UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO CIENCIAS E INGENIERIAS

ShareTime: Solución Tecnológica para la Coordinación de Reuniones

Sistematización de Experiencias prácticas de investigación y/o intervención

Jefferson Alfonso Camacho Soto

Ingeniería en Sistemas

Trabajo de titulación presentado como requisito
para la obtención del título de
Ingeniero en Sistemas

Quito, 20 de diciembre de 2018

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO CIENCIAS E INGENIERIAS

HOJA DE CALIFICACIÓN
DE TRABAJO DE TITULACIÓN

ShareTime: Solución Tecnológica para la Coordinación de Reuniones

Jefferson Alfonso Camacho Soto

Calificación: _____

Nombre del profesor, Título académico

Daniel Riofrío, Doctor of Philosophy
in Computer Science

Firma del profesor _____

Quito, 20 de diciembre de 2018

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Jefferson Alfonso Camacho Soto

Código: 00117710

Cédula de Identidad: 0202009387

Lugar y fecha: Quito, 20 de diciembre de 2018

RESUMEN

El presente proyecto describe la creación de una aplicación móvil desarrollada para la plataforma Android, la cual permite coordinar reuniones en el tiempo libre de los participantes. Este problema común, para cualquier persona, se evidencia con mayor frecuencia en el transcurso de los años de estudios universitarios donde existe la necesidad de reunirse para realizar trabajos en grupo y al tener todos los compañeros diferentes horarios y actividades, resulta tedioso encontrar un momento en común para este fin. Además, usualmente el líder del grupo es quien carga con todo este trabajo. Esta aplicación permite distribuir el trabajo a todos los integrantes, ya que cada participante determina sus tiempos libres y el coordinador es quien indica a la aplicación las condiciones para reunirse (fecha límite para reunirse, coordinar, etc.) y finalmente la aplicación realiza el cálculo de todos los tiempos en común y se encarga de comunicar a los integrantes la resolución tomada.

ShareTime nació con la idea de ser una aplicación distribuida que no dependa de un servidor central, para lo que se investigaron plataformas de mensajería. Es así como, Telegram fue la plataforma de mensajería que se usó para mandar los paquetes de comunicación de ShareTime. Entre estos paquetes se encuentran los de: solicitud de reunión, reunión agendada y tiempos libres. Se seleccionó a Telegram por las limitaciones de WhatsApp en el momento de: conocer los identificadores de usuarios, grupos, enviar archivos y lectura de éstos. Estas características eran vitales para crear el sistema distribuido. Por otro lado, se debe tomar en cuenta que a pesar de que Telegram tiene una API muy completa, hubo mucho tiempo invertido en entender su protocolo debido a su escasa documentación y ejemplos para Android. Teniendo ya la información proporcionada por Telegram (identificadores) se necesitó de una base de datos para poder desarrollar la aplicación distribuida. Con el objetivo de mantener la consistencia de los datos, información de: participantes, grupos, reuniones y tiempos libres. Se seleccionó a SQLite ya que es una base de datos relacional que es soportada por Android.

Los resultados de la aplicación fueron exitosos ya que se logró obtener el tiempo en común para una reunión. Además, se obtuvo una aplicación independiente de un servidor, lo que evita gastos mensuales de mantenimiento. Por otro lado, la aplicación cuenta una interfaz amigable para el usuario con buenas restricciones de ingreso de datos no válidos y bastante retroalimentación para el usuario, para conocer el estado de cada reunión. También es exitosa con respecto a la arquitectura, ya que mediante la utilización del patrón de diseño MVC, no permite tener dependencias rígidas entre módulos de la aplicación. Como resultado, ShareTime puede ser fácilmente extendida a otras plataformas de mensajería, ya que se han identificado los métodos claves para que a futuro implementar una interfaz para abstraer esta funcionalidad.

Palabras clave: sistema distribuido, móvil, Telegram, scheduling problem, Android, UML, MVC

ABSTRACT

This work describes the developing process of an Android mobile application that coordinates meetings based on participants' time availability. This problem is common to many individuals and it is more noticeable during college, where study groups and meetings happen frequently among students with different time availability. A meeting coordination process is tedious, and it is usually the project leader the one who carries this endeavor. The proposed platform simplifies and distributes, among all the members, the process of scheduling events, meetings or appointments. A coordinator sets preconditions to the proposed platform event (deadlines, coordination, etc.) and each participant set their free time on the application. It is finally the application which performs the scheduling of the event and communicates the result to all the members of the event.

ShareTime packets communication is based on Telegram. Among these packets there are meeting requests, confirmations and participants availability information packages. Telegram provides an API that allow the platform to identify a user and a group of users as well as sending and receiving files; those characteristics were essential in order to create this distributed application. A database was also needed to develop the platform. SQLite was selected as the database management system.

The results of the application were successful since it was possible to obtain the time in common for a meeting. In addition, an independent application was obtained from a server, which avoids monthly maintenance costs. On the other hand, the application has a friendly interface for the user with good restrictions of invalid data entry and enough feedback for the user, to know the status of each meeting. It is also successful with respect to the architecture, since by using the MVC design pattern, it does not allow rigid dependencies between modules of the application, with this it is achieved that ShareTime is easily extended to other messaging platforms, since It showed the key methods so that in the future with an interface this extension can be made.

Keywords: distributed system, mobile, Telegram, scheduling problem, Android, UML, MVC

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Descripción del problema

Las aplicaciones móviles han tenido un incremento exponencial en los últimos años. Dentro de estas, las de tipo utilidad son atractivas para todos los usuarios ya que ayudan a optimizar recursos, entre estos el tiempo. Según datos del 2016, Android tuvo 86.2% del mercado de celulares, en comparación con el 12.9% de iOS (iPhone) (Katariya, 2017). Además, las aplicaciones de productividad ocupan el doceavo lugar, de 49 categorías, de las más populares en Google Play (Statista, 2018). Debido a estos antecedentes se propone crear una aplicación móvil en la comunidad Android con el objetivo de llegar a una gran cantidad de usuarios.

Esta aplicación tiene el objetivo de solucionar el problema de coordinación de una reunión. El nicho de mercado inicial (interesados del producto) son los estudiantes universitarios, específicamente los de la comunidad de la Universidad San Francisco de Quito (USFQ). Esto debido a las diferentes clases, clubs, eventos, conferencias y deportes a las que asisten tanto estudiantes como profesores. Adicionalmente, existe una necesidad particular de los profesores cuando tienen que recuperar clases y no encuentran un horario donde coincidir con todos sus estudiantes, entonces la aplicación podrá apoyar a solucionar este inconveniente.

Por otro lado, Telegram es una aplicación de mensajería instantánea que contiene un API muy completo, lo que permite desarrollar grandes aplicaciones con su plataforma. Por lo cual, se propone utilizar la plataforma de Telegram para el registro de usuarios, creación de grupos e intercambio de archivos y mensajes de la aplicación.

Objetivo General

Crear una aplicación móvil distribuida que permita optimizar el tiempo en la coordinación de reuniones mediante una interfaz amigable con el usuario utilizando como canal de mensajería Telegram.

Objetivos Específicos

Modelar el flujo de procesos mediante UML

Conocer el total funcionamiento de la API de Telegram.

