**Introducción**

Entendemos por trabajo a toda aquella actividad mediante el esfuerzo físico o mental que realiza el ser humano con el objetivo de ganar algún sustento a cambio que le permita mantenerse en su día a día. Independientemente de lo que se reciba a cambio, el trabajo denominado comúnmente como empleo aparece como una de las actividades más importantes por estar directamente relacionado con la subsistencia humana.

Los empleos vinculan a las personas con la sociedad y la economía en las que viven. El acceso a un empleo seguro, productivo y remunerado de manera justa puede ser en calidad de asalariado o por cuenta propia, es así que se tiene dos categorías diferentes y que se las clasifica como empleos formales y/o informales.

El empleo formal se caracteriza por ser regulado por diferentes organismos gubernamentales, como ser impuestos, ministerio de trabajo, entre otros.

El empleo informal se caracteriza por no estar regulado por el gobierno, haciendo que la(s) personas realicen alguna actividad u oficio para poder producir dinero para ellos mismos, indiferentemente de como esté la situación económica del país o de la oferta de empleos que existan.

El trabajo informal resulta ser popular por no tener la necesidad de contar con curriculum vitae escrito para trabajar, y que no se cuente siempre con una referencia física, por tanto esta categoría es flexible en cuestión de tiempo y dinero para la persona.

Para obtener y enterarse de requerimientos de trabajos, se usan diferentes herramientas, como ser periódico o teléfono, pero gracias a los avances tecnológicos, hay diferentes formas de poder obtener empleos mediante internet o herramientas especializadas en búsqueda de trabajo, pero estas están especializadas en búsqueda de trabajos formales.

Dado que no existen soluciones tecnológicas para la búsqueda de trabajo informal, se propone una aplicación móvil para la búsqueda y oferta de trabajos informales.

**Antecedentes**

**Antecedentes organizacionales**

En la economía global, el trabajo informal es el más realizado y está en constante crecimiento, en el caso de Bolivia, de igual forma, ya que según la OTI (Organización Internacional del Trabajo), indica que “entre el 60 y 70% de la población tiene un empleo informal, pero la situación es más dramática en el segmento juvenil, donde el porcentaje se eleva hasta 87,4%” (Luna Acevedo, 2014).

La situación actual del empleo en Bolivia por datos que fueron revelados por Bruno Rojas (experto en el tema del Centro de Estudios para el Desarrollo Laboral y Agrario- CEDLA) dice que “alrededor del 65% de la población urbana y ocupada del eje central se encuentran en actividades económicas familiares y semi empresariales” (Luna Acevedo, 2014), además de que “un 3% están ocupados por un sector doméstico (cocineras, jardineros) y otro 12% de la población se encuentran ocupados en el sector estatal de la economía. Y el restante 20% es acaparado apenas por el sector empresarial de la economía” (Luna Acevedo, 2014), considerando la información anterior, un 32% de la actividad económica pertenece al sector formal y un 68% al sector informal.

No existen parámetros estadísticos exactos que predigan el crecimiento de los empleos informales, ya que con el pasar de los años, va aumentando la cantidad de gente que trabaja de esta manera (ver anexo A), además, no se sabe con exactitud a qué tipos de empleos informales se dedican estas personas.

Al ser un mercado creciente, los gobiernos están empleando diferentes medidas para poder controlar y reducir el crecimiento de los empleos informales, ya que estos trabajadores no están ligados a los diferentes derechos y obligaciones que tiene los trabajos formales.

Describiendo algunos empleos informales, se tiene a los pintores, plomeros, jardineros, albañiles, vendedores ambulantes, negocios de ventas independientes, entre otros.

Los trabajadores informales ofertan sus servicios de las siguientes maneras:

* Publican referencia de sus servicios en diferentes periódicos, para que salgan en la sección de los clasificados, si el periódico tiene página web de clasificados, también se publican ahí.

Ejemplos: La Razón, Los Tiempos, El Deber, etc.

* Dejan referencia de ellos entre sus amigos y/o gente para la que trabajaron.
* Dejan volantes o carteles informativos.
* Esperan a que los clientes vayan a buscarlos a su puesto de trabajo o que se pongan en contactos con ellos.

Las personas demandan servicios de trabajadores informales de la siguiente forma:

* Buscan en la sección de clasificados del periódico.
* Publican sus requerimientos en periódicos o clasificados online de estos periódicos.
* Se pregunta a familiares, amigos o conocidos.

Existen diferentes inconvenientes a la hora de contratar trabajadores informales, como pudiendo menncionarse los siguientes:

* No siempre las personas que requieren servicios de trabajadores informales conocen al o los que le realizaran el trabajo, si son de confianza, puntuales, etc.
* No hay la obligación de hacer un contrato por el servicio.
* El cliente tiene que esperar la predisponibilidad del trabajador informal, ya que éste puede tener trabajos por hacer y no puede asistir al momento con el cliente.
* El trabajador informal no siempre conoce dónde es la dirección donde debe ir a realizar su trabajo, ya que pueden estar en calles poco conocidas o tener equivocada de referencias de donde vive el cliente.
* No siempre el cliente describe bien el problema que tiene porque no sabe la terminología correcta para describirlo.

**Antecedentes tecnológicos**

Con el auge de las tecnologías y el crecimiento del uso de internet, se implementaron diferentes soluciones tecnológicas para encontrar empleo.

En Bolivia, “A diciembre de 2014, una de cada dos personas en Bolivia tenía acceso al servicio de internet. Según datos oficiales, el 46,3% de la población boliviana (4.981.684) dispone de una conexión móvil o fija a la red de comunicación” (Quispe Aline, 2015), dando a entender que el acceso a internet es mayor, sobretodo en telefonía móvil, “las 4.981.684 conexiones de internet anotadas en el país, existen 4.801.498 (96,38%) usuarios que tienen conexiones móviles (tecnología 2G, 3G y 4G y terminales)” (Quispe Aline, 2015).

Las diferentes soluciones tecnológicas que se implementaron para la búsqueda de trabajo en Bolivia son:

* Clasificados online.- Son los clasificados similares que los periódicos ofrecen, estas páginas suben publicaciones cada vez que el cliente publica algo. Aunque estos servicios no siempre realizan control de las publicaciones antiguas.

Ejemplos.- http://www.anunico.com.bo/, http://www.evisos.com.bo/, https://clasificados.eldeber.com.bo/contrata-2, etc.

* Sistemas web.- Páginas web que hacen de intermediarios para poder conseguir empleos, diferentes empresas envían sus requerimientos al sistema y este envía a sus suscriptores las ofertas (dependiendo del área de interés del suscriptor), o el suscriptor busca los trabajos por diferentes criterios, sea área de trabajo, por ciudad, por cargo, por *tags*, etc.

Estos sistemas están especializados en trabajos formales, ya que te permiten realizar tu propio curriculum para esa web y casi la totalidad de los que piden trabajadores, son gente del área del empleo formal.

Ejemplos.- http://www.trabajopolis.bo/, http://bo.acciontrabajo.com/, https://bo.linkedin.com/, etc.

* Redes sociales.- Existen grupos donde postean diferentes ofertas de trabajo o de servicios, ya sea de trabajo formal o informal.

Estos anuncios solo muestran información de lo que hacen o requieren, pero no siempre dejan información completa del lugar físico donde trabajan, teléfono, requisitos, etc, generalmente, siempre dejan esa información para gente interesada que se contacte con ellos por mensajes privados, o comúnmente llamados “inbox”, aunque la respuesta no siempre es inmediata o no hay respuesta del que lo solicito.

Ejemplos.- www.facebook.com (TRABAJOS COCHABAMBA, Empleos, trabajos, servicios y más Cochabamba, etc).

**Problema**

**Situación problemática**

* Cuando un trabajador informal deja una referencia en periódicos no asegura que se vaya a obtener empleo en el futuro, dificultando la obtención de trabajo para los trabajadores informales.
* Cuando un trabajador informal deja una referencia entre sus familiares y amigos, no asegura que se vaya a obtener empleo en el futuro, dificultando la obtención de trabajo para los trabajadores informales.
* Cuando un trabajador informal deja la referencia de sus servicios con volantes, no asegura que se vaya a obtener empleo en el futuro, dificultando la obtención de trabajo para los trabajadores informales.
* El cliente no siempre conoce donde encontrar al trabajador informal de forma física, lo cual dificulta la obtención de los servicios que ofrecen los trabajadores informales.
* El cliente no siempre conoce a los trabajadores informales que está contratando, lo cual provoca que se obtenga un mal servicio o una mala resolución del problema.
* El cliente tiene que esperar la pre disponibilidad del trabajador informal con el que se contacta, generando el alargamiento de la necesidad o el problema del cliente.
* La falta de conocimiento o las malas referencias para llegar al lugar donde se tiene que hacer el trabajo provoca pérdida de tiempo y molestias con el cliente.
* Al no tener un contrato formal, no hay forma de asegurar que el trabajador cumpla con su trabajo en el tiempo acordado, horas de trabajo, horas en la que se trabajara o con la calidad acordada.
* La poca o mala referencia del problema que tiene que dar el cliente al trabajador informal, genera pérdida de tiempo y problemas para solucionar el problema.
* Los sistemas web para búsqueda de trabajos están especializados para trabajos formales, dejando excluidos a los trabajadores informales, limitando su área de búsqueda de trabajo.
* Las publicaciones de trabajo de grupos de redes sociales no siempre exponen toda la información requerida para entender el servicio que se oferta o se requiere, provocando pérdida de clientes.

**Formulación del problema**

Los limitantes métodos tradicionales y las pocas soluciones tecnológicas para la búsqueda de oferta y demanda de trabajadores informales provocan pérdida de dinero y tiempo a los trabajadores informales y a los clientes.

**Objetivos**

**Objetivo general**

Desarrollar una aplicación móvil para la gestión de oferta y demanda de trabajadores informales.

**Objetivos específicos**

* Desarrollar el módulo de solicitud del trabajador informal.
* Desarrollar el módulo de búsqueda de trabajos informales.
* Desarrollar el módulo de gestión de perfil, comentarios y puntuación de trabajadores informales.
* Implementar API de uso de mapas en la aplicación móvil.

**Alcances**

**Alcance del proyecto**

* Descripción de historias de usuario.
* Testeo y optimización de la aplicación.
* Implementación de tecnología de encriptación de datos.
* Desarrollo del código fuente.
* Desarrollo del servicio web que usara la aplicación.
* Video explicando cómo se usara el sistema.
* Lista de los servicio web creados.
* Desarrollo aplicación móvil en *Android*.
* Aplicación de código abierto

**Alcance del producto**

* Desarrollar módulos de publicación y búsqueda de trabajos informales.
* Desarrollar módulo de usuarios para el ingreso y salida del sistema.
* Desarrollar módulo de perfil de trabajadores informales.
* Diseño e implementación de algoritmo para búsqueda/solicitud de trabajos informales.
* Aplicación de *API* para uso de mapas.
* Implementación de metodologías de encriptación de datos.
* Diseño e implementación de servicios web.

**Límites**

* Al no tener recursos financieros para poder pagar un servidor web y licencia de funcionamiento para que el proyecto se publique en internet, este proyecto de grado no se lanzara al mercado.
* No se hará una investigación sobre el intervalo de que personas usan internet, se usaran datos de diferentes fuentes de información.
* Se desarrollará la aplicación móvil para el sistema operativo Android, ya que es de libre desarrollo y es el más usado actualmente
* La aplicación será nativa, ya que permitirá correr a la aplicación de forma más rápida y se evitara el problema de fallas en el uso de componentes del dispositivo móvil (el *GPS* como ejemplo).
* El servidor web no enviara correos electrónicos a los usuarios registrados.

**Justificación**

* La aplicación permitirá tener una nueva forma de obtener trabajos a los trabajadores informales, ya que estos no tendrán que esperar que los clientes los ubiquen y que ellos puedan buscar a sus futuros clientes.
* La aplicación permitirá una contratación de trabajadores informales al cliente de forma más simple y rápida, ya que estos no tendrán que esperar la pre disponibilidad de estos cuando se los contactan.
* Facilitará al trabajador informal llegar al lugar donde lo hayan contratado.
* Los clientes podrán conocer referencias del trabajador al que quiere contratar, para determinar si es bueno o malo en su trabajo.

1. **Marco metodológico**

Las metodologías de desarrollo consisten en guías para poder llevar a cabo el desarrollo de cualquier software, constituyéndose en guías para poder:

* Planificar cómo se efectuara el desarrollo del software.
* Obtener una buena estructura lógica y de diseño del software.
* Evitar que el ciclo de vida del desarrollo del software se extienda indefinidamente.
* Control de errores y crecimiento del software.

Las metodologías de desarrollo son importantes, ya que evita que el desarrollo del software sea desordenado, además de ofrecer buenas prácticas para el desarrollo de software.

Un modelo para el desarrollo de software es una representación abstracta de un proceso. Cada modelo representa un proceso desde una perspectiva particular y así proporcione información parcial sobre el proceso. Éstos modelos generales no son descripciones definitivas de los procesos del software más bien son abstracciones de los procesos que se pueden utilizar para el desarrollo del software. Puede pensarse en ellos como marcos de trabajo del proceso y que pueden ser adaptados para crear procesos más específicos. (CENDEJAS, s.f).

Un factor importante de las metodologías de desarrollo de software es que no son definitivas, es decir, no es necesario cumplir a cabalidad con lo que dictan para poder desarrollar el software, sino que es posible combinarlas para así obtener una guía de desarrollo de software, el cual se ajusta a la medida de la situación y el desarrollo.

Como se explicó en el concepto anterior, un modelo para el desarrollo de software es una representación abstracta de un proceso. Cada modelo representa un proceso desde una perspectiva particular y así proporcione información parcial sobre el proceso. Éstos modelos generales no son descripciones definitivas de los procesos del software más bien son abstracciones de los procesos que se pueden utilizar para el desarrollo del software. Puede pensarse en ellos como marcos de trabajo del proceso y que pueden ser adaptados para crear procesos más específicos. (CENDEJAS VALDÉZ, s.f).

La tendencia de las metodologías de desarrollo de software empezaron a ser populares en el año 1960, cuando las grandes empresas necesitaban software para encarar tareas en sus diferentes áreas de trabajo, sean estas grandes o pequeñas. Los desarrolladores necesitaban guías para no perderse o construir un producto con errores o que no respondan a los requerimientos del cliente.

Actualmente para el desarrollo de software, se aplican los siguientes modelos de desarrollo de software:

* Modelos prescriptivos: Modelo incremental, modelo en espiral, entre otros.
* Modelos ágiles: *SCRUM*, *XP*, por citar algunos

A continuación, se describe el modelo en espiral y modelo *SCRUM* como candidatos a ser aplicados al desarrollo del proyecto

* 1. **Modelo en espiral**

Es un modelo de desarrollo de software creado en 1986 por Barry Boehm, la cual consiste en un desarrollo evolutivo del software, ya que:

* No exige que el software concluya una vez que se completaron todas las etapas que describe esta metodología, solo puede ser cierto módulo o funcionalidad del software.
* Cuando se cumple la última etapa de la metodología, se vuelve a la primera etapa, haciendo que el software nunca tenga fin en su desarrollo, dando margen a poderlo ampliarlo y mejorarlo, esto hace que el modelo sea cíclico y tenga forma de espiral, originando ahí el nombre de esta metodología.
* Se pueden repetir todos los pasos cuantas veces sea necesario.
* Da la posibilidad de mostrar el avance al cliente, para en caso de haber algún error en el software, volver a la etapa donde este el error para corregirlo, antes de volver a iniciar un nuevo ciclo de desarrollo de esta metodología.

Es un modelo de proceso de software evolutivo donde se conjuga la naturaleza de construcción de prototipos con los aspectos controlados y sistemáticos del MODELO LINEAL y SECUENCIAL. Proporciona el potencial para el desarrollo rápido de versiones incrementales del software que no se basa en fases claramente definidas y separadas para crear un sistema. (ANONIMO, 2015).

Las etapas que sugiere la metodología en su modelo original., son las siguientes:

1. Planificación o determinación de objetivos: Consiste en resaltar los límites del desarrollo y los objetivos a cumplir en la iteración presente.
2. Análisis de riesgo: Aquí se determina los riesgos o amenazas que puede haber durante el ciclo de desarrollo.
3. Desarrollo y pruebas: Se desarrolla el software, además de realizar las diferentes pruebas para verificar que el software desarrollado funcione bien.
4. Evaluación: Se entrega el software concluido para evaluación por parte del cliente, para determinar los arreglos o mejoras a hacer antes de pasar al siguiente iteración o para considerarlos para el nuevo ciclo de desarrollo.

**Figura 1**

**Representación visual del modelo en espiral**



Fuente: Elaboración propia

* 1. ***SCRUM***

*SCRUM* es “Un marco de trabajo por el cual las personas pueden acometer problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente” (SUTHERLAND y SCHWABER, 2013).

Entre sus característica más importante es la de presentar frecuentemente el avance de la solución del problema, ya que de esa forma se puede obtener un mejor resultado y reducir el margen de error y pérdidas a medida que el proyecto avanza.

