FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL MONITOREO DE LA CALIDAD DEL AIRE EN QUITO

AUTOR

DIEGO ALEJANDRO NARANJO TORRES

AÑO

2019

FACULTAD DE INGENIERÍAS Y CIENCIAS APLICADAS

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA EL MONITOREO DE LA CALIDAD DEL AIRE EN QUITO

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Sistemas de Computación e Informática

Profesor Guía

PhD. Yves Philippe Rybarczyk

Autor

Diego Alejandro Naranjo Torres

Año

2019

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido el trabajo, desarrollo de una aplicación móvil para el monitoreo de la calidad del aire en Quito, a través de reuniones periódicas con el estudiante Diego Alejandro Naranjo Torres, en el semestre 201910, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

_____

Yves Philippe Rybarczyk

Doctor en Informática

CI: 1756950976

DECLARACIÓN DEL PROFESOR CORRECTOR

“Declaro haber revisado este trabajo, desarrollo de una aplicación móvil para el monitoreo de la calidad del aire en Quito, de Diego Alejandro Naranjo Torres, en el semestre 201910, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

_____

Bernarda Cecibel Sandoval Romo

Máster en Ciencias de la Computación

CI: 1709974453

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

_____

Diego Alejandro Naranjo Torres

CI: 1718004045

AGRADECIMIENTOS

A mi tutor Yves Rybarczyk y a la Doctora Rasa Zalakeviciute por toda la ayuda brindada y las explicaciones en el transcurso del desarrollo de este proyecto, de igual manera a mis amigos de la Dirección de Tecnologías de la Ex SH, Jhonny, Andrés, Diego y Robert por toda su ayuda.

DEDICATORIA

A mis padres, mi hermano Paúl y a mi novia Daniela, por ser mi apoyo incondicional, ustedes han sido mi fuerza para seguir adelante y culminar con éxito esta etapa tan importante de mi vida, son sin duda los pilares que me han sostenido para no rendirme y convertir mis sueños en realidad.

RESUMEN

La contaminación del aire es un problema a nivel mundial y es uno de los principales causales de enfermedades de nivel respiratorio con alta mortalidad especialmente en niños y adultos mayores, actualmente el Distrito metropolitano de Quito (DMQ) no cuenta con información más precisa y sectorial sobre el nivel de calidad del aire en la ciudad, a través de la Secretaría del Ambiente y la implementación de 9 estaciones de monitoreo se posee información de la calidad del aire pero en un radio reducido, es así que no se puede conocer los sectores con mayor nivel de contaminación, además es importante informar a los ciudadanos independientemente de su ubicación a que nivel de contaminación están expuestos y de esta manera evitar futuras enfermedades relacionadas con este problema. Recientemente se realizó por parte de la Secretaría del Ambiente una investigación sobre la calidad del aire en Quito, tomando en cuenta su importancia como la capital de Ecuador, este estudio dio como resultado sobre los niveles a largo plazo de la contaminación por partículas finas no solo superan los niveles recomendados de 10 g-m de la Organización mundial de la Salud (OMS), sino que también son más altos que los estándares nacionales de 15 g-m3, esto refleja las tendencias globales de la urbanización. (Lugo, Arias 2018) El desarrollo tecnológico que se vive actualmente, ha propiciado la aparición de dispositivos móviles, con grandes capacidades de cómputo y numerosos sensores. Vivimos en la era de las comunicaciones en red, actualmente no existe en el país una solución como esta, al detectar los eventos de forma local dispositivos móviles. El presente Trabajo de Titulación busca dar una solución al problema del monitoreo y a la predicción de la contaminación del aire. Para obtener la información, se ejecutaron varios módulos en la aplicación para la recolección de datos, los cuales se obtuvieron por los métodos que a continuación se detallan: datos del tiempo de tráfico administrado por Google Maps, el manejo de Inverse Distance Weighting (IDW), además se aplicará un enfoque de tipo Machine Learning con los resultados obtenidos se elaborará un modelo a fin de obtener la técnica de aprendizaje supervisado con mejor desempeño.

ABSTRACT

The effects of the rapid growth of the world population turn into excessive use and scarcity of natural resources, deforestation, climate change and especially environmental pollution. Currently, more than half of the world's population lives in urban areas, and this number is expected to grow to about 66% by 2050, mainly due to urbanization trends in developing countries. A recent study on air quality in Quito, the capital of Ecuador, coincides at long-term levels. Higher than the national standards of 15 g-m3. The situation has diminished due to the efforts of local and national governments in the last decade, in some places of the city. The latter reflects the global trends of urbanization. It is also based on the use of different monitoring stations distributed in certain parts of the city by the Ministry of Environment. Also applies *Machine Learning* to obtain a more accurate model. Currently, there is no such solution in the country, by detecting events locally in the device itself. This is a way of demonstrating that there is a solution for users not to expose themselves to high levels of pollution and in this way to provide greater privacy, as well as a sensible use of resources. To obtain the information, several modules were executed in the application for the data collection, which were obtained by the following methods are detailed: the data of the time administered by Google Maps, the handling of (Inverse Distance Weighting IDW), method used for the interpolation between the user's location and the monitoring stations. With the results, a model was developed and a supervised learning technique with better performance was obtained.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Capítulo I. Introducción

1.1. Antecedentes

La contaminación del aire es un problema a nivel mundial y es uno de los principales causales de enfermedades de nivel respiratorio con alta mortalidad especialmente en niños y adultos mayores, los niveles de contaminación urbana pueden verse influenciados en gran medida por las condiciones meteorológicas y las fuentes de los principales contaminantes. Existe una correlación positiva entre las concentraciones promedio diarias de $PM_{2.5}$ y la HR en áreas centrales con mucho tráfico, y una correlación negativa en las afueras de la ciudad en áreas más industriales. Mientras que en los sitios de tráfico los fuertes eventos de precipitación (> 9 mm) jugaron un papel importante en la eliminación de la contaminación de $PM_{2.5}$, en las afueras de la ciudad, las concentraciones de $PM_{2.5}$ disminuyeron al aumentar la HR independientemente de la acumulación de lluvia. (Lugo, Arias 2018)

Cabe esperar un aumento de las concentraciones de $PM_{2.5}$ en cualquier ciudad altamente motorizada donde hay alta HR y falta de fuertes precipitaciones, especialmente en países de rápido crecimiento y en desarrollo con alta motorización debido a la mala calidad del combustible.

Quito se encuentra entre las cinco ciudades más pobladas en las altas elevaciones del mundo. Se extiende de norte a sur sobre una amplia pendiente del volcán Pichincha (cuya cima es de 4800 msnm), que forma el arte de los Andes, y tiene una elevación promedio de 2815 msnm. La ciudad está experimentando actualmente una expansión del área metropolitana (4218 km2). La ciudad se extiende a lo largo de varias terrazas que varían en elevación entre 2700 y 3000 msnm, y se expande a los valles circundantes que se ubican en 2300-2450 msnm. (Lugo, Arias 2018)

En virtud a lo mencionado anteriormente sobre la expansión de la ciudad y debido a su ubicación en el Ecuador, la altura solar al mediodía es de 66.5 ° F

y 90 ° F durante todo el año con altos niveles de intensidad solar. Sin embargo, gracias a la elevación, tiene un clima primaveral durante todo el año con una temperatura promedio de 14.5 ° C.

Quito se encuentra en una región tropical de gran altitud y, por lo tanto, tiene dos estaciones: seca (junio-agosto, con un promedio de precipitación de alrededor de 14 mm / mes) y lluviosa (septiembre-mayo, con un promedio de precipitación de 59 mm / mes) con La mayor parte de las precipitaciones se producen en períodos prolongados.

La contaminación en la ciudad puede ser generada por diferentes fuentes, como el tráfico, la combustión de madera, la minería, los parques industriales y las centrales termoeléctricas. El centro de Quito está predominado por áreas residenciales con tráfico vehicular pesado.

En este decimosexto año de operaciones en Quito se ejecutó el cambio y la actualización del equipamiento de la Red Metropolitana de Monitoreo Atmosférico de Quito (REMMAQ), cuyas operaciones permitirá continuar con el monitoreo, interpretación de los resultados necesarios y el análisis todo esto para orientar las políticas y resoluciones de reducción de los niveles de aquellos contaminantes que pueden afectar la salud de los grupos más vulnerables de la ciudad.

Para determinar la calidad del aire es importante analizar los principales contaminantes en Quito y sus fuentes los cuales influyen en la determinación del nivel de calidad del aire, los principales son:

Dióxido de Azufre (SO2): El dióxido de azufre en la ciudad viene principalmente de las emisiones de termoeléctricas e industria. Las fuentes móviles en la ciudad representan un porcentaje menor de las mismas. (Lugo, Arias 2018)

Monóxido de Carbono (CO): Las emisiones de monóxido de carbono en la ciudad son en su gran mayoría provenientes del tráfico vehicular de automotores a gasolina. Las mayores concentraciones se las encuentra en las

horas y meses con menores temperaturas, debido a un mayor efecto de los arranques en frío. (Lugo, Arias 2018)

Ozono (O3) El ozono troposférico (O3): Se forma por reacciones químicas en el aire entre los hidrocarburos y los óxidos de nitrógeno bajo la influencia de la luz solar. En Quito, se han registrado las concentraciones de ozono más altas durante el mes de septiembre, coincidiendo con el equinoccio y características meteorológicas propicias para una mayor insolación. Los meses con menores concentraciones de ozono son los correspondientes a período con mayor intensidad de lluvias y días nublados (abril, mayo). (Lugo, Arias 2018)

Óxidos de Nitrógeno (NOx): Los óxidos de nitrógeno (NOx) es la suma de óxido nítrico (NO) y dióxido de nitrógeno (NO2). Las emisiones en ciudad provienen principalmente del tráfico vehicular. Estas emisiones contienen óxidos de nitrógeno donde aproximadamente el 80 % es monóxido de nitrógeno (NO). Sin embargo, este se transforma rápidamente a dióxido de nitrógeno (NO2). La proporción de NO2 de NOx aumenta cuando existe mayor ozono en el ambiente. Debido a que este acelera el proceso químico donde el NO se convierte en NO2. (Lugo, Arias 2018)

Benceno: El benceno forma parte de los compuestos orgánicos volátiles (VOC) y es un carcinógeno humano. Las emisiones se deben principalmente al tráfico vehicular, por autos a gasolina.

El benceno se produce por la combustión incompleta de los combustibles, el aceite lubricante del motor y adicionalmente por la evaporación de los combustibles del sistema del vehículo.

Esto ocurre cuando se realiza una conducción alborotada o luego de finalizar la conducción cuando el vehículo está caliente. La Norma Ecuatoriana NTE INEN 935 establece como contenidos máximos de benceno, el 1 y 2% en volumen, para las gasolinas de 87 octanos (Extra) y de 92 octanos (Súper), respectivamente. (Lugo, Arias 2018)

Los análisis estadísticos de los datos diarios recolectados en 2018 en la ciudad de Quito muestran que el promedio de $PM_{2.5}$ se eleva a 17.1 ± 6.05 | jg / m 3 (más de 17.0 ^ g / g) m 3) con relación a los años anteriores. Quito ha estado experimentando algunos cambios en el clima, con un aumento de la temperatura ambiente de 13.5 ° C a 14.8 ° C en la última década. En el mismo período, el promedio anual de HR disminuyó de un 76% a un 64% en todos los sistemas monitoreados. (Lugo, Arias 2018)

La tendencia general decreciente en las concentraciones de $PM_{2.5}$ se registra en los distritos de Belisario y Camal en un 12,6% y 2,8%, respectivamente, mientras que en el norte de Cotocollao y Carapungo, vemos un aumento constante del 12,9% y el 11,4%, respectivamente. Estos cambios en las concentraciones anuales se deben a las regulaciones de combustible y tráfico implementadas en la ciudad de Quito y todo el Ecuador, y al crecimiento continuo de la ciudad en las afueras.

Los contaminantes de criterio gaseoso tienden a ser elevados dependiendo de la ubicación o la estación, si bien la ciudad sufre de una alta contaminación de $PM_{2.5}$ a largo plazo, principalmente debido a que las fuentes móviles provienen de combustibles de baja calidad y tecnologías antiguas, tiene recursos limitados para estudiar la composición química de las partículas.

Para confirmar la correlación de $PM_{2.5}$ con otras emisiones de fuentes móviles comunes en un área urbana, la Secretaría del Ambiente realizó el análisis de correlación de los datos de concentración de $PM_{2.5}$, estos resultados contrastan en los diferentes sitios de ubicación de las estaciones, la mejor correlación lineal se obtuvo en la estación de Belisario.

Estos resultados apoyan el hecho de que estos contaminantes se originan en gran medida de fuentes creadas por el hombre, siendo los vehículos el principal contribuyente.

Este análisis ayudó a identificar más áreas relacionadas con el tráfico en la ciudad. Específicamente, los altos factores de enriquecimiento de Zinc (Zn),

Vanadio (V), Níquel (Ni), Arsénico (As) y Plomo (Pb) implicaron que las emisiones industriales y vehiculares son una fuente importante en el cobertizo de aire urbano de Quito, la Secretaria del Ambiente monitorea la calidad del aire en sitios puntuales pero también atiende a denuncias de la población.

Para el análisis en primer lugar se toman muestras de aire ambiente las cuales son enviados a analizadores los cuales identifican los diversos contaminantes del aire que se encuentran en la muestra, en la estación se cuenta con analizadores de diversos contaminantes como ozono, PM_{10}, material particular $PM_{2.5}$, temperatura, velocidad del viento, dirección del viento entre otros.

En virtud a lo expuesto anteriormente y con el fin de tomar medidas eficaces para reducir el impacto de la contaminación atmosférica se requiere un análisis del estado de la calidad de aire, sus causas de mayor polución, cómo se transportan los contaminantes y transformados en la atmósfera, y cómo impactan los seres humanos, los ecosistemas y el clima.

Las políticas eficaces de calidad del aire requieren cambios estructurales y cambios de comportamiento en el que están involucrados todos los habitantes del DMQ. El presente trabajo se concentra en desarrollar una aplicación móvil para la visualización de niveles de contaminación alrededor de un área de la ubicación geoespacial de una persona.

1.2. Alcance

El proyecto consiste en desarrollar una aplicación móvil que muestre la concentración de contaminantes atmosféricos en un rango de espacio determinado. Para hacerlo, la aplicación tendrá que calcular en base a algunos datos proporcionados por la Secretaría del Ambiente, así como datos del modelo matemático a implementarse.

Dado que hay un número limitado de estaciones de monitoreo (solo 9 estaciones) se tendrá que usar Inverse Distance Weighting Interpolation (IDW), el cual será nuestro método de cálculo para la estimación de concentración de contaminantes en la ciudad dentro de la aplicación.

Para optimizar el acceso a la información, el cálculo de la interpolación se enfocará en el área en la que se encuentra el usuario, esto significa que la aplicación también tendrá que capturar las coordenadas GPS del dispositivo móvil, además se tomará datos de tráfico para sumarla al modelo matemático y que el resultado sea más preciso, todo esto dará como resultado la visualización de la cantidad de contaminación que rodea al usuario.

