente"] as? Bool ?? f
                if estudiasActualmente2 == true{
                    self.estudiasActualmenteSi.isChecked = true
                    self.estudiasActualmente = true
                }else{
                    self.estudiasActualmenteNo.isChecked = true
                    self.estudiasActualmente = false
                }
                let idiomaExtra = myData?["Idioma Extra"] as? Bool ?? false
                if idiomaExtra == true{
                    self.idiomasExtraSi.isChecked = true
                    self.idioma = true

                    self.idioma1Label.isHidden = false
                    self.idioma2Label.isHidden = false

                    self.idioma1TextField.isHidden = false
                    self.idioma2Textfield.isHidden = false
```

```
}else{
    self.idiomasExtraNo.isChecked = true
}

let nombre = myData?["Nombre"] as? String ?? ""
self.nombreTextField.text = nombre

let edad = myData?["Edad"] as? Int ?? 0
self.edadText.text = "\(edad)"

let correo = myData?["Correo"] as? String ?? ""
self.correoElectronico = correo

let apellido = myData?["Apellido"] as? String ?? ""
self.apellidoTextfield.text = apellido

let telefono = myData?["Telefono"] as? String ?? ""
if telefono != ""{
    self.numeroCelularTextfield.text = "\(telefono)"
}
let direccion = myData?["Calle"] as? String ?? ""
if direccion != ""{
    self.direccionTextfield.text = direccion
}
let colonia = myData?["Colonia"] as? String ?? ""
if colonia != ""{
    self.coloniaTextField.text = colonia
```

```
                }
                let idioma1 = myData?["Idioma1"] as? String ?? ""
                if idioma1 != ""{
                   self.idioma1TextField.text = idioma1
                }
                let idioma2 = myData?["Idioma2"] as? String ?? ""
                if idioma2 != ""{
                    self.idioma1TextField.text = idioma2
                }

            } else {
                print("Document does not exist")
            }
        }
    }

    @IBAction func actualizarAction(_ sender: Any) {
        actualizarDatos()
    }
    //funcion que actualiza los datos de informacion personal
    func actualizarDatos(){
        //comprueba que los textfield no esten vacios
        if nombreTextField.text == "" ||
        apellidoTextfield.text == "" || direccionTextfield.text == "" || edadText.text =
            let alertController = UIAlertController(title: "Error", message: "Por favor
            let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
            alertController.addAction(defaultAction)
```

```
            present(alertController, animated: true, completion: nil)
        }else {
            let washingtonRef2 = db.collection("Usuarios").document(userID)
            washingtonRef2.updateData([
                //datos que se actualizan en la base de tados
                "Perfil":([
                    "Automovil": automovil,
                    "Nombre": self.nombreTextField.text as Any,
                    "Apellido": self.apellidoTextfield.text as Any,
                    "Telefono": self.numeroCelularTextfield.text as Any,
                    "Calle": self.direccionTextfield.text as Any,
                    "Colonia": self.coloniaTextField.text as Any,
                    "Idioma1": self.idioma1TextField.text as Any,
                    "Idioma2": self.idioma2Textfield.text as Any,
                    "Idioma Extra": idioma,
                    "Correo": correoElectronico,
                    "Edad": Int(edadText.text!) ?? 0,
                    "Estudias Actualmente": estudiasActualmente,

                    ])
            ]) { err in
                if let err = err {
                    print("Error updating document: \(err)")
                } else {
                    print("Document successfully updated")
                    if self.deDondeVengo == 1{
                        //avisa al usuario si se actualizaron los datos
```

```
                    let alertController = UIAlertController(title: "Actualizar", mes
                    alertController.addAction(UIAlertAction(title: "Ok", style: .def
                        print("Handle Ok logic here")
                        self.navigationController?.popViewController(animated: true)
                                                    }))
                    self.present(alertController, animated: true, completion: nil)
                }else{
                    //se cambia de view para no volver actualizar por accidente
                    let vc = self.storyboard?.instantiateViewController(withIdentifi
                    self.present(vc!, animated: true, completion: nil)
                }
            }

        }
    }
}
```

Todas las vacantes

```
  //funcion que indica el tamaño que tendra la tabla
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    //indica el tamaño de la tabla con el numero de empresas obtenidas de la base de
    return (empresasArray.count)

}
//funcion que pone la informacion en la tabla
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITab
```

```
let cell = tableView.dequeueReusableCell(withIdentifier: "customCell", for: inde

cell.nombreEmpresa.text = empresasArray[indexPath.row]
cell.direccionEmpresa.text = direccionArray[indexPath.row]
cell.nombreVacante.text = nombreVacanteArray[indexPath.row]
cell.postulantesNumero.text = postulantesArray[indexPath.row]
cell.vacantesNumero.text = vacantesArray[indexPath.row]
cell.fechaPublicado.text = fechaArray[indexPath.row]

let imageURL3 = Storage.storage().reference(forURL: "gs://empleos-juarez-maquila

imageURL3.downloadURL(completion: { (url, error) in
    if error != nil {
        print(error?.localizedDescription as Any)
        return
    }
    URLSession.shared.dataTask(with: url!, completionHandler: { (data, response,
        if error != nil {
            print(error as Any)
            return
        }
        guard let imageData = UIImage(data: data!) else { return }
        DispatchQueue.main.async {
            //                    cell.myButonView.setBackgroundImage(imageData,
            cell.imagenEmpresa.image = imageData
```

```
            }
        }).resume()
    })

    return cell
}
//funcion para agrandar las celdas de la tabla
func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGF
    return 180

}
//funcion que recorre todos los documentos en vacantes y lista todas las vacantes
func traerVacantes(){

    let db = Firestore.firestore()
    db.collection("Vacantes")
                .getDocuments() { (querySnapshot, err) in
                    if let err = err {
                        print("Error getting documents: \(err)")
                    } else {
                        for document in querySnapshot!.documents {

                         let postulantesB = document.data()["Solicitudes"] as? Int ??
                         self.postulantesArray.append("Postulantes: \(postulantesB)")
                         let uidB = document.data()["UID"] as? String ?? ""
                         self.uidArray.append(uidB)
                         if  let detalles = document.data()["DatosEmpresa"] as? [Stri
```

```
                let calleB = detalles["Calle"] as? String ?? ""
                self.direccionArray.append(calleB)
                let nombreEmpresaB = detalles["Nombre"] as? String ?? ""
                self.empresasArray.append(nombreEmpresaB)
                self.idDocumentoArray.append(document.documentID)
                let fechaB = detalles["Fecha"] as? Date ?? Date()
//                  self.fechaArray.append(fechaB)