Implementar la arquitectura MVC con alta cohesión y bajo acoplamiento

Implementar una base de datos eficiente en espacio y velocidad

Desarrollar una interfaz gráfica sencilla, práctica e intuitiva con validaciones y restricciones de datos

Organización del Documento

El documento está organizado de una manera que engloba el desarrollo de software mediante el método cascada. Comenzando con la formulación del problema, análisis de requerimientos, diseño, implementación y pruebas. Al final nos encontramos con imágenes de la aplicación resolviendo un caso en particular, una descripción de las mejoras para el futuro y las conclusiones de este proyecto.

DESARROLLO DEL TEMA

Formulación del problema

Meeting Scheduling Problem (MSP)

El conflicto de coordinación que se va a resolver tiene las siguientes condiciones. Habiendo un grupo de personas (incluyendo parejas) que tiene la necesidad de reunirse, la reunión tendrá algunas condiciones como: fecha límite para reunirse, fecha límite para coordinarla (fecha límite de espera de respuesta de tiempos libres de los participantes) y el tiempo de duración de la sesión; suponiendo que cada persona sabe sus tiempos libres. Se desea maximizar la cantidad de personas que van a asistir en esta reunión y la duración de esta.

Sistema distribuido

Para lograr resolver el problema planteado se ha decidido utilizar un sistema distribuido, el cual según Chaffo y Mariños lo definen como un conjunto de computadoras separadas físicamente, pero conectadas entre sí por una red de comunicaciones distribuida (Chaffo & Mariños, 2013). Dentro de esta clasificación se encuentran las bases de datos distribuidas la cuales tienen como beneficios: compartir datos; distribución y autonomía de entidades; ahorro económico; aumento de eficiencia en acceso a datos; permitir recuperar la base de datos y acceso a información desde diferentes lugares.

Android

Se desarrolló el prototipo en Android para la API 19 (KitKat [4.4 - 4.4.4]) en adelante. Se desarrolló para esta versión ya que, mediante las estadísticas obtenidas de IDE Android Studio, mi aplicación correrá en el 95,3% de los dispositivos Android activos. Además de que con estas características se tiene el potencial de Java SE 7.

Telegram

Telegram es un sistema de mensajería con un API bastante completa que permite desarrollar aplicaciones complejas mediante acceso a información de su base de datos como: identificadores de usuarios, grupos y archivos. Por otro lado, por su versatilidad no se necesita tener instalado Telegram en el celular Android para que funcione ShareTime ya que el acceso a la información de Telegram se hace mediante internet.

Por esta razón es la que se decidió desarrollar el primer prototipo (prueba de concepto) con esta plataforma ya que con esta se facilita la creación de una interfaz, que implemente los métodos necesarios para que el sistema sea descentralizado y a futuro expandirlo con otros sistemas de mensajería. Inicialmente se quiso usar WhatsApp, pero mirando su limitación de no tener una API (actualmente está habilitado para selectas organizaciones y es más una API comercial para crear bots) ésta restringía el acceso a información de los contactos y grupos (se debía programáticamente filtrar los contactos que usaban WhatsApp del celular y no se podía enviar varios archivos al mismo tiempo sin tener que entrar a la aplicación de WhatsApp y hacer click en “enviar” manualmente. A esto se añade que se debía tener un hilo escuchando en la carpeta de WhatsApp para saber si ha llegado un archivo nuevo. En el proceso de busca de más opciones, se presentó la opción de Messenger Facebook la cual, si cuenta con una API, pero esta está diseñada para crear bots, no para crear aplicaciones avanzadas. Esta API si permitía enviar archivos, pero mediante un bot o por la función share. En conclusión, WhatsApp y Messenger tienen un enfoque más comercial que no da las facilidades para desarrollar una aplicación completa con sus plataformas.

Hablando sobre el aspecto de seguridad, Telegram se lo considera más seguro que otras plataformas de mensajería como WhatsApp y Line. Esto es debido a que en el modo servidor- cliente que es el usado en los chat normales y grupales, se tiene un cifrado simétrico basado en AES de 256 bits y RSA de 2048 bits (asimétrico) con intercambio seguro de claves

Diffie-Hellman. Con esto Telegram evita que cualquier mensaje sea descifrado cuando son interceptados (Telegram, s. f.).

Desarrollo del prototipo

Análisis de requerimientos

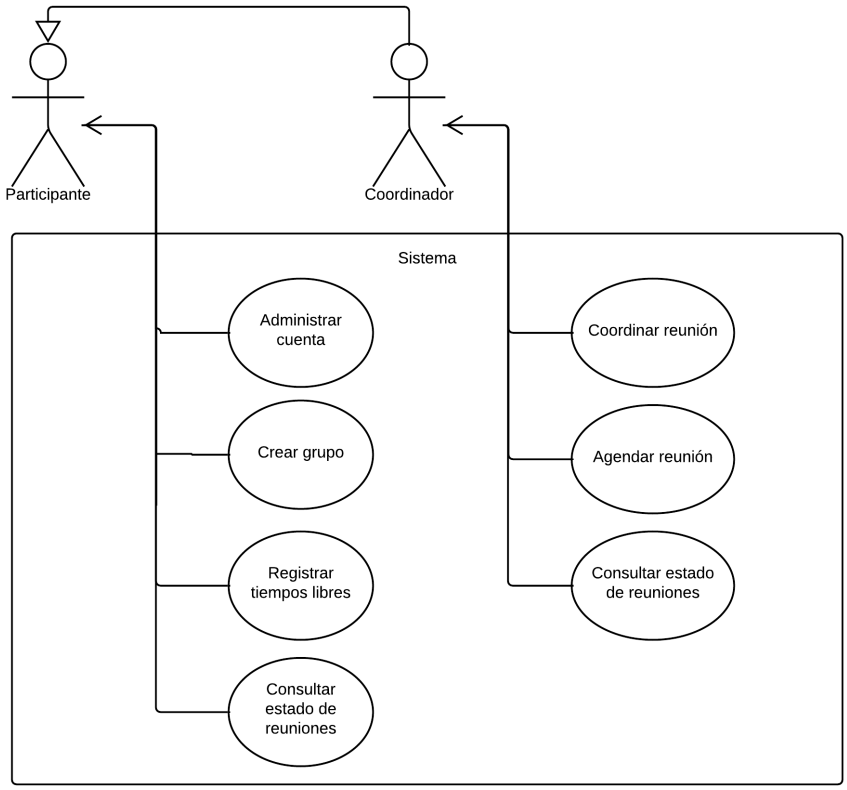

Figura 1: Casos de uso

Después de un análisis de los requerimientos de la aplicación se decidió que en el proceso de coordinación de una reunión existen dos tipos de usuarios: participante y coordinador. En donde cada coordinador también es un participante como se lo evidencia en la figura 1.

Cada participante tiene permitido las funciones de: administrar cuenta, crear grupos, ingresar tiempos libres y consultar el estado de las reuniones. La acción de administrar cuenta engloba a las subactividades: registrarse, en donde el participante se registra en ShareTime y Telegram mediante su número de celular y cerrar sesión, se refiere al proceso cuando el

usuario quiere cambiar de número para esto se elimina todos sus datos guardados hasta el momento. Por otro lado, el participante es el que crea los grupos en Telegram, es decir crea un chat grupal con los integrantes que él seleccione. Cuando el coordinador crea una solicitud de reunión, el participante puede agregar sus tiempos libres mediante una fecha-hora inicial y final. En el proceso de coordinación existe algunos estados que el participante puede evidenciar: “faltas tú”, en donde el participante ingresa sus tiempos libres; “en proceso”, cuando está esperando la respuesta del coordinador; “agendado”, cuando el coordinador ya tomó una resolución e “historial”, que se refiere a las reuniones que ya se realizaron.