Parte del proyecto es la construcción de varios modelos basados en el aprendizaje automático (del más sencillo al más complejo), para determinar el nivel de precisión y las variables más influyentes y descartables para un resultado final afinado.

La precisión de estos modelos serán validadas por comparación con mediciones realizadas en el terreno con la utilización del equipo portátil de monitoreo de la contaminación denominado MicroDust perteneciente a la Universidad de la Américas.

En este contexto la aplicación será desarrollada de manera nativa para dispositivos ANDROID y se usarán APIs y servicios propios para este sistema operativo, cabe recalcar que en su fase inicial el aplicativo no será cargado directamente en la Google Store para su uso masivo ya que será de uso primario en la Secretaría del Ambiente para el análisis de posibles nuevas funcionalidades antes de implementarse.

1.3. Justificación

Las concentraciones de la mayoría de los contaminantes atmosféricos

disminuyeron en la última década, comparando con normas de calidad de aire nacionales Dióxido de Carbono (CO): 54%, Dióxido de Azufre (SO2): 77%, Ozono (O3): 30%, Dióxido Nitroso (NO2): 14,6%, Material Particulado 2.5 (PM10): 44% y Material Particulado 2.5 (PM2.5): 33%).

Varias acciones contribuyeron a esta disminución como son: la Revisión Técnica Vehicular, controles públicos a las industrias de alto impacto y la mejora de los combustibles, lo que permitió el acceso a mejores tecnologías vehiculares. (Zalakeviciute, Rybarczyk, López, Suarez, 2018, p. 66-75)

Actualmente no ha sido posible cumplir con todos los estándares de calidad ambiental. La principal emisión que afecta la calidad del aire en el DMQ es la producida por fuentes móviles, cuyo parque vehicular mantiene un crecimiento continuo anual superior al 7%.

Este incremento provoca un aumento del tráfico vehicular dificultando el cumplimiento de normas de calidad. A nivel internacional, la Organización Mundial de la Salud (OMS) emite directrices sobre Calidad del Aire, las mismas que constituyen el análisis más consensuado y científicamente respaldado sobre los efectos de la contaminación en la salud y en las que se incluyen los parámetros de calidad del aire que se recomiendan para una disminución significativa de los riesgos sanitarios.

Los habitantes de la ciudad de Quito desconocen la magnitud del nivel de contaminación al que se hallan expuestos en el día a día; sin embargo, gracias a la construcción de este modelo de medición podrían conocer los niveles y tomar medidas preventivas o correctivas necesarias.

El uso de modelos basados en aprendizaje supervisado ayudará a predecir la contaminación en el aire a partir de la información del tráfico, relacionado también con datos meteorológicos, podemos observar en la tabla 1 los efectos negativos que provocan todos estos contaminantes en la salud de las personas:

Tabla 1. Componentes químicos en el aire y efectos a la salud

Contaminante	Características	Fuentes Principales	Efectos sobre la Salud
Partículas sedimentables	Material particulado en general de tamaño mayor a 10µm. Partículas gruesas de tierra y polvo tóxicos.	Erosión eólica y tráfico en vías sin pavimento, actividades de construcción, molienda y aplastamiento de rocas.	Exposición continua a altas concentraciones causa irritación de garganta y mucosas.
PM_{10}	Material particulado suspendido de diámetro menor a 10 µm. Partículas de material sólido o gotas líquidas suspendidas en el aire. Puede presentarse como polvo, niebla, aerosoles, humo, hollín, etc.	Erosión eólica, tráfico en vías sin pavimento y actividades de construcción. Procesos de combustión (industria y vehículos de automoción).	Produce irritación de las vías respiratorias, agrava el asma y favorece las enfermedades cardiovasculares. Se relaciona con la silicosis y asbestosis. Causa deterioro de la función respiratoria (corto plazo). Asociado con el desarrollo de enfermedades crónicas, cáncer o muerte prematura (largo plazo).
$PM_{2.5}$	Material particulado suspendido menor a 2.5 µm.	Procesos de combustión (industrias, generación termoeléctrica). Incendios forestales y quemas. Purificación y procesamiento de metales.	Tiene la capacidad de ingresar al espacio alveolar o al torrente sanguíneo incrementando el riesgo de padecer enfermedades crónicas cardiovasculares y muerte prematura.
SO_2	Gas incoloro de olor fuerte. Puede oxidarse hasta SO3 y en presencia de agua formar H2SO4. Importante precursor de sulfatos e importante componente de partículas respirables.	Procesos de combustión. Centrales termoeléctricas, generadores eléctricos. Procesos metalúrgicos. Erupciones volcánicas. Uso de fertilizantes.	Altas concentraciones ocasionan dificultad para respirar, conjuntivitis, irritación severa en vías respiratorias y en pulmones. Causante de bronco constricción, bronquitis, traqueítis y bronco espasmos, agravamiento de enfermedades respiratorias y cardiovasculares existentes y la muerte.
CO	Gas incoloro, inodoro e insípido.	Procesos de combustión incompleta. Los vehículos a gasolina constituyen la fuente más importante.	La hipoxia (falta de oxígeno) producida por inhalación de CO, puede afectar al corazón, cerebro, plaquetas y endotelio de los vasos sanguíneos. Asociado a disminución de la percepción visual, capacidad de trabajo, destreza manual.
NO_2	Gas rojizo marrón, de olor fuerte y penetrante. Puede producir ácido nítrico, nitratos y compuestos orgánicos tóxicos.	Procesos de combustión (vehículos, plantas industriales, centrales térmicas, incineradores).	Causa irritación pulmonar, bronquitis, pulmonía, reducción significativa de la resistencia respiratoria a las infecciones. Exposición continua a altas concentraciones incrementa la incidencia en enfermedades respiratorias en los niños, agravamiento de afecciones en individuos asmáticos y enfermedades respiratorias crónicas.
Benceno	El benceno es un líquido incoloro, que se evapora al aire muy rápidamente, es muy inflamable y de aroma dulce.	Incendios forestales, es un componente natural del petróleo crudo, gasolina, el humo de cigarrillo y otros materiales orgánicos que sean quemados.	Niveles muy altos puede causar la muerte. Niveles bajos pueden causar somnolencia, mareo y taquicardia. Exposición de larga duración puede causar anemia. Puede producir hemorragias y daños en el sistema inmunitario. Es un reconocido cancerígeno.
Cadmio	Metal que por lo general se encuentra combinado con otros componentes como el oxígeno.	Producción de metales, baterías, plásticos, humo de cigarrillo.	Niveles altos de cadmio puede dañar gravemente los pulmones. Exposición prolongada a niveles más bajos de cadmio en el aire, produce acumulación de cadmio en los riñones y posiblemente enfermedad renal. El cadmio y los compuestos de cadmio son carcinogénicos.
Mercurio inorgánico (vapores)	Metal que existe en forma natural en el ambiente y que tiene varias formas químicas.	Extracción de depósitos minerales, al quemar carbón y basura de plantas industriales. Por liberación de mercurio durante tratamientos médicos o dentales.	La inhalación de vapor de mercurio, de ser mortal por inhalación y perjudicial por absorción cutánea. Puede tener efectos perjudiciales en los sistemas nervioso, digestivo, respiratorio e inmunitario y en los riñones, además de provocar daños pulmonares.

Tomado de Secretaría del Ambiente, s.f.

Es importante dar las facilidades a los habitantes de la ciudad de Quito para que puedan informarse sobre los niveles de contaminación presentes a su alrededor para mitigar los efectos negativos en la salud, esto se pretende realizar mediante el desarrollo de la aplicación móvil, esto tomando como base el uso mayoritario de los teléfonos inteligentes.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un aplicativo móvil que muestre la concentración de contaminantes atmosféricos dentro de un área alrededor del posicionamiento del usuario en la ciudad de Quito.

1.4.2. Objetivos específicos

- Realizar mediante Inverse Distance Weighting (IDW) la interpolación de la cantidad de contaminación de manera sectorial.
- Aumentar la precisión de la predicción mediante el consumo de información de tráfico y datos meteorológicos en tiempo real.
- Utilizar el método de aprendizaje automático (Machine Learning) para completar el modelo predictivo en caso de estimación demasiado aproximativo por el IDW.
- Levantar la información de requerimientos funcionales y no funcionales para el diseño de la aplicación.
- Desarrollar una interfaz móvil para observar los niveles de gases contaminantes alrededor del posicionamiento global de los ciudadanos del Distrito Metropolitano de Quito.
- Verificar mediante pruebas de campo el correcto funcionamiento del aplicativo.

1.5. Metodologías por utilizar

En el presente proyecto de titulación se aplicará varias metodologías para el desarrollo del aplicativo así como para la construcción del mejor modelo para determinación del nivel de contaminación alrededor del usuario.

Para el desarrollo del aplicativo se ha tomado a SCRUM como metodología de desarrollo a aplicarse, seguiremos los protocolos y buenas prácticas que esta metodología nos indica para obtener como resultado final el aplicativo totalmente funcional.

Por otro lado usaremos regresiones lineales o ajuste lineal para determinar mediante los diferentes valores y características que se obtendrán posterior a la toma de datos en campo y las proporcionadas por la Secretaría del Ambiente el nivel de relación de dependencia de las variables con respecto a la determinación del nivel de contaminación alrededor del usuario.

Para la determinación de las interpolaciones se hará uso de la ponderación de distancia inversa (Inverse Distance Weighted IDW), este método determinista nos presentará el valor de nivel de contaminación con relación a la ubicación del usuario. El aprendizaje automático o Machine Learning será determinante en la realización de esta tesis ya que de esta manera podremos construir los modelos de predicción de la calidad del aire y determinar el mejor para la implementación en el aplicativo.

2. Capítulo II: Marco Teórico

2.1. Dispositivos móviles.

El desarrollo tecnológico vivido en los últimos años ha propiciado la aparición de dispositivos móviles, con grandes capacidades de cómputo y numerosos sensores.

Los llamados Smartphone surgen en la primera década del siglo XXI. En 1999, la compañía japonesa NTT DoCoMo fabricó el primer Smartphone, capaz de lograr una gran tasa de adopción entre los habitantes de un país. Se denomina Smartphone a los teléfonos inteligentes, con gran capacidad para el manejo de datos, mayor conectividad y mejores prestaciones que los teléfonos móviles convencionales. (Rose, 2001)

Gracias al avance de la tecnología, y lo aprendido con los Smartphone, se han desarrollado distintos dispositivos tales como: Tablet, Smartwatch, e infinidad de dispositivos electrónicos. Nos encontramos en la era del always on, es decir, siempre conectados. Todos estos dispositivos electrónicos tienen algo en común: su gran adopción por parte de los usuarios y su gran capacidad para la obtención y manejo de datos de diferentes sensores (similar a una computadora en miniatura).

En el Ecuador 8,1 millones de personas usan teléfonos inteligentes según datos de la ARCOTEL, en el mercado podemos encontrar dispositivos que cuentan con conectividad a internet, ya sea mediante Wi-Fi o internet móvil (2G, 3G, 4G), sistema de posicionamiento global o GPS para la detección de la ubicación del dispositivo, acelerómetros y brújulas capaces de detectar movimientos y aceleraciones. A continuación, se muestra una breve descripción de aquellos sensores estudiados para la toma de datos y detección de eventos.

2.1.1. Sistema de localización: GPS

Sistema de posicionamiento global, que funciona mediante una red de 24 satélites situados en la órbita del planeta Tierra. Su exactitud permite determinar la posición de un objeto con una precisión del orden de centímetros, en cualquier lugar del planeta. Para determinar la posición de dicho objeto, el receptor se encarga de localizar al menos 3 satélites de la red.

Una vez localizados se envía una señal a cada uno de ellos, sincronizando así el reloj del GPS, para posteriormente medir el retardo entre las distintas señales. Este retardo permite conocer la distancia a cada uno de los satélites. Pudiendo así triangular el lugar exacto donde se sitúa el objeto. (GPS Wiki, 2018)

A pesar de la existencia de otros sistemas de navegación (Galileo desarrollado por la UE o GLONASS gestionado por Rusia), GPS es el más extendido. Su generalización en el mundo de los dispositivos móviles ha permitido el desarrollo de diferentes aplicaciones. La mayoría de estos dispositivos cuentan con un chip GPS, que permite a los desarrolladores utilizar la ubicación en sus aplicaciones, lo cual incrementa las opciones durante el proceso de desarrollo.

2.1.2. Conectividad: Redes inalámbricas y móviles

El Internet móvil es posible gracias a las denominadas redes inalámbricas y móviles. Podemos diferenciar, de forma general, dos tipos de conexiones para la obtención de internet móvil: redes Wi-Fi y redes de Banda Ancha Móvil (BAM). Las redes Wi-Fi permiten la conexión inalámbrica de dispositivos móviles, a través de un punto de acceso de red. El alcance es de aproximadamente 20 metros. (Bernardos, 2015)

Por otro lado, las redes de BAM, funcionan a partir de la cobertura móvil. Según la tecnología que implementen pueden ofrecer distintos anchos de banda (GPRS, 3G, 4G). Ambas tecnologías son ampliamente utilizadas, tanto, que en los últimos años el crecimiento de líneas fijas con acceso a Banda Ancha Fija (potenciales de ofrecer Wi-Fi), y móviles con acceso a Banda Ancha Móvil, ha sido desde 2010 hasta 2014 bastante notable.

A continuación en la figura 1, los datos de penetración en el Ecuador, donde podemos notar las expansión y crecimiento de la conectividad móvil.

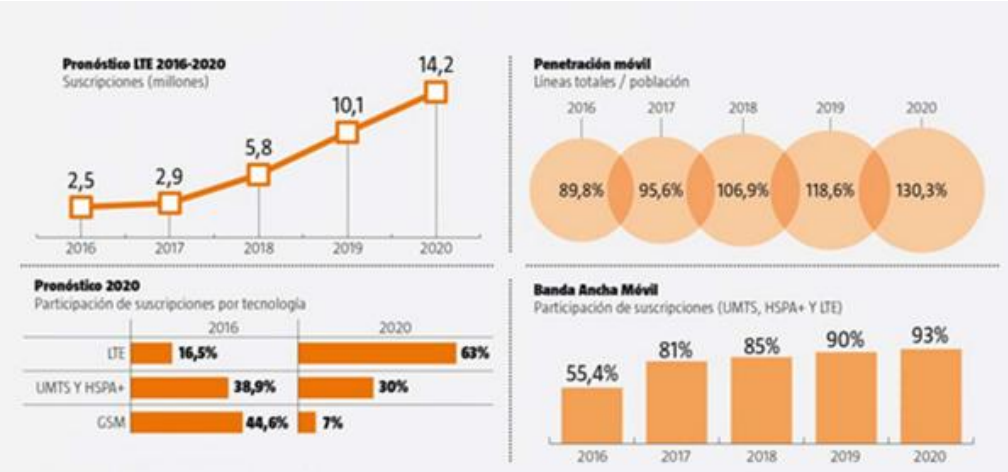


Figura 1. Evolución de la Tecnología Móvil en el Ecuador
Tomado de ITC S.A., s.f.

Además, en función de la tecnología utilizada, se distingue entre diferentes tipos de conexiones móviles, en la tabla 2 y 3 observamos la aclaratoria sobre las características para cada tipo de conexión.

