INGENIERÍA DE SOFTWARE II

PROYECTO DE SEMESTRE

JAVIER EDUARDO SEPULVEDA FIUENTES

UNIVERSIDAD COOPERATIVA DE COLOMBIA

SEDE BOGOTA

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

17 de noviembre de 2016

INGENIERÍA DE SOFTWARE II

PROYECTO DE SEMESTRE

Presentado por:

JAVIER EDUARDO SEPULVEDA FIUENTES

Presentado a:

JOSE FERNANDO SOTELO CUBILLOS

UNIVERSIDAD COOPERATIVA DE COLOMBIA

SEDE BOGOTA

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

17 de noviembre de 2016

**TÍTULO DEL PROYECTO**

¡MOTOOLS!

MO > >MOTO

TOOLS >> HERRAMIENTAS

SQA: El nombre dado al proyecto y el cual se espera conlleve el producto fina es apropiado ya que además de estar relacionado propiamente con el objetivo de la aplicación, es corto y llamativo lo cual augura una buena aceptación por parte del público.

**DESCRIPCIÓN**

Es una aplicación a través de la cual un usuario puede recibir de manera rápida una notificación en caso de que cerca de una ubicación establecida se encuentre rondando la grúa de movilidad o algún agente de tránsito. De manera adicional la aplicación contará con un servicio colaborativo a través del cual los usuarios podrán registrar información sobre ubicación de parqueaderos para motos y tarifas.

**OBJETIVOS**

**PRINCIPAL**

Diseñar una aplicación capaz de notificar a un usuario sobre la cercanía de la grúa de movilidad o algún agente de la policía de tránsito

SQA: El objetivo es claro y conciso, contiene exactamente lo que pretende lograrse con el desarrollo de proyecto

**OBJETIVOS SECUNDARIOS**

* Lograr el envío de alertas en el menor tiempo posible
* Crear dos funciones principales (accesos), una para envío de alertas y otra para acceso a la aplicación en general.
* Guardar y mostrar estadística sobre las alertas notificadas por los usuarios y llegadas reales de los agentes de tránsito.
* Crear un módulo para la información de lugares de parqueo y tarifas

**JUSTIFICACIÓN**

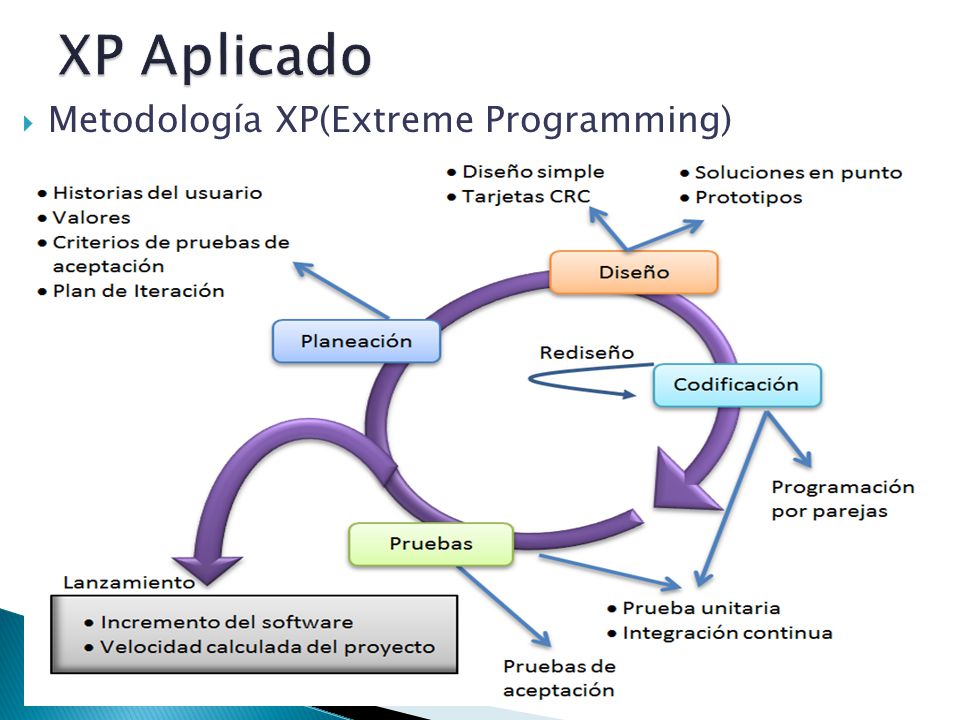
El presente proyecto surge y está dirigido principalmente a los estudiantes motorizados de la Universidad Cooperativa de Colombia sede Bogotá quienes por economía y facilidad deciden estacionar sus vehículos en los alrededores del campus sobre la vía pública mientras acuden a sus clases, exponiéndose de esta manera a adjudicarse una multa por infringir normas de tránsito y a que su vehículo sea inmovilizado por parte de las autoridades de tránsito.