Para resumir como trabaja *SCRUM*, primero se basa en iteraciones (*Sprint* en inglés), las cuales son períodos de tiempo entre 2-4 semanas, donde el equipo define lo que se va a trabajar y repartir estos trabajos entre los miembros del equipo antes de iniciar la iteración, pasado el tiempo de la iteración, se muestra el avance del proyecto a los clientes, estos dan sus modificaciones o agregaciones a hacer y se planifica la siguiente iteración en base al trabajo que se tiene que hacer y las modificaciones a hacer, y todo lo descrito anteriormente se repite la cantidad de veces necesario hasta que el problema sea resuelto.

*SCRUM* no hace jerarquías de trabajo, sino que hace que todos los miembros del equipo trabajen juntos, pero les da diferentes roles, tales como:

* *Scrum Master*: El encargado de que la metodología se cumpla, además de ser el que ayuda a la resolución de diferentes problemas que pueda haber en el equipo.
* *Product Owner*: Es el representante del cliente al que se le está resolviendo el problema, es el puente de comunicación entre el cliente y el equipo en general, además se encarga de supervisar que la solución que se está desarrollando sea la que el cliente desea.
* Equipo de desarrollo: Son el grupo de personas que desarrollan la solución al problema.

El Equipo *SCRUM* consiste en un Dueño de Producto (*Product Owner*), el Equipo de Desarrollo (*Development Team*) y un *Scrum Master*. Los Equipos *SCRUM* son autoorganizados y multifuncionales. Los equipos autoorganizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. El modelo de equipo en *SCRUM* está diseñado para optimizar la flexibilidad, la creatividad y la productividad. (SCHWABER, 2013).

*SCRUM* cuenta con diferentes tipos de eventos para poder coordinar el desarrollo, soluciones de problemas y agregar, modificar o eliminar funcionalidades o aspectos del producto, entre esos eventos tenemos:

* *Sprint Planning Meeting*: Consiste en determinar los trabajos a realizar para solucionar problemas específicos del problema general, para que al final se una todo y se de una posible solución que pueda usar el cliente al finalizar la iteración, esta reunión suele tener una duración de 4 horas como máximo.
* *Daily Scrum*: Son reuniones diarias donde cada miembro del equipo describe que realizara ese día, si tiene dificultades en algo y de reportar lo que hizo hasta antes del *Daily Scrum* actual, esto suele durar entre 15-20 minutos como máximo.
* *Sprint Review*: Consiste en la presentación del avance realizado, para ver qué aspectos se cumplieron o no y ver posibles errores que pueda haber para subsanarlos en la siguiente iteración, esta reunión suele tener una duración de 4 horas como máximo.
* *Sprint Retrospective*: Consiste en la evaluación de la iteración que acabo de finalizar, para ver los errores que se cometieron, solución de problemas, recalcar lo positivo que se realizó y de evaluar el trabajo desempeñado, todo esto con el objetivo de mejorar como equipo para las siguientes iteraciones, esta reunión suele tener una duración de 4 horas como máximo.

En *SCRUM* existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en *SCRUM*. Todos los eventos son bloques de tiempo (time-boxes), de tal modo que todos tienen una duración máxima. Una vez que comienza un *Sprint*, su duración es fija y no puede acortarse o alargarse. Los demás eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso. (SCHWABER, 2013).

*SCRUM* usa historias de usuarios, las cuales son una forma de poder representar una necesidad del cliente de forma de que cualquier persona lea la descripción, pueda entender lo que se quiere llegar a hacer, siendo más específicos, “Una historia de usuario representa una pequeña cantidad de valor empresarial que un equipo puede entregar en una iteración. Mientras que los requisitos tradicionales (como los casos de uso) pretenden ser lo más detallado posibles” (ROTH, s.f.).

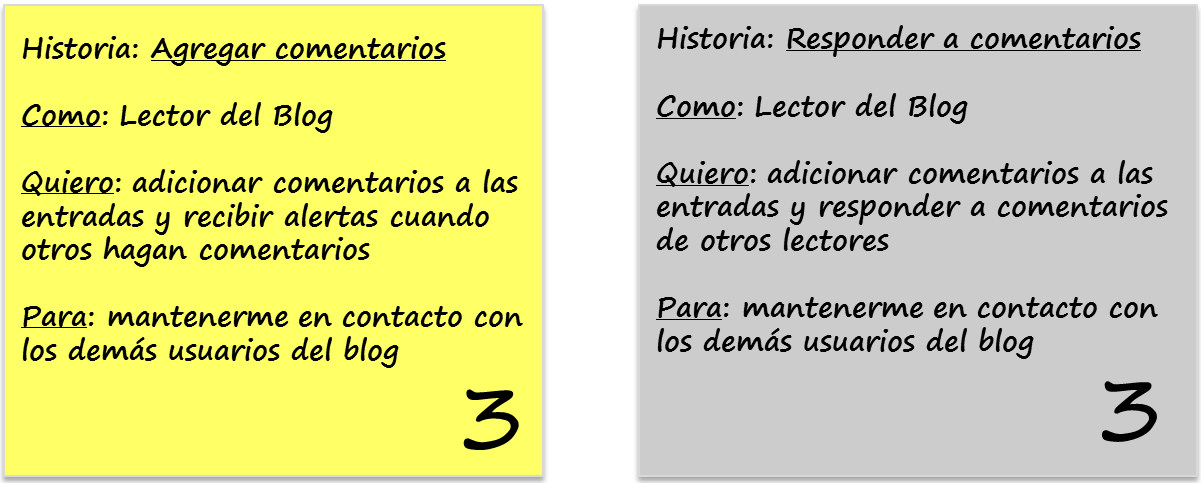
Las historias de usuarios son escritas en lenguaje que las personas puedan entender, así nadie tiene que estar aprendiendo referencias nuevas, su estructura básica es la siguiente:

* Título: Nombre que recibirá la historia de usuario
* Como: Se describen quienes son los que realizarán o requieren esta funcionalidad.
* Quiero: Se describe lo que se va a hacer.
* Para: Se describe por qué se necesita esta función

Aunque se pueden agregar más características para describir la funcionalidad, los elementos descritos anteriormente son indispensables y las necesarias para escribir una historia de usuario.

**Figura 2**

**Estructura de una historia de usuario**



Fuente: www.gazafatonarioit.com

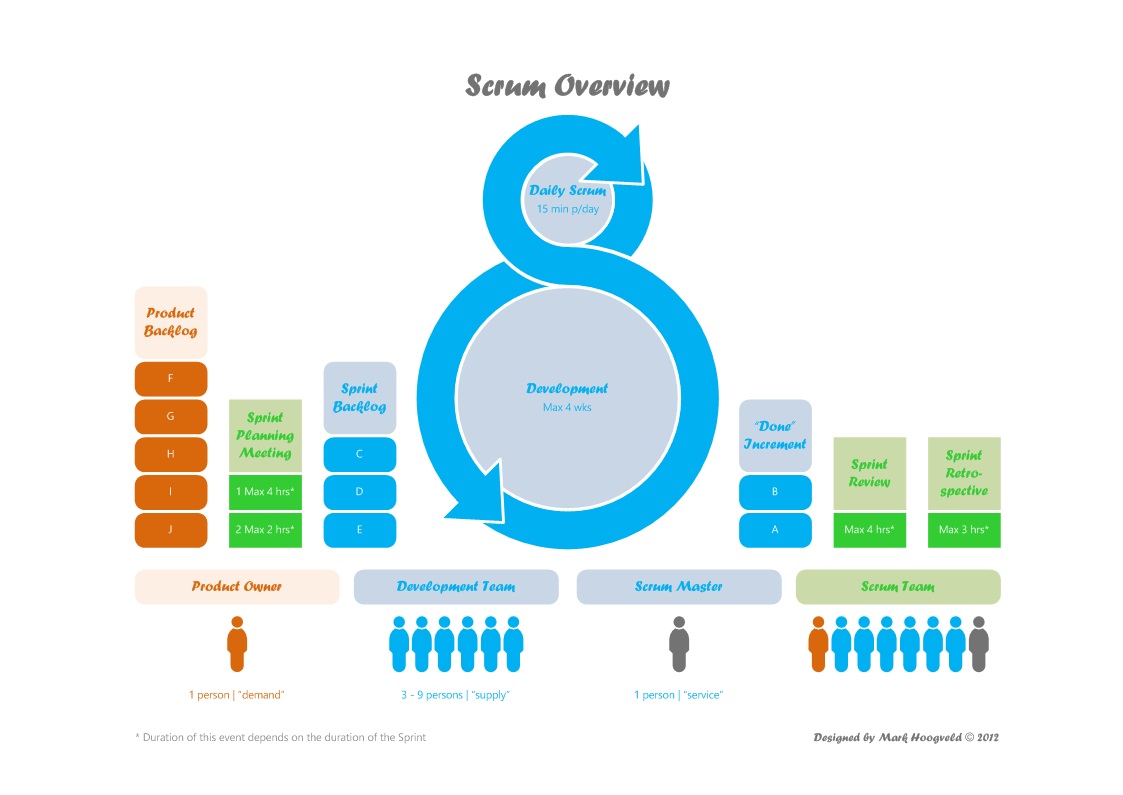
Esta herramienta la aplican las metodologías ágiles y se empezó a popularizar y a desplazar a los diagramas de caso de uso de *UML*, por los siguientes motivos:

* Modificar el diagrama es muy tedioso, ya que cada vez que se quiere añadir una nueva funcionalidad, se tiene que modificar el diagrama, lo mismo sucede cuando se quiere eliminar una funcionalidad.
* Se tiene que hacer especificación constante de cómo están conectados las funciones del sistema en el diagrama de casos de uso, lo cual conlleva al punto descrito anteriormente en caso de querer modificar o añadir funcionalidades.
* Si una persona no conoce los diagramas de caso de uso, es más difícil que entienda lo que se va a desarrollar.

Después de describir los aspectos importantes de *SCRUM*, a continuación, una imagen descriptiva de cómo funciona *SCRUM*.

**Figura 3**

**Representación visual completa del funcionamiento de SCRUM**



Fuente: https://calvinx.com

* 1. **Lenguaje Unificado de Modelado**

El Lenguaje Unificado de Modelado o *UML* es un lenguaje gráfico para poder visualizar, crear y documentar como está construido un sistema en general, para que cualquier persona pueda entender como está estructurado y como está analizado un software.

El Lenguaje de Modelamiento Unificado (*UML* - *Unified Modeling Language*) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. *UML* entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. (SALINAS y HISTCHFELD, s.f).

*UML* no es una metodología de desarrollo de software, por lo cual, no puede ser usado para planificar el proceso de desarrollo. La utilidad del *UML* es la de mostrar de forma gráfica como está diseñado, creado y analizado un sistema.

*UML* cuenta con diferentes diagramas para poder representar sistemas, pero solo se detallan los que se usaron en el proyecto:

* Diagrama de clases: Diagrama que modela un conjunto de objetos que tienen una estructura, comportamiento y relaciones con propiedades parecidas que tiene un sistema.
* Diagrama de contexto: Diagrama que sirve para modelar los límites de usabilidad de un sistema, mostrando la interacción entre componentes y la interacción entre los usuarios de esta.
* Diagrama de despliegue: Diagrama que sirve para modelar el funcionamiento del software en la realidad, mostrar los componentes con los que puede interactuar y como están relacionados estos entre sí.
  1. **Comparación de modelos de desarrollo de software**

A continuación, se compararan las ventajas y desventajas del modelo en espiral y *SCRUM*.

**Tabla 1**

**Lista de ventajas y desventajas del modelo espiral y *SCRUM***



Fuente: Elaboración propia

Se puede observar que ambos enfoques tienen sus debilidades y fortalezas, por tanto, para la elaboración de la propuesta metodológica para el proyecto se tomara como referencia ambos modelos.

* 1. **Propuesta metodológica**

*SCRUM* es una excelente metodología de desarrollo de software, pero está más enfocada para equipos con varias personas y dado que el desarrollo de este proyecto será individual, será muy difícil de implementar lo que dicta la metodología.

*SCRUM* tiene muchas similitudes con el modelo en espiral, ya que maneja iteraciones y se adapta a los cambios que vienen en los desarrollos de software, solo que el modelo en espiral no está especialmente enfocado a equipos de desarrollo de varias personas, siendo más flexible para que solo una persona lo pueda desarrollar.

Con lo descrito anteriormente, se decidió usar el modelo en espiral como metodología de desarrollo de software, por los siguientes motivos:

* Al tener requerimientos iniciales ya definidos, se puede empezar a planear el desarrollo.
* La metodología al manejarse en iteraciones, permitirá poder desarrollar por partes pequeñas el proyecto, para poder tener mejor control de la integración de las nuevas funcionalidades del proyecto, por lo que cada objetivo específico será una iteración

A pesar de usar el modelo en espiral para el desarrollo del proyecto, se decidió conservar una herramienta de *SCRUM* para facilitar el desarrollo de este, para lo cual, se decidió usar las historias de usuario para descripción de requerimientos.

El modo en que se trabajara con el modelo en espiral será el siguiente:

* En la planificación se detallarán los requerimientos necesarios a realizar en la iteración.
* En análisis de riesgo se describirán las diferentes amenazas que puedan haber en el proyecto, con sus vulnerabilidades y soluciones, todo esto reflejado en una matriz de riesgo, para entender cómo se puntuara en la matriz, ver el anexo B.
* En el desarrollo de describirá como se desarrolló el sistema, para lo cual, se dividirá este capítulo en los siguientes subcapítulos:
  + Diseño
  + Sistema web
  + Aplicación móvil

Cada subcapítulo descrito anteriormente expondrá la información relacionada a este, para tener orden de lectura y no mezclar información. Asimismo, se usará diagramas *UML* para describir la estructura lógica del sistema e historias de usuario para marcar las funcionalidades a desarrollar.

* En evaluación se hará una evaluación con respecto al desarrollo del sistema en esa determinada iteración, dando correcciones y mejoras a hacer.

1. **Marco conceptual**
   1. **Servicios web**

Un servicio web (o *Web Service*) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, evitando que la aplicación que solicite la información tenga que obtener y procesar la información requerida, ayudando al rendimiento de la aplicación solicitante.

Existen numerosas definiciones de Servicios Web y esto demuestra, en parte, la gran complejidad de los servicios que se agrupan bajo este término y las implicaciones asociadas a ellos. Hasta ahora la definición más general y convincente es decir que los Servicios Web son el conjunto de aplicaciones o tecnologías con capacidad para interoperar en la Web. Estas tecnologías intercambian datos entre ellas con el fin de ofrecer unos servicios. La *World Wide Web* no es sólo un espacio de información, también es un espacio de interacción. Utilizando la Web como plataforma, los usuarios, de forma remota, pueden solicitar un servicio que algún proveedor ofrezca en la red. Pero para que esta interacción funcione, deben existir unos mecanismos de comunicación estándares entre diferentes aplicaciones. Estos mecanismos deben poder interactuar entre sí para presentar la información de forma dinámica al usuario. Se precisa, pues, una arquitectura de referencia estándar que haga posible la interoperabilidad y extensibilidad entre las distintas aplicaciones y que permita su combinación para realizar operaciones complejas. (LAMARCA, 2013).

Las aplicaciones que consumen servicios web pueden estar desarrolladas en lenguajes de programación diferentes, desde diferentes sistemas operativos y pueden consumir el servicio desde cualquier dispositivo que lo permita, indiferente de su sistema operativo.

Los servicios web también pueden ser utilizados para intercambiar datos en redes de computadoras o Internet, además de que servicios web pueden llamar a otros servicios web bajo el mismo concepto descrito anteriormente.

La información resultante que devuelven los servicios web son archivos *JSON* o *XML* (formatos de archivos ligeros para intercambio de información), los cuales, son estándares para el envío de información entre sistemas, ayudando al desarrollo de sistemas que usen servicios web.

Las ventajas de los servicios web son:

* Mejora el rendimiento de las aplicaciones
* Uso de estándares para recibir y enviar información a los servicios web.
* Disponibilidad para cualquier tipo de software, en cualquier momento y lugar.
* Interconectividad de sistemas de información más simple.
* Ayuda a crear independencia de algunos de los componentes de un sistema de información o de software, simplificación del desarrollo de software.
  + 1. ***RESTFul***

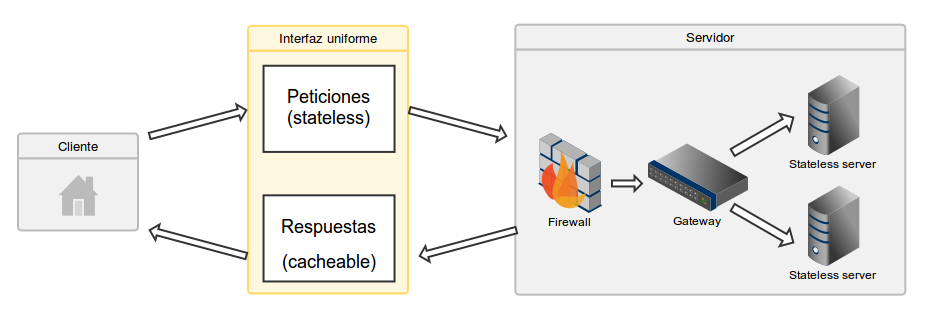
*REST* (*Representational State Transfer*) es un estilo de arquitectura de software para crear servicios web, el cual, tiene la característica de ser *stateless*, (significa que no guarda información de lo que envió en cada petición al servicio web) y de poder enviar y recibir recursos (*image/JPG*, *XML*, *JSON*, *text/html*, etc), mediante uso del protocolo *HTTP*.