Tabla 2. Características estándares de la conexión Wi-Fi

Estándar Wi-Fi	Velocidad máxima	Banda de Frecuencia
802.11	Entre 1 y 2 Mbps	2.4 Ghz
802.11a	54 Mbps	5.15 Ghz
802.11b	11 Mbps	2.4 Ghz
802.11g	54 Mbps	2.4 Ghz
802.11n	> 100 Mbps	2.4 Ghz – 5 Ghz

Tomado de ITC S.A., s.f.

Tabla 3. Características de la tecnología de redes móviles.

Tecnología	Velocidad promedio de bajada	Velocidad promedio de subida
GSM (2G)	1.8 kB/s	1.8 kB/s
GPRS (2.5G)	7.2 kB/s	3.6 kB/s
EDGE (2.75G)	29.6 kB/s	29.6 kB/s
UMTS (3G)	48 kB/s	48 kB/s
EDGE	59.2 kB/s – 237 kB/s	59.2 kB/s – 118 kB/s
HSPA (3.5G)	1.706 kB/s – 5.25 MB/s	720 kB/s – 1.437 MB/s
LTE	21.625 MB/s – 40.750 MB/s	7.25 MB/s – 10.750 MB/s

Tomado de ITC S.A., s.f.

2.1.3. Conclusiones sobre los dispositivos móviles

Tras realizar un análisis sobre los dispositivos móviles, y el estado de la tecnología que incorporan, podemos concluir:

– Que la tecnología móvil y en concreto los Smartphone, tiene una gran acogida en la sociedad, contando en la ciudad de Quito con una de las mayores tasas de penetración.

– Que la tecnología incorporada en los dispositivos móviles está muy desarrollada. Los Smartphone cuentan con numerosos sensores, que les permitirán interaccionar con el medio y recabar información.

– Que el gran desarrollo del acelerómetro, por su precisión, y su bajo consumo de batería, permite que sea la opción elegida para este proyecto, por delante de otras opciones como: La captación de imágenes, y su posterior análisis, o el uso intensivo del GPS.

2.2. Desarrollo de software en dispositivos móviles

Aunque no hay una fecha exacta para datar el desarrollo de las primeras aplicaciones en dispositivos móviles, sí que se puede determinar que su incremento e importancia creció con la aparición de los llamados Smartphone.

Se denomina aplicación móvil o APP, al software desarrollado y diseñado para ser ejecutado en dispositivos móviles, tales como Smartphone, Tablet o todo tipo de Gadget informático. Las APP son programas diseñados para facilitar al usuario realizar una tarea, ya sea de ocio, educativa, trabajo, acceso a diferentes servicios, etc. Este tipo de aplicaciones suelen estar ubicadas en los dispositivos, y están escritas en diferentes lenguajes de programación según su plataforma. En las próximas secciones, se realizará una introducción

a los diferentes sistemas operativos existentes en el mercado del software para dispositivos móviles.

Todo esto para posteriormente centrarse en Android (el SO seleccionado para el desarrollo de este proyecto) y algunas herramientas de software adicional que han resultado interesantes durante el desarrollo de este proyecto.

2.2.1. Introducción al desarrollo en dispositivos móviles

Para entender la magnitud del mercado del desarrollo en dispositivos móviles, es importante conocer los diferentes SO con mayor cuota de mercado en la actualidad.

Android: Lanzado al público por primera vez en septiembre de 2008, surge como un SO móvil basado en el núcleo de Linux y creado como software libre, tanto para usuarios como para desarrolladores.

La principal meta de este SO es mejorar la experiencia de los usuarios. Comprado en 2007 por Google, en la actualidad Android es uno de los agentes más importantes del mercado de los dispositivos móviles. (Cinar, 2012)

IOS: Desarrollado por Apple, sale por primera vez al público en 2007 con el lanzamiento del primer iPhone. Se caracteriza desde su primera versión por la simplicidad en su manejo, haciendo que su interfaz funcione con gran fluidez. (Morrissey, 2010)

Existen además otros SO en el mercado de los dispositivos móviles, Windows Phone de Microsoft, o BlackBerry 6 de BlackBerry, pero cuya implantación es mínima, en la tabla 4 podemos ver una estadística de las ventas de Smartphone, clasificada por SO, durante Diciembre de 2017, en algunos de los mercados mundiales más relevantes. (Kantar World Planet, 2015)

Tabla 4. Cuota de uso de Smartphones según S.O.

Sistema Operativo Móvil	Porcentaje %
Android	58,87
IOS	26,47
Windows Phone	3,09
BlackBerry	7,10

Tomado de ZK, s.f

2.2.2. Sistema Operativo Android

Basado en un núcleo Linux, y desarrollado por Google, Android es sin duda una de las mejores opciones para el desarrollo de software móvil. Si algo caracteriza a este SO es su definición como software libre, ya que uno de los objetivos de Android es llegar al mayor número, tanto de usuarios como de desarrolladores. Podemos destacar a Android por su compatibilidad entre distintos dispositivos. Por tratarse de un proyecto de software libre, es una plataforma a la cual cualquier fabricante de hardware puede acceder. Esto ayuda a los desarrolladores, ya que al lanzar sus aplicaciones no necesitan preocuparse de la compatibilidad con distintos dispositivos hardware.

Las APP Android están escritas en el lenguaje de programación Java. Gracias al SDK de Android y sus herramientas, se puede compilar el código generado de la APP. El SO operativo cuenta con una máquina virtual cuyas funciones son la gestión de memoria y recursos de forma eficiente. (Web desarrolladores, 2015)

Para su compatibilidad con distintos dispositivos, y su definición como código abierto basado en software de Linux, Android cuenta con la siguiente arquitectura de componentes:

Kernel de Linux: Es el núcleo del SO. Son los cimientos de Android sobre los que se gestionan recursos como la memoria, la capa de protocolos, los procesos y los drivers o controladores de dispositivos.

Hardware abstraction layer: Capa de abstracción del hardware. Provee una interfaz estándar para diferentes componentes hardware, como puede ser la cámara o el módulo Bluetooth.

Android Runtime: Se trata del entorno de ejecución. Cada aplicación lanza su propia instancia de Android Runtime (ART). Las características de ART principales son:

– Capacidad para la compilación antes de la ejecución, o en tiempo de ejecución.

– Optimización del recolector de basura (que se encarga de gestionar la memoria).

– Soporte para la puesta a punto, con herramientas de diagnóstico de software.

Bibliotecas: Nativas de C/C++. Se utilizan por distintos componentes del sistema. Entre sus librerías se encuentra la del sistema C, librerías multimedia para la reproducción y grabación de formatos de audio y sonido, gestores de contenidos visuales tanto en 2D como en 3D, motores de navegación web o el motor de BBDD relacionales SQLite.

Framework de aplicaciones: Son APIs de Java. Basado en APIs de código en lenguaje Java, es la puerta de acceso a todas las características disponibles en el SO Android.

Estás APIs son como bloques, que proveen lo necesario para el desarrollo de APPs, simplificando el uso de los diferentes componentes o servicios. Entre ellos:

– Un sistema de interfaz gráfica, que incluye listas, grids, cajas de texto o botones para la construcción de la interfaz de usuario.

– Un manager de recursos, que provee acceso a recursos fuera del código como, textos, gráficos y ficheros.

– Un manager de actividades, que maneja el ciclo de vida de las aplicaciones y provee una pila de navegación común.

Aplicaciones: Android cuenta con un núcleo de aplicaciones por defecto como email, SMS, calendarios, navegador, contactos y otras. Esto conlleva a que el usuario podrá instalar las aplicaciones que desee, las cuales estarán basadas en los servicios provistos por capas inferiores. Android está en constante actualización y cada día aumenta su campo de interrelación con los diferentes dispositivos móviles del mercado.

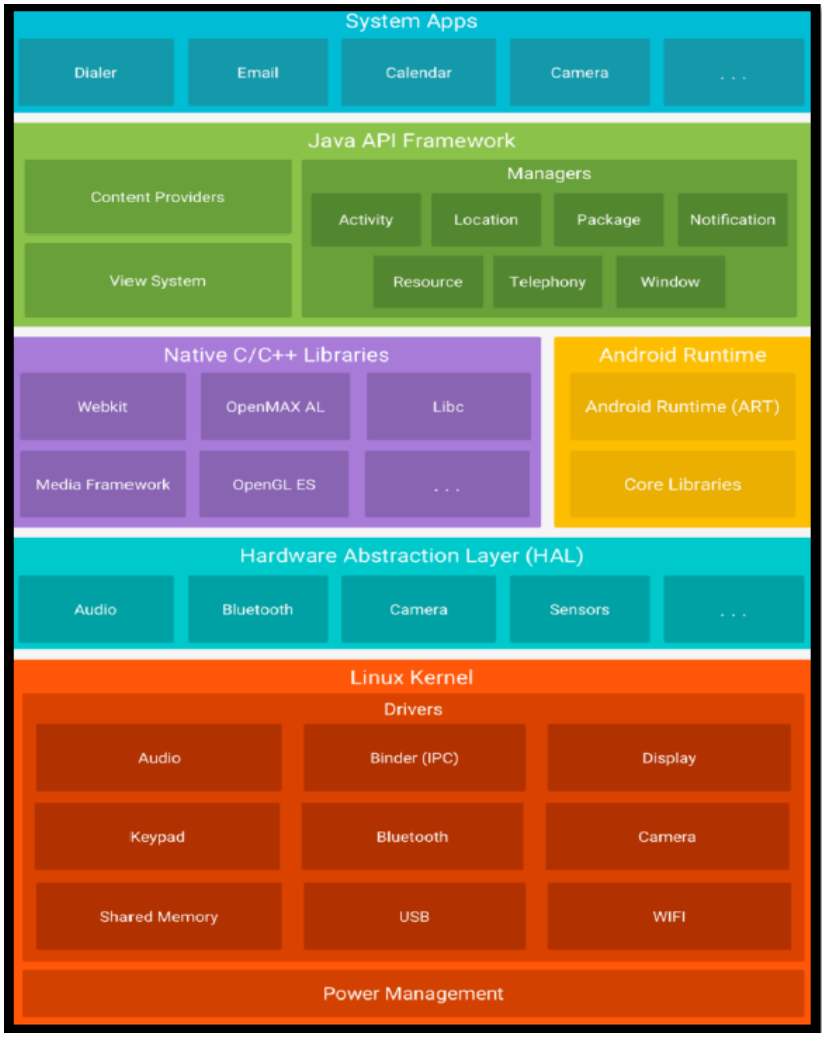


Figura 2. Pilas de capas de software por las que está formado Android. Tomado de Mundo Android, s.f.

Durante el desarrollo de una aplicación en Android se debe tener en cuenta su gestión automática de procesos. Existen tres niveles como se observa en la figura 3 que determinan la prioridad de un proceso, e indican al SO si debe eliminarse o no.

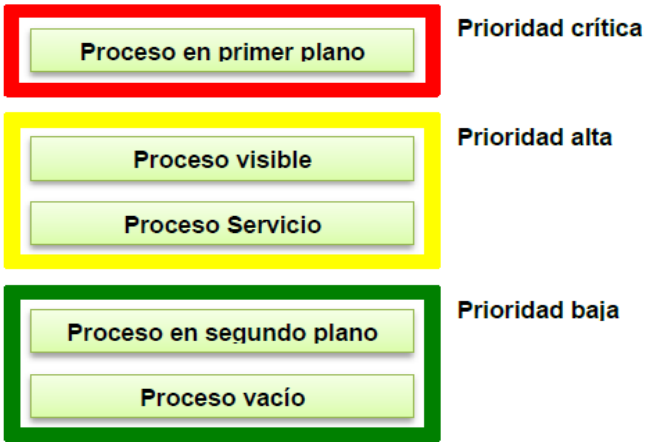


Figura 3. Jerarquía de procesos en Android.
Tomado de Mundo Android, s.f.

Existe una serie de elementos, que son clave, por resultar imprescindibles para el desarrollo en Android. Son cuatro: Activity, Service, Content Provider y Broadcast Receiver.

– Service: Permite llevar a cabo operaciones en segundo plano o background y, por tanto, no proporcionan al usuario una interfaz.

– Content Provider: Proporciona facilidades de acceso a datos estructurados. Además de encapsular los datos, proporcionan mecanismos para definir la seguridad de los datos.

– Broadcast Receiver: Permiten que se reciban notificaciones y otros datos, desde ciertos componentes. El componente que envía la notificación lo hacen a través de un sendbroadcast. Los componentes que la reciben se asocian o suscriben a un Receiver.

– Activity: Es el componente básico que permite crear módulos que interaccionan con el usuario. Cada activity tiene asociada una vista. La vista está constituida por un conjunto de elementos gráficos.

Además de estos componentes, cabe destacar los intent, que permiten el envío de mensajes entre componentes y el manifest, que es un archivo de configuración donde podemos aplicar las configuraciones básicas de una aplicación. Una aplicación está formada por uno o varios componentes y siempre al menos por un activity. Una vez lanzada la aplicación, esta pasará por diferentes estados dependiendo del uso que se le esté dando. Esta serie de estados conforman el llamado ciclo de vida de la APP como se pude apreciar en la figura 4. (Web desarrolladores, 2015)

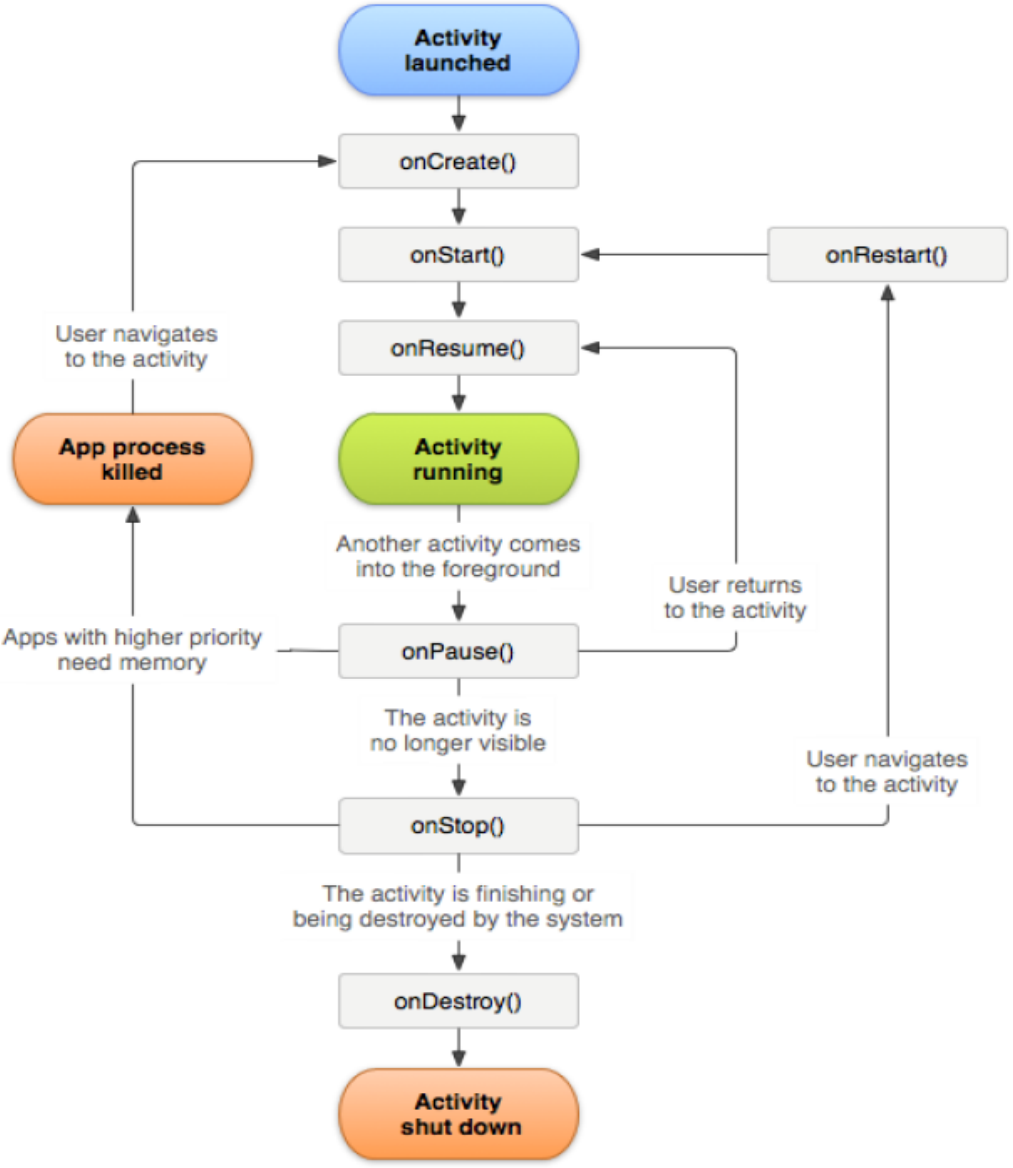


Figura 4. Ciclo de vida de una aplicación Android.
Tomado de Desarrolladores Android, s.f.