Por otro lado, el coordinador será la persona que: coordine, agenda y consulte el estado de las reuniones. La actividad de coordinar comienza cuando el usuario selecciona un grupo y crea una solicitud de reunión con la fecha límite para reunirse, coordinar, etc. Esta información se envía a todos los participantes y después de recibir todos los tiempos disponibles de ellos o se haya vencido la fecha para coordinar, se procede a calcular los tiempos en común. Aquí es donde aparece la actividad “agendar”, donde se obtiene todos los tiempos libres de la base de datos, se ejecuta el algoritmo recursivo, se crea una lista con los resultados y después de seleccionar una de estas, se les notifica a todos los participantes la resolución. En todo este proceso se evidencia los estados de la reunión para el coordinador: “en proceso”, cuando se está en espera de la respuesta de todos los participantes o no se ha vencido la fecha límite de coordinación; “listo”, cuando ya se ha recibido todos los tiempos libres de los participantes o ya se venció la fecha límite de coordinación y “agendado”, cuando ya se a seleccionado una solución de tiempo en común.

Diseño

Interacción

Se usó los diagramas UML para mostrar el flujo de la aplicación. Existe un diagrama de actividad por cada caso de uso como lo explica Larman. Entonces se comenzará con los diagramas de actividades de cada caso de uso del participante y luego del coordinador.

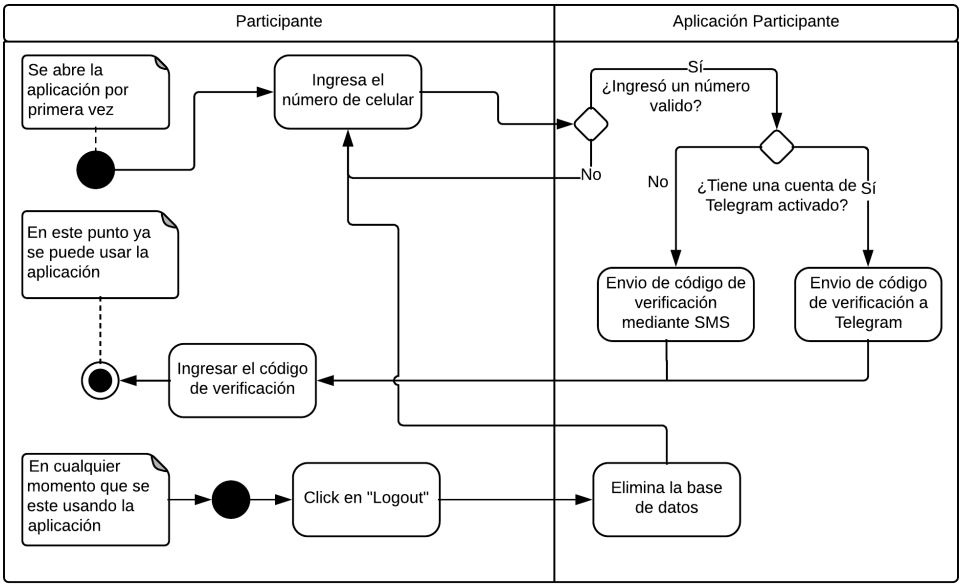

Figura 2: Diagrama de actividad de administrar cuenta

La actividad de la figura 2 muestra dos subactividades: registro y cerrar sesión. En la primera, el usuario debe ingresar un número de teléfono para que le llegue un código de verificación mediante Telegram (si ese número ya esta relacionado con una cuenta de Telegram) o SMS, (si se va a registrar el usuario). Mientras que cuando el usuario cierra sesión, la aplicación se encarga de borrar toda su información y regresa al estado de registro.

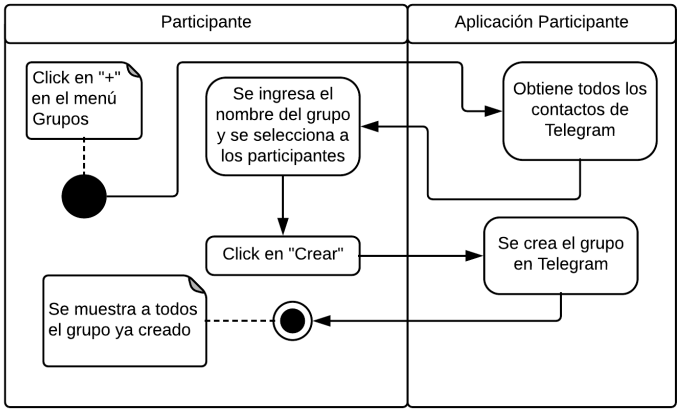

Figura 3: Diagrama de actividad de crear grupo

La actividad de “crear grupo” lo hace cualquier participante, ya que, si un participante crea un grupo, no significa que sea coordinador del mismo. La excepción es en la primera vez que se crea un grupo ya que se supone que el participante quiere crear una solicitud de

reunión y al haber enviado esta solicitud es cuando los demás participantes se dan cuenta de la existencia del grupo y la primera solicitud de reunión.

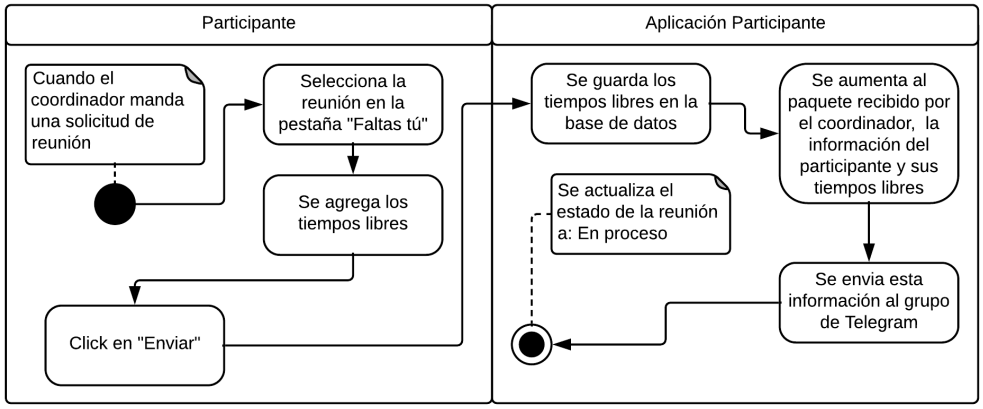

Figura 4: Diagrama de actividad de registrar tiempos libres

La actividad de la figura 4 se da solo cuando se ha recibido una solicitud de reunión por parte del coordinador. En donde el participante agrega sus tiempos libres y al accionar el botón “Enviar” se crea un paquete y se envía a el grupo de Telegram. Esta actividad finaliza cambiando el estado de la reunión a “En proceso”.