En lo transcurrido de la semana del 12 de septiembre de 2016 son varios los casos en los cuales la policía de tránsito a arribado a los alrededores de la sede principal de la Universidad en mención mientras los estudiantes de esta misma realizan sus labores académicas. Desafortunadamente el voz a voz no es para nada efectivo en este caso ya que cuando la noticia llega al estudiante se corre un enorme riesgo de que al llegar al lugar de los hechos ya sea demasiado tarde (como en algunos de los casos de esta semana) y su vehículo ya haya sido recogido por la grúa de movilidad y de manera adicional se convierta en acreedor de un gasto inesperado de mínimo CO$ 310.000.

Con el fin de brindar a este grupo de estudiantes una herramienta capaz de notificar de manera ágil cuando sus vehículos automotores corran peligro se pretende desarrollar este proyecto de aplicación el cual son toda seguridad y con el apoyo de todos los usuarios logrará evitar en gran medida estas situaciones, ahorrándonos de ésta manera tiempo y dinero.

**METODOLOGÍA A USAR**

**eXtreme Programming**



SQA: La metodología a utilizar si es apropiada ya que se requiere la terminación del proyecto en un corto plazo, además se pretende empezar a realizar pequeñas entregas funcionales cada dos semanas a partir del 01 de octubre de 2016. Debe tenerse cuidado en el aspecto de la codificación, ya que en XP es de suma importancia la programación en parejas lo cual es algo que no se puede dar en el desarrollo de este proyecto

PRINCIPIOS BÁSICOS

Además de los cuatro principios fundamentales que caracterizan a toda metodología ágil de desarrollo, se tendrán en cuenta los siguientes principios propios de la metodología XP:

**Retroalimentación**

* **Principio de pruebas**: lo primero que se debe hacer es establecer un período de pruebas de aceptación del programa, en el cual se definirán las entradas y salidas del sistema. Básicamente se define lo que debe hacer el software desarrollado. Como si fuese una caja negra.
* **Planificación**: el cliente (o su representante) escribirá sus necesidades para definir concretamente las actividades que el sistema debe realizar. En esta fase se creará un documento que contendrá **historias de usuario** que forman el **plan de liberación**, el cual define los tiempos de entrega de la aplicación para poder recibir *feedback* por parte del cliente.
* **Cliente *in-situ***: **el cliente (o su representante) deberá formar parte del equipo de desarrollo**. Se le dará poder para determinar los requisitos de la aplicación, definir la funcionalidad y dar prioridad a determinadas cosas. Gracias a esto, habrá una fuerte interacción con los programadores, disminuyendo así el tiempo de comunicación y la cantidad de documentación a redactar. **El cliente estará con el equipo durante todo el proceso de desarrollo del proyecto**.
* ***Pair-programming***: este punto junto con el anterior son los más radicales de esta metodología. Consiste en **escribir código en parejas compartiendo una sola máquina**. Según los experimentos ya realizados sobre este método, se producen mejores y más consistentes aplicaciones a igual o menor coste.

**Proceso continuo en lugar de por bloques**

* **Integración continua**: consiste en implementar progresivamente las nuevas características del software. En lugar de crear versiones estables en función de una planificación previamente realizada, los programadores reúnen su código y reconstruyen el proyecto varias veces al día si hace falta.
* **Refactorización**: mediante la constante eliminación de código duplicado y/o ineficiente los equipos de programación mejoran el diseño del sistema. El código se evalúa continuamente para ofrecer la mayor calidad posible.
* **Entregas pequeñas**: el producto es evaluado en un ambiente real mediante la colocación de un sistema sencillo en producción el cual se actualizará rápidamente, es decir, cada 2 semanas (3 como máximo) el software será puesto en producción.

**Entendimiento compartido**

* **Diseño simple**: el mejor programa será aquel que cumpla con los requisitos y sea más simple. Es importante proporcionar un software que cubra las necesidades de un cliente. Ni más ni menos.
* **Metáfora**: expresa la visión evolutiva del proyecto y define los objetivos del sistema mediante una historia.
* **Propiedad colectiva del código**: el código tiene propiedad compartida. Nadie es propietario de nada, ni siquiera de lo que ha desarrollado. Todos los programadores son “dueños” de todo el código. Según esta metodología, cuantos más programadores trabajen en una parte de código, menos errores tendrá.
* **Estándar de programación**: define las reglas para escribir y documentar código, además de cómo se comunican las diferentes piezas de código desarrolladas por diferentes equipos. El objetivo de esto es que parezca que el código ha sido escrito por una única persona.

**Bienestar del programador**

* **Semana de 40 horas**: los programadores cansados escriben peor código.Es importante minimizar las horas extras y mantener a los programadores frescos y descansados. De esta manera, se generará mejor código. Si es necesario hacer horas extras, quiere decir que el proyecto está mal planificado.