Se pueden controlar los diferentes estados gracias a una serie de métodos predefinidos, donde el usuario podrá añadir los comportamientos deseados en su aplicación.

A continuación, una breve explicación de los métodos y su uso.

– onCreate(): Implementa las acciones a realizar cuando se crea la aplicación.

– onStart(): Implementa las acciones a realizar cuando la aplicación pasa a primer plano.

– onResume(): Implementa las acciones a realizar cuando la aplicación pasa a primer plano, tras haber estado en pausa.

– onStop(): Implementa las acciones a realizar cuando se para la aplicación.

– onRestart(): Implementa las acciones a realizar cuando se reinicia la aplicación, tras haber sido parada.

– onPause(): Implementa las acciones a realizar cuando se pausa la aplicación.

– onDestroy(): Implementa las acciones a realizar cuando se cierra la aplicación.

2.2.3. Librerías de datos sobre tráfico (APIs)

Como parte de su apuesta por el desarrollo de software libre, Android incluye la posibilidad de utilizar e implementar librerías de terceros. Aunque los datos proporcionados por Android, mediante el acceso al GPS del dispositivo móvil, son bastante amplios (incluyendo coordenadas de localización, altura, velocidad, etc.), en ocasiones es necesario conocer otro tipo de datos ajenos al dispositivo.

Es el caso de los datos sobre tráfico, durante el desarrollo de este proyecto, surgirá la necesidad de conocer datos como la velocidad máxima de la vía por la que se circula, ya que es necesario conocer para nuestro modelo el nivel de tráfico al alrededor de una ubicación en nuestro caso de la ubicación del usuario.

Para ello será necesario el uso de librerías de datos sobre tráfico de terceros, pasamos a analizar algunas de ellas.

Google API: Desarrollada por Google, provee una interfaz de programación para aplicaciones, con acceso a los servicios provistos por esta empresa. (API Mapas, 2015)

Para el problema que nos ocupa, el análisis de la congestión del tráfico, la API de Google nos proporciona algunas de las siguientes herramientas:

– Capa de datos de tráfico.

– Mapas.

– Dibujado y marcado de mapas.

– Eventos.

HERE WeGo API: Creada por Nokia, en la actualidad su propietaria es un consorcio formado por empresas potentes del sector automovilístico (como Audi o BMW). Con su API, provee servicios relacionados con mapas, cartografía y estado del tráfico. Una de las características más interesantes es su base de datos de límites de velocidad en diferentes vías. (API Tráfico, 2015)

Para la utilización de sus servicios, HERE WeGo ofrece una conexión con los datos necesarios a través del protocolo HTTP. La respuesta a esta petición contendrá los datos, en formatos XML o JSON. Android cuenta con métodos implementados en su API para el tratamiento de ambos lenguajes, lo cual simplifica la tarea de implementación de esta API de Here WeGo. (API Tráfico 2015)

2.2.4. Conclusiones sobre el desarrollo de software en dispositivos móviles

Tras realizar un análisis sobre el estado del desarrollo de software, en dispositivos móviles, podemos concluir:

– Que existen varios SO potentes en el mercado, con grandes herramientas para los desarrolladores, pero destaca fuertemente la posición de Android, por ser líder en número de dispositivos compatibles y el actor con mayor número de usuarios.

– Que existen librerías de terceros, compatibles con Android, y que permiten la implementación de aquellas características, que este SO no tiene por sí mismo. Algunas de estas proporcionan información sobre tráfico que nos será útil durante el desarrollo de este proyecto.

2.3. Inverse Distance Weighting (IDW)

La ponderación a distancia inversa es una técnica determinista de interpolación no lineal que utiliza un promedio ponderado de los valores de atributos (es decir, fenómenos) de puntos de muestra cercanos para estimar la magnitud de ese atributo en ubicaciones no muestreadas.

El peso que se asigna a un punto en particular en el cálculo del promedio depende de la distancia del punto muestreado a la ubicación no muestreada como se observa en la figura 5.

El método se denomina ponderación de distancia inversa porque, de acuerdo con la primera ley de geografía de Tobler, la similitud de dos ubicaciones debería disminuir al aumentar la distancia.

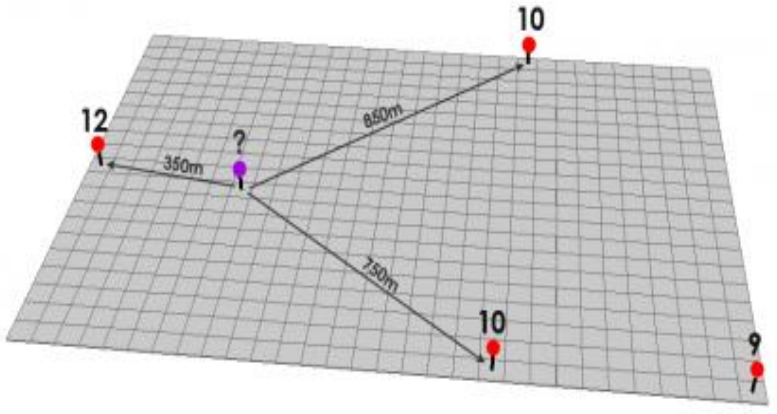


Figura 5. Gráfico espacial sobre IDW.
Tomado de ArcGis, s.f.

Para utilizar la interpolación de ponderación de distancia inversa para crear una superficie, hay varios factores que los cartógrafos deben tener en cuenta. Una pregunta importante es el tipo de relación que tiene el fenómeno con la distancia (por ejemplo, ¿disminuye dramáticamente con la distancia, o incluso los puntos relativamente distantes tienen algún grado de similitud con la ubicación no muestreada?).

Muchos cartógrafos eligen especificar una relación de distancia inversa al cuadrado, donde el peso de un punto difiere con el cuadrado inverso de la distancia (es decir, 1 / distance2) en lugar de una simple distancia inversa (es decir, 1 / distancia).

Para lo cual existen dos maneras de calcular IDW, las dos difieren en el hecho de elevar al cuadrado la distancia entre la ubicación actual y el punto de referencia como se observa a continuación:

$$((12/350) + (10/750) + (10/850)) / ((1/350) + (1/750) + (1/850)) = 11.1$$

$$= ((12/350^2) + (10/750^2) + (10/850^2)) / ((1/350^2) + (1/750^2) + (1/850^2)) = 11.4$$

Para el desarrollo del proyecto utilizaremos la primera opción ya que es la más efectiva para obtener resultados mucho más precisos. De manera general la fórmula se puede resumir de la siguiente manera:

$$\hat{v} = \frac{\sum_{i=1}^{n} \frac{1}{d_i} v_i}{\sum_{i=1}^{n} \frac{1}{d_i}}$$

Un segundo factor importante es determinar qué tan grande debe ser el vecindario de influencia: todos los puntos dentro de una distancia fija de la ubicación no muestreada, o si el vecindario debe consistir en un número particular de puntos, independientemente de su distancia a la no muestreada ubicación.

Una variación de este segundo método podría ser especificar alguna combinación de distancia y número de puntos (por ejemplo, seleccione el número n más cercano de puntos dentro de los 10 km de la ubicación no muestreada) como se observa en la figura 6 cada una de estas decisiones puede tener un impacto en la apariencia final de la superficie interpolada y, por lo tanto, debe considerarse cuidadosamente.

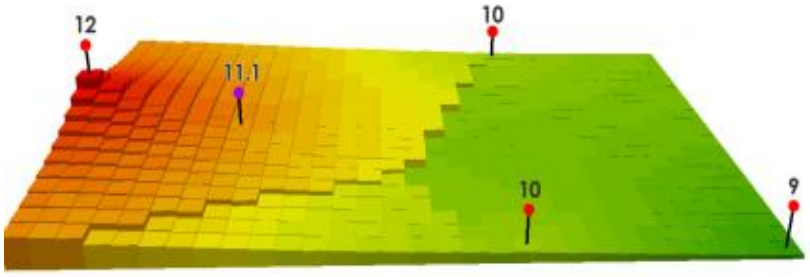


Figura 6. Gráfico espacial de obtención de resultados sobre IDW. Tomado de ArcGis, s.f.

Debido a que la ponderación inversa de la distancia es una técnica determinista, no tiene en cuenta la estructura espacial (es decir, la disposición) de los puntos de muestra.

Por lo tanto, los resultados que obtiene al utilizar esta técnica pueden verse influenciados por el espaciado y la densidad de las muestras, y es bueno tener cuidado con la precisión de los valores interpolados. Además, debido a que la ponderación inversa de la distancia calcula un valor promedio, el valor que

calcula para un punto no muestreado nunca puede ser más alto que el valor máximo para un punto de muestra o más bajo que el valor mínimo del punto de muestra, por lo tanto, si los picos y valles de los datos no están representados en su muestra, esta técnica puede ser muy inexacta en algunas ubicaciones.

2.4. Desarrollo ágil

Durante el proceso de creación de este proyecto, se ha utilizado una metodología ágil. Desde la perspectiva del software, se define agilidad como un comportamiento persistente que se caracteriza por su flexibilidad para adaptarse a los cambios, esperados o inesperados, de forma rápida. Además, busca lo económico, una duración corta de tiempo, sin perder en calidad. Se utilizan la experiencia y el conocimiento como instrumentos que completan esta mejora. (Casos de Uso Wiki, 2018)

Las metodologías de tipo ágil se caracterizan por tener un ciclo de vida iterativo. Algunas otras características de este tipo de metodología son:

– Procesos iterativos e incrementales.

– Mitigación del riesgo.

– Mejora continua.

– Calidad desde la primera iteración

– Se priorizan los requerimientos según su valor.

– Colaboración continúa con el cliente

– Incorporación de cambios.

Existen diversos tipos de metodologías de desarrollo ágil, atendiendo a diferentes filosofías, la elegida para este desarrollo es Scrum.

Scrum: Se trata de una metodología de gestión y control para productos, cuyo objetivo es restar complejidad a dicha gestión. Se trata de focalizar los esfuerzos en un desarrollo de software que cumpla con los requisitos. Se caracteriza por ser una metodología simple, lo que la convierte en fácilmente escalable para distintos tipos de proyecto. (Scrum Alliance, 2015)

Entre sus características más destacables están:

– Adopción de una estrategia de desarrollo incremental.

– Basar la calidad del resultado en el conocimiento e interacción del equipo desarrollador.

– Solapamiento de las distintas fases de desarrollo.

Para entender el proceso iterativo de Scrum, es necesario comprender que su mayor utilidad es durante el desarrollo de proyectos de investigación y desarrollo como se observa en la figura 7 cada iteración permitirá tener un producto de calidad, y además la introducción de características nuevas.

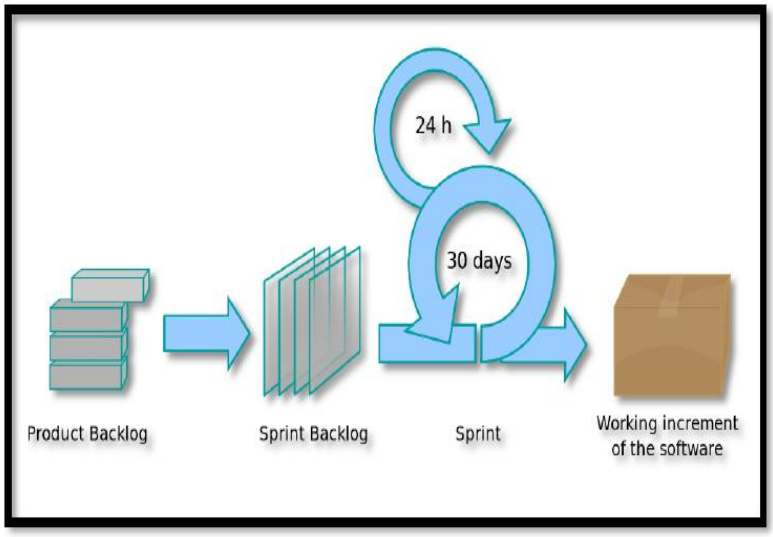


Figura 7. Diagrama de la metodología Scrum.
Tomado de Metodología Scrum, 2018.

1. Al comenzar cada iteración, el equipo decidirá las características a implementar.

2. Con las características elegidas, se plantea la mejor forma de implementarlas.

3. Se realiza el desarrollo.

4. Una vez terminado el desarrollo, se hace una revisión de lo implementado, y se analiza la siguiente iteración.

A pesar de la baja complejidad de este proyecto, y del reducido número de actores, la decisión de este tipo de metodología se basa en:

– Ir consiguiendo objetivos a la vez que el desarrollador adquiere conocimientos.

– El tutor podrá comprobar el avance y los resultados tras cada iteración.

– Entre el desarrollador y el tutor se establecen nuevos objetivos a perseguir en las siguientes iteraciones.

– Se consigue desde el principio software de calidad que se podrá reutilizar tras cada iteración.

En base a lo explicado, se definen las siguientes secciones que dividen de forma bastante fiel, cada una de las iteraciones utilizadas durante el desarrollo del proyecto.

2.5. Regresiones lineales aplicadas

El modelo de pronóstico de regresión lineal permite hallar el valor esperado de una variable aleatoria a cuando b toma un valor específico.