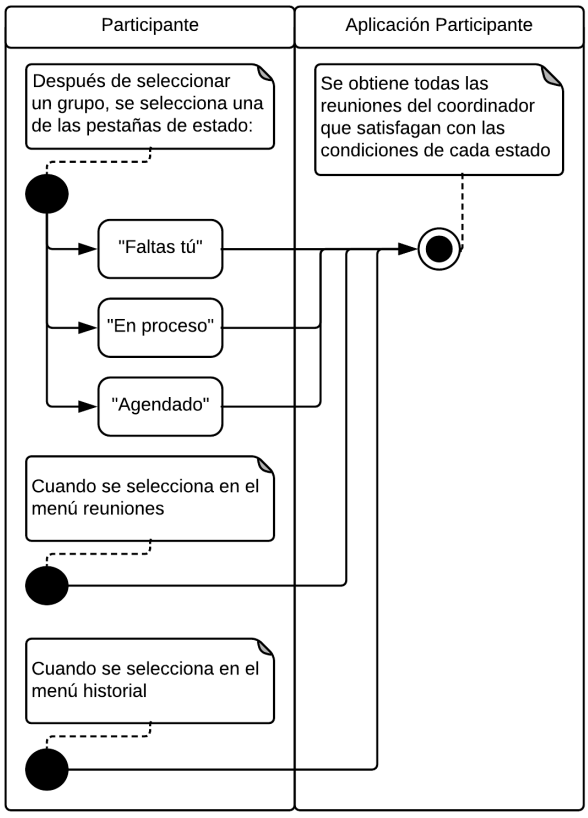

Figura 5: Diagrama de actividad de consultar estado de reuniones (participante)

La actividad de “consultar estado de reuniones” se da cuando se ha recibido previamente una solicitud de reunión, en donde el estado inicial de la reunión es “Faltas tú”. Cuando el usuario selecciona una reunión con este estado comienza la actividad “registrar tiempos libres”. Después de enviar el paquete, el estado de la reunión pasa a “En proceso”, en dónde el participante al seleccionar un grupo, solo le aparece el mensaje de que se esta en espera de la respuesta del coordinador. Finalmente, se logra el estado “Agendado”, cuando el coordinador ha enviado un paquete con la información de la reunión. Por otro lado, el participante puede acceder a todas sus reuniones agendadas (sin necesidad de seleccionar un grupo), a través del menú “Reuniones”. Además, existe un acceso directo en la barra de menú: “Historial”, que permite mirar las reuniones que ya fueron realizadas. Vale recalcar que estos dos elementos del menú también se pueden ver como coordinador.

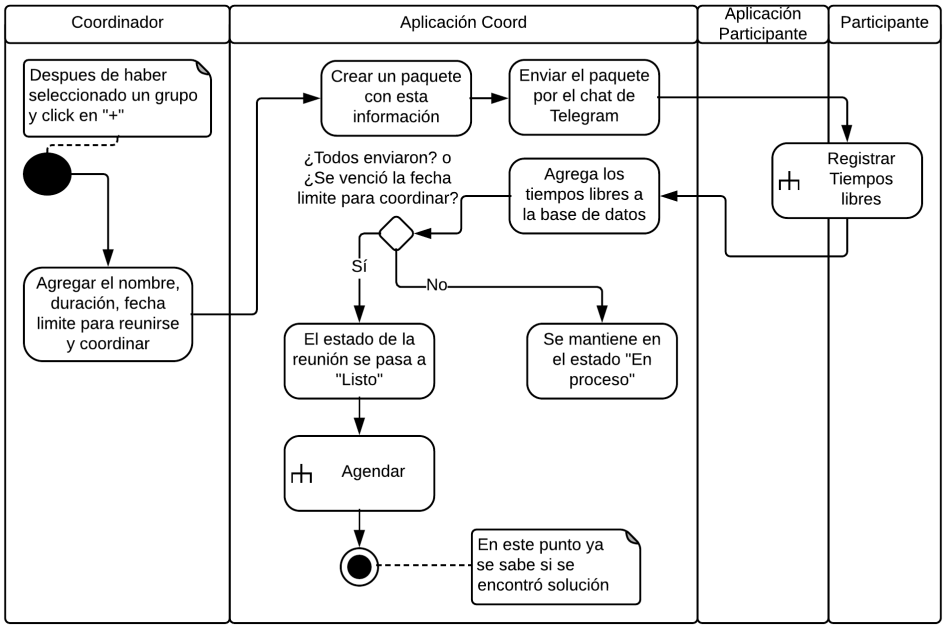

Figura 6: Diagrama de actividad de coordinar reunión

La actividad de la figura 6 la puede hacer cualquier participante de un grupo, pero cuando la realiza se convierte en el coordinador de la reunión. Inicialmente se agrega los requisitos de la reunión, luego se envía como paquete al grupo de Telegram. Después de haber recibido todos los tiempos libres de todos participantes o vencido el tiempo limite de

coordinación, se procede a cambiar el estado de la reunión a “Listo” y finalmente se llega a la actividad agendar, que se muestra la imagen siguiente:

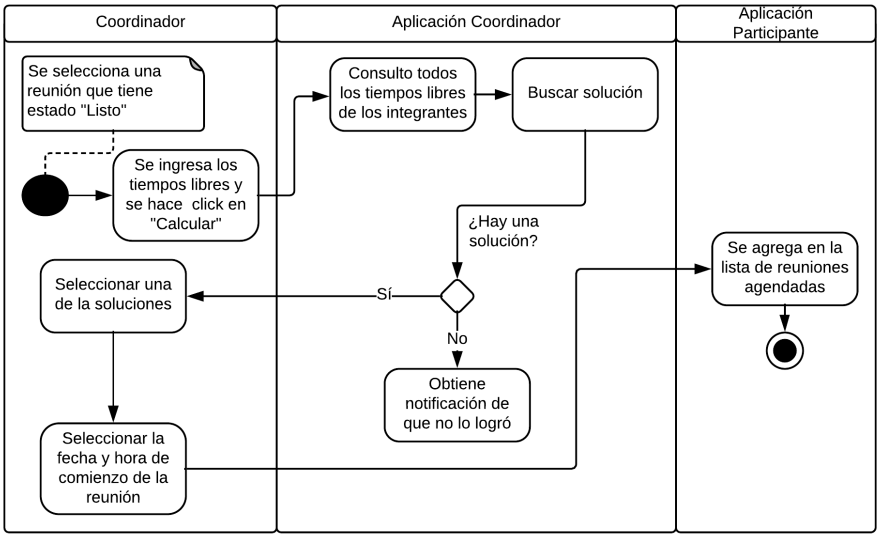

Figura 7: Diagrama de actividad de agendar reunión

En la actividad de “agendar reunión”, el coordinador comienza seleccionando el botón “Calcular”, con lo que se obtiene todos los tiempos libres de los participantes desde la base de datos y luego se ejecuta el algoritmo para encontrar los tiempos en común. Si se encuentra una solución aparece una lista con las distintas opciones y al seleccionar una de estas, se comprueba la fecha y hora de inicio de la reunión para ayudar en el caso que haya soluciones con mayor duración de la deseada. Finalmente, se envía un paquete con la información de la reunión agendada.