//                 funcion para comparar la fecha de publicacion con la fe
                let someDateComponentsFormatter = DateComponentsFormatte

                someDateComponentsFormatter.unitsStyle = .positional
                //        someDateComponentsFormatter.includesApproximat
                //        someDateComponentsFormatter.includesTimeRemain
                someDateComponentsFormatter.allowedUnits = [.day, .hour,

                let remainingTime = someDateComponentsFormatter.string(f
                self.fechaArray.append("Publicado Hace \(remainingTime)"
                //                                      cuentaAtras.text
            }
            if  let detalles = document.data()["Detalles"] as? [String:
                let vacantesB = detalles["Vacantes"] as? Int ?? 0
                self.vacantesArray.append("Vacantes: \(vacantesB)")
                let nombreVacanteB = detalles["Puesto"] as? String ?? ""
                self.nombreVacanteArray.append(nombreVacanteB)
```

```
                        }


                       }
                    //funcion que recarga los datos de la tabla una vez que se obtuv
                       self.tableView?.reloadData();
                   }
            }
    }
    //funcion que se ejecuta solo una vez al iniciar la vista
    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.delegate = self
        tableView.dataSource = self
        traerVacantes()
    }

//funcion para mandar el id del documento a la siguiente vista
    override func prepare(for segue: UIStoryboardSegue, sender: Any!) {
            if (segue.identifier == "VerVacanteUsuarios") {
                let destino = segue.destination as! informacionVacanteViewController
                let blogIndex = tableView.indexPathForSelectedRow?.row

                destino.documentoId = idDocumentoArray[blogIndex!]
    //              destino.jornadaNombre = jornadaNombreArray[blogIndex!]
    //              destino.idUsuario1 = idUsuario
            }
```

```
        }
```

Todas las empresas

```
    //Funcion que rrecorre todos los documentos de empresas en la base de tados y las li
    func traerVacantes(){

           let db = Firestore.firestore()
           db.collection("Empresas")
                      .getDocuments() { (querySnapshot, err) in
                          if let err = err {
                              print("Error getting documents: \(err)")
                          } else {
                              for document in querySnapshot!.documents {
                                  //  print("\(document.documentID) => \(document.data()

                               if  let detalles = document.data()["DatosEmpresa"] as? [S

                                let nombreEmpresa = detalles["Nombre"] as? String ?? ""
                                let calleB = detalles["Calle"] as? String ?? ""
                                let telefono = detalles["Telefono"] as? String ?? ""

                                   self.empresasUIDArray.append(document.documentID)
                                   self.nombreEmpresaArray.append(nombreEmpresa)
                                self.calleArray.append(calleB)
                                self.telefonoArray.append(telefono)
                               }
```

```
                                }
//                              self.refreshControl.endRefreshing()
                                self.tableView?.reloadData();
                            }
                    }
        }
```

Información de la vacante

```
        func traerDatos (){

            let db = Firestore.firestore()
            print("documento ID Trae \(documentoId)")
            if(documentoId != ""){
                let docRef = db.collection("Vacantes").document(documentoId)

                docRef.getDocument { (document, error) in
                    if let document = document {

                        let myData = document.data()
                        let uid = myData["UID"] as? String ?? ""
                        let solicitudesB = myData["Solicitudes"] as? Int ?? 0
                        self.numeroSolicitudes = solicitudesB

                            let imageURL3 = Storage.storage().reference(forURL: "gs://em
```

```
                imageURL3.downloadURL(completion: { (url, err
                    if error != nil {
                        print(error?.localizedDescription as
                        return
                    }
                    URLSession.shared.dataTask(with: url!, co
                        if error != nil {
                            print(error as Any)
                            return
                        }
                        guard let imageData = UIImage(data: d
                        DispatchQueue.main.async {
                            //                    cell.myButo
                             self.empresaImage.image = imageData


                        }
                    }).resume()
                })
            if  let variable = myData["SolicitudesMapa"] as? [String: Any]{

                self.solicitudesDiccionario = variable
                print("lo que trae variable\(variable)")
                print("lo que trae diccionario\(self.solicitudesDiccionario)
                self.revisarSolicitud()
            }
```

```
if  let detalles = myData["Detalles"] as? [String: Any]{
    let descripcionB = detalles["Descripcion"] as? String ?? ""
    self.descripcionVacante.text = descripcionB
    let documentosDescrpcionB = detalles["DocumentosDescripcion"
    self.documentosDescripcion.text = documentosDescrpcionB

    let generoB = detalles["Genero"] as? String ?? ""
    if generoB == "Hombre"{
        self.hombreCheck.isChecked = true
    }else if generoB == "Mujer"{
        self.mujerCheck.isChecked = true
    }else{
        self.ambosCheck.isChecked = true
    }
    let puesto = detalles["Puesto"] as? String ?? ""
    self.puestoNombre.text = puesto
    let salarioInicialB = detalles["SalarioInicial"] as? Int ??
    self.salarioInicial.text = "\(salarioInicialB)"
    let salarioMaximoB = detalles["SalarioMaximo"] as? Int ?? 0
    self.salarioMaximo.text = "\(salarioMaximoB)"
    let vacantesB = detalles["Vacantes"] as? Int ?? 0
    self.vacantesDisponibles.text = "\(vacantesB)"
}

if  let detalles = myData["DatosEmpresa"] as? [String: Any]{
    let calleB = detalles["Calle"] as? String ?? ""
    self.direccionEmpresa.text = calleB
```

```
                let fabricacionB = detalles["Fabricacion"] as? String ?? ""
                self.dedicadaEmpresa.text = fabricacionB
                let telefonoB = detalles["Telefono"] as? String ?? ""
                self.telefonoEmpresa.text = telefonoB
                let nombreB = detalles["Nombre"] as? String ?? ""
                self.nombreEmpresa.text = nombreB

            }

        if  let variable = myData["Aptitudes"] as? [String: Any]{

            let aptitudB1 = variable["Aptitud1"] as? String ?? ""
            if aptitudB1 == ""{
                self.aptitud1.isHidden = true
            }else{
                self.aptitud1.text = aptitudB1
            }
            let aptitudB2 = variable["Aptitud2"] as? String ?? ""
            if aptitudB2 == ""{
                self.aptitud2.isHidden = true
            }else{
                self.aptitud2.text = aptitudB2
            }
            let aptitudB3 = variable["Aptitud3"] as? String ?? ""
            if aptitudB3 == ""{
                self.aptitud3.isHidden = true
            }else{
```

```
                self.aptitud3.text = aptitudB3
            }
            let aptitudB4 = variable["Aptitud4"] as? String ?? ""
            if aptitudB4 == ""{
                self.aptitud4.isHidden = true
            }else{
                self.aptitud4.text = aptitudB4
            }
            let aptitudB5 = variable["Aptitud5"] as? String ?? ""
            if aptitudB5 == ""{
                self.aptitud5.isHidden = true
            }else{
                self.aptitud5.text = aptitudB5
            }
            let aptitudB6 = variable["Aptitud6"] as? String ?? ""
            if aptitudB6 == ""{
                self.aptitud6.isHidden = true
            }else{
                self.aptitud6.text = aptitudB6
            }
        }

        if  let variable = myData["Beneficios"] as? [String: Any]{

            let cafeteria = variable["Cafeteria"] as? Bool ?? false
            if cafeteria == true{
                self.cafeteria.text = "Si"
```