La aplicación de este método implica un supuesto de linealidad cuando la demanda presenta un comportamiento creciente o decreciente, por tal razón, se hace indispensable que previo a la selección de este método exista un análisis de regresión que determine la intensidad de las relaciones entre las

variables que componen el modelo.

Existen medidas de la intensidad de la relación que presentan las variables que son fundamentales para determinar en qué momento es conveniente utilizar regresión lineal.

Utilizando este método determinaremos la relación que existe entre una variable dependiente y una o más variables independientes.

Para poder realizar esta relación, se debe postular una relación funcional entre las variables. Cuando se trata de una variable independiente, la forma funcional que más se utiliza en la práctica es la relación lineal.

2.6. Machine Learning (aprendizaje automático)

El aprendizaje automático (ML) es una categoría de algoritmo que permite que las aplicaciones de software sean más precisas para predecir resultados sin ser programadas explícitamente. La premisa básica del aprendizaje automático es construir algoritmos que puedan recibir datos de entrada y usar análisis estadísticos para predecir una salida mientras se actualizan las salidas a medida que se dispone de nuevos datos.

Los procesos que se involucran en Machine Learning son en su mayoría muy similares a los de la minería de datos y también a los del modelado predictivo, esto quiere decir, los dos necesitan buscar en los datos para encontrar patrones y ajustar los procesos del programa en consecuencia. Varias personas conocen de los conceptos del aprendizaje automático, por ejemplo las compras en Internet y reciben anuncios relacionados con su compra.

El motivo más importante es que los motores de recomendación utilizan el Machine Learning para personalizar cada publicación de anuncios en línea en prácticamente en tiempo real, otros casos comunes de Machine Learning

incluyen la detección de fraudes, la detección de spam, el filtrado de amenazas a la seguridad de la red, el mantenimiento predictivo y por supuesto la creación de fuentes de noticias.

2.6.1. Funcionamiento del aprendizaje automático

Los diferentes algoritmos de Machine Learning por lo general se clasifican como supervisados o no supervisados. Los algoritmos supervisados solicitan una persona muy calificada en analizar todos los datos con habilidades de Machine Learning para proveer tanto la entrada como la salida que requerimos, por otro lado proporciona información sobre la precisión de las predicciones durante el entrenamiento de los algoritmos.

Los expertos en el análisis de los datos determinan las variables, características, el modelo que se debe analizar y usar para desarrollar predicciones. Posterior completa el entrenamiento, el algoritmo aplicará lo aprendido para los nuevos datos. Estos algoritmos no supervisados son más complejos de desarrollar ya que necesitan ser entrenados con los datos de resultados esperados. Para solucionar este problema en menor tiempo se utiliza un enfoque iterativo denominado aprendizaje profundo para revisar los datos y llegar a conclusiones.

Los algoritmos de aprendizaje no supervisados son mayormente utilizados para actividades de procesamiento mucho más complejas que los sistemas de aprendizaje supervisado se incluyen el reconocimiento de imágenes, la voz a texto y por último la generación de lenguaje natural.

Por ejemplo, las redes neuronales en su principio funcionan combinando miles de millones de ejemplos de datos de entrenamiento y de esta manera identifican automáticamente correlaciones muy vagas entre muchas variables.

Posteriormente a la ejecución del entrenamiento, el algoritmo puede usar su

base de datos de asociaciones para interpretar los nuevos datos. Todos estos algoritmos solo se han hecho factibles en la era de la información con grandes datos, ya que requieren grandes cantidades de datos de entrenamiento para su funcionamiento como se observa en la figura 8.

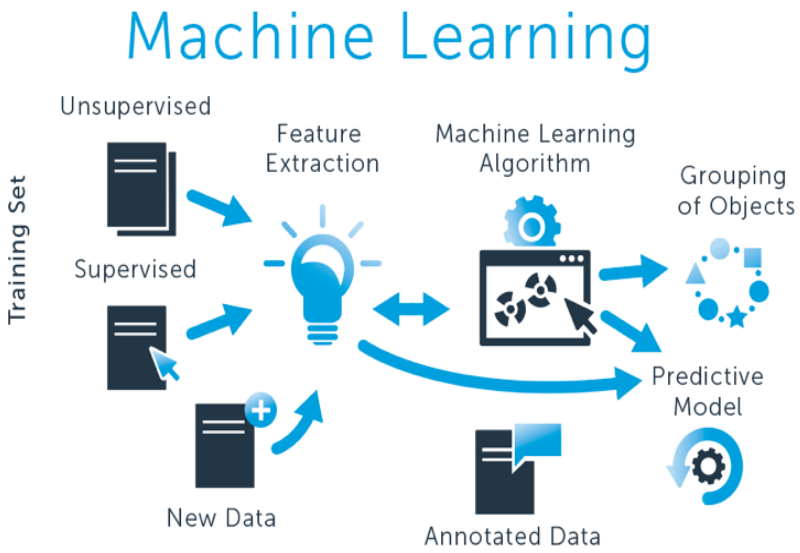


Figura 8. Conjunto de entrenamiento de ML.
Tomado de Link2Solution, s.f.

2.6.2. Tipos de algoritmos de Machine Learning.

Efectivamente existe usos casi ilimitados para Machine Learning, no hay escasez de algoritmos de Machine Learning.

Estos van desde lo muy simple a lo extremadamente complejo. Estos son varios de los modelos más utilizados hoy en día:

Las siguientes clases de algoritmo de Machine Learning implican que se debe identificar una correlación, muchas veces entre dos variables, y de esta manera usar esa correlación para hacer predicciones sobre tópicos de datos futuros.

Árboles de decisión: Este tipo de modelos utilizan observaciones sobre ciertas

acciones y de esta manera identifican una ruta óptima para llegar a un resultado esperado.

K-significa agrupamiento: Este tipo modelos agrupan un gran número específico de puntos de datos en un número específico de grupos en función de características similares.

Redes neuronales: Este tipo de modelos de aprendizaje profundo son los que utilizan grandes cantidades de datos para el entrenamiento y poder identificar correlaciones entre muchas variables, es decir, aprender a procesar datos entrantes en el futuro.

Aprendizaje de refuerzo: Este tipo de aprendizaje profunda involucra modelos que repiten muchos intentos para completar un proceso determinado, todos los pasos que producen resultados favorables son recompensados y los pasos que producen resultados que no son los deseados se penalizan hasta que el algoritmo aprende el proceso óptimo.

2.6.3. Ejemplos de aprendizaje automático.

El aprendizaje automático actualmente en varios aspectos y áreas se está utilizando en un amplio abanico de aplicaciones. El ejemplo más conocido es el servicio de noticias de Facebook. El servicio de noticias utiliza el aprendizaje automático para personalizar el contenido de cada usuario de la red social. Si un usuario deja el uso con frecuencia para leer o recibir publicaciones de un amigo en particular, el servicio de noticias comenzará a mostrar más de la actividad de ese amigo anteriormente en el canal.

Detrás de escena, el software simplemente utiliza el análisis estadístico y el análisis predictivo para identificar patrones en los datos del usuario y usar esos patrones para poblar la fuente de noticias.

Si un usuario de Facebook ya no se detiene a leer o a comentar las

publicaciones de uno de sus amigos, la nueva información se incluirán en el grupo de datos y la fuente de noticias se ajustará como consecuencia, en este sentido el aprendizaje automático no es de sorprenderse que también está ingresando a varias aplicaciones empresariales.

En virtud a lo anteriormente dicho los sistemas de gestión de la relación con el cliente (CRM) utilizan modelos de aprendizaje, esto con la finalidad de analizar el correo electrónico y hacer que los miembros del equipo de ventas respondan primero a los mensajes más importantes, sistemas más avanzados pueden incluso recomendar respuestas potencialmente efectivas, en este sentido los proveedores de inteligencia empresarial (BI) y analíticos utilizan el aprendizaje automático en su software para ayudar a los usuarios a identificar automáticamente los puntos de datos potencialmente importantes.

En muchas organizaciones los sistemas de recursos humanos (HR) en su mayoría usan modelos de aprendizaje para identificar las características de los futuros empleados y confían en este conocimiento para encontrar los mejores candidatos para puestos vacantes en las empresas.

De igual manera el uso para el desarrollo de aplicaciones móviles ha ido en crecimiento y enfocado al manejo de los datos para obtener un modelo confiable de procesamiento, es así, que lo implementaremos en nuestro proyecto de manera que sea nuestro modelo lo más preciso posible en los resultados.

3. Capítulo III: Estado del arte

3.1. Soluciones/Aplicaciones actuales

Actualmente existen diferentes soluciones en forma de aplicación (APP) que ofrecen información del tráfico en tiempo real pero no determinan el nivel de contaminación usando estos datos, por otra parte, existen pocas aplicaciones las cuales ofrecen el nivel de contaminación en una ciudad pero no se

determina como y de donde obtienen esos datos además que la información presentada cubre una gran zona y no se muestra el nivel de contaminación de zonas más pequeñas y en tiempo real, lo cual es el objetivo de esta tesis mediante el desarrollo de una aplicación muestre en tiempo real el nivel de contaminación al que está expuesto el usuario en un radio de 500m.

A continuación, se realiza un análisis de estas aplicaciones y sus algoritmos, lo cual nos ayuda a definir los aspectos diferenciales de nuestra solución.

3.1.1. Google Maps

Lanzado en 2005 por Google, surge como un servicio de mapas en la web para, posteriormente, pasar a ser también una aplicación móvil. En la actualidad presta servicios de localización, de búsqueda de direcciones, de navegación (en distintos medios de transporte), estado del tráfico (lanzado en 2007), es importante recalcar que para la obtención de datos de tráfico es necesario pagar una suscripción mensual.

Para la obtención de información sobre el estado del tráfico en las carreteras, Google cuenta principalmente con cuatro vías:

Sensores: En las primeras versiones tras su lanzamiento, Google Maps dependía completamente de los sensores instalados en carreteras o vehículos que circulaban por ellas.

Base de datos: Se utiliza información basada en historial de tráfico, que gracias a los algoritmos de Google podía prever el nivel de congestión en las carreteras. (Google Traffic Wiki, 2018)(InfoMapas, 2018)

Para la distinción del nivel de tráfico, Maps dibuja sobre la carretera un color que indica el nivel de congestión previsto como se observa en la figura 9.

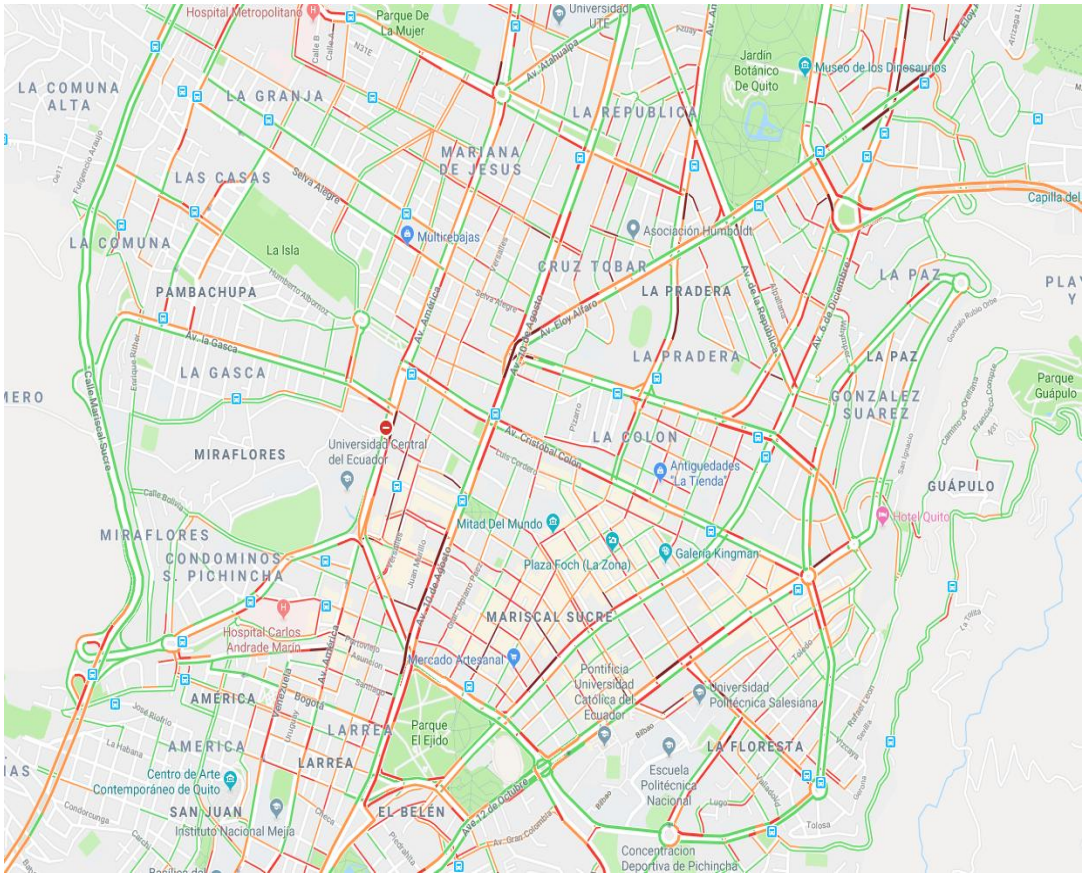


Figura 9. Vista de la predicción de tráfico basada en el histórico de datos. Tomado de Google Maps, s.f.

Usuarios: En 2007, con la adquisición de ZipDash's technology por parte de Google, se integra un nuevo método de detección de información de tráfico, que permite en tiempo real determinar el estado de una carretera. (Bazeley, 2005)

Para ello, tal y como informa Dave Barth, en un artículo de agosto de 2009, se utiliza la información recibida por todos aquellos usuarios de la aplicación con GPS y datos activados. (Barth, 2009)

La ubicación del usuario junto con otros datos como velocidad, altitud o posición, es enviada continuamente a los servidores de Google. Gracias a la combinación de los datos obtenidos de diferentes Smartphone, y su comparación con la velocidad máxima de la vía, se puede determinar el estado de congestión de la carretera.

De esta forma el usuario actúa como sensor que envía datos, que Google se encarga de analizar y actualizar en tiempo real.

Gracias a este análisis del tráfico en tiempo real, Google Maps es capaz de ofrecer el tiempo aproximado en rutas de navegación, o rutas alternativas más rápidas en caso de congestión en nuestra ruta original como se observa en la figura 10.