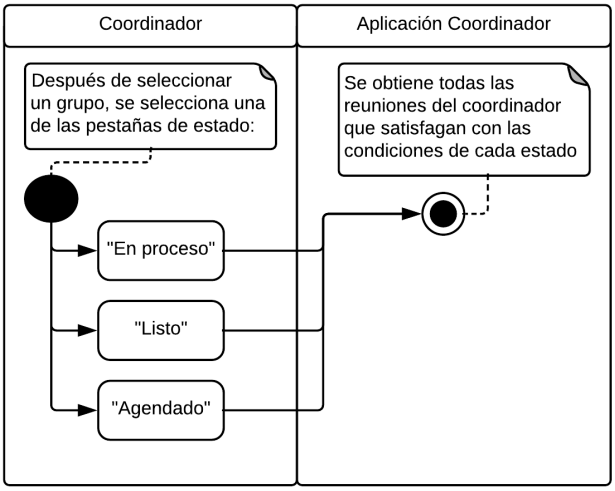

Figura 8: Diagrama de actividad de consultar estado de reuniones (coordinador)

Esta última actividad comienza cuando el coordinador crea una reunión, en este punto la reunión tendrá el estado “En proceso”, que corresponde cuando la reunión todavía no ha recibido los tiempos libres de todos los participantes o no ha superado la fecha límite de coordinación. Al seleccionar una reunión en este estado, se puede visualizar los participantes que han enviado sus tiempos libres hasta ese momento. Después de que hayan enviado todos los participantes sus tiempos libres o se haya vencido la fecha límite de coordinación, el estado de la reunión pasa a “Listo”, en donde el coordinador puede ingresar sus tiempos libres. Después de realizar esta acción y calcular los tiempos en común, el estado de la reunión pasa a “Agendado”, en donde se puede mirar: la fecha, duración y participantes que van a asistir.

Arquitectura del prototipo

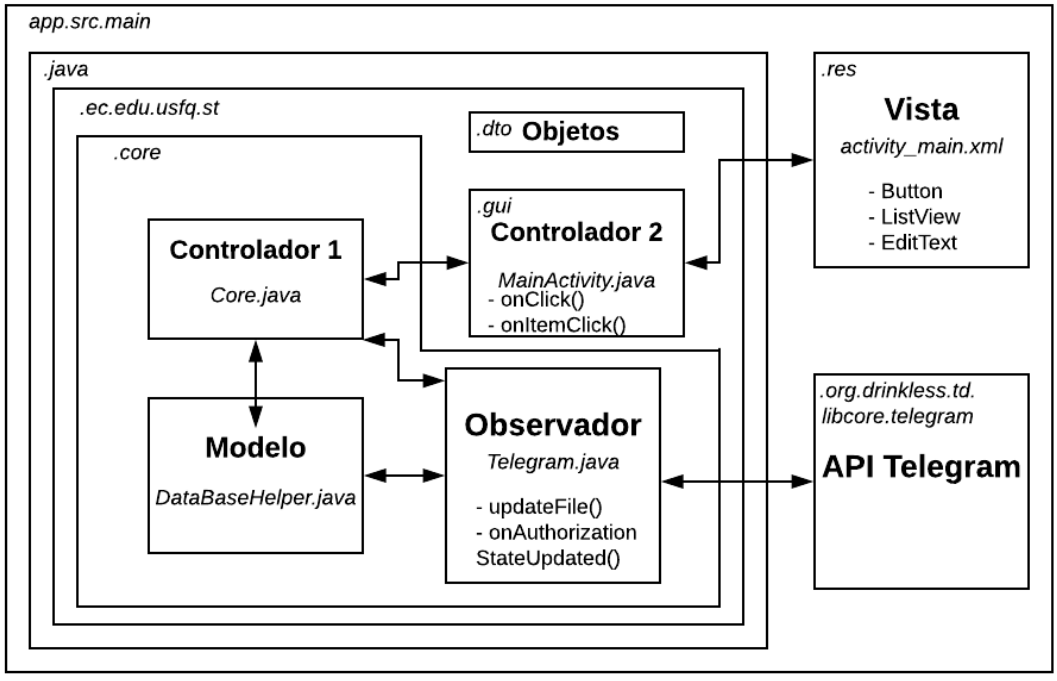

Figura 9: Diagrama de componentes de software

La figura ilustra la distribución de paquetes de la aplicación siendo “app.src.main.” la estructura básica de Android. Dentro de esta se encuentra los paquetes: “.res”, que

corresponde a los archivos xml (vistas); “org.drinkless.td.libcore.telegram”, que corresponde a la API de Telegram con todos sus métodos y finalmente “.ec.edu.usfq.st” que corresponde a el paquete principal de la aplicación. Este contiene las siguientes secciones: “core”, la que el módulo principal de la aplicación; “dto”, que contiene los objetos de transferencia y “gui”, que contiene todas las actividades, fragmentos y adaptadores (java).

Por otro lado, se usó el patrón de arquitectura MVC (Model View Controller) en modo activo en donde: el modelo, es la base de datos en sí; la vista, son los archivos xml de las actividades, fragmentos, diálogos y vistas de Android; el controlador principal (1), es la clase “Core”, que es la encargada de insertar, borrar y modificar datos; el controlador secundario, son los archivos java que maneja las actividades y fragmentos, y finalmente, el observador, es la clase “Telegram”, encargada de monitorear los cambios en la plataforma de mensajería y cuando estos existan (por ejemplo, un nuevo usuario). Cuando sucede esto, la clase Telegram notifica al controlador principal “Core”. Se debe mencionar que se usó además el patrón singleton para manejar correctamente los objetos de las clases Telegram (observador) y Core (controlador principal) para que exista una sola instancia dentro de toda la vida de la aplicación (Gamma, Helm, Johnson & Vlissides, 1995). La aplicación además tiene la arquitectura modular, es la que permite tener el concepto de “caja negra”, en donde cada objeto hace su funcionamiento independiente de los demás objetos. Y para poder usar uno de estos objetos se necesita conocer sus inputs y outputs. Con toda esta estructura de aplicación se logró tener alta cohesión y bajo acoplamiento. Siendo cohesión el grado de relación y acoplamiento el de dependencias entre los componentes o clases (Pressman, 2010).

Core

Este es el primer modulo de la aplicación que contiene los módulos: Telegram, base de datos, el flujo de la aplicación (protocolo) y algoritmo para la selección de tiempos libres.

Telegram

Este módulo es el encargado de toda la API de Telegram, es decir es el cliente de Telegram, que permite actualizar la información de los contactos y grupos. Además, es el encargado de las funciones de envío y recepción de archivos. Se debe mencionar que la API de Telegram es nativamente escrita en C++, pero con la ayuda JNI se puede utilizar en Java, por lo tanto, en Android. La API fue descargada de la pagina oficial de Telegram con el nombre de enlace (Prebuilt library for Android), teniendo una estructura de paquete "org.drinkless.td.libcore.telegram" y no se pudo mover, ni cambiar de nombre ya que hay muchas dependencias en código C++ que lo complica. Para aprender a usar la herramienta se comenzó leyendo un tutorial de cómo compilar la versión desktop y posteriormente se ejecutó un ejemplo. Después de esto se interpolo este a Android es decir se transformó de programación secuencial a eventos. Se debe recalcar que fue una limitante que no hubiera un ejemplo específicamente en Android. Ya que se invirtió mucho tiempo en esto.