DESARROLLO DE LA METODOLOGÍA

**Planificación**

* Historias de usuarios
* Plan de entregas (“Release Plan”)
* Plan de iteraciones (“Iteration Plan”)
* Reuniones diarias de seguimiento (“Stand-up meeting”)

**Diseño**

* Simplicidad
* Soluciones “spike”
* Recodificación
* Metáforas

**Desarrollo del código**

* Disponibilidad del cliente
* Uso de estándares
* Programación dirigida por las pruebas (“Test-driven programming”)
* Programación en pares
* Integraciones permanentes
* Propiedad colectiva del código
* Ritmo sostenido

**Pruebas**

* Pruebas unitarias
* Detección y corrección de errores
* Pruebas de aceptación

**REQUISITOS**

**FUNCIONALES**

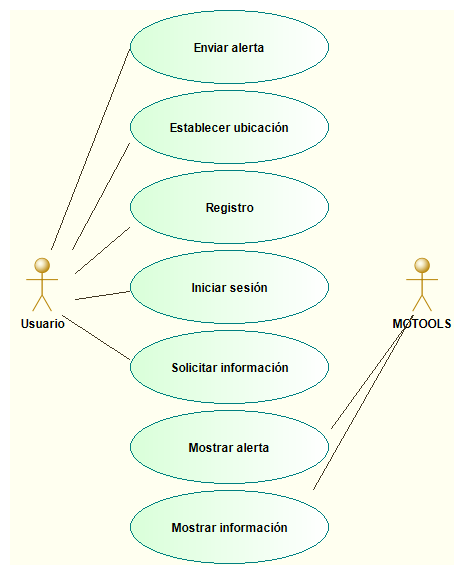
* Registro en la aplicación: El usuario luego de la instalación deberá registrarse sus identificadores serán número de celular y el número de placa de su vehículo, además deberá rellenar otros datos de información básica
* Inicio de sesión: Se requiere que el usuario inicie sesión con los datos de registro para poder hacer uso de las funciones de la aplicación
* Envío de notificaciones: El usuario contará con una función para el envío de alertas a los demás usuarios.
* Establecer ubicación Cuando un usuario va a estacionar su vehículo establece una ubicación la cual será de referencia para que la aplicación le muestre las alertas pertinentes. También se requiere establecer ubicación para registrar sitio de parqueo en el módulo de información.
* Visualizar estadísticas: El usuario podrá ver un registro estadístico sobre la totalidad de alertas enviadas por los usuarios y las ubicaciones con mayores alertas recibidas.
* Visualizar sitios de parqueo y tarifas: La aplicación debe disponer de un módulo a través del cual se pueda informar sobre sitios de parque y tarifas.

**NO FUNCIONALES**

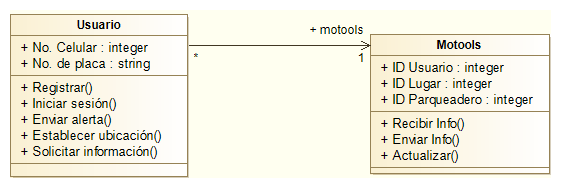
* Ejecución de aplicación en segundo plano: La aplicación luego de ser ejecutada por primera vez deberá seguir ejecutándose en segundo plano.
* Mantener sesión: Luego de que el usuario ha iniciado sesión esta deberá permanecer iniciada para que el usuario pueda recibir las notificaciones.
* Uso de GPS: Cuando el usuario va establecer una ubicación o enviar una alerta la aplicación debe asegurar el funcionamiento del GPS.
* Conexión a internet: El dispositivo debe contar con una conexión permanente a la red de datos móviles.
* Se requiere un tiempo de respuesta mínimo tanto para el envío de la alerta como para la recepción de la notificación.