Información de terceros: Acuerdos con distintos organismos encargados del tráfico a todos los niveles de la administración pública, así como adquisición de empresas privadas, proveen a Google Maps de información del tráfico. Un ejemplo de ello es Waze, adquirida por Google en 2013. Waze es una aplicación colaborativa, cuyo algoritmo de detección será tratado en sucesivas secciones. (Web Waze, 2018)

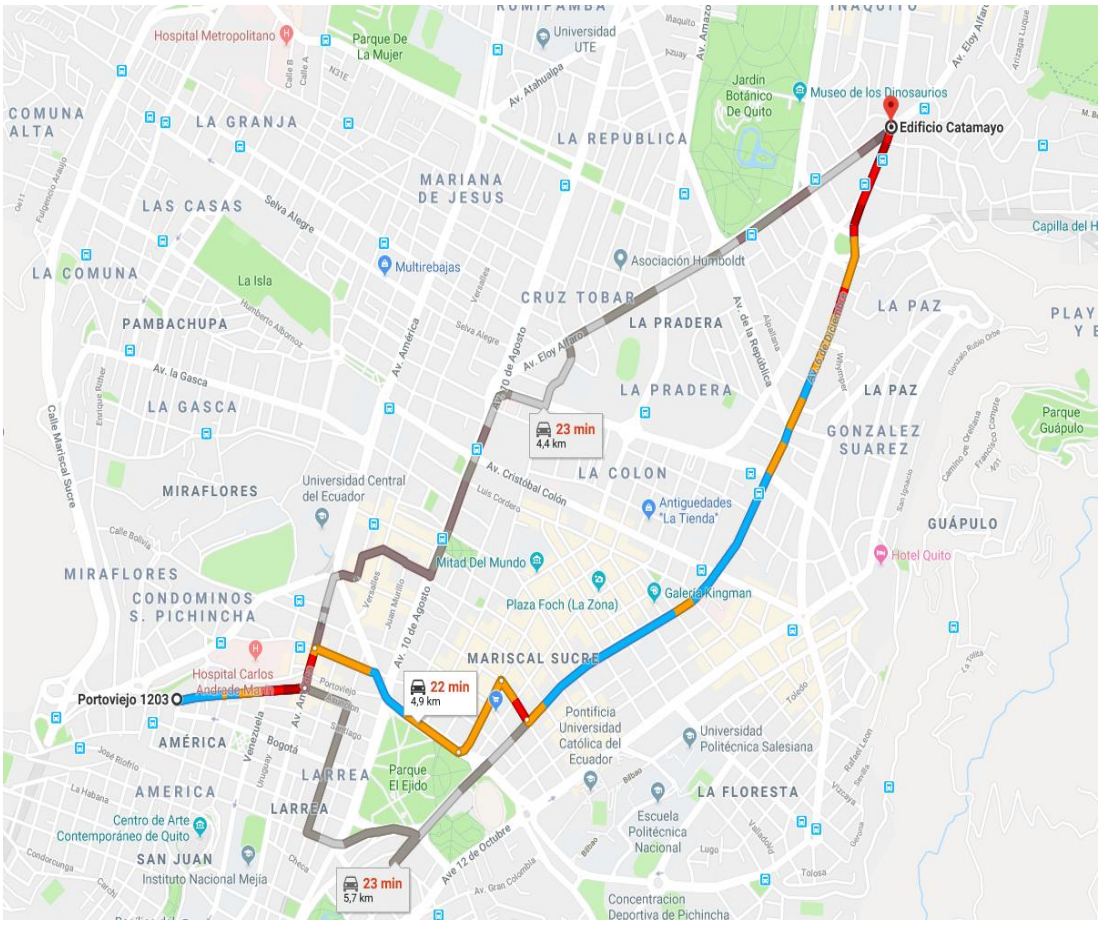


Figura 10. Tráfico en tiempo real, y tiempo estimado de ruta.
Tomado de Google Maps, s.f.

3.1.2. Aplicaciones colaborativas: Waze

Fundada en 2008, Waze es una aplicación social que permite a sus usuarios la navegación asistida por GPS y la detección del tráfico en tiempo real. (Web Waze, 2018)

Se basa en la premisa de ser una aplicación mantenida por el usuario y que aprende del mismo. Waze pertenece a una nueva generación de aplicaciones llamadas colaborativas. Estas son aplicaciones interactivas, que buscan un intercambio activo del usuario para captar información del mismo, con el fin de mejorar la aplicación.

En el caso de Waze, la interacción se efectúa mediante el establecimiento de avisos por parte del usuario como se observa en la figura 11. Durante la navegación hacia su destino, el usuario cuenta con un apartado de alertas por diferentes causas, entre ellas encontramos:

– Tráfico: Puede ser moderado, denso o detenido.

– Policía: Puede ser visible o no visible.

– Accidente: Puede ser leve o grave.

– Peligro: Puede ser en la vía, en el arcén o climatológico.

– Cierres: Por cierre de una vía o calle.

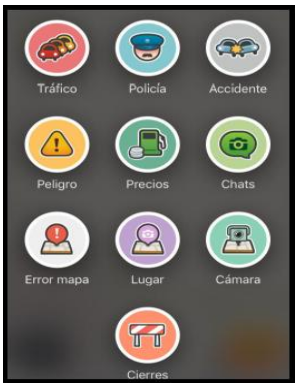


Figura 11. Distintos tipos de alertas para el usuario.
Tomado de la App Waze, s.f.

El usuario únicamente deberá pulsar la alerta correspondiente, y una notificación será enviada a los servidores de Waze. A partir de ese momento, y gracias a diferentes algoritmos que procesan el número de alertas de un suceso, el número de usuarios en el lugar y otros datos, los servidores son capaces de determinar si existe algún tipo de congestión en la vía.

Una vez determinado el alcance del suceso, dicha alerta podrá ser enviada a otros usuarios Waze. Estos datos podrán considerados en la elección de su ruta de navegación más óptima como se observa en la figura 12.

Figura 12. Detalle de alertas informadas en ruta.
Tomado de App Waze, s.f.

3.1.3. Trabajos de investigación

Previo a la realización de este proyecto, se han investigado diferentes soluciones alternativas. Todas ellas buscan dar una solución al problema de la obtención de datos de tráfico en tiempo real para el uso de esta información en nuestro modelo para el monitoreo de los niveles de contaminación en un área alrededor de una persona.

3.1.3.1. TrafficSense Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones

Se caracteriza por incorporar un punto de vista, donde los dispositivos móviles son protagonistas. Se trata de utilizar estos dispositivos y sus sensores para la captura de datos, y su posterior detección de eventos. (Mohan, Padmanabhan, Ramjee, 2018)

El dispositivo con mayor importancia es el acelerómetro, capaz de detectar eventos de frenado y aceleración. En cuanto al uso del acelerómetro, este artículo propone una metodología para su reorientación y la captura de aceleraciones en los diferentes ejes.

3.1.3.2. Sensor Fusion Method for Smartphone Orientation Estimation

Este artículo no propone una solución al problema de la congestión diaria, propone un método para la orientación del acelerómetro de un Smartphone. Puesto que la reorientación de este sensor, es fundamental para la detección de eventos de aceleración/frenado. Los dispositivos utilizados en él son: el magnetómetro y el giroscopio. (Ayub, Bahraminisaab, Honary, 2012)

3.1.3.3. Video Based Vehicle Detection and Its Application in Intelligent Transportation System

En este artículo del departamento de ingeniería eléctrica y de computadores de la Universidad de Nevada, se analiza un sistema de detección de vehículos

inteligente, basado en video.

La idea principal es capturar mediante cámaras, imágenes o videos, el estado de las carreteras. Se automatizará el mecanismo de análisis para poder detectar distintos patrones, que indiquen eventos en las carreteras. (Chintalacheruvu, Muthukumar, 2012)

3.1.3.4. Android Smartphone Application for Driving Style Recognition

En él, su autor utiliza la fusión de los sensores acelerómetro, giroscopio, y magnetómetro, incorporados en los dispositivos móviles, para tratar de definir distintos patrones de conducción. (Stoichkov, 2013)

3.1.4. Conclusión sobre las soluciones analizadas

Tras realizar un análisis de las principales soluciones del mercado, y los principales artículos que proponen soluciones alternativas, podemos concluir:

Que ninguna de las soluciones existentes en el mercado cumple con los requisitos impuestos para este proyecto dado que como objetivo principal queremos determinar el nivel de contaminación alrededor del usuario en un radio de 500m.

Haremos uso parcial de Google Maps dónde mediante el desarrollo de varios controladores podremos obtener datos de tráfico, aunque la privacidad queda en entredicho por el envío continuo de datos, pero mediante la utilización de la API de Google Maps y utilizando el envío de datos mediante url a este servicio podemos obtener como resultado un archivo Json con la información de tráfico alrededor de un punto podemos trazar estas rutas y de esta manera obtener el

nivel de tráfico alrededor de una ubicación.

Para probar este recurso usamos la configuración de request por url agregando una ubicación inicial y una final más nuestra ApiKey dónde veremos como resultado datos de distancia y tiempo como se muestra en la figura 13.


```
{
   "geocoded_waypoints" : [
      {
         "geocoder_status" : "OK",
         "place_id" : "EitBdi4gMTAgZGUgQWdvc3RvIDEyMiwgUXVpdG8gMTcwMTM2LCBFY3VhZG9yIhoSGAoUChIJ268K6Caa1ZERk25ZZGZ_8tYQeg",
         "types" : [ "street_address" ]
      },
      {
         "geocoder_status" : "OK",
         "place_id" : "ChIJOZhkKz2a1ZERtKX8sglUKy4",
         "types" : [
            "bus_station",
            "establishment",
            "point_of_interest",
            "transit_station"
         ]
      }
   ],
   "routes" : [
      {
         "bounds" : {
            "northeast" : {
               "lat" : -0.2106951,
               "lng" : -78.50090470000001
            },
            "southwest" : {
               "lat" : -0.2172136,
               "lng" : -78.50646069999999
            }
         },
         "copyrights" : "Datos de mapas ©2019 Google",
         "legs" : [
            {
               "distance" : {
                  "text" : "1,2 km",
                  "value" : 1194
               },
               "duration" : {
                  "text" : "5 min",
                  "value" : 287
               },
               "end_address" : "Buenos Aires (Parque Ejido), Quito 170136, Ecuador",
               "end_location" : {
                  "lat" : -0.2106951,
                  "lng" : -78.50090470000001
               },
               "start_address" : "Av. 10 de Agosto 122, Quito 170136, Ecuador",
               "start_location" : {
                  "lat" : -0.2172136,
                  "lng" : -78.5055734
               },
```


Figura 13. Respuesta Json de Google Maps con información de tráfico. Tomado de Google Maps, s.f.

4. Capítulo IV: Análisis y diseño de la solución técnica

En este capítulo se analiza y se describe el diseño de la solución técnica. Se realiza a través de la descripción de la arquitectura para, a continuación, presentar los requisitos a resolver y los casos de uso a tratar.

Una vez definido el tema de este proyecto, la obtención de información de contaminantes en el aire de la ciudad de Quito desde las estaciones de monitoreo. Para la obtención de los datos sobre el tráfico en las carreteras, es necesario definir una solución técnica eficaz. Se define como eje principal de la APP, el desarrollo de un algoritmo de monitoreo de contaminantes en el aire en base a las fuentes antes mencionadas, cuyas características permitan:

– Información en tiempo real.

– Interfaz amigable con el usuario.

– Un menor consumo de recursos.

– Una eficacia probada.

– Escalable.

– De fácil integración.

4.1. Instrumentación y análisis de datos

Para tener en cuenta la variabilidad del terreno complejo de Quito, se recopilaron datos meteorológicos y de calidad del aire de cinco sitios durante un período de nueve años, 2007-2016. Los sitios seleccionados varían según la altitud y las densidades de tráfico de vehículos.

1) San Antonio:

- Latitud = -0.0121152

- Longitud = -78.45

2) Carapungo:

- Latitud = -0.095686

- Longitud = -78.4497

3) Cotocollao:

- Latitud = -0.1114389
- Longitud = -78.5

4) Belisario:

- Latitud = -0.1848222
- Longitud = -78.496

5) Centro:

- Latitud = -0.2206528
- Longitud = -78.5134

6) El Camal:

- Latitud = -0.2515083
- Longitud = -78.5172

7) Guamaní:

- Latitud = -0.334
- Longitud = -78.5534

8) Tumbaco:

- Latitud = -0.215192
- Longitud = -78.4

9) Los Chillos:

- Latitud = -0.2987465
- Longitud = -78.456

La última estación solo ha estado operando desde 2014, las estaciones de

medición se ubicaron en los techos de edificios relativamente altos de acuerdo con los criterios para el monitoreo de la calidad del aire establecidos por la Agencia de Protección Ambiental de los Estados Unidos (USEPA, Washington, DC, EE. UU.) como se observa en la figura 14.

Figura 14. Ubicación de las estaciones de monitoreo en la ciudad de Quito
Tomado de Secretaría del Ambiente, s.f.

Para los datos de concentración de $PM_{2.5}$, se utilizó el monitor de partículas de ambiente continuo MicroDust perteneciente a la Universidad de las Américas, equipo con el cual se realizaron la toma de mediciones desde la estación de Belisario hacia la estación del Centro.

Se aplicará un enfoque de minería de datos para clasificar las concentraciones de $PM_{2.5}$ entre los valores por encima y por debajo de la mediana a largo plazo en relación con el tráfico y los datos del equipo de monitoreo. La clasificación de los datos se obtuvo a través del proceso de Machine Learning.

4.2. Arquitectura

Para la implementación de la aplicación se ha considerado la arquitectura definida por el equipo de desarrolladores de InfoAndroid, existen otras

arquitecturas pero la que se adapta de mejor manera a los requerimientos de implementación es la de la empresa antes mencionada.

Como se observa en la figura 14, la arquitectura consta de 4 capas, que interactúan entre ellas de forma jerárquicas, esta forma modulable permite la fácil sustitución de una de las capas. Por ejemplo, si decidiésemos utilizar otra capa para la comunicación en lugar de internet, o si sustituyésemos Android por otro SO móvil, en la figura 15 se observa una breve descripción de estas capas, sus componentes, y su función dentro de la arquitectura del sistema.

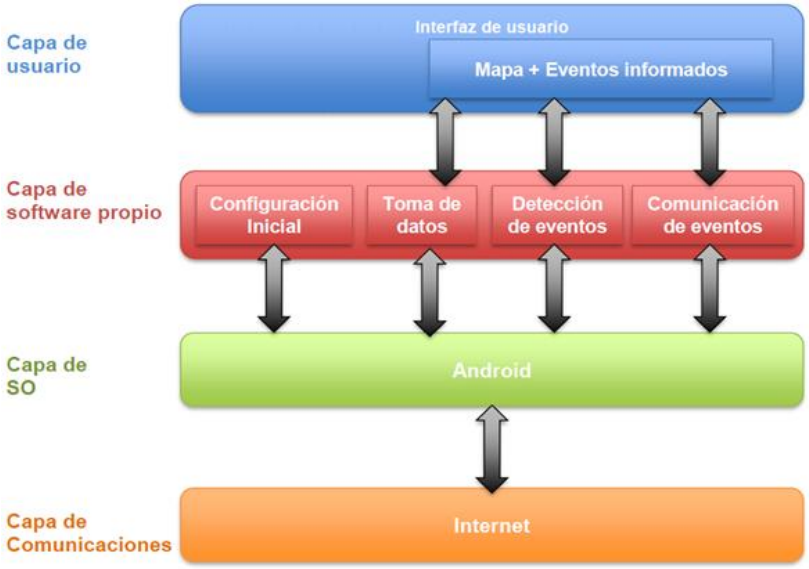


Figura 15. Arquitectura de la APP.
Tomado de InfoAndroid, s.f.