*REST* es un estilo de arquitectura que abstrae los elementos de dicha arquitectura dentro de un sistema *hypermedia* distribuido. Pero esto de por sí no nos dice demasiado a no ser que nos dediquemos al estudio teórico de este tipo de cosas. Así que, de nuevo, ¿qué es *REST*? Utilizando palabras llanas, podríamos decir que *REST* es un conjunto de principios, o maneras de hacer las cosas, que define la interacción entre distintos componentes, es decir, las reglas que dichos componentes tienen que seguir. El protocolo más usado que cumple esta definición, es el protocolo *HTTP*. (FERNÁNDEZ, 2013).

Al basarse en *HTTP*, *RESTful* puede ser implementado en sistemas web y en los servidores que lo almacena, permitiendo ser accesibles mediante *URL* y usa los métodos *HTTP* conocidos para obtener o devolver información (*GET*, *POST*, *DELETE*, etc), todo esto hace que el desarrollo, testeo e implementación de servicios web con *RESTful* sea fácil, práctico y de mejor integración con diferentes sistemas o a peticiones de diferentes clientes que requieran estos servicios.

**Figura 4**

**Diagrama de funcionamiento de servicios *RESTful***



Fuente: http://www.adwe.es/

*RESTful* es el de cumplir con los siguientes requerimientos:

* Escalabilidad: A medida que la web va crecido, no tiene que afectar la usabilidad del sistema y seguir siendo accesible por medio de internet.
* Generalidad de interfaces: Dar la facilidad de que cualquier cliente pueda interactuar con cualquier servidor que ofrezca el servicio web sin ninguna configuración especial.
* Funcionamiento independiente del cliente: Los servicios web deben ser capaces de poder interactuar con usuarios actuales (que usan tecnologías más avanzadas o actuales) o antiguos (que usan tecnologías antiguas).
* Compatibilidad con componentes intermediarios: Independientemente de cuantos servicios web o sistemas se tiene que acceder para obtener el resultado deseado, estos tienen que devolver el resultado deseado.
  + 1. ***SOAP***

*SOAP* es un protocolo que permite la comunicación entre diferentes sistemas mediante intercambio de archivos *XML* vía internet, el cual se volvió estándar para intercambio de información en servicios web y que esta implementado en varios lenguajes de programación.

*SOAP* es un protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido. Es un protocolo basado en *XML* que consta de tres partes: un sobre que define un marco para describir lo que está en un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos por la aplicación, y una convención para representar las llamadas a procedimiento remoto y respuestas. Con *SOAP* potencialmente se puede utilizar en combinación con una variedad de otros protocolos; sin embargo, los únicos fijaciones definidas en este documento se describe cómo utilizar *SOAP* en combinación con *HTTP* y *HTTP* *Extensión Framework*. (W3C, 2000).

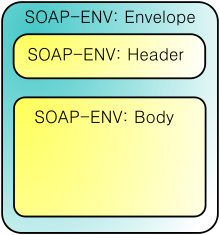
La estructura de SOAP es la siguiente:

* *Envelope*: Campo que contiene la información que permite identificar el mensaje y declara el inicio y el fin del archivo.
* *Header*: Campo que contiene la información opcional que permite declarar la forma de procesar la información, intermediaros que pueden haber, entre otros.
* *Body*: Contiene la información que se envía.
* *Faul*: Campo opcional donde se describe los errores que se pudieron obtener durante él envió de la información o de la respuesta del destinatario.

Se deja una imagen con el esquema descrito anteriormente.

**Figura 5**

**Estructura de *SOAP***



Fuente: es.wikipedia.org

Cualquier servicio web que use *SOAP*, tiene que cumplir con las siguientes características:

* Extensibilidad (facilidad de poder ampliar los servicios y su seguridad).
* Neutralidad (Se tiene que poder usarse en *HTTP*, *SMTP*, *JMS*, *TCP* o cualquier protocolo de transmisión de texto).
* Independencia (Se tiene que poder implementar en cualquier lenguaje de programación).
  + 1. **Selección de servicio web**

Aunque *SOAP* es un estándar para creación de servicios web, este perdió popularidad en contra de *RESTful*, porque *RESTful* es más rápido de procesar, ya que no tiene que estar creando la estructura tradicional de los archivos *XML* que deben crear *SOAP* para el intercambio de información y porque soporta *JSON* sin estructura base que tiene que tener el archivo *XML* de *SOAP*.

*Ruby on Rails* es un *framework* que permite escribir servicios web *RESTful* para el sistema web que se esté desarrollando, pudiendo funcionar como servicios web y sistema web al mismo tiempo, siendo muy versátil a la hora de desarrollar cualquier sistema.

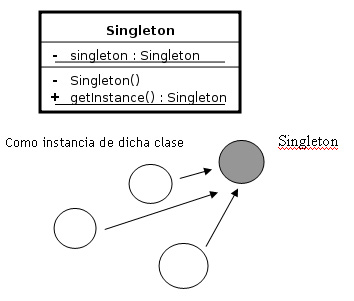
Por lo descrito anteriormente, se decidió que los servicios web serán *RESTful* y se usara *Ruby on Rails* para desarrollo del sistema web.

* 1. **Patrón *Singleton***

El patrón *Singleton* “es un patrón diseñado para limitar la creación de objetos pertenecientes a una clase. El objetivo de este patrón es el de garantizar que una clase solo tenga una instancia (o ejemplar) y proporcionar un punto de acceso global a ella. Esta patrón; por ejemplo, suele ser utilizado para las conexiones a bases de datos” (MOYA, 2015.).

**Figura 6**

**Explicación visual del patrón *Singleton***



Fuente: www.palentino.es

La facilidad de poder instanciar solo una vez la clase y que se puede acceder a esta desde diferentes partes del programa, evita la instancia de clases constantemente, ahorrando tiempo de ejecución y espacio en memoria.

El patrón *Singleton* se puede implementar en varios lenguajes de programación como *Java*, *C#* o *Ruby*, permitiendo la facilidad de poder implementar este patrón en el proyecto.

* 1. **Matriz de riesgos**

Las matrices de riesgo son esquemas que sirven para especificar las diferentes amenazas que puede tener un proyecto, considerando la probabilidad y el impacto que pueda tener en el proyecto, además de una posible solución o forma de mitigar las amenazas.

Estas matrices son usadas principalmente en empresas para determinar las amenazas que pueda tener un proyecto, determinar los costos necesarios para subsanar las amenazas y dar un análisis de que si es fiable hacer el proyecto o no.

Una matriz de riesgo constituye una herramienta de control y de gestión normalmente utilizada para identificar las actividades (procesos y productos) más importantes de una empresa, el tipo y nivel de riesgos inherentes a estas actividades y los factores exógenos y endógenos relacionados con estos riesgos (factores de riesgo). Igualmente, una matriz de riesgo permite evaluar la efectividad de una adecuada gestión y administración de los riesgos financieros que pudieran impactar los resultados y por ende al logro de los objetivos de una organización. (SIGWEB, s.f.).

**Figura 7**

**Matriz de riesgo**



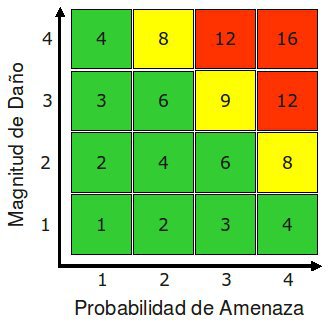
Fuente: aisha-osorio.webnode.com.co

El impacto y la probabilidad reciben una calificación, el que realiza el análisis de riesgo da un puntaje determinado en un rango impuesto de valores, que generalmente son de 1 al 5, siendo 1 el más bajo y 5 el más alto en cuestión de que si puede suceder y cuanto afectaría al proyecto.

El impacto y la probabilidad se los puede graficar y tratar como un mapa de coordenadas de 2 dimensiones, siendo un mejor entendimiento de que amenazas son más importantes de tratar.

**Figura 8**

**Diagrama de impacto y probabilidad en mapa de coordenadas de 2 dimensiones**



Fuente: https://protejete.wordpress.com

Hay ocasiones que también el valor de la probabilidad y el impacto se los multiplican, para que el resultado se pueda colocar en una tabla de prioridades o de riesgo, que tiene un rango de valores, donde los valores bajos son amenazas que no afectaran mucho al proyecto y los valores altos los que si afectaran mucho al proyecto.

**Figura 9**

**Diagrama de impacto y probabilidad con tabla de riesgo**



Fuente: Elaboración propia

No hay una forma determinada para crear matrices de riesgo, pero generalmente, tienen que estar la amenaza, la probabilidad que suceda, el impacto que generara y la forma en la que se puede solventar.

Tampoco hay una forma definitiva para determinar la puntuación, se puede considerar manejar ambas características por separado, el rango de valores puede ser diferente y que en vez de manejar números, se manejen características como bajo, medio y alto.

Se usara un diagrama de impacto y probabilidad con tabla de riesgo para puntuar las vulnerabilidades y se manejaran las siguientes características para describir la matriz de riesgo de este proyecto:

* Amenaza: Riesgo o posible peligro que puede generar determinada por una acción o suceso.
* Vulnerabilidad: Factores o consecuencias que hacen que una determinada amenaza pueda suceder.
* Probabilidad: Valor numérico del 1 al 5 que determina si determinada vulnerabilidad pueda suceder.
* Impacto: Valor numérico del 1 al 5 que cuantifica que tanto afectara de forma negativa la vulnerabilidad al proyecto.
* Riesgo: Multiplicación de Probabilidad con Impacto, el resultado determinara la importancia de solucionar la vulnerabilidad.
* Medida preventiva: Solución que se da para solventar la vulnerabilidad.

1. **Tecnologías de desarrollo**
   1. **Tecnologías de desarrollo de aplicación móvil**
      1. ***Java***

Es un lenguaje de programación seminterpretado, orientado a objetos e inspirado en C/C++. Tiene la característica de que solo se tiene que compilar una vez para que pueda correr en diferentes equipos, máquinas o sistemas operativos sin necesidad de depender de limitaciones o recursos exclusivos del equipo.

El lenguaje *Java* se creó bajo los siguientes objetivos:

* Sencillo y familiar de usar.
* Orientado a objetos.
* Robusto y seguro.
* Independiente de la arquitectura.
* Portable.
* Alto rendimiento.
* Interpretado.
* Multi hilo.

*Java* es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o *write once, run anywhere*), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. *Java* es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados. (ANÓNIMO, 2015).

Una de las características más importante de *Java* es que es un lenguaje de programación compilado e interpretado, haciendo que se pueda compilar el código y ser interpretado por una máquina virtual (*Java Virtual Machine*), haciendo posible que se puede ejecutar en cualquier equipo.

*Java* fue diseñado para desarrollo de sistemas web, pero además de eso, se le añadió diferentes funcionalidades para que se pueda desarrollarse en diferentes sistemas.

Por las características anteriores, además de ser de libre desarrollo (no se tiene que pagar para poder desarrollar aplicaciones en *Java*), este lenguaje de programación gano mucha fama con el pasar de los años, siendo en 2005 uno de los mejores años para *Java*, ya que contaba con 4,5 millones de desarrolladores y 2.500 millones de dispositivos habilitados con tecnología J*ava*.

Para programar aplicaciones nativas en *Android* se necesita *Java*, por lo cual, se decidió usar *Java* para el desarrollo de la aplicación móvil.

* + 1. ***Android***

*Android* es uno de los sistema operativo para dispositivos móviles más popular del mercado, por el hecho de que está basado en Linux y es software libre, *Android t*ambién funcionan en *tablets*, celulares, televisores, etc, el cual, fue lanzado al público en el año 2007.

*Android* es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, *tablets* o tabléfonos; y también para relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por *Android* Inc., empresa que Google respaldó económicamente y más tarde, en 2005, la compró.9 *Android* fue presentado en 2007 junto la fundación del *Open Handset Alliance* (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles.10 El primer móvil con el sistema operativo *Android* fue el *HTC Dream* y se vendió en octubre de 2008.11 Los dispositivos de *Android* venden más que las ventas combinadas de *Windows Phone* e *IOS*. (ANÓNIMO, 2015).

*Android* tiene una gran comunidad de desarrolles de aplicaciones, la cual, ya desarrollo más de 1000000, los cuales usan la aplicación oficial de *Android* para poder descargar las aplicaciones, la cual se llama *Google Play*, o usar otras aplicaciones no oficiales como *Samsung Apps*, *Slideme* o *Amazon Appstore*.

*Android* es uno de los sistemas operativos más populares en el mercado para dispositivos móviles, teniendo en E.E.U.U un 61.9 % y a nivel mundial un 82.3 % en uso de dispositivos con *Android* en el año 2014.

*Android* tiene las siguientes características:

* Soportes para gráficos en 2D y 3D (los cuales están basados en *OpenGL* ES 2.0).
* Adaptabilidad de la pantalla a diferentes resoluciones y dispositivos.
* Soporte para múltiples formas de interconectividad y transmisión de datos, sea *Wi-fi*, *bluetooth*, etc.
* Tiene implementado los estándares para realizar llamadas y envió de *SMS*.
* Tiene soporte a *Java*, haciendo que la mayoría de las aplicaciones para *Android* sean desarrolladas en ese lenguaje de programación.
* No implementa una máquina virtual java para ejecutar sus aplicaciones, sino usa una máquina virtual llamada *Dalvik*, la cual está diseñada específicamente para *Android* y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.
* Tiene implementado por defecto aplicaciones de:
  + Tienda de aplicaciones.
  + Mapas y geolocalización.
  + Correo electrónico.
  + Navegación por internet.
  + Etc.

*Android* es de código abierto, lo cual significa que diferentes personas pueden descargar el código fuente del sistema operativo y sacar diferentes versiones de *Android* o mejorarlo para su distribución gratuita o de pago.

Aunque una de las desventajas notorias de *Android* es que se necesita controladores para que ciertas versiones de *Android* funcionen en Hardware antiguo o nuevo, para lo cual, los fabricantes de hardware tienen que dar un firmware para que *Android* funcione correctamente.

La empresa que desarrolla las versiones comerciales de *Android*, subió a internet su propia interfaz de desarrollo para crear aplicaciones para *Android*, la cual es gratuita y se llama *Android Studio*.

Cuando se crea proyectos con *Android Studio*, se tienen las siguientes ventajas:

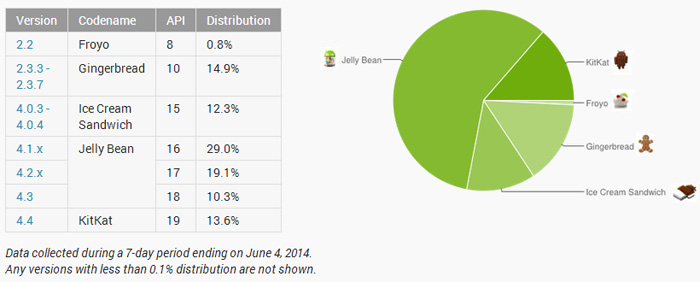
* Se crea de manera automática la estructura de las carpetas que se usaran.
* Se configura todos los parámetros y complementos que se fueran a añadir al proyecto de forma automática.
* Permite descargar complementos de *Android* para el desarrollo de proyectos más complejos y referenciarse a estos de manera automática

Por los motivos mencionados anteriormente, se usara *Android Studio* como interfaz de desarrollo para la aplicación móvil.

La versión mínima en la cual funcionara la aplicación móvil será en la 4.0.3, ya que es esta versión tiene aspectos visuales y funcionales que son usados en versiones superiores o actuales, evitando tener incompatibilidades de componentes entre versiones.

**Figura 10**

**Diagrama de uso de las versiones de *Android***



Fuente: www.androidporelmundo.com

* 1. **Tecnologías de desarrollo del sistema web**
     1. ***Ruby***

Es un lenguaje de programación interpretado y orientado a objetos, que tiene la característica de que su sintaxis trata de llegar o igualar al del lenguaje natural, permitiendo que cualquier usuario pueda programar en ella y que sea fácil de entender, para poder atraer a más gente a usar este lenguaje de programación.

Se empezó a popularizar en 2006 para adelante, ya que *Ruby* es de licencia de software libre, es bastante flexible a la hora de programar y por el nacimiento de *Ruby on Rails*, un *framework* para el desarrollo de sistema web usando Ruby como lenguaje base, dando facilidad a la hora de programar sistemas web.

Para desarrollar sistemas web con *Ruby on Rails*, se requiere *Ruby*, por lo cual, se decidió usar *Ruby* para el desarrollo del sistema web.