Base de datos

El tipo de base de datos que se uso fue SQLite, ya que es el que se usa en los dispositivos Android. Cada actividad y fragmento de la aplicación que interactúa con la base de datos utiliza la clase "DataBaseHelper", que es una subclase de "SQLiteOpenHelper" que simplifica la creación de la base de datos y permite la obtención del objeto "SQLiteDatabase" para manipular los elementos de la base de datos. Las consultas de la base de datos se realizaron con SQL (Structured Query Language) y los resultados de estas son manejadas mediante el objeto "Cursor" (Deitel, Deitel & Deitel, 2015). Por otro lado, la clase "Core", es la que inicializa a el objeto "DataBaseHelper "y lo maneja como un singleton para utilizar en todas las actividades y fragmentos. Cuando se quiere realizar consultas, ingreso y actualización de datos en la base de datos, se manda como parámetros los datos por lo que no permite inyección SQL.

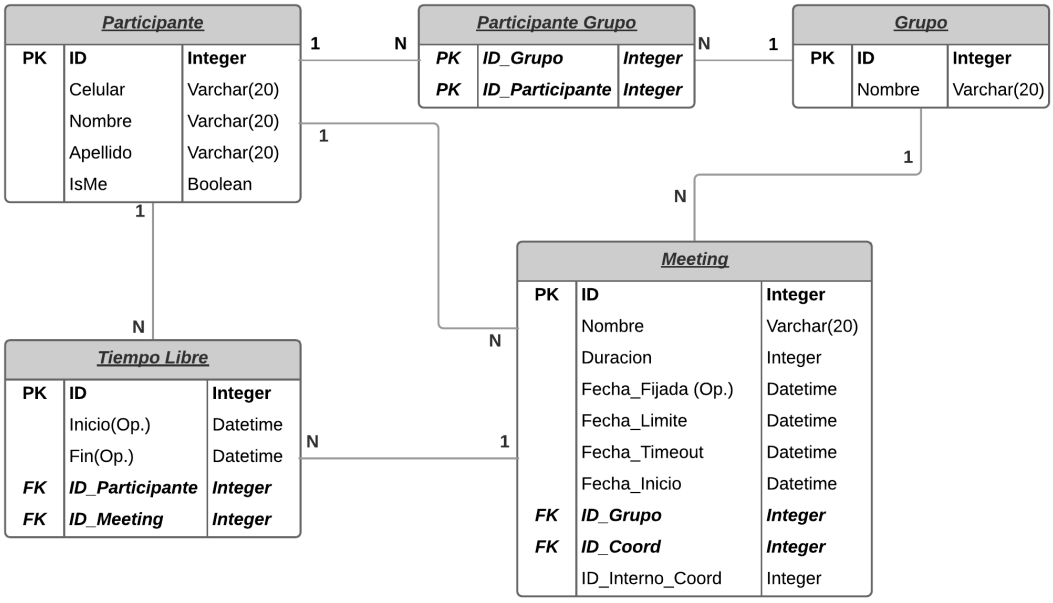

Figura 10: Diagrama de entidad relación

Este módulo es el encargado de gestionar la base de datos. En el diagrama de entidad se muestra las siguientes tablas:

Participante: son los usuarios que se registran en ShareTime, aquí hay un campo especial, “isMe” el cual usa el usuario para identificarse a el mismo.

Grupo: son los grupos.

Participante_Grupo: contiene la relación muchos a muchos entre participantes y grupo en donde se registra los participantes de cada grupo.

Meeting: esta tabla sirve para dos funciones: primero, para generar la necesidad de reunirse (solicitud de reunión) con todas las restricciones como: la fecha límite para coordinar la reunión, la fecha máxima para reunirse, etc. y segundo, para agendar la fecha fija para los mismos.

Tiempo_Libre: son los tiempos libres de un Participante. Esta tabla también se usa para conocer los participantes de una reunión cuando ya es agendada. Esto se lo hace mediante identificar que los campos “inicio” y “fin” estén vacíos.

Protocolo

Esta sección corresponde a todos los flujos de la aplicación que están dentro de la clase “Core”. El cual comienza cuando un participante crea un grupo, estos datos son guardados en su dispositivo y mandados a todos los participantes mediante el chat grupal de Telegram.

El protocolo es dependiente de la estructura de la base de datos. En la tabla “Meeting”, cuando el coordinador crea una solicitud de reunión es cuando este manda un paquete a los participantes y es así como cada participante sabe que pertenece a ese grupo. En las posteriores veces este solo agrega la información de la solicitud de reunión. Por otro lado, la tabla “Meeting” a pesar de ya tener su secuencial (“ID”*)* se agregó un campo “ID_Interno_Coord” para que haya sincronización de datos. Este identificador se usa en el lado del participante, para identificar la reunión creada por el coordinador, es decir para que el participante pueda responder sus tiempos libres y no haya confusión en el lado del coordinador.

Hablando sobre la tabla “Tiempos_Libres”, esta tiene dos funcionalidades: primero, sirve para obtener todos los tiempos libres de los participantes del grupo y segundo, para conocer cuales son los participantes que van a asistir a la reunión agendada (solución).

Se debe tomar en cuenta que toda la transferencia de paquetes se hizo mediante la serialización del objeto “Paquete”. Se debe recalcar que se identifican los archivos de la aplicación mediante el mensaje “Enviado por ShareTime”. Los archivos tienen la siguiente estructura en su nombre: “ST-397478695-699091930-639832218.stime”. Siendo la primera combinación de números, el identificador del grupo (negativo); el segundo, el identificador del coordinador (positivo) y tercero, el identificador del participante emisor (positivo). Se debe tomar en cuenta que todos los participantes siempre están leyendo todos los archivos que llegan al chat grupal. De esta manera hay 3 tipos de paquetes, diferenciados por su contenido: el primero, es para solicitud de una reunión; el segundo, es para la respuesta de un

participante a esta solicitud y el tercero, para una enviar una reunión agendada. El primero contiene un objeto meeting, un objeto participante (coordinador) y un objeto grupo. El segundo, contiene los mismos del anterior, pero agregando un objeto participante (para identificar quien responde a la solicitud de reunión) y con una lista de tiempos libres. El tercero, contiene los mismos objetos que el primer tipo de paquete, más una lista de participantes (reunión agendada).

Algoritmo para la selección de tiempos libres

El algoritmo se encuentra dentro de la clase "Core", el cual tiene un proceso recursivo. Para resolver el problema de coordinación de grupos se diseño este algoritmo que tiene la estructura de un seudoarbol, que es implementado mediante arreglos y listas. Este árbol se va poblando mediante un proceso recursivo. Este proceso comienza con el nodo padre que es el coordinador de la reunión, al cual se le da el privilegio de siempre asistir y continua con sus nodos hijos (los demás participantes del grupo) comenzando con el primer nodo y preguntándole si hay algún tiempo en común entre ellos (simulando una coordinación verbal entre dos personas) si este responde que sí, continua hacia abajo del árbol haciendo la pregunta entre este y el primero de sus nodos hijos (los participantes faltantes). Si este responde que sí, se repite el proceso, entonces el árbol va creciendo de esta manera. Esto se lo hace hasta llegar a la mayor profundidad del árbol (donde ya no existan más participantes) siendo esto el mejor caso ya que significa que se pueden reunir todos (sin haber iterado por todas las posibilidades).