Capa de usuario: Su componente principal es la interfaz de usuario. Provee al usuario de un medio para comunicarse con las distintas funcionalidades de la aplicación, de una forma intuitiva y clara, esta capa de usuario se divide principalmente en dos funcionalidades:

– Sensor nivel de contaminación: Permite al usuario visualizar el nivel de tráfico a su alrededor en un radio de 500m, esta información se presentará en forma de radar alrededor de la ubicación del usuario, dependiendo del nivel de contaminación el radar se pintará del color correspondiente, esto se explicará más adelante.

– Sitios de las estaciones de monitoreo: Permite visualizar el lugar geoespacial donde se encuentran ubicados los diferentes sensores de monitoreo provistos por la Secretaría del Ambiente.

Capa de software propio: Es aquella referida a los componentes software que han sido desarrollados para esta APP. Se apoyan en los servicios proporcionados por el SO Android, y los servicios que se obtiene de Google para la realización de diferentes tareas:

– Carga inicial: Tiene principalmente el objetivo de cargar el mapa geo referencial y todos los servicios de geolocalización del mismo en la aplicación mediante el Api de Google Maps.

– Toma de datos: El sistema tomará de forma continua medidas del acelerómetro. Además, se podrá hacer uso del sistema GPS para la ubicación del usuario y de las estaciones.

– Comunicación de eventos: Si se detecta un evento, este podrá ser obtenido mediante el Web Service de Google Maps mediante el envío de datos a un servidor y la respuesta del mismo. Es decir, si por ejemplo nuestro algoritmo de detección detecta cambios en el tráfico o en el desplazamiento del usuario este procederá a realizar el recalculo de todos los procedimientos para obtener un resultado en vivo. Es importante aclarar, que la comunicación de datos únicamente se producirá al detectar un evento de forma local en nuestro dispositivo.

Capa de SO: En nuestro caso se trata del SO para dispositivos móviles Android. Como ya vimos durante el capítulo 2, este provee de herramientas para facilitar el acceso a los servicios proporcionados, por lo que mediará entre el hardware y nuestra solución de software para la obtención de datos.

Capa de comunicaciones: Este provee la comunicación de eventos mediante la red a un servidor. Esto permitirá la obtención de información en vivo sobre tráfico y desplazamiento del usuario, así como los marcadores de las estaciones de monitoreo. Cabe destacar que la importancia de este aplicativo

radica en el estudio de análisis de datos mediante aprendizaje automático para tener el mejor modelo predictivo para los niveles de contaminación.

4.3. Requisitos

La especificación de requisitos de software consiste en una descripción del comportamiento del sistema desarrollado. Para la descripción de las interacciones entre usuario y aplicación se utiliza un conjunto de casos de uso. La definición de los requisitos de software (ERS) se realiza dirigida por el cliente y el equipo de desarrollo.

Por su importancia, es necesario el uso de un lenguaje y estructura adecuada, de forma que sea comprensible para cualquier parte involucrada en el proyecto. Para este aplicativo, se ha decidido realizar una distinción en:

– Requisitos funcionales.

– Requisitos de restricción

– Requisitos no funcionales.

Como método para una mejor comprensión de los requisitos, así como de su importancia, necesidad y prioridad, se ha decidido el desarrollo de la siguiente tabla. Se utiliza los componentes de la tabla 5 para la definición de cada requisito.

Tabla 5. Ejemplo tabla requerimientos de software.

Identificador:	RS-XX
Nombre:	
Descripción:	
Fuente:	
Prioridad:	**Necesidad:**
Estabilidad:	**Verificabilidad:**
Prerrequisitos:	

Tomado de Instituto de Ingeniería Eléctrica y Electrónica, s.f.

– Identificador: Para la identificación de cada requisito se le asignará un número de identificador único. Este número empezará por RS (requisito de software) seguido de un número.

– Nombre: Titular que resume el requisito.

– Descripción: Breve descripción sobre el objetivo del requisito.

– Fuente: Es el titular o persona que establece este requisito. En este aplicativo se considera fuente al usuario, para aquellos requisitos que cumplen las funciones del proyecto, y al desarrollador, para aquellos requisitos que implican la verificación del correcto funcionamiento del sistema.

– Prioridad: Nivel de importancia del requisito. Los posibles valores son alta, media o baja.

– Necesidad: Nivel de necesidad declarado por el usuario que establece el requisito. Los posibles valores son alta, media o baja.

– Estabilidad: Indicador de la posible variabilidad del requisito a lo largo del proyecto. Los posibles valores son alta, media o baja.

– Verificabilidad: Indicador de la facilidad de comprobación de un requisito. Los posibles valores son alta, media o baja.

– Prerrequisito: Posibles requisitos de los que depende el requisito definido.

4.3.1. Requisitos funcionales

Establecen los comportamientos del sistema. Un requisito funcional define una función del sistema o sus componentes. Pueden ser requisitos funcionales: los cálculos, los detalles técnicos, la manipulación de datos y otras funcionalidades específicas que el sistema deba cumplir como se observa en las tablas 6, 7, 8, 9 y 10.

Tabla 6. RS-01: Utilización del API Google Maps.

Identificador:	RS-01		
Nombre:	Uso API Google Maps		
Descripción:	El aplicativo hará uso de la API de Google Maps para la obtención de datos de georeferenciación así como también los datos para el cálculo de la velocidad promedio alrededor del usuario para el dato de tráfico.		
Fuente:	Desarrollador		
Prioridad:	Alta	**Necesidad:**	Alta
Estabilidad:	Alta	**Verificabilidad:**	Alta
Prerrequisitos:	No procede		

Tabla 7. RS-02: Animación radar.

Identificador:	RS-02		
Nombre:	Animación radar		
Descripción:	La aplicación mostrará en forma de radar alrededor de la ubicación del usuario el nivel de contaminación mediante código de colores.		
Fuente:	Desarrollador		
Prioridad:	Alta	**Necesidad:**	Alta
Estabilidad:	Alta	**Verificabilidad:**	Alta
Prerrequisitos:	RS-01		

Tabla 8. RS-03: Cálculo de IDW.

Identificador:	RS-03		
Nombre:	Cálculo de IDW		
Descripción:	El aplicativo tomará la ubicación de las estaciones de monitoreo y el usuario con estos datos en vivo se calculará la interpolación utilizando las fórmula de IDW.		
Fuente:	Desarrollador		
Prioridad:	Alta	**Necesidad:**	Alta
Estabilidad:	Alta	**Verificabilidad:**	Alta
Prerrequisitos:	RS-01		

Tabla 9. RS-04: Cálculo velocidad promedio.

Identificador:	RS-04
Nombre:	Cálculo velocidad promedio
Descripción:	Mediante el uso del web service de google maps el aplicativo enviará su posicionamiento global y en un archivo Json obtendremos la distancia y tiempo para el cálculo de velocidad promedio en un radio de 500m alrededor del usuario.
Fuente:	Desarrollador
Prioridad: Alta	**Necesidad:** Alta
Estabilidad: Alta	**Verificabilidad:** Alta
Prerrequisitos:	RS-01

Tabla 10. RS-05: Cálculo resultado del modelo.

Identificador:	RS-05
Nombre:	Cálculo resultado del modelo
Descripción:	Madiante la construcción del mejor modelo para a obtención del nivel de contaminación se hará uso de todas las variables involucradas para el cálculo final y representar este resultado en el radar mediante código de colores.
Fuente:	Desarrollador
Prioridad: Alta	**Necesidad:** Alta
Estabilidad: Alta	**Verificabilidad:** Alta
Prerrequisitos:	RS-01, RS-02, RS-03, RS-04

4.3.2. Requisitos de restricción

Los requisitos de restricción son aquellos que deben cumplirse. De no hacerlo, el comportamiento de la aplicación se podría volver inestable e incorrecto, cabe recalcar que para el desarrollo en dispositivos Android hay que tener muy en cuenta la forma en que cada versión del sistema operativo hace el control y manejo de la petición de permisos para de las aplicaciones emitidas por fuentes desconocidas, como se observa en las tablas 11, 12, 13 y 14.

Tabla 11. RS-06: Permisos utilización de recursos.

Identificador:	RS-06		
Nombre:	Permisos utilización de recursos		
Descripción:	El aplicativo deberá solicitar al usuario el acceso a los recursos del sistema, de no ser concedidos se deberá impedir el uso de la aplicación, además estos permiso deben manejar las distintas versiones de android en las que se instale la app.		
Fuente:	Desarrollador		
Prioridad:	Alta	**Necesidad:**	Media
Estabilidad:	Media	**Verificabilidad:**	Alta
Prerrequisitos:	No procede		

Tabla 12. RS-07: Sensores GPS y acelerómetro.

Identificador:	RS-07		
Nombre:	Sensores GPS y acelerómetro		
Descripción:	El aplicativo necesita del uso de los sensores de GPS y acelerómetro, por lo que el dispositivo hardware utilizado deberá disponer de ellos.		
Fuente:	Desarrollador		
Prioridad:	Alta	**Necesidad:**	Alta
Estabilidad:	Alta	**Verificabilidad:**	Alta
Prerrequisitos:	No procede		

Tabla 13. RS-08: Conexión internet móvil.

Identificador:	RS-08		
Nombre:	Conexión internet móvil		
Descripción:	El aplicativo deberá ser provisto de conexión a internet móvil, que posibilite la utilización de bibliotecas externas y el envío de datos.		
Fuente:	Desarrollador		
Prioridad:	Alta	**Necesidad:**	Alta
Estabilidad:	Media	**Verificabilidad:**	Alta
Prerrequisitos:	No procede		

Tabla 14 RS-09: Cobertura móvil y GPS.

Identificador:	RS-09		
Nombre:	Cobertura móvil y GPS		
Descripción:	El aplicativo se utilizará en zonas donde la cobertura móvil y GPS sean lo suficientemente fuertes.		
Fuente:	Desarrollador		
Prioridad:	Media	**Necesidad:**	Media
Estabilidad:	Media	**Verificabilidad:**	Alta
Prerrequisitos:	No procede		

4.3.3. Requisitos no funcionales

También denominados atributos de calidad, es un requisito que especifica criterios o características de funcionamiento. Definen propiedades del sistema, como el tiempo de respuesta, necesidades de almacenamiento, fiabilidad, el uso de un tipo de herramienta, un lenguaje concreto o método de desarrollo, como se observa en las tablas 15, 16 y 17.

Tabla 15. RS-10: Privacidad.

Identificador:	RS-10		
Nombre:	Privacidad		
Descripción:	La privacidad del usuario será fundamental , no se revelará ningún dato del mismo.		
Fuente:	Desarrollador		
Prioridad:	Alta	**Necesidad:**	Media
Estabilidad:	Alta	**Verificabilidad:**	Media
Prerrequisitos:	No procede		

Tabla 16. RS-11: Algoritmo de detección.

Identificador:	RS-11
Nombre:	Algoritmo de detección
Descripción:	El algoritmo desarrollado para la detección de la velocidad promedio del tráfico vehicular alrededor de la ubicación del usuario, se caracterizará por detectar este tipo de eventos de forma local en el dispositivo del usuario.
Fuente:	Desarrollador
Prioridad: Alta	**Necesidad:** Alta
Estabilidad: Alta	**Verificabilidad:** Media
Prerrequisitos:	No procede

Tabla 17. RS-12: Desarrollo en Android Nativo.

Identificador:	RS-12
Nombre:	Desarrollo en Android Nativo
Descripción:	El desarrollo de la aplicación móvil, se efectuará bajo la plataforma Android. Se utilizará además el entorno de desarrollo Android Studio.
Fuente:	Desarrollador y Usuario
Prioridad: Media	**Necesidad:** Baja
Estabilidad: Alta	**Verificabilidad:** Alta
Prerrequisitos:	No procede

4.4. Casos de uso

Es una descripción de los pasos o actividades a tener en cuenta para llevar a cabo cualquier tipo de proceso. En software, esto implica una secuencia de interacciones entre un sistema y sus actores, en respuesta a un evento iniciado. Para el establecimiento de los casos de uso, se puede realizar un diagrama que especifica las comunicaciones y el comportamiento del

aplicativo, mediante su interacción con los usuarios, y otros sistemas. (Casos de Uso Wiki, 2018)

Partiendo de los requisitos de usuario definidos en el apartado anterior, se procederá a describir los casos de uso aplicables a este proyecto. Cada caso de uso podrá satisfacer uno o varios de los requisitos. Como método para una mejor comprensión de los casos de uso, y de forma adicional al desarrollo de diagramas vamos a utilizar los componentes de la tabla 18 para la descripción de cada caso de uso:

Tabla 18. Ejemplo tabla requerimientos de software.

Identificador:	CU-XX
Nombre:	
Actor/es:	
Descripción:	
Precondiciones:	
Postcondiciones:	
Secuencia principal	

Tomado de Instituto de Ingeniería Eléctrica y Electrónica Adaptación, s.f.

– Identificador: Para la identificación de cada caso de uso se le asignará un número de identificador único. Este número empezará por CU (caso de uso) seguido de un número.

– Nombre: Titular que resume el caso de uso.

– Actor/es: Representa el rol que toman los usuarios.

– Descripción: Breve descripción sobre el objetivo del caso de uso.

– Precondiciones: Especifica las condiciones que se deben cumplir previamente, para el cumplimiento del caso de uso actual.

– Postcondiciones: Especifica el estado en que queda el sistema tras el cumplimiento del caso de uso.

– Secuencia principal: Representa a la secuencia de pasos que componen

principalmente al caso de uso. Estos pasos irán numerados en orden de ejecución.

El diagrama de usos general de la aplicación separada por actores es un apartado importante dónde definimos los casos de usos generales, las figuras 16 y 17 desvelan que hay 4 posibles casos de uso generales a partir de los cuales derivan otros secundarios. Además, aparecen dos actores bien distinguidos:

– Usuario: Es aquel actor hacia el que va dirigido el proyecto. Es el usuario general, quien hará uso de la aplicación. Más adelante se explicarán los diferentes casos de uso que afectan a este actor.

– Desarrollador: Es aquel actor quién construirá el proyecto, sus casos de uso están definidos de cara al desarrollo y prueba del software. Más adelante se explicarán los diferentes casos de uso que afectan a este actor.

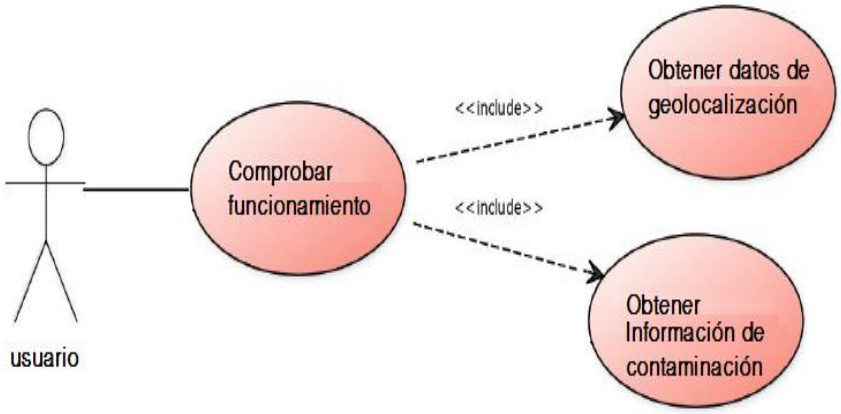


Figura 16. Diagrama casos de uso centrado en el actor usuario.