* + 1. ***Ruby on Rails***

Es un *framework* gratuito para el desarrollo de sistemas web usando *Ruby* y una arquitectura *MVC* (Modelo-Vista-Controlador) para dar estructura al sistema web.

*Ruby on Rails* permite la inclusión de librerías de forma simple (las cuales se llaman gemas), las cuales son de libre acceso y descarga, además de poder configurar que gemas se usaran si están en desarrollo, pruebas unitarias o en producción.

Un aspecto importante es el uso del *ActiveRecord*, utilidad que permite acceder a diferentes base de datos y obtener información de estas sin necesidad de estar escribiendo constantemente las sentencias *SQL*, solo basta con una línea de comando para recolectar información.

*Ruby on Rails* es bastante personalizable, por los siguientes motivos:

* Permite configurar el manejador de base de datos que se usara durante el desarrollo, durante las pruebas unitarias y durante producción, facilitando su implementación.
* Permite configurar rutas a los controladores de forma simple.
* Configuración de parámetros de seguridad que tendrá el sistema web.

Otra de las características importantes es que se puede crear las bases de datos que necesite el sistema sin necesidad de hacer correr *scripts* en el manejador de base de datos que se tenga configurado, agilizando el desarrollo del software.

* + 1. ***PostgreSQL***

Es un sistema gestor de base de datos de código abierto, el cual es bastante popular, siendo implementado en varios servidores web y teniendo soporte en diferentes sistemas operativos.

*PostgreSQL* implementa muchas opciones de seguridad para que se reduzcan los problemas de ejecución, teniendo un soporte grande de información por tabla (pudiendo soportar 32 terabytes de datos por tabla) y el soporte para *triggers*.

Además de unas de sus características importantes es que tiene soporte a múltiples consultas a tablas, para evitar problemas de concurrencia y de acceso a estas por medio de direcciones IP en su versión 4 o 6.

Muchos servidores web para sistemas web desarrollados en *Ruby on Rails* implementan *PostgreSQL*, citando como ejemplo *Heroku*, por lo cual, se decidió usar este sistema gestor de base de datos para el servidor web.

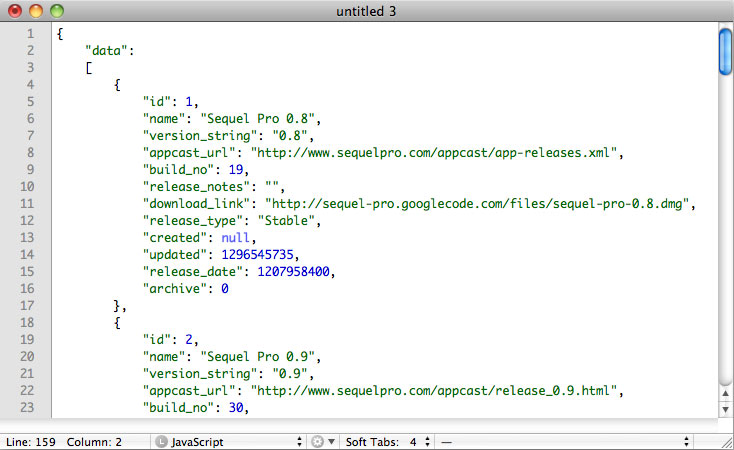
* 1. **Formato de intercambio de datos *JSON***

*JSON* es un lenguaje de etiqueta bastante simple de crear y leer, el cual sirve para hacer envió de información entre diferentes aplicaciones y sistemas, siendo usado comúnmente en servicios web.

*JSON* (*JavaScript Object Notation*) es un formato para el intercambios de datos, básicamente *JSON* describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. *JSON* nació como una alternativa a *XML*, el fácil uso en *javascript* ha generado un gran número de seguidores de esta alternativa. Una de las mayores ventajas que tiene el uso de *JSON* es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías. (ESQUIVA, 2013).

**Figura 11**

**Ejemplo del contenido de un archivo *JSON***



Fuente: www.sequelpro.com

El contenido de los archivos *JSON* es muy similar a *Javascript*, siendo más intuitivo crear estos archivos, además de que *Javascript* y *Java* tiene soporte para crear y leer archivos *JSON*.

*JSON* tiene soporte con diferentes servicios web, en especial con *RESTFul,* que es con el cual trabajaremos, además considerando de que se usara *Java* para el desarrollo de la aplicación móvil, se decidió usar *JSON* como lenguaje para crear los archivos que van a transferir.

1. **Ingeniería del proyecto**

Como se explicó en el marco metodológico, el desarrollo del proyecto sigue el flujo principal del modelo en espiral, por esta razón, considerando el desarrollo iterativo establecido por el modelo y objetivos específicos planteados para el proyecto, se ha establecido que la cantidad de iteraciones a realizar serán cuatro, número que coincide con la cantidad de objetivos específicos de este proyecto.

* 1. **Análisis inicial**

A continuación, se muestra la tabla con la información de las iteraciones planificadas, los objetivos específicos establecidos y requerimientos funcionales principales de cada iteración

**Tabla 2**

**Iteraciones a desarrollar**

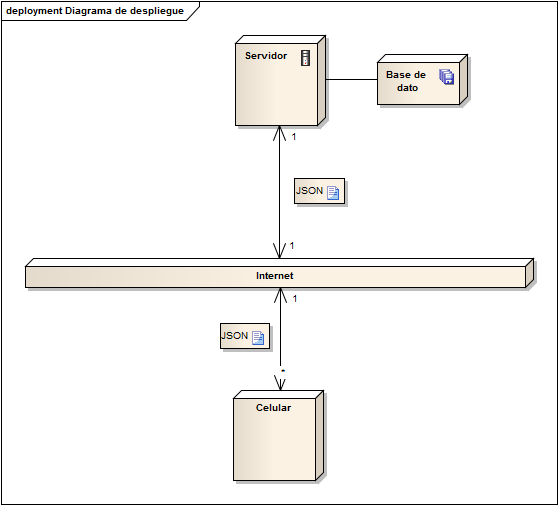


Fuente: Elaboración propia

Además de describir lo que se hará en cada iteración, se creó un diagrama de contexto y de despliegue, los cuales muestran como funcionara a nivel técnico la aplicación

**Figura 12**

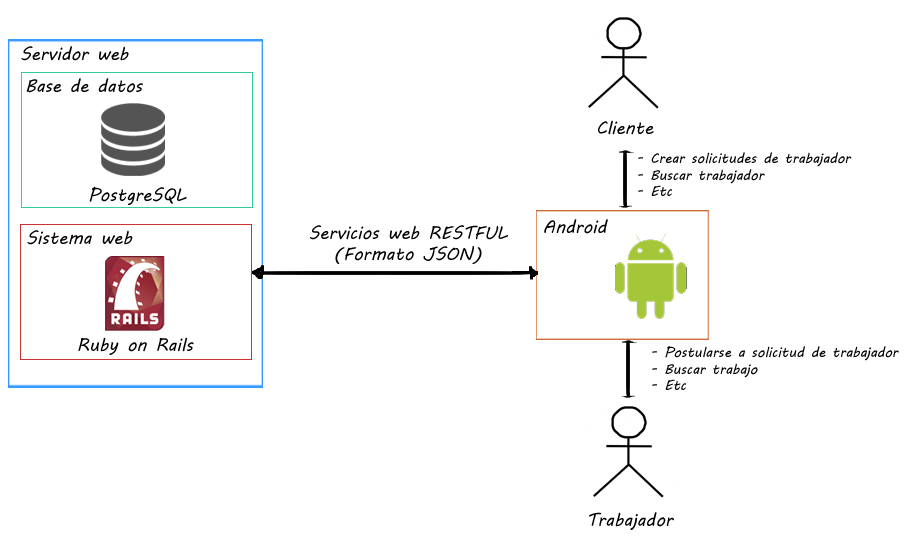
**Diagrama de despliegue del funcionamiento del sistema**



Fuente: Elaboración propia

**Figura 13**

**Diagrama de contexto del funcionamiento del sistema**



Fuente: Elaboración propia

Una vez concluido el análisis inicial, se procede al desarrollo de las iteraciones, usando la forma de trabajo descrita con anterioridad.

* 1. **Desarrollo del módulo de solicitud de trabajador informal**
     1. **Planificación**

Se determinaron las siguientes funcionalidades a desarrollar:

* Ingreso al sistema.
* Salir del sistema.
* Registrarse.
* Registrar solicitud de trabajador informal.
* Obtener todas las publicaciones de solicitud de trabajadores informales hechas.

Cada requerimiento descrito tiene que desarrollarse tanto para el servicio web, como para la aplicación móvil.

No se considerara la subida de fotos a las solicitudes de trabajadores informales, ya que no son indispensables en esta etapa del desarrollo del proyecto.

* + 1. **Análisis de riesgo**

En base a la experiencia y los problemas que se puede tener, se determinó las siguientes amenazas dentro de esta iteración:

* Enviar información del dispositivo móvil al servidor web.
* Recibir información del servidor web al dispositivo móvil.
* Guardar información de forma permanente en el dispositivo móvil.

A continuación, se muestra un diagrama de riesgo con las amenazas descritas.

**Tabla 3**

**Análisis de riesgo para el objetivo: “Desarrollo del módulo de solicitud de trabajador informal”**



Fuente: Elaboración propia

* + 1. **Desarrollo**
       1. **Diseño**

Antes de desarrollar el software, primero se documentó las historias de usuario que se desarrollarán en esta iteración para recalcar lo que se desarrollará tanto para el servicio web, como para la aplicación móvil.

**Tabla 4**

**Historias de usuario para el objetivo: “Desarrollo del módulo de solicitud de trabajador informal.”**



Fuente: Elaboración propia

Se determinó que los datos más importantes para implementar el acceso al sistema web serán los siguientes:

* Correo electrónico.
* Contraseña.

Se procedió a diseñar la base de datos, para el cual se determinó que se necesitaría los siguientes datos a guardar para esta iteración:

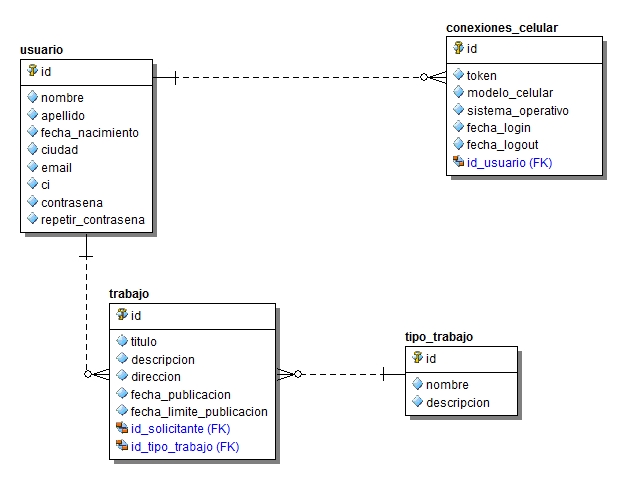
* Información del usuario (nombre, apellido, CI teléfono y ciudad donde vive actualmente).
* Datos de las conexiones activas que tenga el usuario en diferentes dispositivos.
* La solicitud de trabajador informal (ubicación, título, descripción, fecha donde se requiere al trabajador, etc).

Las solicitudes de trabajo, una vez creadas, no se podrán modificar, para evitar problemas de actualización de información entre los postulantes al trabajo y al cliente

En base a esos requerimientos, se diseñó la base de datos a usar.

**Figura 14**

**Diagrama del diseño de la base de datos de la iteración # 1**



Fuente: Elaboración propia

La forma en la cual la aplicación se conectaría con el servidor será a través de archivos *JSON*, los cuales se enviarían por servicios *RESTFul*, para facilitar el desarrollo de ampliaciones del software en distintas plataformas (*IOS*, *Windows Phone*, etc), además de facilitar el mantenimiento del sistema en general.

Pero para que no cualquiera pueda utilizar los servicios *RESTFul*, se pensó en que cuando se haga el inicio de sesión al sistema se proporcione un *token*, para que cada vez que se realice esta acción, se pida el *token* y si este existe, se pueda realizar la función requerida, caso contrario, devuelva un mensaje de error.

* + - 1. **Sistema web**

Se desarrollaron los servicios web, los cuales son los métodos definidos en los controladores, estos devuelven los resultados esperados mediante archivos *JSON,* los cuales son generados por la gema *Responder*.

Para él envió de los archivos *JSON*, los servicios web son métodos *POST*, ya que este método no le permite ver al usuario la información que se está enviando y recibiendo del servidor, dando más de seguridad al sistema.

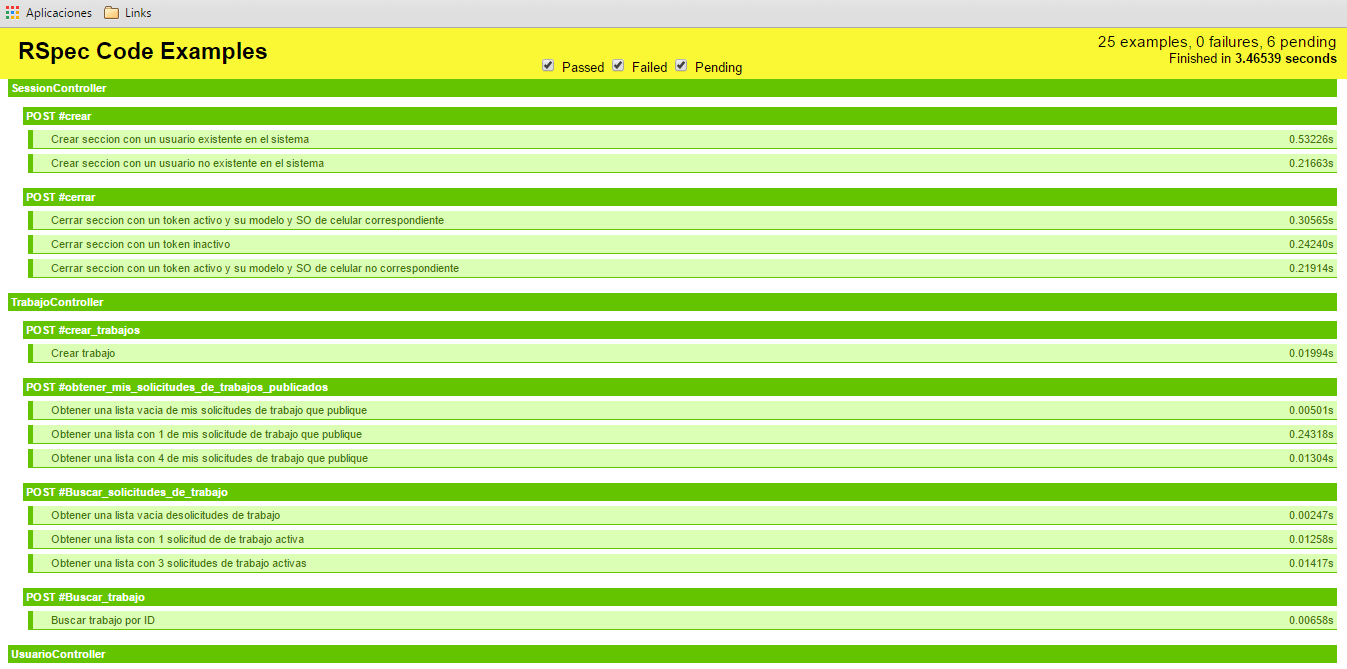
Para la autentificación dentro del sistema web, se usó la gema *Sorcery*. Esta gema implementa la verificación de la existencia de un usuario y el cifrado de su contraseña.

Para la generación del *token*, se decidió usar el CI del usuario, modelo del dispositivo desde donde se accede y su sistema operativo, de los cuales se convierte a *ASCII* todo sus caracteres y se toma la mitad de cada uno para concatenarlos en 1 solo, en caso de que el resultado sea mayor a 100 caracteres, solo se utilizan los 100 primeros caracteres de esta concatenación, pero en caso de ser menos de 100 caracteres, se va a añadir al final del *token* el mismo resultado obtenido, hasta que sea mayor a 100, para después tomar los primeros 100 caracteres del resultado.

A medida que se terminaba de desarrollar los servicios web, se empezó a crear sus respectivos test, para el cual se usó *RSPEC*, una gema de Ruby on Rails que permite la creación y ejecución de *Unit Test* en Ruby, se hizo esto para verificar que devuelva los resultados esperados y poder corregir errores en caso de modificar o ampliar los servicios web.

**Figura 15**

**Resultado de las pruebas del sistema web en la iteración # 1**



Fuente: Elaboración propia

* + - 1. **Aplicación móvil**

Primero se procedió con la configuración del “androidmanifest.xml” para que la aplicación móvil tenga acceso a internet y pueda recibir y enviar información.