```
}else{
    self.cafeteria.text = "No"
}
let fondoAhorroB = variable["Fondo de ahorro"] as? Bool ?? false
if fondoAhorroB == true{
    self.fondoAhorro.text = "Si"
}else{
    self.fondoAhorro.text = "No"
}
let tiempoExtraB = variable["Tiempo Extra"] as? Bool ?? false
if tiempoExtraB == true{
    self.tiempoExtra.text = "Si"
}else{
    self.tiempoExtra.text = "No"
}
let transporteB = variable["Transporte"] as? Bool ?? false
if transporteB == true{
    self.transporte.text = "Si"
}else{
    self.transporte.text = "No"
}
let valesDespensaB = variable["Vales de despensa"] as? Bool ?? f
if valesDespensaB == true{
    self.valesDespensa.text = "Si"
}else{
    self.valesDespensa.text = "No"
}
```

```
            }

        if  let variable = myData["Documentacion"] as? [String: Any]{
            let actaNacimientoB = variable["Acta de Nacimiento"] as? Bool ?? false
            if actaNacimientoB == false{
                self.actaNacimiento.isHidden = true
            }
            let curpB = variable["CURP"] as? Bool ?? false
            if curpB == false{
                self.curp.isHidden = true
            }
            let cartaRecomendacionB = variable["Carta de recomendación"] as? Bool ?? fal
            if cartaRecomendacionB == false{
                self.cartaRecomendacion.isHidden = true
            }
            let cartaRetencionB = variable["Carta de retención de Infonavit"] as? Bool ?
            if cartaRetencionB == false{
                self.cartaRetencionInfonavit.isHidden = true
            }
            let comprobanteDomicilioB = variable["Comprobante de domicilio"] as? Bool ??
            if comprobanteDomicilioB == false{
                self.comprobanteDomicilio.isHidden = true
            }
            let comprobanteEstudiosB = variable["Comprobante de estudios"] as? Bool ?? f
            if comprobanteEstudiosB == false{
                self.comprobanteEstudios.isHidden = true
            }
```

```
            let identificacionOficialB = variable["Identificación oficial"] as? Bool ??
            if identificacionOficialB == false{
                self.identificacionOficial.isHidden = true
            }
            let nssB = variable["NSS (Emitida por el imss)"] as? Bool ?? false
            if nssB == false{
                self.nss.isHidden = true
            }
            let noPenalesB = variable["No antecedentes penales"] as? Bool ?? false
            if noPenalesB == false{
                self.cartaNoAntecedentes.isHidden = true
            }
            let rfcB = variable["RFC"] as? Bool ?? false
            if rfcB == false{
                self.rfc.isHidden = true
            }
            let solicitudLlenaB = variable["Solicitud Llenada"] as? Bool ?? false
            if solicitudLlenaB == false{
                self.solicitudLlena.isHidden = true
            }
        }
        if  let variable = myData["Turnos"] as? [String: Any]{
            if  let turno = variable["Turno1"] as? [String: Any]{
            if  let lunes = turno["Lunes"] as? [String: Any]{
                let laboral = lunes["Laboral"] as? Bool ?? false
                if laboral == true{
                    self.primerTurnoCheck.isChecked = true
```

```
            }else{
                self.primerTurnoStack.isHidden = true
            }
        }
    }
    if  let turno = variable["Turno2"] as? [String: Any]{
        if  let lunes = turno["Lunes"] as? [String: Any]{
            let laboral = lunes["Laboral"] as? Bool ?? false
            if laboral == true{
                self.segundoTurnoCheck.isChecked = true
            }else{
                self.segundoTurnoStack.isHidden = true
            }
        }
    }
    if  let turno = variable["Turno3"] as? [String: Any]{
        if  let lunes = turno["Lunes"] as? [String: Any]{
            let laboral = lunes["Laboral"] as? Bool ?? false
            if laboral == true{
                self.tercerTurnoCheck.isChecked = true
            }else{
                self.tercerTurnoStack.isHidden = true
            }
        }
    }
    if  let turno = variable["Especial1"] as? [String: Any]{
        if  let dia = turno["Lunes"] as? [String: Any]{
```

```
            let laboral = dia["Laboral"] as? Bool ?? false
            if laboral == true{
                self.turnoEspecialCheck1.isChecked = true
                self.turnoEspecialStack1.isHidden = false
                self.lunesespecial1.backgroundColor = #colorLiteral(red: 0.36057
                let entradaB = dia["Entrada"] as? String ?? ""
                let salidaB = dia["Salida"] as? String ?? ""
                self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
            }
        }
        if  let dia = turno["Martes"] as? [String: Any]{
            let laboral = dia["Laboral"] as? Bool ?? false
            if laboral == true{
                self.turnoEspecialCheck1.isChecked = true
                self.turnoEspecialStack1.isHidden = false
                self.martespecial1.backgroundColor = #colorLiteral(red: 0.360575
                let entradaB = dia["Entrada"] as? String ?? ""
                let salidaB = dia["Salida"] as? String ?? ""
                self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
            }
        }
        if  let dia = turno["Miercoles"] as? [String: Any]{
            let laboral = dia["Laboral"] as? Bool ?? false
            if laboral == true{
                self.turnoEspecialCheck1.isChecked = true
                self.turnoEspecialStack1.isHidden = false
                self.miercolespecial1.backgroundColor = #colorLiteral(red: 0.360
```

```
            let entradaB = dia["Entrada"] as? String ?? ""
            let salidaB = dia["Salida"] as? String ?? ""
            self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
        }
    }
    if  let dia = turno["Jueves"] as? [String: Any]{
        let laboral = dia["Laboral"] as? Bool ?? false
        if laboral == true{
            self.turnoEspecialCheck1.isChecked = true
            self.turnoEspecialStack1.isHidden = false
            self.juevespecial1.backgroundColor = #colorLiteral(red: 0.360575
            let entradaB = dia["Entrada"] as? String ?? ""
            let salidaB = dia["Salida"] as? String ?? ""
            self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
        }
    }
    if  let dia = turno["Viernes"] as? [String: Any]{
        let laboral = dia["Laboral"] as? Bool ?? false
        if laboral == true{
            self.turnoEspecialCheck1.isChecked = true
            self.turnoEspecialStack1.isHidden = false
            self.viernespecial1.backgroundColor = #colorLiteral(red: 0.36057
            let entradaB = dia["Entrada"] as? String ?? ""
            let salidaB = dia["Salida"] as? String ?? ""
            self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
        }
    }
```