**DIAGRAMAS**

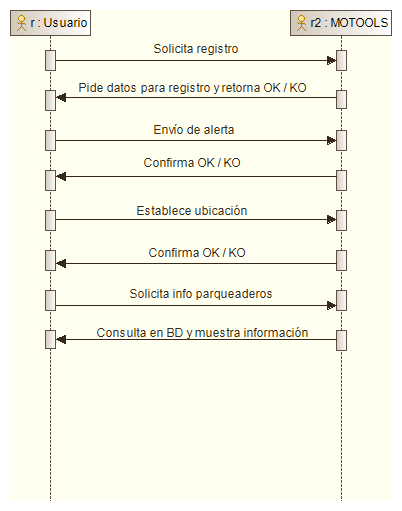
**DIAGRAMA DE CASOS DE USO**



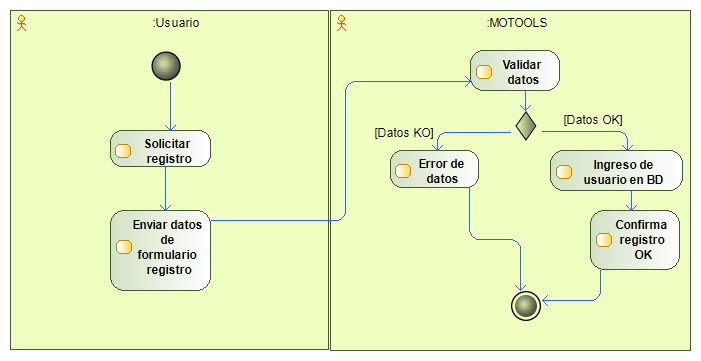
**DIAGRAMA DE CLASES**



**DIAGRAMA DE SECUENCIA**



**DIAGRAMA DE ACTIVIDAD**

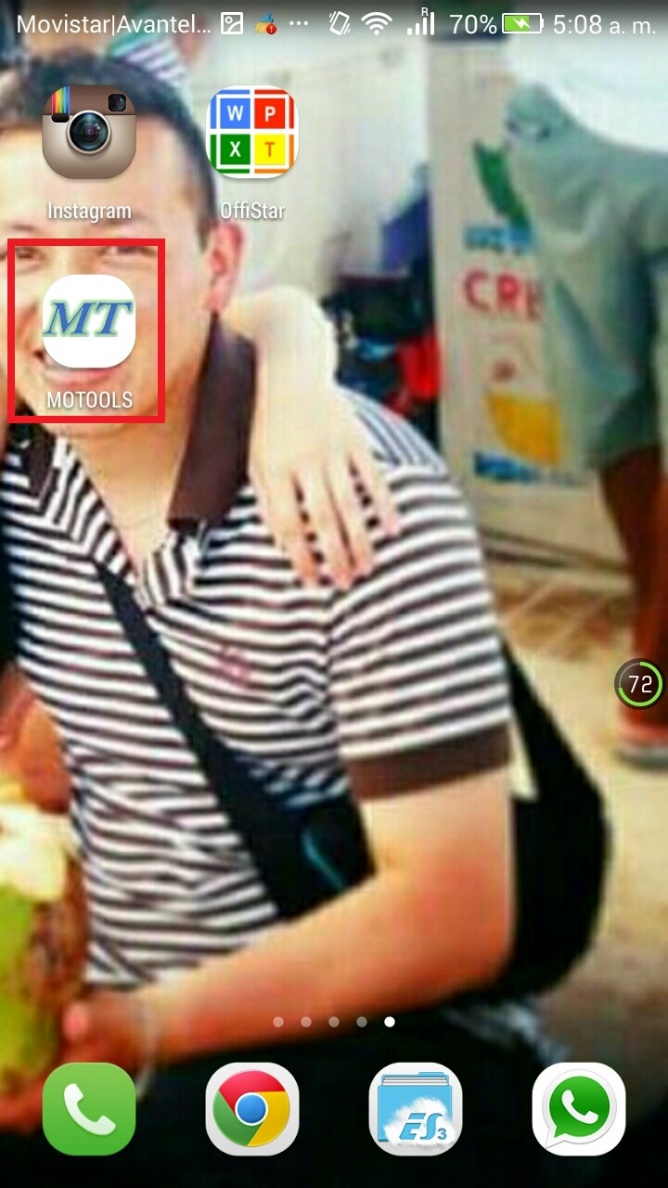


**HISTORIAS DE USUARIO**

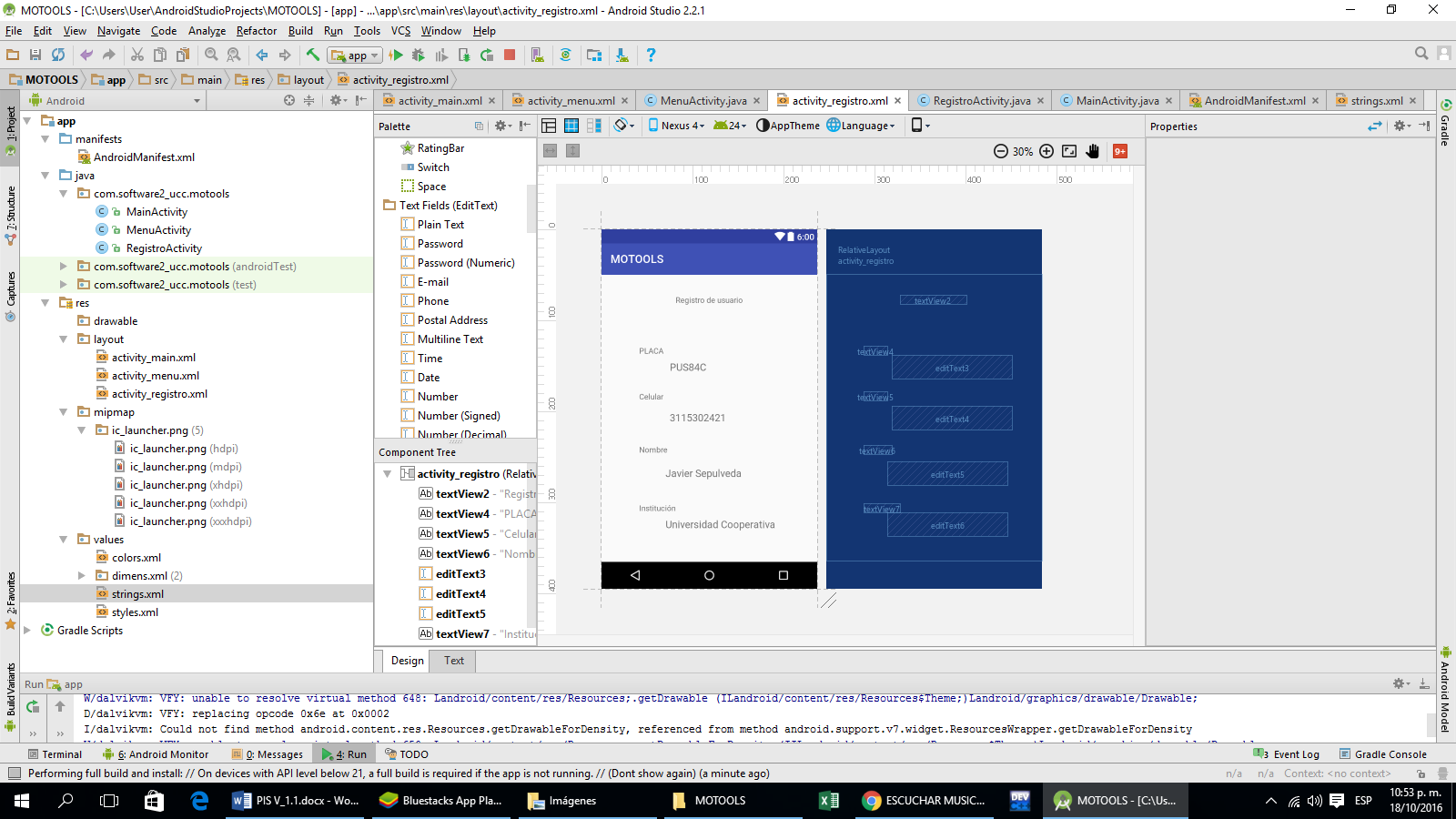
* Como usuario quiero disponer de una función sencilla mediante la cual pueda enviar una alerta.
* Como usuario necesito garantizar que la aplicación se siga ejecutando en segundo plano mientras hago uso de otra función del Smartphone.
* Como usuario solicito una función que me permita corroborar la conectividad de la aplicación a la red.
* Como usuario solicito una función que me permita corroborar la fecha de actualización de la información sobre parqueaderos y tarifas mostrada por la aplicación.