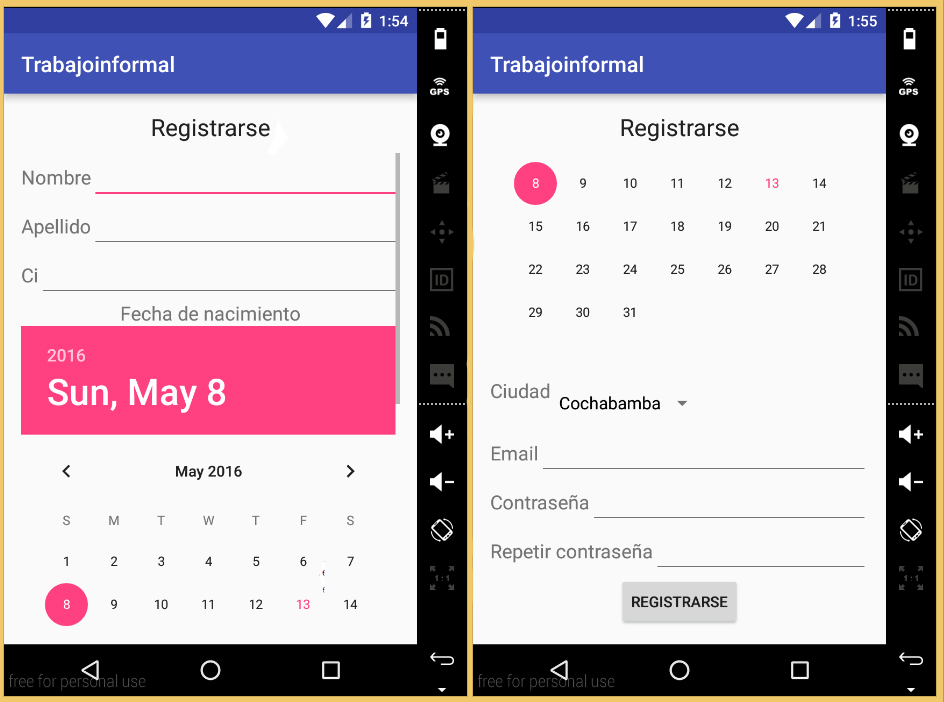
Para que la aplicación pueda usar los servicios *RESTFul*, se implementó el *AsyncTask*, una librería de *Android* que permite ejecutar múltiples tareas en la misma actividad actual, esto permite llamar a los servicios *RESTFul*, ya que no se pueden ejecutar acciones que requieran uso de internet en el hilo principal de la actividad.

El código de implementado de esta tarea, se encuentra en el anexo H.

Después se procedió al desarrollo de los *layouts* de las vistas, tratando de que solo se desarrolle un único *layout* para todos los tamaños de pantalla disponible, además de usar los colores predeterminados que ofrece *Android Studio* cuando se crea la aplicación.

**Figura 16**

**Vista “Registrarse”**



Fuente: Elaboración propia

Para la representación de la información, se decidió usar *ListView*, el cual es un componente para la interfaz de usuario que permite mostrar información en forma de una lista que tiene implementado su propio *scroll*, ya que nos facilitara poder mostrar grandes cantidades de información en cualquier tipo de pantalla de cualquier dispositivo.

Por cuestión de lógica, una vez que el usuario haya iniciado sesión en el sistema, al cerrar la aplicación, el usuario no tiene que volver a ingresar su información para acceder nuevamente al sistema (almenos que se haya cerrado sesión del sistema), para solventar ese problema, se decidió guardar la información del *token* que se genera al iniciar una sesión en el Smartphone, para realizar eso, se usó la librería *SharedPreferences* de *Android*, que permite guardar determinada información de manera permanente en el Smartphone.

**Figura 17**

**Vista principal después de iniciar sesión**



Fuente: Elaboración propia

Aunque el *SharedPreferences* permite guardar información permanentemente en el Smartphone, tener que estar accediendo al archivo constantemente para obtener/guardar información que se obtiene de los servicios web es muy molestoso, para solventar ese problema, se implementó el patrón *Singleton*, para que toda la información que se necesite usar/almacenar mientras la aplicación está en ejecución este siempre disponible,

Cuando se vaya a crear una solicitud de trabajador informal, se tiene que seleccionar el tipo de trabajador que se solicita, pero en vez de poner todos los trabajos disponibles en un *Combobox*, se decidió crear una vista donde estén los diferentes tipos de trabajos que existen en el sistema, considerando que será más atractivo de esa forma.

**Figura 18**

**Vista “Crear solicitud de trabajo informal”**



Fuente: Elaboración propia

A continuación, se dejará algunas figuras mostrando las vistas generadas, las cuales, ya cuentan con el código implementado para su funcionamiento.

**Figura 19**

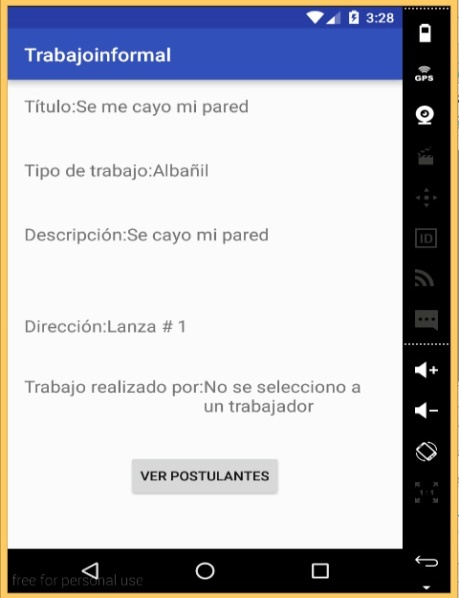
**Vista “Iniciar sesión”**



Fuente: Elaboración propia

**Figura 20**

**Vista “Ver trabajo como creador”**

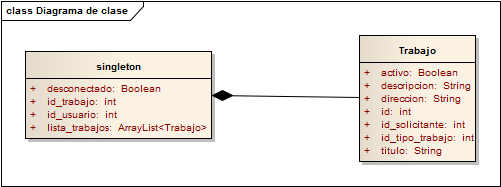


Fuente: Elaboración propia

Después de terminar de programar la aplicación móvil, se procedió a crear los diagrama de clases para la aplicación móvil, donde se mostraran las diferentes clases que se usaron para el funcionamiento de cada aplicación, pero no se agregaran las clases que se extienden de *AppCompatActivity* en caso del diagrama de clases de la aplicación móvil, ya que en estas implementan la lógica que tendrá un determinado *layout*.

**Figura 21**

**Diagrama de clases de la aplicación móvil de la iteración # 1**



Fuente: Elaboración propia

* + 1. **Evaluación**

Pero para tener algo de retroalimentación, se mostró la aplicación a otras personas para que la evalúen, para lo cual, se probó la aplicación con 6 personas al azar, las cuales dieron las siguientes observaciones a considerar para mejorar la aplicación:

* El aspecto visual es malo, se tiene que mejorar.
* Las letras son pequeñas.
* Usar otro tipo de letras.
* El selector de fechas no es amigable, hay que mejorarlo o cambiarlo.
* Hay botones muy grandes y otros pequeños en una misma vista, que sean de tamaño uniforme.
* En iniciar sesión, cuando se escribe el correo electrónico, debería poder desplegarse una lista de correos que iniciaron sesión con el dispositivo, para poder simplificar la escritura del correo.
  1. **Desarrollo del módulo de búsqueda de trabajos informales**
     1. **Planificación**

Se determinaron las siguientes funcionalidades a desarrollar:

* Postularse a la solicitud de trabajo informal.
* Buscar solicitudes de trabajo informal.
* Ver postulaciones a solicitud de trabajo informal.
* Aceptar o rechazar la solicitud de un trabajador informal.
* Ver solicitud de trabajo informal.

Cada requerimiento descrito tiene que desarrollarse tanto para el servicio web, como para la aplicación móvil.

No se considerara la subida de fotos a las solicitudes de trabajadores informales, ya que no son indispensables en esta etapa del desarrollo del proyecto.

* + 1. **Análisis de riesgo**

En base a la experiencia y los problemas que se puede tener, se determinó que la amenaza actual seria la mala implementación de los componentes visuales y funcionales de *Android*, lo cual generaría problemas de ejecución del programa.

Entre esos componentes se tiene:

* *Listview*.
* Uso del *AndroidManifest* para nombrar actividades, que actividad iniciara primero, si tendrá algún estilo visual determinado, etc.
* *ImageView*.

A continuación, se muestra un diagrama de riesgo con las amenazas descritas.

**Tabla 5**

**Análisis de riesgo para el objetivo: “Desarrollo del módulo de búsqueda de trabajos informales”**



Fuente: Elaboración propia

* + 1. **Desarrollo**
       1. **Diseño**

Primero se documentó las historias de usuario que se desarrollarán en esta iteración para recalcar lo que se desarrollará tanto para el servicio web, como para la aplicación móvil.

**Tabla 6**

**Historias de usuario para el objetivo: “Desarrollo del módulo de búsqueda de trabajos informales”**



Fuente: Elaboración propia

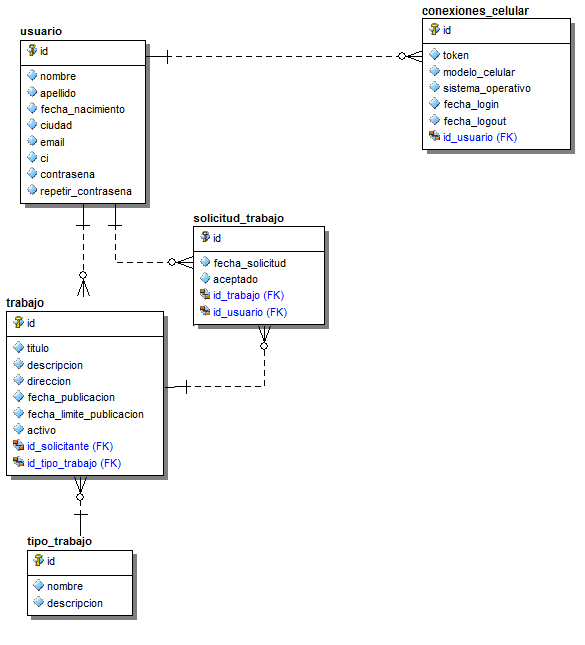
Se procedió a la ampliación de la base de datos, para el cual se determinó que se necesitaría los siguientes datos a guardar para esta iteración:

* ¿Quién se está postulando a un trabajo?
* ¿Qué trabajador fue escogido por el cliente?

En base a esos requerimientos, se amplió la base de datos.

**Figura 22**

**Diagrama del diseño de la base de datos iteración # 2**



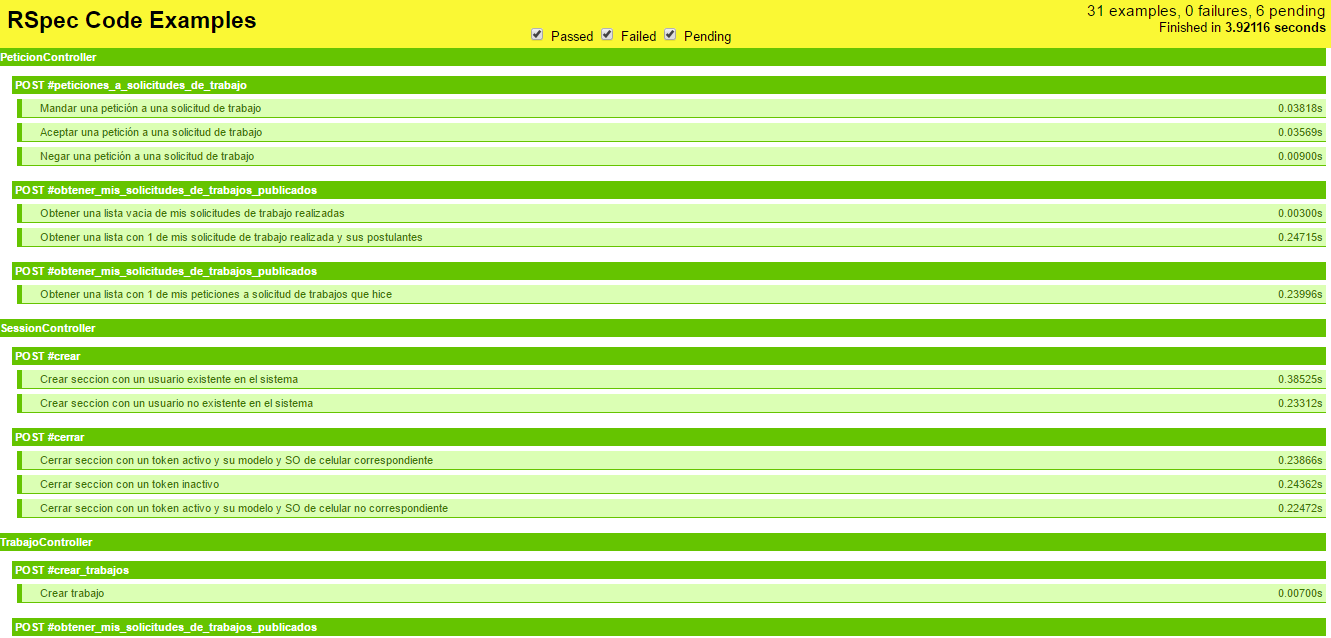
Fuente: Elaboración propia

* + - 1. **Sistema web**

Se procedió a desarrollar los servicios web necesarios, incluyendo sus respectivos *Unit Test* para probar y controlar el correcto funcionamiento de este en caso de modificación o cuando se agreguen nuevos servicios web.

**Figura 23**

**Resultado de las pruebas del sistema web en la iteración # 2**



Fuente: Elaboración propia

* + - 1. **Aplicación móvil**

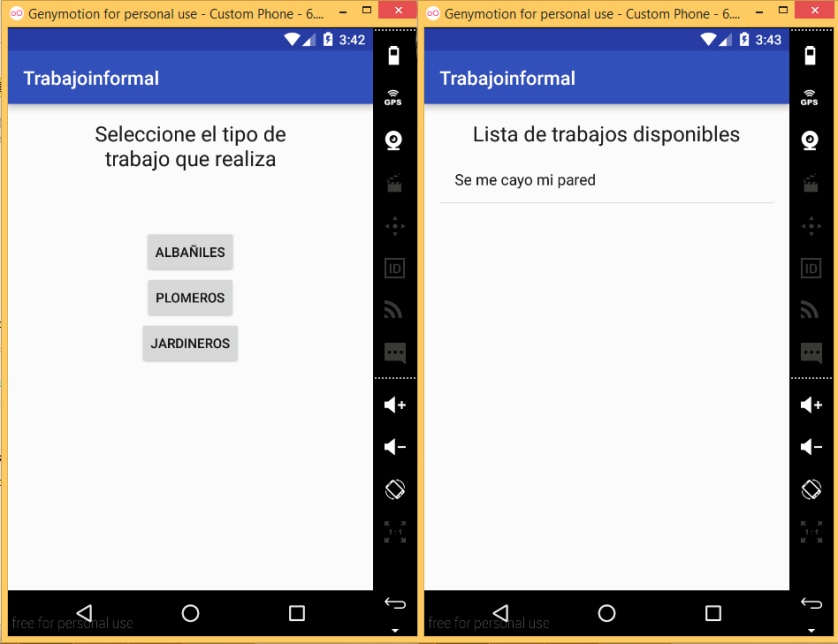
Se procedió al desarrollo de los *layouts* de las diferentes vistas, desarrollando un único *layout* para todos los tamaños de pantalla disponible, además de usar los colores predeterminados que ofrece *Android Studio* cuando se crea el sistema.

Se siguió usando los *Listview* para mostrar la información descritos en la iteración anterior, además de seguir usando el patrón *Singleton* para almacenar la información que se obtiene de los servicios web.

Para la vista donde se implementará las opciones de búsqueda de trabajo, se decidió que se pondrán en botones todos los tipos de trabajo que hay en el sistema y al presionar alguno, debajo de este se podrán ver todos los trabajos disponibles en ese momento.

**Figura 24**

**Vista “Buscar trabajo”**

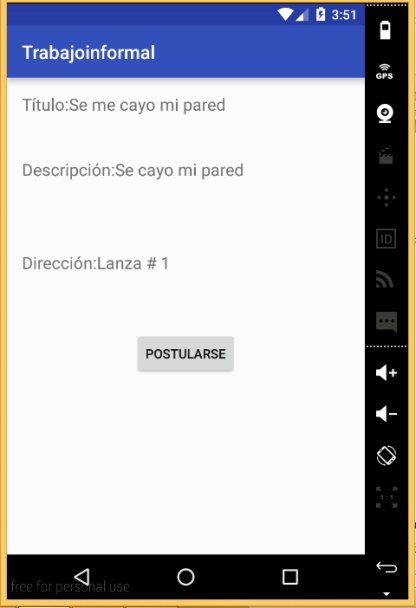


Fuente: Elaboración propia

A continuación, se dejará algunas figuras mostrando las vistas generadas, las cuales, ya cuentan con el código implementado para su funcionamiento.