```
            if  let dia = turno["Sabado"] as? [String: Any]{
                let laboral = dia["Laboral"] as? Bool ?? false
                if laboral == true{
                    self.turnoEspecialCheck1.isChecked = true
                    self.turnoEspecialStack1.isHidden = false
                    self.sabadopecial1.backgroundColor = #colorLiteral(red: 0.360575
                    let entradaB = dia["Entrada"] as? String ?? ""
                    let salidaB = dia["Salida"] as? String ?? ""
                    self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
                }
            }
            if  let dia = turno["Domingo"] as? [String: Any]{
                let laboral = dia["Laboral"] as? Bool ?? false
                if laboral == true{
                    self.turnoEspecialCheck1.isChecked = true
                    self.turnoEspecialStack1.isHidden = false
                    self.domingoppecial1.backgroundColor = #colorLiteral(red: 0.3605
                    let entradaB = dia["Entrada"] as? String ?? ""
                    let salidaB = dia["Salida"] as? String ?? ""
                    self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
                }
            }
        }
        if  let turno = variable["Especial2"] as? [String: Any]{
        if  let dia = turno["Lunes"] as? [String: Any]{
            let laboral = dia["Laboral"] as? Bool ?? false
            if laboral == true{
```

```
                self.turnoEspecialCheck2.isChecked = true
                self.turnoEspecialStack2.isHidden = false
                self.lunesespecial2.backgroundColor = #colorLiteral(red: 0.360575407
                let entradaB = dia["Entrada"] as? String ?? ""
                let salidaB = dia["Salida"] as? String ?? ""
                self.horarioEspecial2.text = "\(entradaB) a \(salidaB)"
            }
        }
        if  let dia = turno["Martes"] as? [String: Any]{
            let laboral = dia["Laboral"] as? Bool ?? false
            if laboral == true{
                self.turnoEspecialCheck2.isChecked = true
                self.turnoEspecialStack2.isHidden = false
                self.martespecial2.backgroundColor = #colorLiteral(red: 0.3605754077
                let entradaB = dia["Entrada"] as? String ?? ""
                let salidaB = dia["Salida"] as? String ?? ""
                self.horarioEspecial2.text = "\(entradaB) a \(salidaB)"
            }
        }
        if  let dia = turno["Miercoles"] as? [String: Any]{
            let laboral = dia["Laboral"] as? Bool ?? false
            if laboral == true{
                self.turnoEspecialCheck2.isChecked = true
                self.turnoEspecialStack2.isHidden = false
                self.miercolespecial2.backgroundColor = #colorLiteral(red: 0.3605754
                let entradaB = dia["Entrada"] as? String ?? ""
                let salidaB = dia["Salida"] as? String ?? ""
```

```
            self.horarioEspecial2.text = "\(entradaB) a \(salidaB)"
        }
    }
    if  let dia = turno["Jueves"] as? [String: Any]{
        let laboral = dia["Laboral"] as? Bool ?? false
        if laboral == true{
            self.turnoEspecialCheck2.isChecked = true
            self.turnoEspecialStack2.isHidden = false
            self.juevespecial2.backgroundColor = #colorLiteral(red: 0.3605754077
            let entradaB = dia["Entrada"] as? String ?? ""
            let salidaB = dia["Salida"] as? String ?? ""
            self.horarioEspecial2.text = "\(entradaB) a \(salidaB)"
        }
    }
    if  let dia = turno["Viernes"] as? [String: Any]{
        let laboral = dia["Laboral"] as? Bool ?? false
        if laboral == true{
            self.turnoEspecialCheck2.isChecked = true
            self.turnoEspecialStack2.isHidden = false
            self.viernespecial2.backgroundColor = #colorLiteral(red: 0.360575407
            let entradaB = dia["Entrada"] as? String ?? ""
            let salidaB = dia["Salida"] as? String ?? ""
            self.horarioEspecial2.text = "\(entradaB) a \(salidaB)"
        }
    }
    if  let dia = turno["Sabado"] as? [String: Any]{
        let laboral = dia["Laboral"] as? Bool ?? false
```

```
                if laboral == true{
                    self.turnoEspecialCheck2.isChecked = true
                    self.turnoEspecialStack2.isHidden = false
                    self.sabadopecial2.backgroundColor = #colorLiteral(red: 0.3605754077
                    let entradaB = dia["Entrada"] as? String ?? ""
                    let salidaB = dia["Salida"] as? String ?? ""
                    self.horarioEspecial2.text = "\(entradaB) a \(salidaB)"
                }
            }
            if  let dia = turno["Domingo"] as? [String: Any]{
                let laboral = dia["Laboral"] as? Bool ?? false
                if laboral == true{
                    self.turnoEspecialCheck1.isChecked = true
                    self.turnoEspecialStack1.isHidden = false
                    self.domingoppecial1.backgroundColor = #colorLiteral(red: 0.36057540
                    let entradaB = dia["Entrada"] as? String ?? ""
                    let salidaB = dia["Salida"] as? String ?? ""
                    self.horarioEspecial1.text = "\(entradaB) a \(salidaB)"
                }
            }
        }

        }

        }
        }
    }
```

```
}

func revisarSolicitud(){
    if solicitudesDiccionario.count > 0 {
        let userID = Auth.auth().currentUser!.uid
        for i in 0 ... solicitudesDiccionario.count-1{

            if Array(solicitudesDiccionario)[i].key == userID{
                solicitudEnviada = true
            }
        }
    }
}


@IBAction func enviarSolicitudAction(_ sender: Any) {

    if solicitudEnviada == false{
        solicitudEnviada = true
        numeroSolicitudes += 1
          let userID = Auth.auth().currentUser!.uid
          solicitudesDiccionario[userID] = userID
          print("lo que trae diccionario despues de agregar \(solicitudesDiccionario
                  let db = Firestore.firestore()
                 let washingtonRef2 = db.collection("Vacantes").document(documentoId
                 washingtonRef2.updateData([
                  "Solicitudes": numeroSolicitudes,
```

```
                    "SolicitudesMapa":self.solicitudesDiccionario,
                ]) { err in
                    if let err = err {
                        print("Error updating document: \(err)")
                    } else {
                        print("Document successfully updated")
                        let alertController = UIAlertController(title: "Solicitud",
                            alertController.addAction(UIAlertAction(title: "Ok", sty
                                print("Handle Ok logic here")
                        }))
                        self.present(alertController, animated: true, completion: ni
                    }
                }
        }else{
            let alertController = UIAlertController(title: "Solicitud", message: "Ya has
                alertController.addAction(UIAlertAction(title: "Ok", style: .default, ha
                    print("Handle Ok logic here")
            }))
            self.present(alertController, animated: true, completion: nil)
        }
    }
```

Perfil de usuario

```
    func traerDatos(){
        let docRef = db.collection("Usuarios").document("\(userID)")
        docRef.getDocument { (document, error) in
```