Se debe mencionar que cada nodo padre va guardando la solución óptima hasta ese momento (los tiempos en común donde se pueden reunir) de su respectivo subárbol hasta que se encuentre una mejor solución. En el caso que no encuentre una mejor, devuelve la información existente a el nodo superior (padre) y así sucesivamente hasta llegar al nodo coordinador. Este algoritmo maximiza la cantidad de personas. Y sino encuentra la duración requerida, devuelve las opciones en donde se pueden reunir menos tiempo.

DTO (Objeto de transferencia de datos)

Este módulo se refiere a los objetos que se usan como entidades en la aplicación. Estas están relacionadas con las entidades de la base de datos, pero no usa copia de los registros, solo llevando información requerida o temporal para actualizar o enviar en el paquete. Esta contiene las clases: Participante, Grupo, Tiempos Libres, Meeting y Paquete.

GUI

Este módulo es el que contiene todas las actividades, fragmentos, diálogos y adaptadores (que sirvieron para manejar las vistas personalizadas: listas y submenús). Se debe mencionar que fue importante el uso de fragmentos para tener una interfaz dinámica ya que se agrego submenús. Además, se debe recalcar que se reusó muchas de las actividades y fragmentos, teniendo al final: 7 actividades y 6 fragmentos. Lo que se considera aceptable basado en la complejidad del flujo de la aplicación. Al inicio se estaba implementando todo con fragmentos, pero se dificultó por la complejidad del ciclo de vida de fragmentos dentro de fragmentos. Cada actividad nueva se creó como un flujo independiente para que cada uno esté relacionado con el proceso que llevan, así hay actividades para la creación de una solicitud de reunión, respuesta de tiempos libre, etc.

Implementación

Se utilizó el IDE Android Studio 3.2.1 para la programación de Android. Por otro lado, una de las partes más desafiantes fue la API de Telegram ya que al ser código de terceros hubo dificultad en averiguar todas las funcionalidades, además de la escasa documentación previa de como usar la biblioteca. Hubo confusión de que esta se debía compilar con C++, pero esto solo era necesario para la versión desktop con java. Lo que se hizo es entender primero un ejemplo en versión Java para interpolarlo a programación de eventos. Uno de los problemas con esto fue que casi toda la clase tenia elementos estáticos que no se podían transferir a Android de manera correcta por lo que se necesitó transformar a

un singleton para que mantengan constancia en memoria. Un paso importante fue registrar a ShareTime en la Telegram, con lo cual se obtuvo un identificador para poder usar su API.

Pruebas

Se realizaron algunas pruebas con de la aplicación, una de estas fue la siguiente:

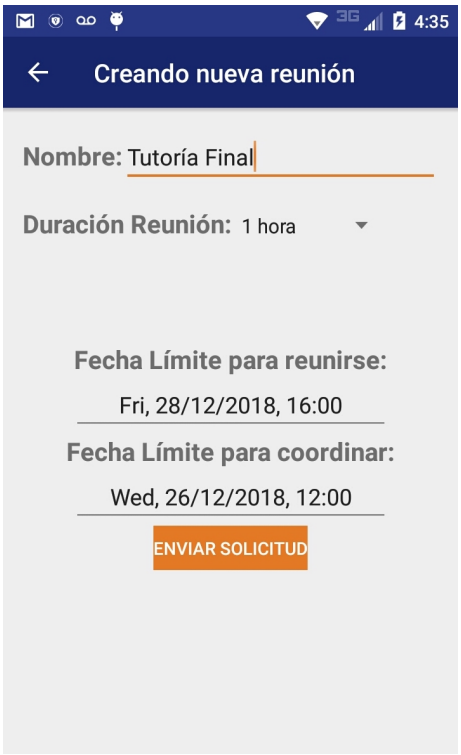

Figura 11: Creando una solicitud de reunión

En esta figura se ilustra la necesidad de encontrar un tiempo en común para una ultima tutoría de mi tesis. La figura 10, es la captura de pantalla del celular del coordinador. Se necesita una duración de sesión de una hora y que la fecha límite que podemos reunirnos con mi tutor, es hasta el 28 de diciembre a las 16:00. También se agrega la fecha límite para coordinar: el 28 de diciembre a las 12:00. Después de seleccionar la opción “Enviar solicitud”, se envía el paquete al participante.

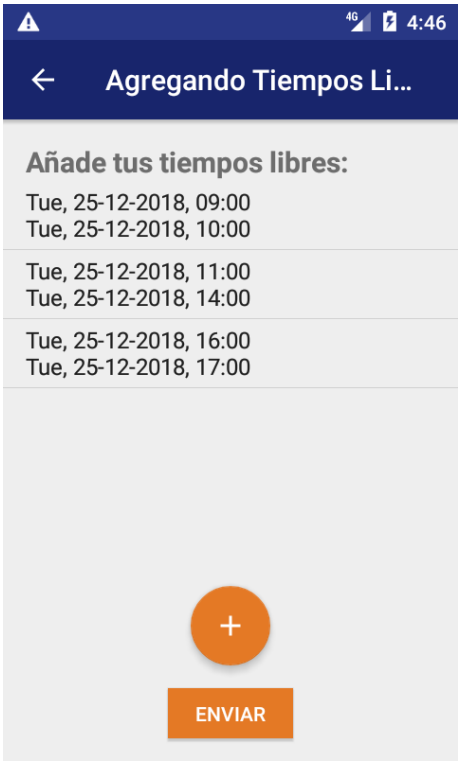

Figura 12: Agregando tiempos libres el participante

El participante al recibir el paquete tiene la opción de agregar sus tiempos libres (en el estado de la reunión "Faltas tú") y ingresa 3 tiempos disponibles como lo ilustra la figura 11. Al finalizar, el selecciona la opción "enviar", donde envía los datos al coordinador.

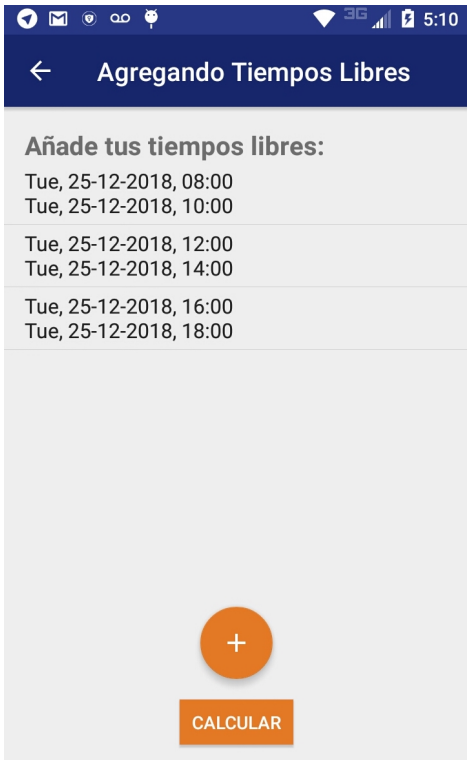

Figura 13: Agregando tiempos libres el coordinador

Al recibir el coordinador, los tiempos libres del participante, el coordinador ingresa sus tiempos libres y selecciona la opción “calcular”, en donde se ejecuta el algoritmo para encontrar los tiempos en común en el background y muestra los resultados de la siguiente imagen.