**Figura 25**

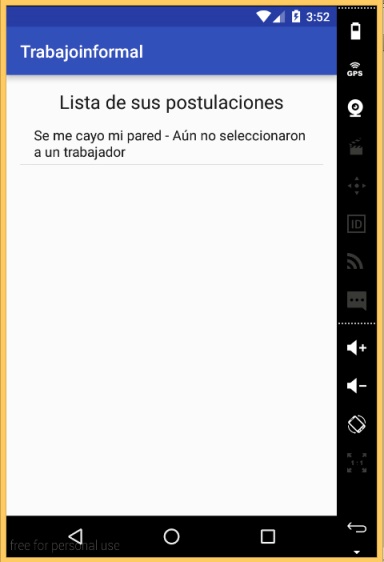
**Vista “Ver trabajo como postulante”**



Fuente: Elaboración propia

**Figura 26**

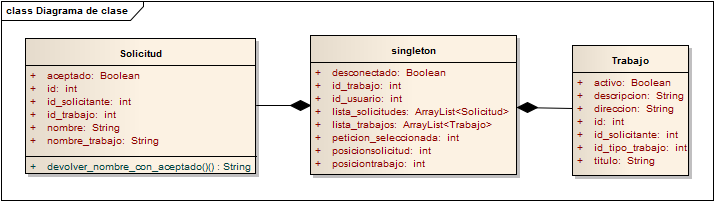
**Vista “Ver mis postulaciones”**



Fuente: Elaboración propia

Después de terminar de programar la aplicación móvil, se procedió a modificar y ampliar el diagrama de clases de la aplicación móvil.

**Figura 27**

**Diagrama de clases de la aplicación móvil de la iteración # 2**

Fuente: Elaboración propia

* + 1. **Evaluación**

Se validó la aplicación móvil en reiteradas oportunidades junto al asesor de este proyecto, tanto en funcionalidad, como en estética.

En la revisión del sistema web, estuvo satisfecho con los test implementados y en la programación en general.

En la revisión de la aplicación móvil, se determinó que se tiene que seguir más convenios de java para desarrollar, además de que no estuvo de acuerdo con el aspecto visual que se estaba manejando en la aplicación móvil, ya que era muy básica su presentación y tiene mucha referencia a una aplicación de escritorio, lo cual rompe con los diseños de aplicaciones móvil e existentes en la actualidad

Ya que el aspecto visual de la aplicación móvil es mala y considerando las opiniones de las personas que probaron la aplicación en la iteración # 1, se tuvo sugerencias de cómo se tiene que desarrollar el aspecto visual de las aplicaciones móviles, se acordó también de que se va a rehacer el aspecto visual de la aplicación móvil para las futuras iteraciones de este proyecto.

* 1. **Desarrollo de módulos de gestión de perfil, comentarios y puntuación de trabajadores informales**
     1. **Planificación**

Para el inicio del desarrollo, primero se tiene que rehacer el aspecto visual desde cero, ya que el aspecto actual es malo.

Para evitar tener múltiples archivos en un mismo proyecto y errores de adaptación de la interfaz antigua a la nueva, se decidió rehacer todo el proyecto, reutilizando todo el código creado.

Después de rehacer la aplicación móvil, se vio los requerimientos mínimos que se pueden obtener del perfil de este proyecto para determinar las funcionalidades a desarrollar, para lo cual, se pudo determinar las siguientes funcionalidades a desarrollar:

* Modificar información del usuario.
* Agregar imagen al perfil del usuario.
* Agregar imagen a la solicitud de trabajo.
* Buscar trabajadores informales.
* ¿Qué tipo de trabajos realizo?
* Comentar a trabajador informal.
* Dar una puntuación a un trabajador informal.

Cada requerimiento descrito tiene que desarrollarse tanto para el servicio web, como para la aplicación móvil.

Considerando de que almacenar varias fotos en el mismo servidor web puede llegar a saturarlo, se optó por que las imágenes que se fueran a subir al sistema, se hagan en un host de imágenes, para el cual, se optó por *Cloudinary*, el cual, tiene librerías externas a implementar para subir imágenes a su servidor.

Y para evitar tener que escribir código que descargue imágenes, se optó por usar herramientas de terceros llamada *Picasso*.

* + 1. **Análisis de riesgo**

Considerando que se volverá a rehacer la aplicación móvil, se determinaron las nuevas amenazas para el proyecto

* No saber implementar la librería de *Cloudinary* para subir archivos a internet.
* No saber usar la librería de Picasso para descargar imágenes de internet.
* Falta de conocimientos en la implementación de los componentes visuales y funcionales para desarrollar la aplicación nuevamente, tales como:
  + Fragmentos.
  + Menús desplegables.
  + *Scrollview*.
  + Botones flotantes.

En cuestión del aspecto visual, se usara el modelo acordado con el tutor de este proyecto de grado y de seguir las interfaces visuales de *Material Desing* de Google, las cuales, ya están implementadas en *Android Studio*.

A continuación, se muestra un diagrama de riesgo con las amenazas descritas.

**Tabla 7**

**Análisis de riesgo para el objetivo: “Desarrollo de módulos de gestión de perfil, comentarios y puntuación de trabajadores informales”**



Fuente: Elaboración propia

* + 1. **Desarrollo**
       1. **Diseño**

Primero se documentó las historias de usuario que se desarrollarán en esta iteración para recalcar lo que se desarrollará tanto para el servicio web, como para la aplicación móvil.

**Tabla 8**

**Historias de usuario para el objetivo: “Desarrollo de módulos de gestión de perfil, comentarios y puntuación de trabajadores informales”**



Fuente: Elaboración propia

Se procedió al análisis del diseño de la base de datos, para el cual se determinó que se necesitaría los siguientes datos a guardar para esta iteración:

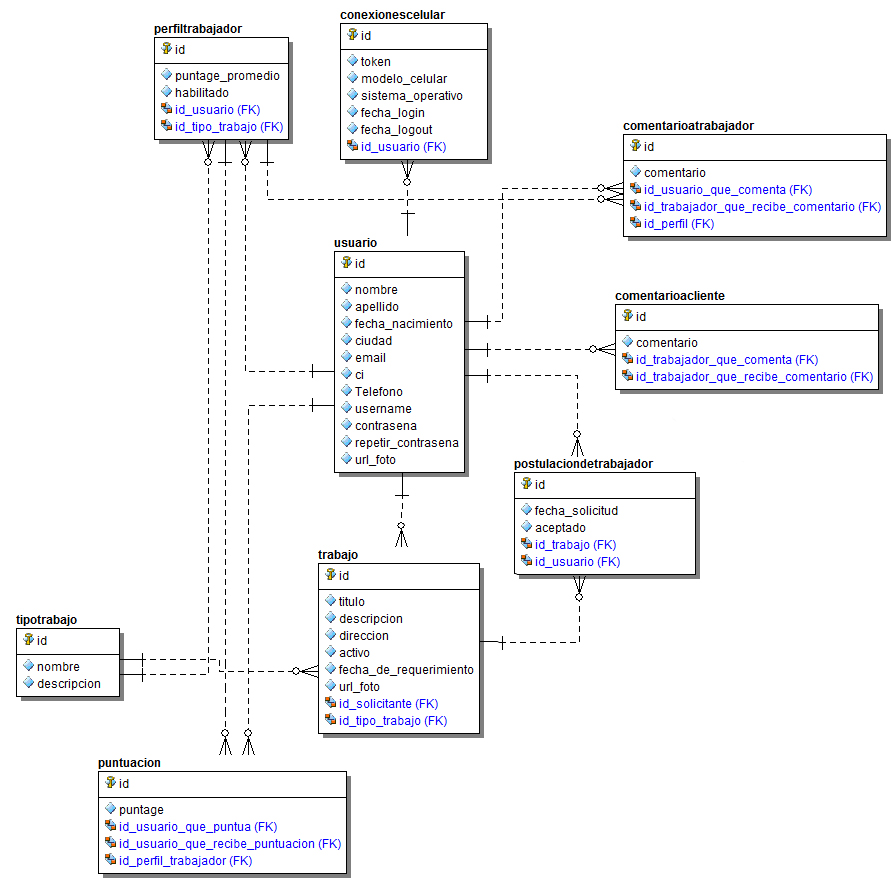
* Un perfil de un tipo de trabajo disponible en el sistema.
* Comentarios que recibió un usuario por perfil de trabajo que tenga.
* Las puntuaciones que se le hicieron por perfil de trabajo que tenga.
* *URL* de la foto de un usuario o de una solicitud de trabajo.
* La solicitud de trabajador informal (ubicación, título, descripción, fecha donde se requiere al trabajador, etc).

Se decidió que los comentarios que se pueden hacer a un trabajador sean de acuerdo al tipo de trabajo que realizan, ya que puede ser que sea malo en un tipo de trabajo, pero en otro no y que los comentarios negativos de donde es malo, afecte en donde es bueno.

Se decidió guardar las puntuaciones que se realizan al perfil del tipo de trabajo que realiza un trabajador, para conocer quien da la puntuación y mostrar al trabajador si es necesario, pero el promedio de puntaje que se le asigno se mantendrá en su perfil del tipo de trabajo, para agilizar el procesamiento del sistema.

**Figura 28**

**Diagrama del diseño de la base de datos de la iteración # 3**



Fuente: Elaboración propia

* + - 1. **Sistema web**

Se procedió a desarrollar los servicios web necesarios, incluyendo sus respectivas pruebas para probar y controlar su funcionalidad en caso de modificación o cuando se agreguen nuevos servicios web.

**Figura 29**

**Resultado de las pruebas del sistema web en la iteración # 3**



Fuente: Elaboración propia

* + - 1. **Aplicación móvil**

Primero se rehízo la aplicación móvil, considerando usar las normativas de diseño de interfaces de aplicaciones móvil de Google *Material Desing*, ya que estas están siendo implementadas en la mayoría de aplicaciones móviles existentes.

Se decidió cambiar los esquemas de colores de la aplicación, evitando usar las predeterminadas que ofrece *Android Studio* por ser muy oscuras, para lo cual, el esquema de colores que se usara serán verdes claros.

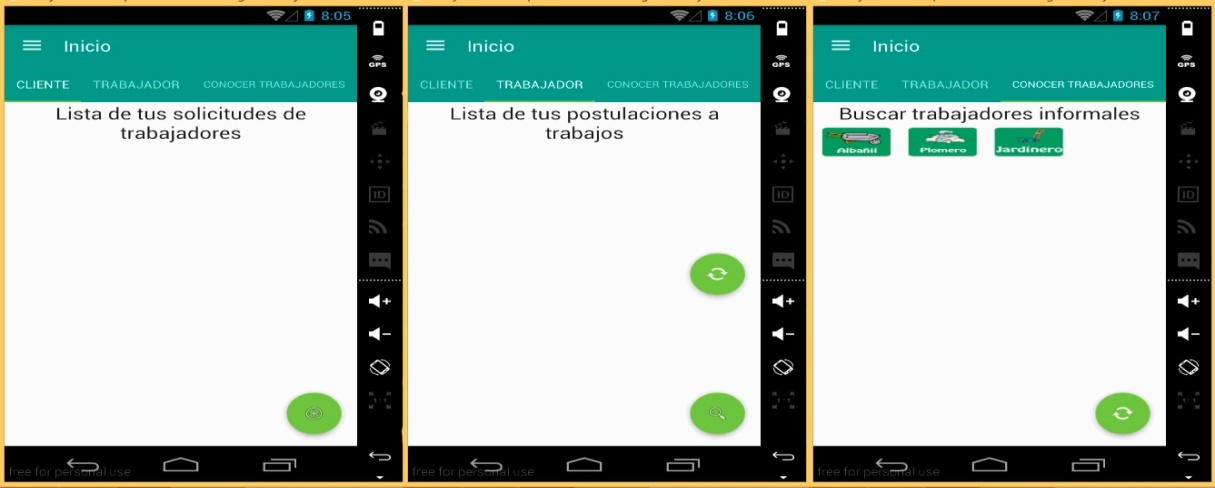
Se crearon imágenes de los diferentes tipos de trabajo disponible, para que el botón parezca más una imagen y sea más atractivo.

Para la pantalla inicial, una vez que el usuario se haya iniciado sesión, la pantalla principal tendrá una barra extra en la parte superior, la cual estará dividida en 3 palabras, las cuales al pulsarlas, llevaran a una vista diferente, cada una independiente de la otra.

Se hizo esto para dar un aspecto más simple, con menos botones, además de que tenga una similitud con la aplicación *Whatsapp* y sea más intuitivo el uso de la aplicación para los usuarios finales.

**Figura 30**

**Pantallas y división que existe en la pantalla de inicio después de iniciar sesión**



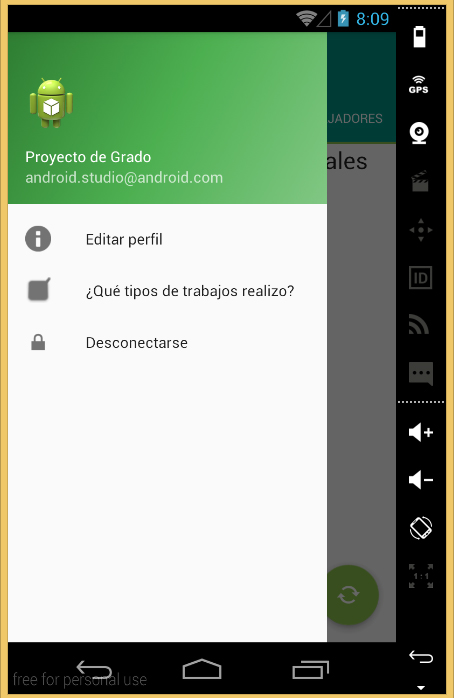
Fuente: Elaboración propia

Para no tener que usar botones comunes en la mayoría de vistas, se decidió usar botones flotantes con una imagen lo más representativa posible de lo que realizara ese botón al pulsarlo.

Solo en la pantalla principal de inicio después de iniciar sesión, existirá un botón en la esquina izquierda superior, el cual abrirá el menú principal de la aplicación, que tendrá diferentes opciones o funcionalidades extra que no encajan en la pantalla principal, para dar más orden a la aplicación, como el desconectarse de la aplicación, editar perfil, etc.

**Figura 31**

**Menú principal de la pantalla principal después de iniciar sesión**

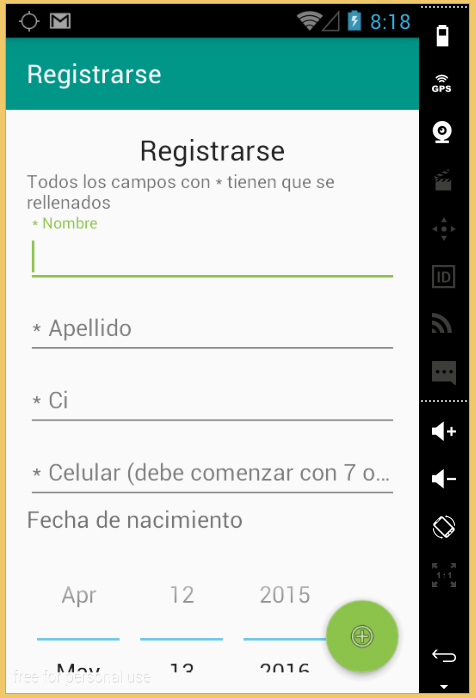


Fuente: Elaboración propia

Para terminar, se cambiaron varias de las interfaces de usuario disponibles, haciendo que la aplicación sea más atractiva para su uso.

**Figura 32**

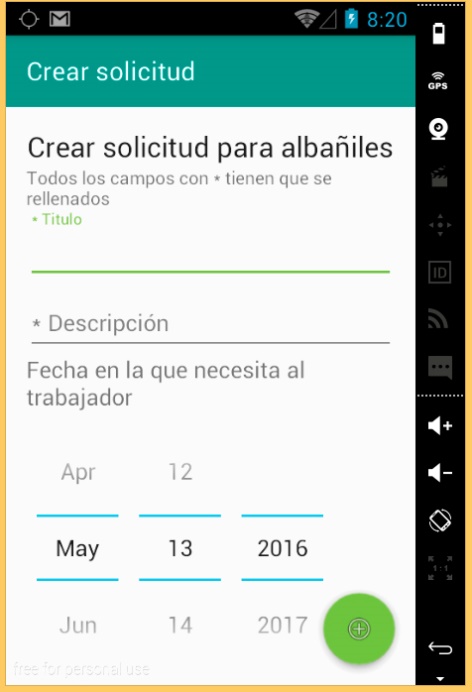
**Vista “Registrarse”**



Fuente: Elaboración propia

**Figura 33**

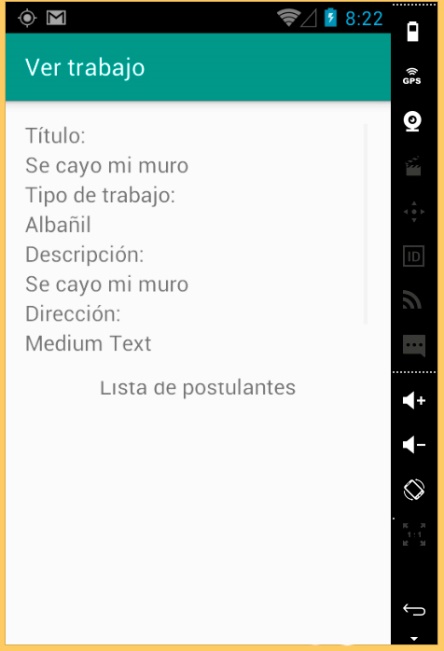
**Vista “Crear solicitud de trabajador”**



Fuente: Elaboración propia

**Figura 34**

**Vista “Ver solicitud de trabajador como creador”**



Fuente: Elaboración propia

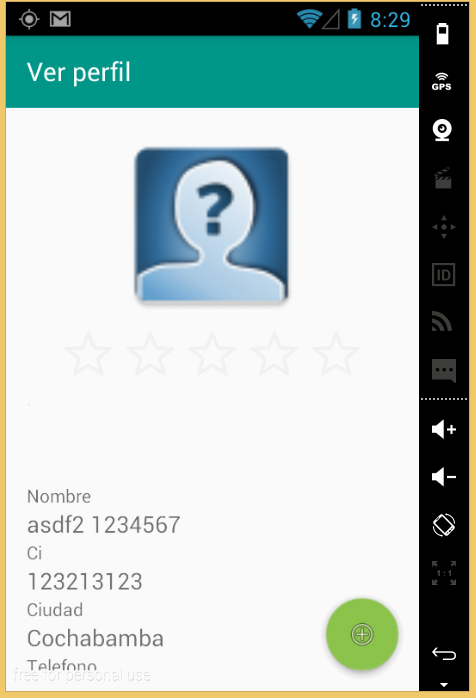
Después de rehacer la aplicación móvil, se procedió al desarrollo normal de la aplicación, para lo cual, se desarrolló los *layouts* de las diferentes vistas, tratando de que solo se desarrolle un único *layout* para todos los tamaños de pantalla disponible.