```
if let document = document {
    let myData = document.data()
    if  let perfil = myData["Perfil"] as? [String: Any]{
        let imageURL3 = Storage.storage().reference(forURL: "gs://empleos-ju

        imageURL3.downloadURL(completion: { (url, error) in
            if error != nil {
                print(error?.localizedDescription as Any)
                return
            }
            URLSession.shared.dataTask(with: url!, completionHandler: { (dat
                if error != nil {
                    print(error as Any)
                    return
                }
                guard let imageData = UIImage(data: data!) else { return }
                DispatchQueue.main.async {
                    //                    cell.myButonView.setBackgroundImag
                    self.fotoUsuario.setBackgroundImage(imageData, for: .nor


                }
            }).resume()
        })
        let nombre = perfil["Nombre"] as? String ?? ""
        let apellido = perfil["Apellido"] as? String ?? ""
        self.nombre.text = "\(nombre) \(apellido)"
        let colonia = perfil["Colonia"] as? String ?? ""
```

```
if colonia != ""{
  self.colonia.text = colonia
}
let estudiasactualmente = perfil["Estudias Actualmente"] as? Bool
if estudiasactualmente == true{
    self.ternimoDEstudiar.text = "Estudia Actualmente: Si"
}else{
    self.ternimoDEstudiar.text = "Estudia Actualmente: No"
}
let correo = perfil["Correo"] as? String ?? ""
if correo != ""{
    self.correo.text = correo
}
let calle = perfil["Calle"] as? String ?? ""
if calle != ""{
    self.calle.text = calle
}
let idioma1 = perfil["Idioma1"] as? String ?? ""
if calle != ""{
    self.idiomaDerech1.text = " \(idioma1)"
    self.idioma1.text = "Idioma Extra:"
}
let idioma2 = perfil["Idioma2"] as? String ?? ""
if calle != ""{
    self.idiomaDerech2.text = " \(idioma2)"
    self.idioma2.text = "Idioma Extra:"
}
```

```
                }
                if  let estudios = myData["Estudios"] as? [String: Any]{
                 if let primaria = estudios["Primaria"] as? [String: Any]{
//                    self.ultimoSemestre.isHidden = true
//                    self.ternimoDEstudiar.isHidden = true
                      let nombre = primaria["Nombre"] as? String ?? ""
                   if nombre != ""{
                       self.nombreEscuela.text = nombre
                   }
                   let certificado = primaria["Certificado"] as? Bool
                   if certificado == true{
                       self.carrera.text = "Cuento con el certificado de primaria: Si"
                   }else{
                       self.carrera.text = "Cuento con el certificado de primaria: No"
                   }}
                    if let secundaria = estudios["Secundaria"] as? [String: Any]{
//                      self.ultimoSemestre.isHidden = true
//                      self.ternimoDEstudiar.isHidden = true
                      let nombre = secundaria["Nombre"] as? String ?? ""
                      if nombre != ""{
                          self.nombreEscuela.text = nombre
                      }
                      let certificado = secundaria["Certificado"] as? Bool
                      if certificado == true{
                          self.carrera.text = "Cuento con el certificado de Secundaria
                      }else{
                          self.carrera.text = "Cuento con el certificado de Secundaria
```

```
        }}
    if let prepa = estudios["Preparatoria"] as? [String: Any]{
        self.ultimoSemestre.isHidden = false
        self.ternimoDEstudiar.isHidden = false
        let nombre = prepa["Nombre"] as? String ?? ""
        if nombre != ""{
            self.nombreEscuela.text = nombre
        }
        let certificado = prepa["Certificado"] as? Bool
        if certificado == true{
            self.carrera.text = "Cuento con el certificado de Preparator
            let certificadoTecnico = prepa["Certificado Tecnico"] as? Bo
            if certificadoTecnico == true{
                let certificadoNombre = prepa["Certificado Nombre"] as?
                self.ultimoSemestre.text = "Certificado Tecnico: Si, \(c
            }else{
                self.ultimoSemestre.text = "Certificado Tecnico: No"
            }

        }else{
            self.carrera.text = "Cuento con el certificado de Preparator
            let ultimosemestre = prepa["Semestre Actual"] as? String ??
            self.ultimoSemestre.text = "Semestre: \(ultimosemestre)"
        }}
    if let universida = estudios["Universidad"] as? [String: Any]{
        self.ultimoSemestre.isHidden = false
        self.ternimoDEstudiar.isHidden = false
```

```
                let nombre = universida["Nombre"] as? String ?? ""
                if nombre != ""{
                    self.nombreEscuela.text = nombre
                }
                let carrera = universida["Carrera"] as? String ?? ""
                if carrera != ""{
                    self.carrera.text = carrera
                }
                let certificado = universida["Facultad Terminada"] as? Bool
                if certificado == true{

                    let certificadoTecnico = universida["Especialidad"] as? Bool
                    if certificadoTecnico == true{
                        let certificadoNombre = universida["Especialidad Nombre"
                        self.ultimoSemestre.text = "Especialidad: Si, \(certific
                    }else{
                        self.ultimoSemestre.text = "Especialidad: No"
                    }
                }else{

                    let ultimosemestre = universida["Semestre Actual"] as? Strin
                    self.ultimoSemestre.text = "Semestre: \(ultimosemestre)"
                }}

        }
        if  let experienciaLaboral = myData["Experiencia Laboral"] as? [String:
            if let Empleo1 = experienciaLaboral["Empleo 1"] as? [String: Any]{
```

```
                        let nombre = Empleo1["Nombre"] as? String ?? ""
                        if nombre != ""{
                           self.trabajo1.text = nombre
                        }
                        let telefono = Empleo1["Telefono"] as? String ?? ""
                        if telefono != ""{
//                            self.telefonoEmpresa[0] = telefono
                        }}
                    if let Empleo2 = experienciaLaboral["Empleo 2"] as? [String: Any]{
                        let nombre = Empleo2["Nombre"] as? String ?? ""
                        if nombre != ""{
                            self.trabajo2.text = nombre
                        }
                        let telefono = Empleo2["Telefono"] as? String ?? ""
                        if telefono != ""{
                            //                              self.telefonoEmpresa[0] = tele
                        }}
                    if let Empleo3 = experienciaLaboral["Empleo 3"] as? [String: Any]{
                        let nombre = Empleo3["Nombre"] as? String ?? ""
                        if nombre != ""{
                            self.trabajo3.text = nombre
                        }
                        let telefono = Empleo3["Telefono"] as? String ?? ""
                        if telefono != ""{
                            //                              self.telefonoEmpresa[0] = tele
                        }}
```

```
            }
        }
    }
}

//accion que permite a los usuarios subir su foto a la base de datos
@IBAction func subirFoto(_ sender: Any) {

    imagePicker.delegate = self
    imagePicker.sourceType = UIImagePickerController.SourceType.photoLibrary

    self.present(imagePicker, animated: true, completion: nil)

}
```

Solicitudes

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int

    return (empresasArray.count)

}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITab

    let cell = tableView.dequeueReusableCell(withIdentifier: "customCell", for: inde
```

```
        cell.nombreEmpresa.text = empresasArray[indexPath.row]

        return cell
    }
    func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGF
        return 180

    }

    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.delegate = self
        tableView.dataSource = self
    }
```