Resultados

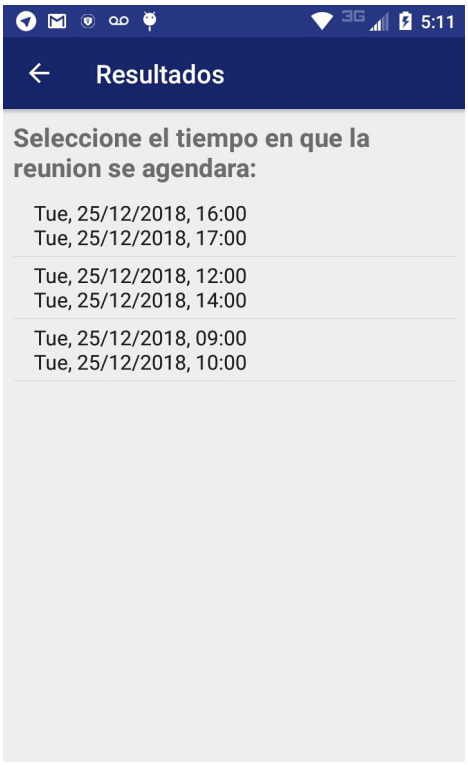

Figura 14: Resultados de la reunión

Mediante: las restricciones de la reunión evidenciados en la figura 10, los tiempos libres ingresados por el participante en la figura 11 y los del coordinador en la figura 12, se obtiene los resultados en la figura 13. Los resultados son tres, los cuales son coherentes. Vale recalcar que la segunda opción da una solución mayor a la necesitada (1 hora) por lo tanto, el coordinador al seleccionar esta opción se despliega un dialogo para seleccionar la fecha y hora de inicio. También vale mencionar que se implemento una animación para que el

usuario espere mientras se ejecute el algoritmo, pero esta nunca se visualizó ya que fue instantáneo el procesamiento.

Trabajo a futuro

A pesar de que la aplicación abarca un gran segmento de mercado tecnológico en Android, se puede exportar a: versiones desktops (Windows, Linux y macOS) y dispositivos Apple (iPhone, iPad) ya que Telegram cuenta con una API para estas plataformas. Sería ideal también ampliar las plataformas de mensajería para el envío de archivos como: WhatsApp, Messenger Facebook, Viber, etc. Esto es posible ya que cuenta con una sólida arquitectura que permite fácilmente exportarlo cuando estas compañías desarrollen APIs más completas.

Con respecto a funcionalidad se puede agregar la opción de coordinar una reunión mediante prioridades donde exista 3 tipos: alta, media y baja. En donde alta, es obligatorio su asistencia; media, deben estar la mayoría de esta sección (es importante que vayan) y baja, donde no es necesario su presencia. También se puede añadir la función de agregar las reuniones agendadas directamente en el calendario (nativo del celular u otro). Y para mejorar la eficiencia, se puede aumentar la funcionalidad de obtener tiempos libres mediante importar las reuniones de nuestro calendario personal. Y con respecto al algoritmo se puede mejorar la eficiencia eliminando las interacciones por participantes que ya fueron pasados antes por el algoritmo.

La interacción del usuario con la aplicación es importante, por lo que se podría desarrollar una interfaz gráfica más sencilla: seleccionando una fecha y duración del tiempo libre para agilitar el proceso de agregar los tiempos libres. Si se desea una interfaz más interactiva se debería buscar una librería o implementarla para que permita al usuario agregar tiempos libres con un scroll o touch. Estas mejoras en la interfaz facilitarían la retroalimentación: en el proceso de registro en la aplicación, notificando al usuario que se

está creando una cuenta de Telegram, en el caso que no la tenga y sobre lo que esta pasando en la creación de la reunión.

Un gran aporte a futuro sería poder editar y eliminar: los grupos, tiempos libres y reuniones en cualquier momento y notificar los cambios a los interesados. Con esto se podría habilitar la opción de que si alguien no puede asistir a ultimo momento, todos estén actualizados de este dato. Con esto se agregaría nuevos estados como: “Cancelación”, “Modificación de tiempos libres”; con lo que se necesitaría agregar un elemento en el menú, llamado “Notificaciones”, en donde se pueda tener retroalimentación del flujo de la aplicación, por ejemplo, cambios de los grupos y reuniones. Con estas funcionalidades se facilita generar más iteraciones de pedir tiempos libres en la aplicación cuando no se haya encontrado solución. Es decir, pedir a los participantes que aumenten sus tiempos disponibles para que haya más probabilidad de encontrar un tiempo en común para reunirse.

CONCLUSIONES

La aplicación ShareTime fue exitosa ya que resuelve el problema de coordinación de reuniones grupales. Gracias a los beneficios del modelo MVC, la aplicación tiene una estructura sólida que permite exportarlo con facilidad a otras plataformas de mensajería. Mediante la plataforma Telegram, la aplicación logró ser distribuida y para consistencia de los datos se usó SQLite, lo cual permite evitar costos mensuales por un servidor. Por otro lado, tiene una interfaz amigable para el usuario y con control de ingreso de datos para no se corrompa el sistema. Además, cuenta con una buena retroalimentación para que el usuario conozca la situación actual de cada reunión.

El reto más importante de la aplicación fue manejar la API de Telegram ya que contiene más de 40000 líneas de código, abarcando todos los conceptos adquiridos a lo largo de la carrera como herencia, sincronización, hilos, patrones de diseño, etc. En ShareTime, al ser una aplicación completa, se evidencia con mayor claridad el ciclo de desarrollo en cascada y se identifica los problemas más grandes de este, como el gran costo de regresar de la implementación al diseño. Esto ocurrió ya que se invirtió mucho tiempo en realizar el diseño basado en WhatsApp sin darse cuenta de que no existía una API con las funcionalidades que se necesitaba.

Además, mientras se avanzó en el proyecto se necesitó un mayor orden, documentación, consistencia y optimización en el código ya que una modificación en alguna parte cambiaba al sistema completo. Esto logró que haya menos líneas de código y dependencias entre todos los componentes en la versión final.

REFERENCIAS BIBLIOGRÁFICAS

Chaffo, G. & Mariños, J. (2013). *Modelado E Implementación De Un Sistema Distribuido Para La Gestión De Consultas En Las Bibliotecas De La Universidad Nacional De Trujillo*. Extraído de: http://www.inf.unitru.edu.pe/revista/22.pdf

Deitel, P., Deitel, H. & Deitel, A. (2015). *Android How to Program, 2nd Edition.* (pp. 304-345)

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). *Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston.*

Katariya, J. (2017). *Apple Vs Android—A comparative study*. Extraído el 17 de diciembre 2018 de: https://android.jlelse.eu/apple-vs-android-a-comparative-study-2017-c5799a0a1683

Larman, C. (2005). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development Third Edition*. (pp. 477-482).

Pressman, R. (2010). *Ingeniería del software: Un enfoque práctico séptima edición.* (pp. 243-245).

Statista (2018). *Most popular Google Play app categories as of 1st quarter 2018, by share of available apps.* Extraído de: https://www.statista.com/statistics/279286/google-play-android-app-categories/

Telegram (s.f.). *Telegram FAQ*. Extraído el 17 de diciembre 2018 de: https://telegram.org/faq#q-how-secure-is-telegram