Para dar calificación a los diferentes trabajadores, se usó la interfaz de usuario *RatingBar*, el cual implementa estrellas visualmente y el control de puntaje de este.

Para los puntajes, se decidió usar un rango de calificación de 1-5 estrellas, ya que es más fácil de puntuar para el usuario.

**Figura 35**

**Vista “ver perfil de trabajador”**

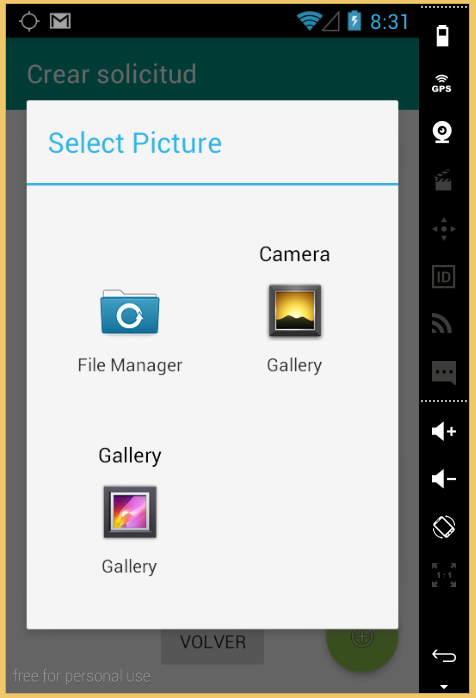


Fuente: Elaboración propia

Para la búsqueda de imágenes dentro del dispositivo, se usó el recurso “Intent.ACTION\_GET\_CONTENT” de la clase *Intent* de *Android*, el cual permite abrir ventanas nuevas en la misma actividad, sin necesidad de cerrarla, para el cual, se la configuro para que pueda buscar imágenes en el dispositivo, ya sea que este en su galería de imágenes o que lo busque manualmente en todos los directorios del dispositivo.

**Figura 36**

**Buscador de imágenes**

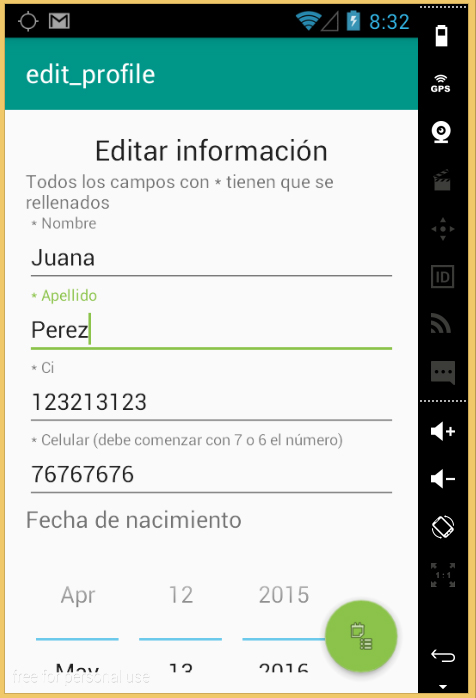


Fuente: Elaboración propia

Para la edición de perfil, se decidió hacer una copia casi exacta de lo que es la vista registrarse, solo que se le agregara un botón y la interfaz para buscar imágenes dentro del dispositivo.

**Figura 37**

**Vista “Editar perfil de trabajador”**



Fuente: Elaboración propia

Se implementó el mismo botón y la interfaz para buscar imágenes dentro del dispositivo cuando se va a crear una solicitud de trabajador informal.

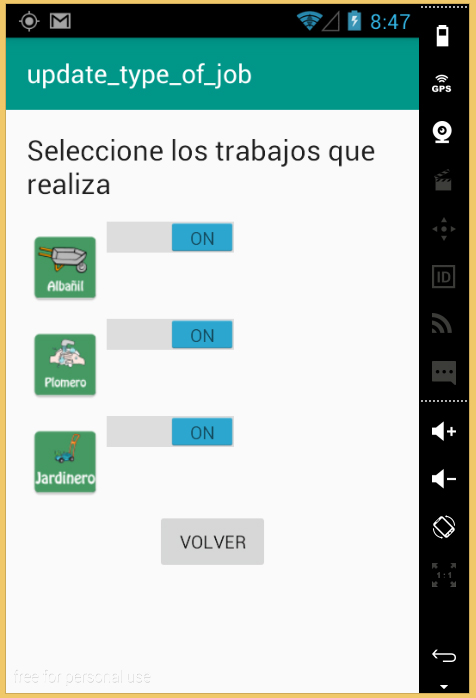
Para que se puedan subir imágenes al host de imágenes *Cloudinary*, se implementó su API que distribuye de forma gratuita, de la cual, permite obtener el *URL* de la foto para su visualización, evitando que se suba imágenes al servidor web.

Para visualizar las imágenes que se subieron a *Cloudinary*, se decidió usar Picasso, una herramienta de terceros gratuita que permite descargar imágenes de internet, pudiendo redefinir sus dimensiones antes de insertar la imagen descargada en un *ImageView*.

A continuación, se dejara algunas figuras mostrando los *layouts* generados, las cuales, ya cuentan con el código implementado para su funcionamiento.

**Figura 38**

**Vista “Seleccionar tipo de trabajos que hago”**

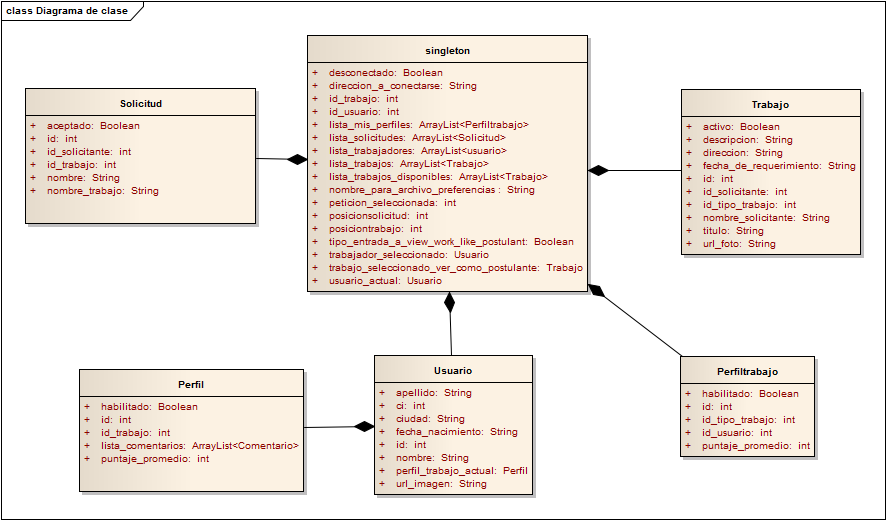


Fuente: Elaboración propia

Después de terminar de programar la aplicación móvil, se procedió a modificar y ampliar los diagrama de clases de la aplicación móvil, pero no se agregaran las clases que se extienden de *AppCompatActivity* en caso del diagrama de clases de la aplicación móvil, ya que en estas implementan la lógica que tendrá un determinado *layout*.

**Figura 39**

**Diagrama de clases de la aplicación móvil de la iteración # 3**



Fuente: Elaboración propia

* + 1. **Evaluación**

Se validó la aplicación concluida por parte del Asesor del proyecto, tanto en funcionalidad, como en estética, se aceptó el nuevo aspecto de la aplicación, pero tuvo las siguientes observaciones para corregir o aumentar:

* Cuando se busca un trabajo, tiene que haber un *textbox* que permita al escribir, poder filtrar todas las propuestas de acuerdo a que el título del trabajo coincida con lo que se escriba en el *textbox*.
* Cuando se busca un trabajo, los resultados tienen que ser en la ciudad que el trabajador este actualmente.
* Usar otro tipo de iconos en el menú principal.
* Las postulaciones a una oferta de trabajo debería tener un precio medio por el cual el trabajador realizara el trabajo.
* En la pantalla donde se ve el trabajo creado, que este toda la información para mostrar en la misma vista (información del trabajo, postulantes en caso de ser el que lo creo y la futura implementación del mapa) puede llegar a confundir al usuario por la gran cantidad de información que se muestra, se recomienda dividir toda la información como la pantalla principal después de iniciar sesión.
* En la pantalla donde se ve el perfil del trabajador, que este toda la información para mostrar en la misma vista (información del trabajador y sus comentarios) puede llegar a confundir al usuario por la gran cantidad de información que se muestra, se recomienda dividir toda la información como la pantalla principal después de iniciar sesión.
* Se tiene que poder agregar N tipos de trabajos de manera automática a la aplicación, sin necesidad de tener que estar actualizando la aplicación para agregar diferentes tipos de trabajos.
  1. **Implementación de API para el uso de mapas en la aplicación móvil.**
     1. **Planificación**

Para esta iteración, lo principal que se tiene que desarrollar es la implementación de mapas en la aplicación web, siendo la opción más viable usar *Google Maps*, porque *Android* tiene soporte y compatibilidad para *Google Maps*, ya que ambas fueron desarrolladas por Google.

Pero además, se tiene que hacer las correcciones descritas en la evaluación de la iteración anterior y terminar de implementar características que no se implementaron en las anteriores.

* + 1. **Análisis de riesgo**

La amenaza más grande que se tiene para el proyecto es la implementación de la *API* de *Google Maps* en la aplicación móvil, para el cual, se hizo su diagrama de riesgo.

**Tabla 9**

**Análisis de riesgo para el objetivo: “Implementación de *API* para el uso de mapas en la aplicación móvil”**



Fuente: Elaboración propia

* + 1. **Desarrollo**
       1. **Diseño**

Antes de desarrollar el software, primero se documentó las historias de usuario que se desarrollarán en esta iteración para recalcar lo que se desarrollará tanto para el servicio web, como para la aplicación móvil.

**Tabla 10**

**Historias de usuario para el objetivo: “Implementación de *API* para el uso de mapas en la aplicación móvil”**



Fuente: Elaboración propia

Primero se procedió a la ampliación de la base de datos, ya que solo se necesita realizar la ampliación de la información existente para que se pueda almacenar ubicaciones para *Google Maps*.

*Google Maps* usa latitud y longitud, pero ya que son términos poco conocidos, se usara “coordenadax” para latitud y “coordenaday” para longitud.

**Figura 40**

**Diagrama del diseño de la base de datos de la iteración # 4**



Fuente: Elaboración propia

* + - 1. **Sistema web**

Solo se actualizaron los servicios web existentes, ya que no hay necesidad de crear más servicios en esta iteración.

* + - 1. **Aplicación móvil**

Para el desarrollo de la aplicación móvil, se descargó las librerías necesarias para que *Android Studio* pueda usar la *API* de *Google Maps*, después se realizó el registro en Google para poder obtener su servicio de *Google Maps*, para obtener una *API key* que nos permitirá acceder a los servicios de *Google Maps*.

Siguiendo las observaciones sobre la visualización de la información cuando se crea una solicitud del trabajador, se consideró hacer un diseño similar al de la pantalla principal después de iniciar sesión, siguiendo la siguiente división:

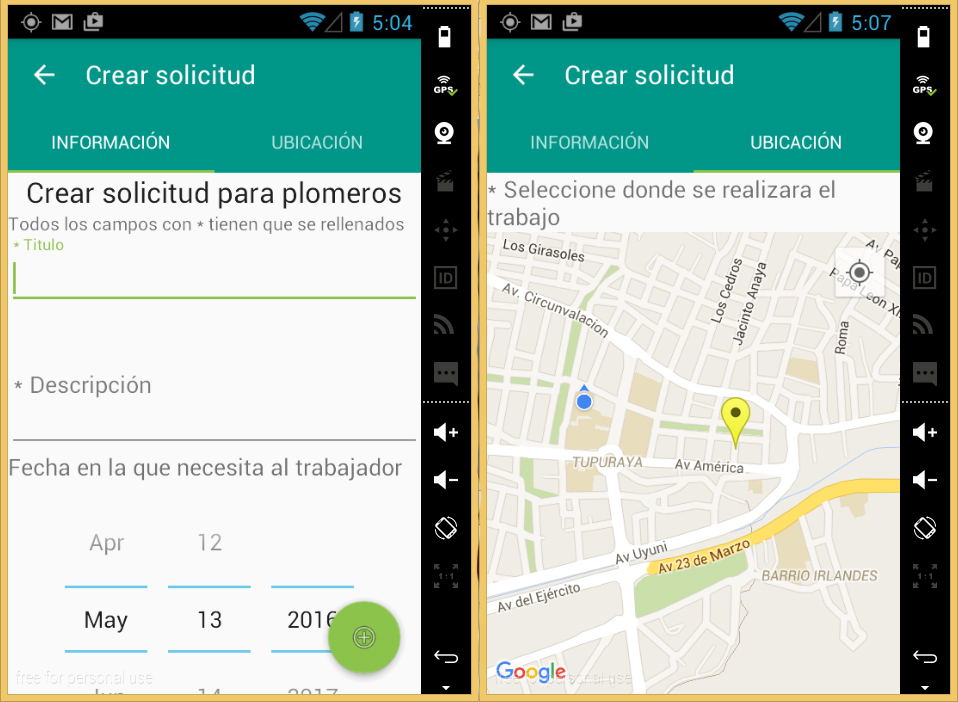
* Información de la solicitud a realizar.
* Mapa para señalar la ubicación de donde se realizara el trabajo.

Cuando se acceda al mapa, este enfocara en donde esté ubicado el usuario en el mapa, dando facilidad a este para ubicar donde se debe realizar el trabajo que se solicitara, pero para que esto funcione, se necesita que el *GPS* este activado, ya que si no, mandara al mapa mundial.

Se pondrá el botón para enfocar directamente en la ubicación actual del usuario, pero esta función depende del *GPS* para que pueda funcionar.

**Figura 41**

**Vista “Crear solicitud de trabajador”**



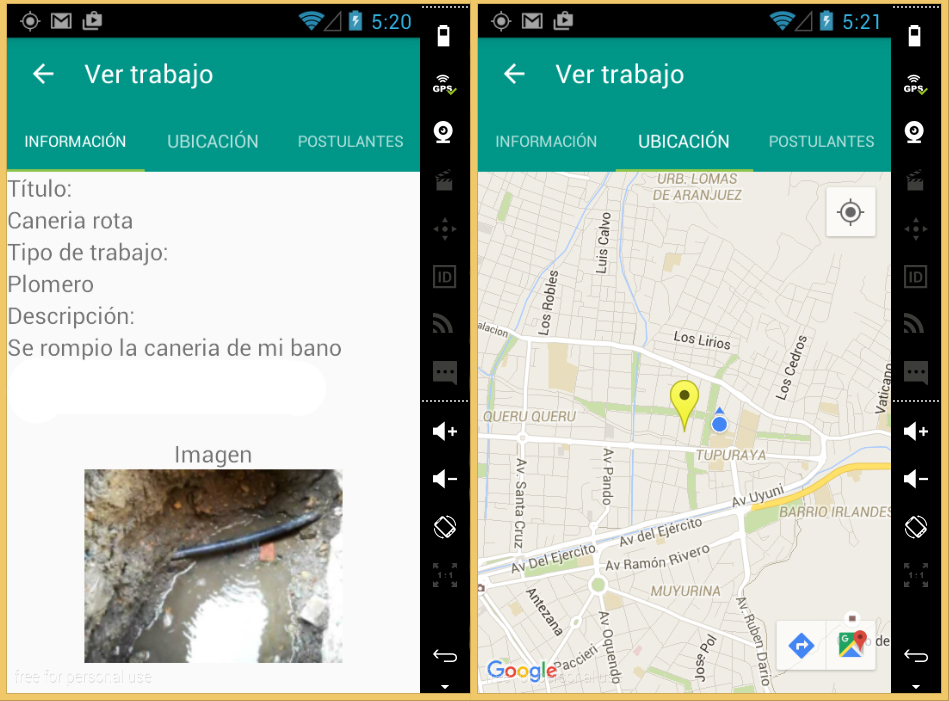
Fuente: Elaboración propia

El diseño de la pantalla principal después de iniciar sesión también fue usado para reestructura la vista de la solicitud de trabajador, para el cual se siguió el siguiente criterio de división:

* Información del trabajo.
* Mapa para señalar la ubicación de donde se realizara el trabajo.
* Lista de los postulantes a la solicitud (solo aparecerá esto si eres el creador de la solicitud).