Ayuda

```
     override func viewDidLoad() {
        super.viewDidLoad()

        ayudaLabel.layer.cornerRadius = 10
        ayudaLabel.layer.borderWidth = 2
        ayudaLabel.layer.masksToBounds = true
        ayudaLabel.layer.borderColor = #colorLiteral(red: 0.05911582708, green: 0.706059
        avisoPrivacidad.layer.borderWidth = 3.0
        avisoPrivacidad.layer.cornerRadius = 8
        avisoPrivacidad.backgroundColor = #colorLiteral(red: 0.3605754077, green: 0.8106
```

```
        avisoPrivacidad.layer.borderColor = #colorLiteral(red: 0.390620172, green: 0.879
        avisoPrivacidad.layer.masksToBounds = true

        continuarButton.layer.borderWidth = 3.0
        continuarButton.layer.cornerRadius = 8
        continuarButton.backgroundColor = #colorLiteral(red: 0.8768144846, green: 0.1553
        continuarButton.layer.borderColor = #colorLiteral(red: 1, green: 0.1491314173, b
        continuarButton.layer.masksToBounds = true
        // Do any additional setup after loading the view.
    }

        @IBAction func cerrarSesion(_ sender: Any) {
            if Auth.auth().currentUser != nil {
                do {
                    try Auth.auth().signOut()
                    let vc = UIStoryboard(name: "Main", bundle: nil).instantiateViewCont
                    present(vc, animated: true, completion: nil)
                } catch let error as NSError {
                    print(error.localizedDescription)
                }
            }
        }
```

6.0.4. Código Interfaz de empresas

Registrar empresa

```
@IBAction func registrarEmpresaAction(_ sender: Any) {

   if nombreEmpresa.text == "" || calleNumero.text == "" || coloniaFraccionamiento.
       let alertController = UIAlertController(title: "Error", message: "Por favor
                           let defaultAction = UIAlertAction(title: "Continuar", sty
                           alertController.addAction(defaultAction)
                           present(alertController, animated: true, completion: nil)
   }else{
       if deDondeVengo == 1{
          let userID = Auth.auth().currentUser!.uid
               print("Te has registrado correctamente")

               db.collection("Empresas").document(userID).setData([
                "DatosEmpresa":([
                   "Nombre": self.nombreEmpresa.text!,
                   "NombreReclutador": self.nombreReclutador.text!,
                   "Calle": self.calleNumero.text!,
                   "Colonia": self.coloniaFraccionamiento.text!,
                   "Fabricacion": self.seDedica.text!,
                   "TelefonoReclutador": self.telefono.text!,
                   "Telefono": self.telefonoEmpresa.text!,
                   "Correo": self.correo.text!,
                   "Verificado": false,
                   "UID": "\(userID)"
```

```
                ])
            ]) { err in
                if let err = err {
                    print("Error writing document: \(err)")
                    let alertController = UIAlertController(title: "Error", mess
                    let defaultAction = UIAlertAction(title: "Ok", style: .cance
                    alertController.addAction(defaultAction)
                    self.present(alertController, animated: true, completion: ni
                } else {
                    print("Document successfully written!")
                     let alertController = UIAlertController(title: "Exito", mes
                alertController.addAction(UIAlertAction(title: "Ok", style: .def
                    print("Handle Ok logic here")
                    self.navigationController?.popViewController(animated: true)
                    }))
                    self.present(alertController, animated: true, completion: ni
                }
            }


        }else{
            if contra.text == ""{
                let alertController = UIAlertController(title: "Error", message: "Po
                                    let defaultAction = UIAlertAction(title: "Ok", st
                                    alertController.addAction(defaultAction)
                                    present(alertController, animated: true, completi
```

```
}else{
    Auth.auth().createUser(withEmail: correo.text!, password: contra.tex
           if error == nil{
               print("Te has registrado correctamente")
               let userID = Auth.auth().currentUser!.uid
               let db = Firestore.firestore()
               db.collection("Empresas").document(userID).setData([
                "DatosEmpresa":([
                   "Nombre": self.nombreEmpresa.text!,
                   "NombreReclutador": self.nombreReclutador.text!,
                   "Calle": self.calleNumero.text!,
                   "Colonia": self.coloniaFraccionamiento.text!,
                   "Fabricacion": self.seDedica.text!,
                   "TelefonoReclutador": self.telefono.text!,
                   "Telefono": self.telefonoEmpresa.text!,
                   "Correo": self.correo.text!,
                   "UID": "\(userID)"
                   ])
               ]) { err in
                   if let err = err {
                       print("Error writing document: \(err)")
                   } else {
                       print("Document successfully written!")
                       print("UID \(userID)")
                       let vc = self.storyboard?.instantiateViewControll
                       self.present(vc!, animated: true, completion: nil
                   }
```

```
                            }
                        }else {
                            let alertController = UIAlertController(title: "Error", m
                            let defaultAction = UIAlertAction(title: "OK", style: .ca
                            alertController.addAction(defaultAction)
                            self.present(alertController, animated: true, completion:
                        }
                    }
                }
            }// segunfo else
        }//primer else
    }
```

Iniciar sesión empresa

```
//funcion para iniciar sesion
    @IBAction func iniciarSesionAction(_ sender: Any) {
        if self.contra.text == "" || self.contra.text == "" {
            let alertController = UIAlertController(title: "Error", message: "Por favor

            let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)
            alertController.addAction(defaultAction)
            self.present(alertController, animated: true, completion: nil)
        } else {

            Auth.auth().signIn(withEmail: self.correo.text!, password: self.contra.text!
                if error == nil {
```

```
                print("iniciaste sesion correctamente")

                let vc = self.storyboard?.instantiateViewController(withIdentifier:
                self.present(vc!, animated: true, completion: nil)

            } else {
                let alertController = UIAlertController(title: "Error", message: err

                let defaultAction = UIAlertAction(title: "OK", style: .cancel, handl
                alertController.addAction(defaultAction)

                self.present(alertController, animated: true, completion: nil)
            }
        }
    }
}
```