**DISEÑO**

Ícono de acceso



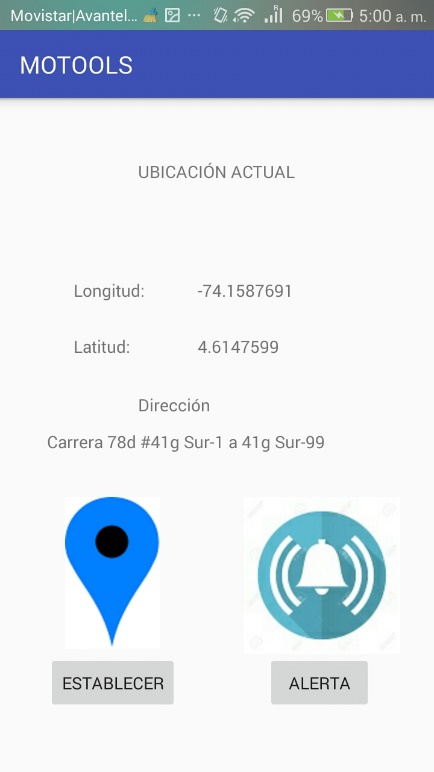
Interfaz de registro:



Interfaz de inicio



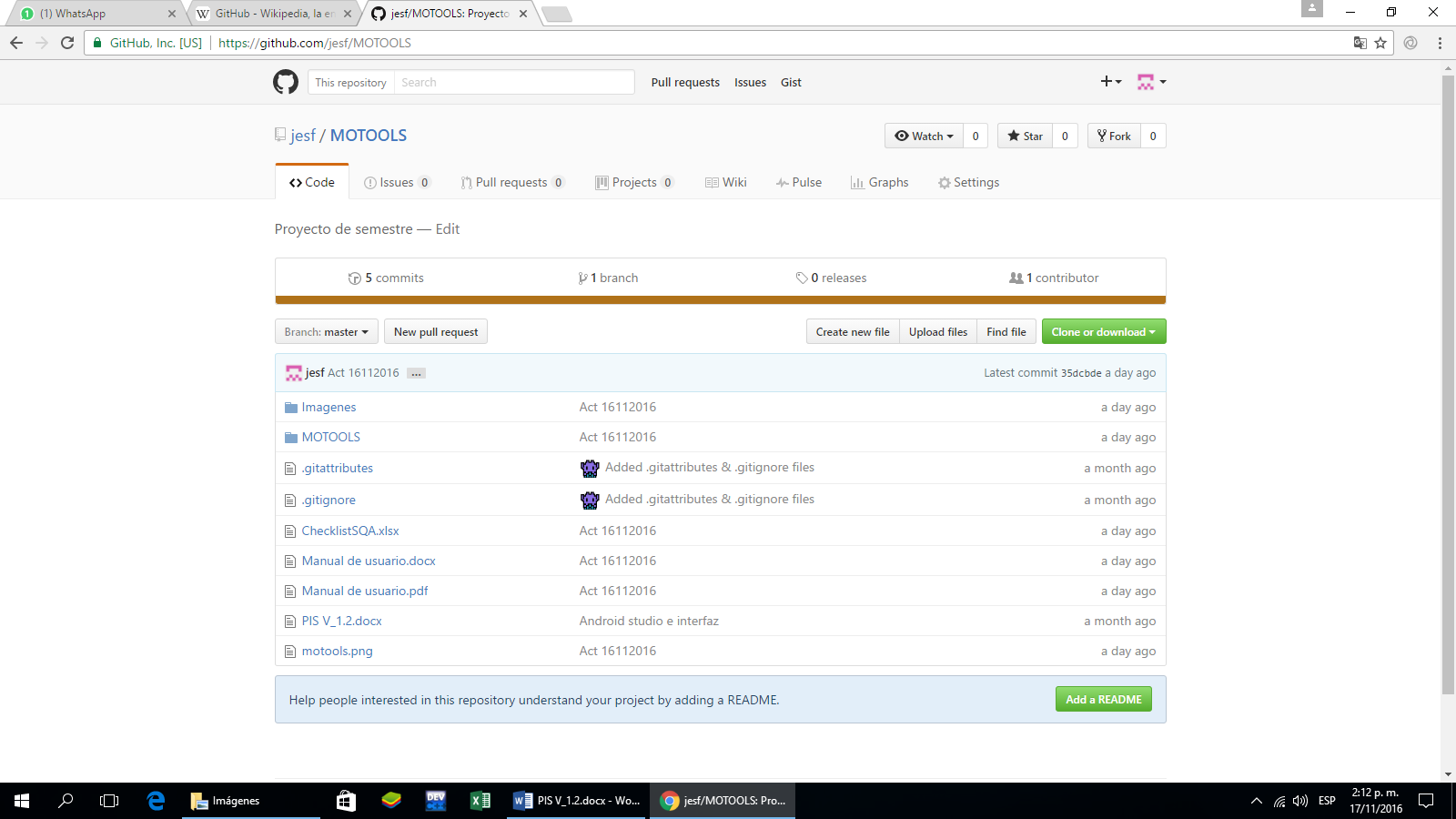
Interfaz de Ubicación:



**CONTROL DE VERSIONES**

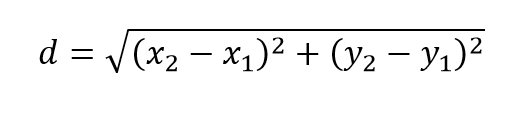
El control de versiones se realizará por medio de GITHUB, plataforma de desarrollo colaborativo para alojar proyectos. Esta plataforma permite de manera fácil y organizada llevar un registro y un control sobre cada uno de los cambios que se van realizando al proyecto en cada una de las etapas de desarrollo.

El proyecto se encuentra alojado en esta plataforma bajo el nombre “MOTOOLS” usuario jesf:



**FUNCIONAMIENTO DE LA APLICACIÓN**

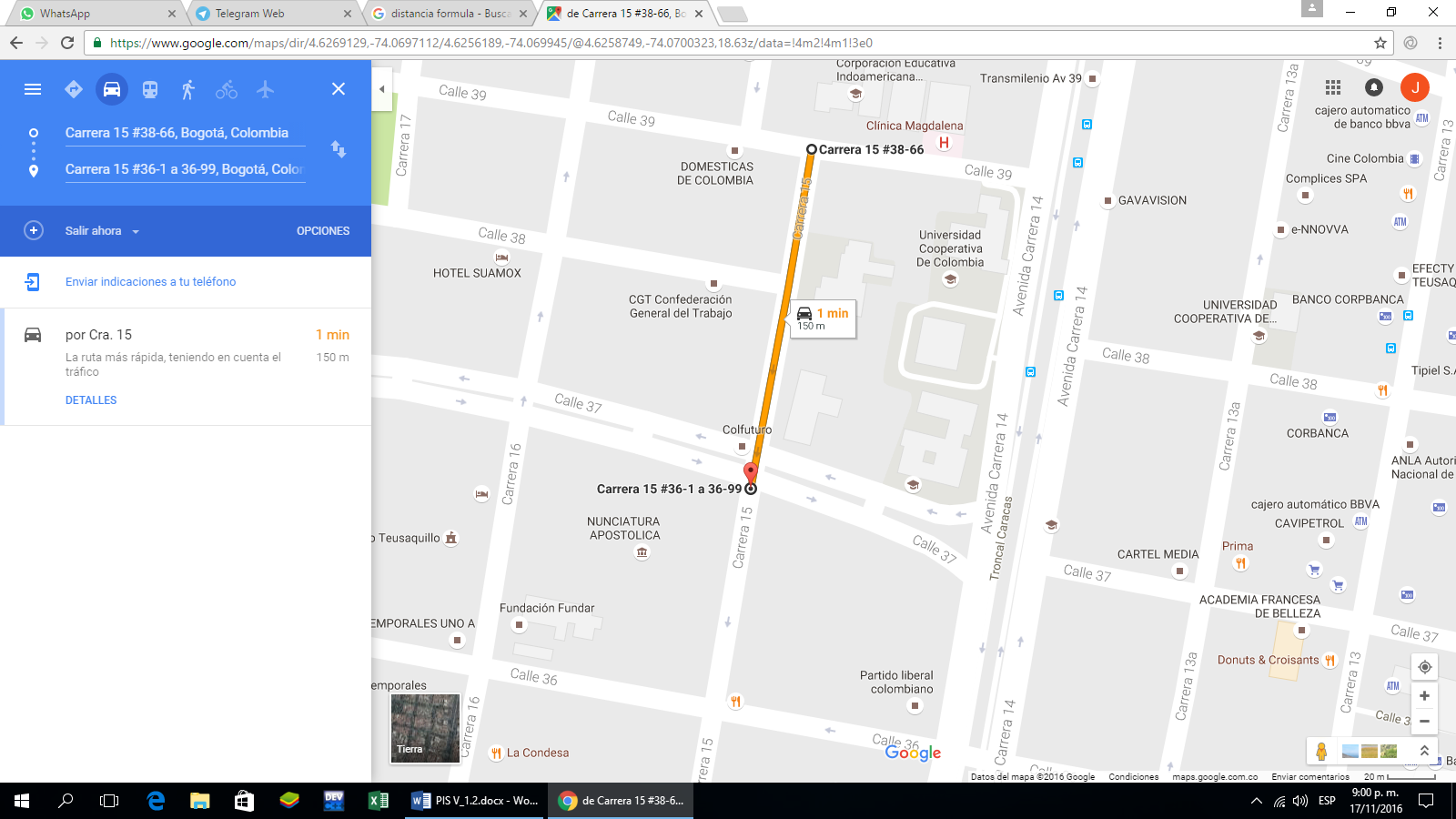
La aplicación funciona gracias a la geolocalización, de tal manera que internamente y para el envío de alertas sólo a los usuarios que puedan estar afectados realiza un cálculo con la fórmula de distancia:

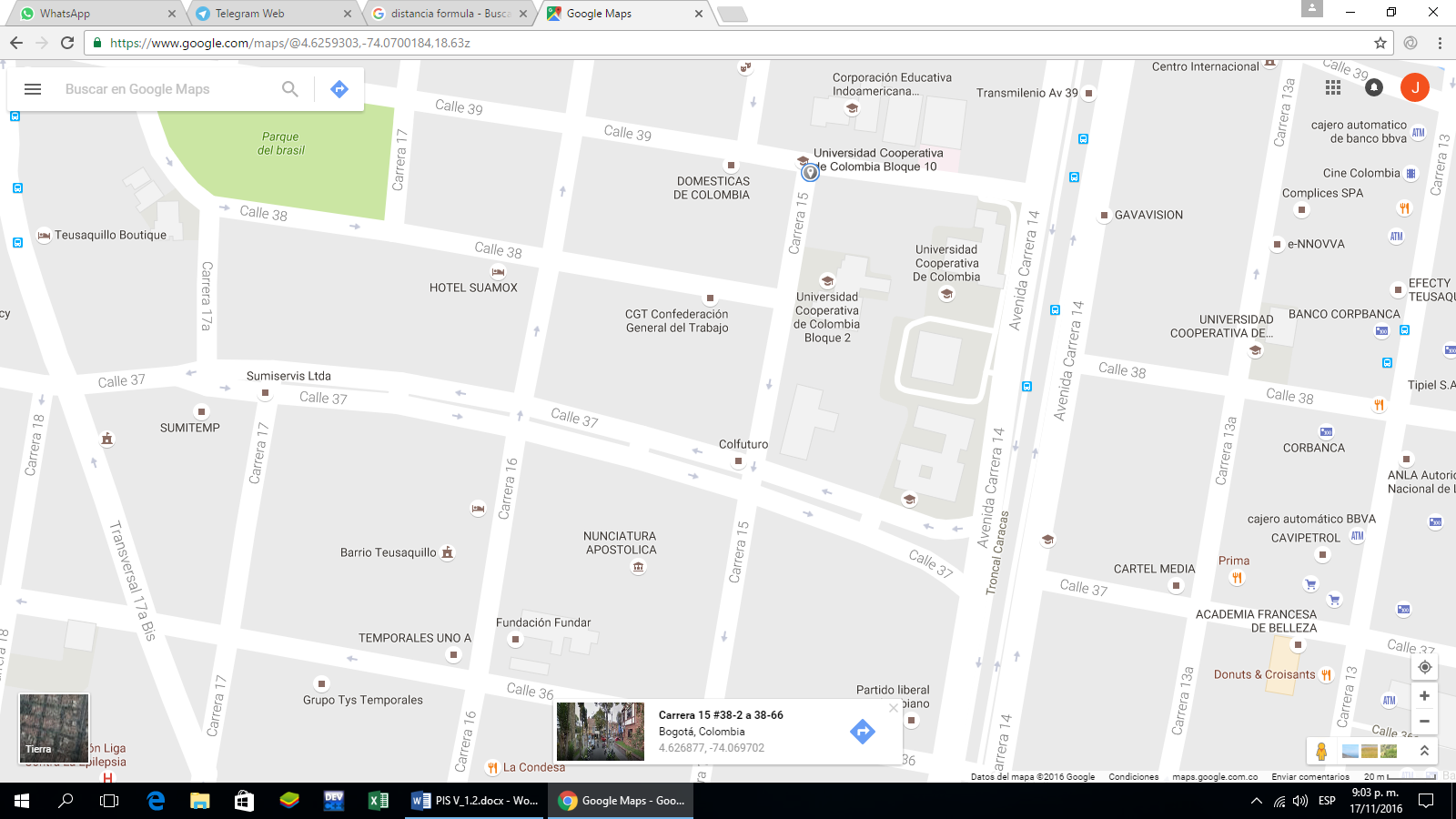


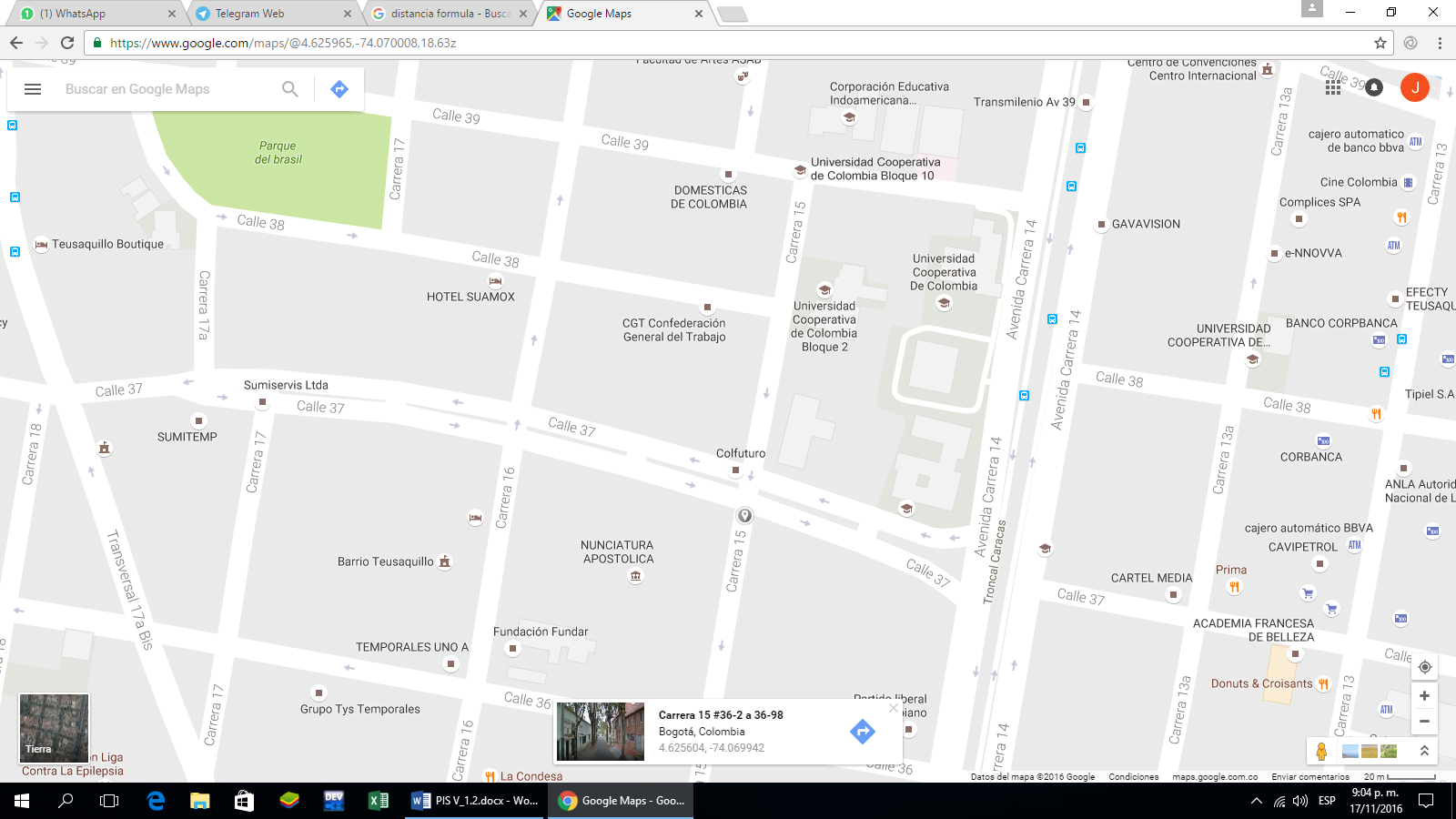
En donde un punto es la ubicación establecida por el usuario y el otro punto es la ubicación de alguna alerta.

Realizando pruebas se ha obtenido una constante la cual se llamara “constante de distancia”, la cual según las pruebas realziadas tiene un valor de 0.0012954. Éste valor se apróxima a 150 metros de distancia, por lo cuial, se en algún caso el cálculo de distancia da menor a éste valor, la aplicación deberá alertar al usuario.

Ejemplo:







En la parte inferior de estas dos imágenes observamos las coordenadas de cada uno de los puntos, usando la fórmula de distancia encontramos lo siguiente:

Este valor se usa como referencia, cualqioer cálculo de distancia que resulte inferior a este número significará que la ubicación de la alerta es menor a 150 metros de la ubicación establecida por el usuario por lo cual la aplicación debe generar la alerta.