**Figura 42**

**Vista “Ver solicitud de trabajador”**



Fuente: Elaboración propia

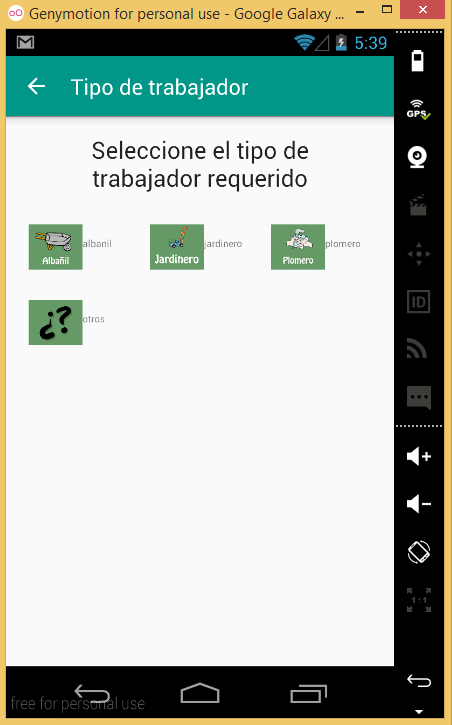
Para poder mostrar que la aplicación se pueda mostrar varios tipos de trabajos informales sin necesidad de tener que estar actualizando la aplicación, se creó una clase llamada *CustomGrid*, el cual es una extensión de la clase *BaseAdapter*, el cual permite crear componentes que se adaptan a los diferentes componentes visuales que tiene *Android*.

Esta clase obtendrá los identificadores de las diferentes imágenes que contenga la aplicación y la asociara a un texto determinado, con lo descrito anteriormente, se usara toda la información contenida en la tabla “Tipotrabajo” de la base de datos del servidor web para rellenar de información a la clase *CustomGrid* y se la ingresara en un *GridView*, permitiéndole mostrar múltiples imágenes en una vista y que esta pueda agrupar en filas de forma automática las imágenes que se carguen.

Una limitante al implementar este código, es que se tiene que tener las imágenes a mostrar si o si cargadas en la aplicación, para lo cual, si se agregan más tipos de trabajo, se les asociara a una imagen default si es que no encuentra su representación visual en la aplicación.

**Figura 43**

**Lista de los trabajos informales disponibles**

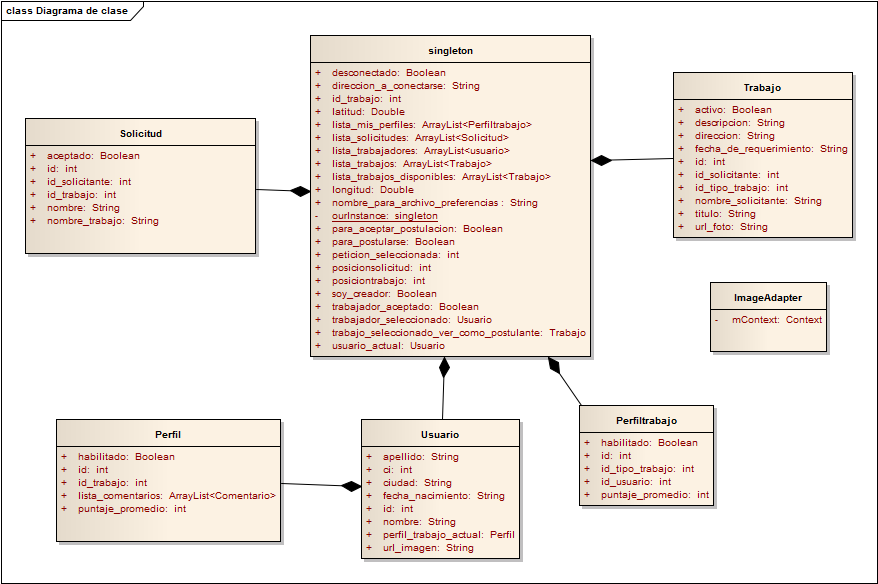


Fuente: Elaboración propia

Después de terminar de programar la aplicación móvil, se procedió a modificar y ampliar los diagrama de clases de la aplicación móvil, pero no se agregaran las clases que se extienden de *AppCompatActivity* en caso del diagrama de clases de la aplicación móvil, ya que en estas implementan la lógica que tendrá un determinado *layout*.

**Figura 44**

**Diagrama de clases de la aplicación móvil de la iteración # 4**



Fuente: Elaboración propia

Tras finalizar el desarrollo, se empezó a corregir los detalles mencionados en la iteración anterior, además de completar algunos aspectos faltantes en la aplicación.

* + 1. **Evaluación**

Se validó los ajustes, tanto en funcionalidad, como en estética, para finalmente ser aprobada la aplicación.

**Conclusiones**

Este proyecto tiene alto grado de escalabilidad, ya que al usar *RESTful*, se puede desarrollar esta aplicación en diferentes plataformas y solo se tienen que usar los servicios ya implementados, ahorrando tiempo de desarrollo y de ampliación y mantenimiento de los programas que usan estos servicios, descritos en anexo C.

Aunque desarrollar aplicaciones nativas es una buena idea de desarrollo de aplicaciones, el tiempo que llevaría para poder exportar el proyecto a otra plataforma es excesivo, por lo cual, se determinó que las herramientas de desarrollo para múltiples plataformas son útiles en caso de que se tenga que difundir una aplicación en múltiples plataformas.

No obstante, realizar páginas web que tengan soporte a resolución de dispositivos móviles, es una buena opción de desarrollo, pero solo se debe realizar esto si es que no se va a necesitar usar los componentes del dispositivo, tales como el *GPS*, *bluetooth*, cámara, etc.

El modelo en espiral como metodología de desarrollo de software sigue siendo una opción viable, aunque es burocrático en ciertos momentos del desarrollo tener que evaluar y documentar los posibles riesgos del proyecto.

Los servicios web son una herramienta muy poderosa, tanto para hacer escalable un proyecto de sistemas informáticos, como para la ayuda en el procesamiento y guardado de información, siendo este el presente y futuro del desarrollo de software y de cualquier tipo de sistema.

No se realizó la investigación de mercado, por lo cual, se lo deja como marco de referencia para que los estudiantes e interesados algún proyecto similar a este, puedan tomar de referencia de este proyecto, por tanto este proyecto estará disponible en internet bajo la licencia de software libre.

**Recomendaciones**

Al manejar un servidor web donde se procesa la información que se envía, en el mismo servidor se puede desarrollar una página web que realice las mismas características que este proyecto, dando nuevas formas a las personas de poder buscar trabajo o trabajador informal.

Como el proyecto solo se desarrolló para *Android*, se sugiere que se implemente esta aplicación móvil para *IOS* y *Windows Phone*, ya que al tener desarrollado los servicios web, solo se tiene que desarrollar ese aspecto visual y el cómo procesara la información que se obtenga de los servicios web.

Una mejor forma de poder evaluar a los trabajadores informales sea por una especialidad del trabajo que realiza, ya que este puede ser bueno en ciertos aspectos de su trabajo, pero en otros no, citando un ejemplo tenemos en jardinería las especialidades de cortar pasto, podar, plantar, etc.

Para facilitar la implementación de lo descrito anteriormente, se dejó un diseño de base de datos en el anexo F, el cual muestra la base de datos a usar para implementar las especialidades en el sistema web, junto con su diccionario de datos en el anexo G.

Otra mejora que se puede implementar, es la de poder abrir el *Google maps* y que aparezcan todos los trabajadores disponibles que estén cerca del cliente, para poder comunicarme directamente con ellos, sin necesidad de publicar una solicitud de trabajador.

El proyecto se encuentra en el siguiente repositorio:

https://github.com/sergioarispejulio/trabajosinformalesapp

**Bibliografía**

* ANÓNIMO (s.f.). " Qué es SCRUM", En:

https://proyectosagiles.org/que-es-scrum/, (fecha de consulta 20/08/2015)

* ANÓNIMO (s.f.). " Significado de Empleo informal ", En:

http://www.significados.com/empleo-informal/, (fecha de consulta 20/08/2015)

* ANÓNIMO (05/10/2015). "Modelo espiral", En:

http://modeloespiral.blogspot.com/, (fecha de consulta 20/12/2015)

* ANÓNIMO (10/12/2015). " Java (lenguaje de programación)", En:

https://es.wikipedia.org/wiki/Java\_(lenguaje\_de\_programaci%C3%B3n), (fecha de consulta 20/12/2015)

* ANÓNIMO (01/03/2009). "El Estado fomenta el empleo eventual y así vulnera la ley laboral", En:

http://industriabolivia.blogspot.com/2009/03/el-estado-fomenta-el-empleo-eventual-y.html, (fecha de consulta 22/08/2015)

* ANÓNIMO (01/12/2015.). " Representational State Transfer", En:

https://es.wikipedia.org/wiki/Representational\_State\_Transfer, (fecha de consulta 15/12/2015)

* ANÓNIMO (s.f.). " RESTful Web Services - Introduction", En:

http://www.tutorialspoint.com/restful/restful\_introduction.htm, (fecha de consulta 15/12/2015)

* ANÓNIMO (28/12/2015). " Android", En:

https://es.wikipedia.org/wiki/Android, (fecha de consulta 05/01/2016)

* Baez, Sergio (20/10/2012). " Sistemas Web", En:

http://www.knowdo.org/knowledge/39-sistemas-web, (fecha de consulta 12/07/2016)

* Bel Puchol, Antonio (26/01/2001). " Apuntes Lenguaje Java", En:

http://abelp.arrakis.es/ApuntesJava/Introduccion.htm, (fecha de consulta 30/08/2015)

* Belmonte Fernández, Oscar (06/06/2005). "En Bolivia el uso de internet en celulares casi se triplica en un año", En:

http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf, (fecha de consulta 30/08/2015)

* Cendejas Valdéz, José Luis (s.f.). " Modelos y metodologías para el desarrollo de software", En:

http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm, (fecha de consulta 10/09/2015)

* Esquiva Rodríguez, Alejandro (13/10/2013). " JSON I – ¿Qué es y para qué sirve JSON?", En:

https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/, (fecha de consulta 12/07/2016)

* Erb, Markus (s.f.). " Matriz para el Análisis de Riesgo", En:

https://protejete.wordpress.com/gdr\_principal/matriz\_riesgo/, (fecha de consulta 25/05/2016)

* Fernández, Alberto (12/11/2013). "Servicios web RESTful con HTTP. Parte I: Introducción y bases téoricas", En:

http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas, (fecha de consulta 05/04/2016)

* framerper1 (02/12/2014). " iOS y Android: rentabilidad contra cuota de mercado, en un gráfico", En:

http://hipertextual.com/archivo/2014/12/ios-y-android-rentabilidad-contra-cuota-mercado-grafico/, (fecha de consulta 20/09/2015)

* Fundación Milenio (s.f.). "Coy 217 - El problema del empleo en Bolivia", En:

http://www.fundacion-milenio.org/Informe-Nacional-de-Coyuntura/coy-217-el-problema-del-empleo-en-bolivia.html, (fecha de consulta 22/08/2015)

* Lamarca Lapuente, María Jesús (08/12/2013.). "Servicios Web", En:

http://www.hipertexto.info/documentos/serv\_web.htm, (fecha de consulta 25/11/2015)

* Luna Acevedo, Ismael (15/02/2014). "Bolivia con nivel más alto de empleo informal juvenil", En:

http://eldia.com.bo/index.php?cat=357&pla=3&id\_articulo=138697, (fecha de consulta 20/08/2015)

* Luna Acevedo, Ismael (21/04/2014). " La barrera infranqueable del empleo", En:

http://www.eldia.com.bo/index.php?cat=357&pla=3&id\_articulo=143546, (fecha de consulta 22/08/2015)

* Mattassi, Joshua (20/08/2015). " Android sigue dominando el mercado de smartphones", En:

http://www.fundacion-milenio.org/Informe-Nacional-de-Coyuntura/coy-217-el-problema-del-empleo-en-bolivia.html, (fecha de consulta 22/08/2015)

* Mendoza, Luz (16/11/2014). "En Bolivia el uso de internet en celulares casi se triplica en un año", En:

http://eju.tv/2014/11/en-bolivia-el-uso-de-internet-en-celulares-casi-se-triplica-en-un-ao/, (fecha de consulta 30/08/2015)

* Moreno, Gerardo Martínez (s.f.). " Ingenieria de Software UML", En:

http://www.monografias.com/trabajos5/insof/insof.shtml, (fecha de consulta 20/02/2016)

* Moya, Ricardo (21/12/2015). " Patrón Singleton en Java, con ejemplos", En:

http://jarroba.com/patron-singleton-en-java-con-ejemplos/, (fecha de consulta 20/04/2016)

* Navarro Marset, Rafael (01/03/2015). "REST vs Web Services", En:

http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf, (fecha de consulta 20/11/2015)

* Netdreams (18/08/2014). " Smartphones en el segundo trimestre del 2014: Android sigue creciendo, Blackberry se hunde y en Perú crece con fuerza Firefox OS", En:

http://netdreams.pe/blog/mundo-movil/smartphones-en-el-segundo-trimestre-del-2014-android-sigue-creciendo-blackberry-se-hunde-y-en-peru-crece-con-fuerza-firefox-os/, (fecha de consulta 07/01/2016)

* Nolasco, Jorge (2014). Desarrollo de aplicaciones móviles con Android. Lima - Peru: Editorial Macro.
* Pereyra, Jiménez Pereyra Jiménez (08/11/2009). "Más mujeres forman parte de la economía informal", En:

http://www.lostiempos.com/diario/actualidad/economia/20091108/mas-mujeres-forman-parte-de-la-economia-informal\_44258\_76174.html, (fecha de consulta 20/08/2015)

* Pinto, Maria (15/10/2004). "Base de datos", En:

http://www.mariapinto.es/e-coms/bases-de-datos/, (fecha de consulta 10/11/2015)

* Roth, Ronica (16/11/2010). "¿Cómo funciona el sistema web?", En:

http://altenwald.org/2010/11/16/como-funciona-el-sistema-web/, (fecha de consulta 12/07/2016)

* Rubio, Manuel (s.f.). "Escribir una fantástica historia de usuario", En:

https://help.rallydev.com/es/writing-great-user-story, (fecha de consulta 20/05/2016)

* Quispe, Aline (01/03/2015). "Cerca de la mitad de la población en Bolivia tiene acceso a internet", En:

http://www.la-razon.com/economia/Cerca-poblacion-Bolivia-acceso-internet\_0\_2226377345.html, (fecha de consulta 30/08/2015)

* Quijano, Juan (28/02/2012). " Historias de usuario, una forma natural de análisis funcional", En:

http://www.genbetadev.com/metodologias-de-programacion/historias-de-usuario-una-forma-natural-de-analisis-funcional, (fecha de consulta 25/04/2016)

* Salinas, Patricio Caro; Histchfeld, Nancy (s.f.). " Tutorial de UML", En:

http://users.dcc.uchile.cl/~psalinas/uml/introduccion.html, (fecha de consulta 05/03/2016)

* Schwaber, Ken; Sutherland, Jeff (07/2013). " La Guía Definitiva de Scrum: Las Reglas del Juego", En:

http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf, (fecha de consulta 15/02/2016)

* SIGWEB (s.f.). " Matriz de Riesgo, Evaluación y Gestión de Riesgos", En:

http://www.sigweb.cl/biblioteca/MatrizdeRiesgo.pdf, (fecha de consulta 25/05/2016)

* TutorialsPoint (s.f.). " What is SOAP?", En:

http://www.tutorialspoint.com/soap/what\_is\_soap.htm, (fecha de consulta 01/06/2016)

* Vásquez, Wálter (24/03/2013). "OIT prevé un 2013 con récord de desempleados en el mundo", En:

http://www.la-razon.com/index.php?\_url=/mundo/OIT-preve-record-desempleados-mundo\_0\_1766823306.html, (fecha de consulta 22/08/2015)

* W3C (08/05/2000). " Simple Object Access Protocol (SOAP) 1.1", En:

https://www.w3.org/TR/2000/NOTE-SOAP-20000508/, (fecha de consulta 25/05/2016)