Crear vacante

```
func traerDatos(){
    let db = Firestore.firestore()
            let docRef = db.collection("Empresas").document(userID)

            docRef.getDocument { (document, error) in
                if let document = document {
```

```
                    let myData = document.data()

                    if  let datosEmpresa = myData["DatosEmpresa"] as? [String: Any]{
                        let calle = datosEmpresa["Calle"] as? String ?? ""
                        self.calleEmpresa = calle
                        let fabricacionB = datosEmpresa["Fabricacion"] as? String ??
                        self.fabircacion = fabricacionB
                        let nombreB = datosEmpresa["Nombre"] as? String ?? ""
                        self.nombreEmpresa = nombreB
                        let telefonoB = datosEmpresa["Telefono"] as? String ?? ""
                        self.telefonoEmpresa = telefonoB
                        self.ponerEnBsaseDeDatos()
                    }
                } else {
                    print("Document does not exist")
                }
            }


}


func ponerEnBsaseDeDatos(){
    self.publicarVacante.isEnabled = false
   let db = Firestore.firestore()
            var ref: DocumentReference? = nil
            ref = db.collection("Vacantes").addDocument(data: [
```

```
"EmpVacantes": numeroVacantesFirebase,
"UID": userID,
"Activa": true,
"Usuario": "",
        "Aptitudes":([
            "Aptitud1": aptitudText1.text!,
            "Aptitud2": aptitudText2.text!,
            "Aptitud3": aptitudText3.text!,
            "Aptitud4": aptitudText4.text!,
            "Aptitud5": aptitudText5.text!,
            ]),
        "Documentacion":([
            "Acta de Nacimiento": actaNacimientoBool,
            "CURP": curpBool,
            "Carta de recomendación": cartaRecomendacionBool,
            "Carta de retención de Infonavit": cartaRetencionInfonavitB
            "Comprobante de domicilio": comprobanteDomicilioBool,
            "Comprobante de estudios": compobanteEstudiosBool,
            "Identificación oficial": identificacionOficialBool,
            "NSS (Emitida por el imss)": nssBool,
            "No antecedentes penales": cartaNoAntecedentesBool,
            "RFC": rfcBool,
            "Solicitud Llenada": solicitudLlenaBool,
            ]),
        "Detalles":([
           "Descripcion": descripcionVacante.text!,
            "DocumentosDescripcion": comentariosDocumentacion.text!,
```

```
        "Genero": genero,
        "Puesto": puestoVacante.text!,
        "SalarioInicial": Int(salarioInicial.text!) ?? 0,
        "SalarioMaximo": Int(salarioTentativo.text!) ?? 0,
        "Vacantes": Int(numeroVacantes.text!) ?? 0,
        ]),
    "DatosEmpresa":([
       "Calle": self.calleEmpresa,
       "Nombre": self.nombreEmpresa,
       "Fabricacion": self.fabircacion,
       "Telefono": self.telefonoEmpresa,
       "Fecha": Date(),
    ]),
    "Beneficios":([
        "Tiempo Extra": tiempoExtraBool,
        "Cafeteria": cafeteriaBool,
        "Transporte": transporteBool,
        "Vales de despensa": valesDespensaBool,
        "Fondo de ahorro": fondoAhorroBool,
        ]),
    "Turnos":([
        "Turno1":([
            "Lunes":([
                "Entrada": entradaTurnosArray[0],
                "Laboral": primerTurnoBool,
                "Salida": salidaTurnosArray[0],
                ]),
```

```
"Martes":([
    "Entrada": entradaTurnosArray[0],
    "Laboral": primerTurnoBool,
    "Salida": salidaTurnosArray[0],
    ]),
"Miercoles":([
    "Entrada": entradaTurnosArray[0],
    "Laboral": primerTurnoBool,
    "Salida": salidaTurnosArray[0],
    ]),
"Jueves":([
    "Entrada": entradaTurnosArray[0],
    "Laboral": primerTurnoBool,
    "Salida": salidaTurnosArray[0],
    ]),
"Viernes":([
    "Entrada": entradaTurnosArray[0],
    "Laboral": primerTurnoBool,
    "Salida": salidaTurnosArray[0],
    ]),
"Sabado":([
    "Entrada": "",
    "Laboral": false,
    "Salida": "",
    ]),
"Domingo":([
    "Entrada": "",
```

```
                        "Laboral": false,
                        "Salida": "",
                        ]),
                    ]),//Turno 1
                ]),
        ]) { err in
            if let err = err {
                print("Error adding document: \(err)")
              self.publicarVacante.isEnabled = true
              let alertController = UIAlertController(title: "Ooops", message:
              alertController.addAction(UIAlertAction(title: "Ok", style: .def
                  print("Handle Ok logic here")

                                                }))
              self.present(alertController, animated: true, completion: nil)

            } else {
                print("Document added with ID: \(ref!.documentID)")
              self.publicarVacante.isEnabled = true
                                          print("Document successfully update
                                        let alertController = UIAlertControll
                                        alertController.addAction(UIAlertActi
                                            print("Handle Ok logic here")
                                            self.navigationController?.popVie
                                                                        }))
                                        self.present(alertController, animate
```

```
                }

    }}
```

Lista de vacantes creadas

```
func traerVacantes(){
    //funcion que revisa todos los documentos de vacantes y verifica si el id del us
    let db = Firestore.firestore()
    db.collection("Vacantes").whereField("UID", isEqualTo: userID)
                .getDocuments() { (querySnapshot, err) in
                    if let err = err {
                        print("Error getting documents: \(err)")
                    } else {
                        for document in querySnapshot!.documents {
                            //  print("\(document.documentID) => \(document.data())")
                         let solicitudesB = document.data()["Solicitudes"] as? Int ??

                         self.solicitudesArray.append("Solicitudes Recibidas: \(solic

                         let activaB = document.data()["Activa"] as? Bool ?? true

                         if activaB == true{
                             self.activaArray.append("Vacante Activa: Si")
                         }else{
```

```
                self.activaArray.append("Vacante Activa: No")
            }

            if  let detalles = document.data()["DatosEmpresa"] as? [Stri

                let fechaB = detalles["Fecha"] as? Date ?? Date()

                let someDateComponentsFormatter = DateComponentsFormatte

                someDateComponentsFormatter.unitsStyle = .positional
                //        someDateComponentsFormatter.includesApproximat
                //        someDateComponentsFormatter.includesTimeRemain
                someDateComponentsFormatter.allowedUnits = [.day, .hour,

                let remainingTime = someDateComponentsFormatter.string(f
                self.fechaArray.append("Publicado Hace \(remainingTime)"

            }

            if  let detalles = document.data()["Detalles"] as? [String:

                let nombreVacante = detalles["Puesto"] as! String

                self.idDocumentoArray.append(document.documentID)
                self.nombreVacanteArray.append(nombreVacante)
            }
```

```
                        }
                    self.refreshControl.endRefreshing()
                        self.tableView?.reloadData();
                    }
            }
    }

    //funcion para refrescar las vacantes
    @objc func refresh(_ sender: Any) {
         nombreVacanteArray = [String]()
         idDocumentoArray = [String]()
         fechaArray = [String]()
        activaArray = [String]()
        solicitudesArray = [String]()
        traerVacantes()
    }
```

Perfil empresa

```
    func traerDatos(){
        let docRef = db.collection("Empresas").document("\(userID)")
        docRef.getDocument { (document, error) in
            if let document = document {
                let myData = document.data()
                //                self.comida = myData["Publicidad"] as? String ?? ""

                let imageURL3 = Storage.storage().reference(forURL: "gs://empleos-juarez
```

```
imageURL3.downloadURL(completion: { (url, error) in
    if error != nil {
        print(error?.localizedDescription as Any)
        return
    }
    URLSession.shared.dataTask(with: url!, completionHandler: { (data, r
        if error != nil {
            print(error as Any)
            return
        }
        guard let imageData = UIImage(data: data!) else { return }
        DispatchQueue.main.async {
            //                    cell.myButonView.setBackgroundImage(im
            self.fotoUsuario.setBackgroundImage(imageData, for: .normal)

        }
    }).resume()
})
if  let datosEmpresa = myData["DatosEmpresa"] as? [String: Any]{
    let calle = datosEmpresa["Calle"] as? String ?? ""

    var string = NSMutableAttributedString(string: "Calle y numero: \(ca
    string.setColorForText("Calle y numero:", with: #colorLiteral(red: 0
    self.calleNumero.attributedText = string

    let colonia = datosEmpresa["Colonia"] as? String ?? ""
```

```
string = NSMutableAttributedString(string: "Col./Fraccionamiento: \(
string.setColorForText("Col./Fraccionamiento:", with: #colorLiteral(
self.coloniaFraccionamiento.attributedText = string

let correo = datosEmpresa["Correo"] as? String ?? ""
string = NSMutableAttributedString(string: "Correo: \(correo)")
string.setColorForText("Correo:", with: #colorLiteral(red: 0, green:
self.correo.attributedText = string

let fabricacion = datosEmpresa["Fabricacion"] as? String ?? ""
string = NSMutableAttributedString(string: "Dedicada a: \(fabricacio
string.setColorForText("Dedicada a:", with: #colorLiteral(red: 0, gr
self.seDedica.attributedText = string

let nombre = datosEmpresa["Nombre"] as? String ?? ""
string = NSMutableAttributedString(string: "Nombre de la empresa: \(
string.setColorForText("Nombre de la empresa:", with: #colorLiteral(
self.nombreEmpresa.attributedText = string


let nombreReclutador = datosEmpresa["NombreReclutador"] as? String ?
string = NSMutableAttributedString(string: "Nombre del reclutador: \
string.setColorForText("Nombre del reclutador:", with: #colorLiteral
self.nombreReclutador.attributedText = string

let telefono = datosEmpresa["Telefono"] as? String ?? ""
string = NSMutableAttributedString(string: "Telefono de la empresa:
```

```
                string.setColorForText("Telefono de la empresa:", with: #colorLitera
                self.telefonoEmpresa.attributedText = string

                let telefonoReclutador = datosEmpresa["TelefonoReclutador"] as? Stri
                string = NSMutableAttributedString(string: "Telefono: \(telefonoRecl
                string.setColorForText("Telefono:", with: #colorLiteral(red: 0, gree
                self.telefonoReclutador.attributedText = string

            }

        }

    }

}
```

Solicitudes general

```
    func traerVacantes(){

        let db = Firestore.firestore()
        db.collection("Vacantes").whereField("UID", isEqualTo: userID)
                  .getDocuments() { (querySnapshot, err) in
                      if let err = err {
                          print("Error getting documents: \(err)")
                      } else {
                          for document in querySnapshot!.documents {
```

```
let activa = document.data()["Activa"] as? Bool ?? true

if activa == true{

    //  print("\(document.documentID) => \(document.data
    let soli = document.data()["Solicitudes"] as? Int ??
    self.solicitudesNumeroArray.append(soli)

    if  let detalles = document.data()["DatosEmpresa"] a

        let fecha = detalles["Fecha"] as? Date ?? Date()
        let someDateComponentsFormatter = DateComponents

        someDateComponentsFormatter.unitsStyle = .positi
        //        someDateComponentsFormatter.includesAp
        //        someDateComponentsFormatter.includesTi
        someDateComponentsFormatter.allowedUnits = [.day

        let remainingTime = someDateComponentsFormatter.
        self.fechaArray.append("Publicado Hace \(remaini


    }

    if  let detalles = document.data()["Detalles"] as? [
```

```
                                        let nombreVacante = detalles["Puesto"] as? Strin

                                        self.idDocumentoArray.append(document.documentID
                                        self.nombreVacanteArray.append(nombreVacante)
                                    }

                                 }

                                }
                             self.refreshControl.endRefreshing()
                                self.tableView?.reloadData();
                            }
                    }
        }

        @objc func refresh(_ sender: Any) {
             nombreVacanteArray = [String]()
             idDocumentoArray = [String]()
            traerVacantes()
        }
```

Solicitudes de cada vacante

```
//  funcion que trae las solicitu de la informacion de la vacante
        func traerVacantes(){

            let userID = Auth.auth().currentUser!.uid
```

```
        let db = Firestore.firestore()
                print("documento ID Trae \(userID)")
                if( userID != ""){
                    let docRef = db.collection("Vacantes").document(documentoId)

                    docRef.getDocument { (document, error) in
                        if let document = document {

                            let myData = document.data()
                            if  let detalles = myData["SolicitudesMapa"] as? [Strin
                               self.solicitudesDiccionario = detalles
                               self.traerInformacionSolicitud()
                          }

                        } else {
                            print("Document does not exist")
                        }
                    }
                }
    }
// funcion que trae la informacion de cada usuario para listarla
func traerInformacionSolicitud(){
    if solicitudesDiccionario.count > 0 {
        print("lo que trae el count del diccionario \(solicitudesDiccionario.count)"
        for i in 0 ... self.solicitudesDiccionario.count-1{
            let db = Firestore.firestore()
```

```
let docRef = db.collection("Usuarios").document("\(Array(solicitudesDicc

docRef.getDocument { (document, error) in
    if let document = document {

        let myData2 = document.data()

        if let myData = myData2["Perfil"] as? [String: Any]{

            let nombre = myData["Nombre"] as? String ?? ""
            let apellidoB = myData["Apellido"] as? String ?? ""
            let edadB = myData["Edad"] as? Int ?? 0
            let coloniaB = myData["Colonia"] as? String ?? ""
            let calleB = myData["Calle"] as? String ?? ""
            let correoB = myData["Correo"] as? String ?? ""
            let automovilB = myData["Apellido"] as? Bool ?? false
            self.nombreUsuarioArray.append(nombre)
            self.apellidoUsuarioArray.append(apellidoB)
            self.direccionArray.append("Direccion: \(calleB) Col. \(colo
            self.edadArray.append("Edad: \(edadB)")
            self.correoArray.append("Correo: \(correoB)")
            self.idDocumentoArray.append(Array(self.solicitudesDiccionar
            if automovilB == true{
                self.automovilArray.append("Automovil propio: Si")
            }else{
                self.automovilArray.append("Automovil propio: No")
                }
```

```
                    }
                }
                print("Lo que trae i es: \(i) y el uid es : \(Array(self.solicitudes

                    self.tableView?.reloadData();

            }

        }


    }

}
```