4.4.1. Casos de uso referentes al usuario

A continuación, pasamos a definir los casos de uso del diagrama anterior que afectan al actor usuario, como se observa en las tablas 19, 20 y 21.

Tabla 19. CU-01: Comprobar funcionamiento.

Identificador:	CU-01
Nombre:	Comprobar funcionamiento
Actor/es:	Usuario
Descripción:	El usuario comprueba la ejecución del aplicativo en sus dispositivos
Precondiciones:	Ejecutar la aplicación
Postcondiciones:	Se abrá detectado el correcto funcionamiento.
Secuencia principal	1. Ejecutar la aplicación 2. Comprobar aplicación desplegada

Tabla 20. CU-02: Obtener datos de geolocalización.

Identificador:	CU-02
Nombre:	Obtener datos de geolocalización
Actor/es:	Usuario
Descripción:	El usuario comprueba en el mapa desplegado su ubicación
Precondiciones:	Ejecutar la aplicación
Postcondiciones:	Se habrá detectado el correcto funcionamiento.
Secuencia principal	1. Ejecutar aplicación 2. Comprobar despliegue de ubicación actual

Tabla 21. CU-03: Obtener información de contaminación.

Identificador:	CU-03
Nombre:	Obtener información de contaminación
Actor/es:	Usuario
Descripción:	El usuario identifica mediante el radar el nivel de contaminación a su alrededor
Precondiciones:	Ejecutar la aplicación
Postcondiciones:	Se abrá detectado el correcto funcionamiento.
Secuencia principal	1. Ejecutar la aplicación 2. Comprobar despliegue del radar

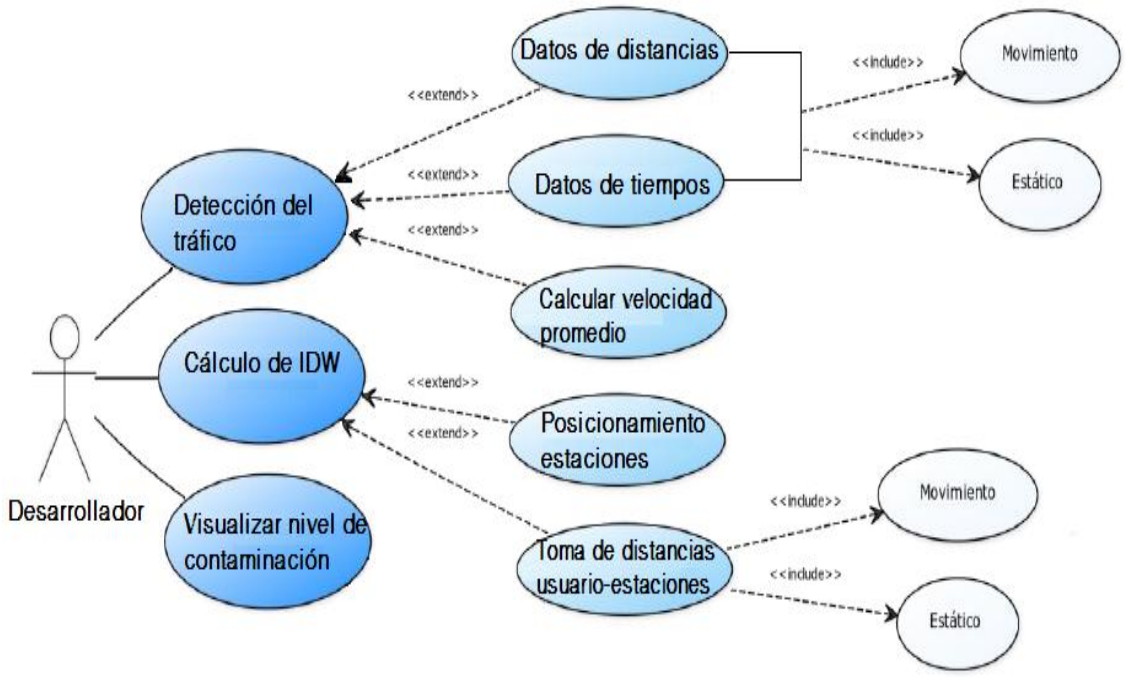


Figura 17. Diagrama casos de uso centrado en el actor desarrollador.

4.4.2. Casos de uso referentes al desarrollador

A continuación, pasamos a definir los casos de uso del diagrama anterior que afectan al actor desarrollador, como se observa en las tablas de la 22 a la 31.

Tabla 22. CU-04: Movimiento.

Identificador:	**CU-04**
Nombre:	**Movimiento**
Actor/es:	**Desarrollador**
Descripción:	El desarrollador detectará el desplazamiento del usuario mediante geolocalización
Precondiciones:	Iniciado todos los servicios de georeferenciación
Postcondiciones:	El aplicativo tomará los datos de desplazamiento como son: latitud y longitud
Secuencia principal	1. Servicios activos 2. Nuevo desplazamiento 3. Recolección de datos

Tabla 23. CU-05: Estático.

Identificador:	CU-05
Nombre:	Estático
Actor/es:	Desarrollador
Descripción:	El desarrollador captará si el usuario no se está desplazando para no generar recálculos
Precondiciones:	Iniciando todos los servicios de georeferenciación
Postcondiciones:	El aplicativo detectará si existe variación en la latitud y longitud dónde se encuentra el usuario
Secuencia principal	1. Servicios activos 2. Detección de no despazamiento 3. No genera nuevos cálculos

Tabla 24. CU-06: Datos de distancias.

Identificador:	CU-06
Nombre:	Datos de distancias
Actor/es:	Desarrollador
Descripción:	El desarrollador tomará a través de la web services de google las distancias en un radio de 500m alrededor del usuario en las diferentes rutas
Precondiciones:	CU-04 Y CU-05
Postcondiciones:	El aplicativo tomará todos los datos de distancias obtenidos del archivo Json generado por el web service de google
Secuencia principal	1. Servicios activos 2. Detección movimiento 3. Detección Estático 4. Obtener distancias

Tabla 25. CU-07: Datos de Tiempos.

Identificador:	CU- 07
Nombre:	Datos de Tiempos
Actor/es:	Desarrollador
Descripción:	El desarrollador tomará a través de la web service de google el tiempo de llegada desde su ubicación hacia los puntos destinados en un radio de 500m
Precondiciones:	CU-04 Y CU-05
Postcondiciones:	El aplicativo tomará todos los datos de tiempos obtenidos del archivo Json generado por el web service de google
Secuencia principal	1. Servicios activos 2. Detección movimiento 3. Detección Estático 4. Obtener tiempos

Tabla 26. CU-08: Calcular velocidad promedio.

Identificador:	CU- 08
Nombre:	Calcular velocidad promedio
Actor/es:	Desarrollador
Descripción:	El desarrollador mediante los datos obtenidos de distancia y tiempo sumado al desplazamiento o no del usuario creará el recálculo para la determinación de la velocidad del tráfico
Precondiciones:	CU-04, CU-05, CU-06 Y CU-07
Postcondiciones:	El aplicativo obtendrá la velocidad promedio alrededor del usuario en un radio de 500m
Secuencia principal	1. Servicios activos 2. Detección movimiento 3. Detección Estático 4. Obtener distancias 5. Obtener tiempos 6. Calcular velocidad promedio

Tabla 27. CU-09: Posicionamiento estaciones.

Identificador:	CU-09
Nombre:	Posicionamiento estaciones
Actor/es:	Desarrollador
Descripción:	Mediante la manipulación de la Api de Google Maps se implementará marcadores sobre el mapa con el posicionamiento de las estaciones.
Precondiciones:	Iniciado todos los servicios de georeferenciación
Postcondiciones:	En el aplicativo se observará mediante marcadores de diferente color la ubicación de las estaciones en toda la ciudad.
Secuencia principal	1. Servicios activos 2. Comprobar visualización de los marcadores de posicionamiento

Tabla 28. CU-10: Toma de distancias usuario-estaciones.

Identificador:	CU-10
Nombre:	Toma de distancias usuario - estaciones
Actor/es:	Desarrollador
Descripción:	El desarrollador captará los cambios de latitud y longitud del usuario y hará el cálculo de la distancia entre la ubicación actual y la de las estaciones
Precondiciones:	CU-04, CU-05 Y CU-09
Postcondiciones:	El aplicativo hará el cálculo de la distancia entre la ubicación del usuario y la de las 9 estaciones de monitoreo.
Secuencia principal	1. Servicios activos 2. Detección movimiento 3. Detección estático 4. Carga capa marcadores 5. Cálculo distancias

Tabla 29. CU-11: Detección del tráfico.

Identificador:	CU-11
Nombre:	Detección del tráfico
Actor/es:	Desarrollador
Descripción:	El desarrollador mediante la obtención de la velocidad promedio realizará el análisis del mismo para reflejar mediante condicionantes el nivel de tráfico existente.
Precondiciones:	CU-04, CU-05, CU-06, CU-07 Y CU-08
Postcondiciones:	El aplicativo determinará el nivel de tráfico alrededor del usuario en un radio de 500m
Secuencia principal	1. Servicios activos 2. Detección movimiento 3. Detección Estático 4. Obtener distancias 5. Obtener tiempos 6. Calcular velocidad promedio 7. Determinar tráfico

Tabla 30. CU-12: Cálculo de IDW.

Identificador:	CU-12
Nombre:	Cálculo de IDW
Actor/es:	
Descripción:	El desarrollador mediante los datos obtenidos de distancias entre el usuario y las estaciones aplicará la fórmula para la interpolación de IDW
Precondiciones:	CU-04, CU-05, CU-09 Y CU-10
Postcondiciones:	El aplicativo realiza correctamente el cálculo de la interpolación IDW
Secuencia principal	1. Servicios activos 2. Detección movimiento 3. Detección estático 4. Carga capa marcadores 5. Cálculo distancias 6. Cálculo de IDW

Tabla 31. CU-13: Visualizar nivel de contaminación.

Identificador:	CU-13
Nombre:	Visualizar nivel de contaminación
Actor/es:	Desarrollador
Descripción:	El desarrollador mediante los datos obtenidos de contaminación implementará la visualización de los mismos en forma de radar alrededor del usuario con su respectivo color de notificación.
Precondiciones:	CU-11 Y C-12
Postcondiciones:	Ela plicativo muestra la contaminación en forma de rada con el color respectivo al nivel de contaminación existente alrededor del usuario.
Secuencia principal	1. Servicios activos 2. Detección movimiento 3. Detección Estático 4. Obtener distancias 5. Obtener tiempos 6. Calcular velocidad promedio 7. Determinar tráfico 8. Carga capa marcadores 9. Cálculo distancia usuario-estaciones 10. Cálculo IDW 11. Muestra de resultado final en forma de radar

5. Capítulo V: Implementación de la aplicación

En este capítulo se muestra la implementación de la solución técnica. Para ello se utilizará todo lo descrito en capítulos anteriores, así como la experiencia técnica adquirida en el lenguaje Android y sobre todo la creación del mejor modelo para detectar el nivel de contaminación mediante el uso de Machine Learning.

Se seguirá la arquitectura descrita, y se cumplirán los objetivos y requisitos fijados. Además, se describirá la metodología utilizada para el desarrollo del software.

5.1. Aplicación GPS

El desarrollo y aplicación de este apartado está enfocado en el manejo del sensor GPS de los dispositivos móviles Android. Este sensor, como ya vimos en capítulos anteriores, permite la geo-localización.

Utilizaremos los datos obtenidos del sensor GPS para ubicar aquellos puntos de la red viaria donde se registren eventos. Para el manejo de datos de localización, Android cuenta con varias clases. La clase principal se denomina *LocationManager*.

La clase anteriormente mencionada provee acceso al servicio de localización del sistema, permitiendo a las aplicaciones obtener actualizaciones periódicas de la localización geográfica. Como método para minimizar los recursos del sistema consumidos, esta clase cuenta con distintos dispositivos para proporcionar la localización, algunos de ellos son: GPS o red.

El desarrollador podrá escoger en función de sus necesidades de precisión, consumo de batería, número de actualizaciones en el tiempo, etc. Este

proyecto requiere de una localización precisa, que permita ubicar perfectamente los eventos en un punto concreto de una vía concreta, por lo que se decide configurar LocationManager para obtener la ubicación mediante GPS.

5.1.1. Aplicación acelerómetro

Tras el desarrollo de una APP capaz de manejar el sensor GPS, y que servirá como módulo para la construcción de la APP final, se utilizará a nivel de hardware para la aplicación el acelerómetro.

Al igual que ocurre con el sensor GPS, Android cuenta con varias clases para el manejo del acelerómetro y los eventos de este sensor. Durante la implementación de esta aplicación, es muy importante tener en cuenta el sistema de coordenadas definido.

El sistema de coordenadas definido por la API de Android es el que podemos observar en la figura 18. Está definido por la posición de la pantalla. Los ejes no son intercambiables aunque haya cambios en la orientación del dispositivo. (Web Desarrolladores Android, 2015)

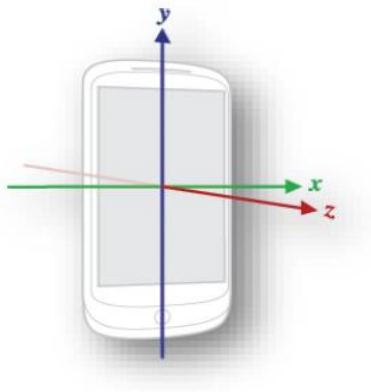


Figura 18. Sistema de coordenadas en Android.
Tomado Web desarrolladores Android.

El eje X es horizontal y apunta hacia la derecha. El eje Y es vertical y apunta hacia arriba. El eje Z es trasversal y apunta hacia el fondo de la pantalla. Como veremos en sucesivas secciones, este aplicativo centra su algoritmo de detección, en la medición de la aceleración para la detección de eventos, esto hace que el acelerómetro sea un dispositivo de gran importancia.

En la aplicación desarrollada se utiliza la API de Android para obtener la medida de este sensor en los tres ejes. Está permitido la configuración del tiempo entre eventos, y otros parámetros (aunque estos no nos serán útiles por estar más dirigidos a aplicaciones como juegos).

Debemos considerar que al tratarse de una aplicación dónde la ubicación del usuario es clave para el cálculo del nivel de contaminación a su alrededor, el detectar el desplazamiento mediante el cambio de coordenadas dentro de la aplicación es de vital importancia, como el desarrollo se lo realizará de forma nativa podemos manipular los sensores a nuestra necesidad.

5.2. Diagrama de Flujo

También denominado diagrama de actividades, sirve como representación gráfica de un algoritmo o proceso, en la figura 19 se muestra un diagrama de flujo general para la aplicación, este pretende ilustrar las acciones principales del aplicativo.

Es importante señalar, que este diagrama de flujo es tan sólo una aproximación, ya que un aplicativo de estas características puede no tener un flujo lineal debido a ser manejado por el usuario en su fase inicial, es decir, el usuario puede realizar las acciones en otro orden, o cerrar la aplicación cuando él decida.

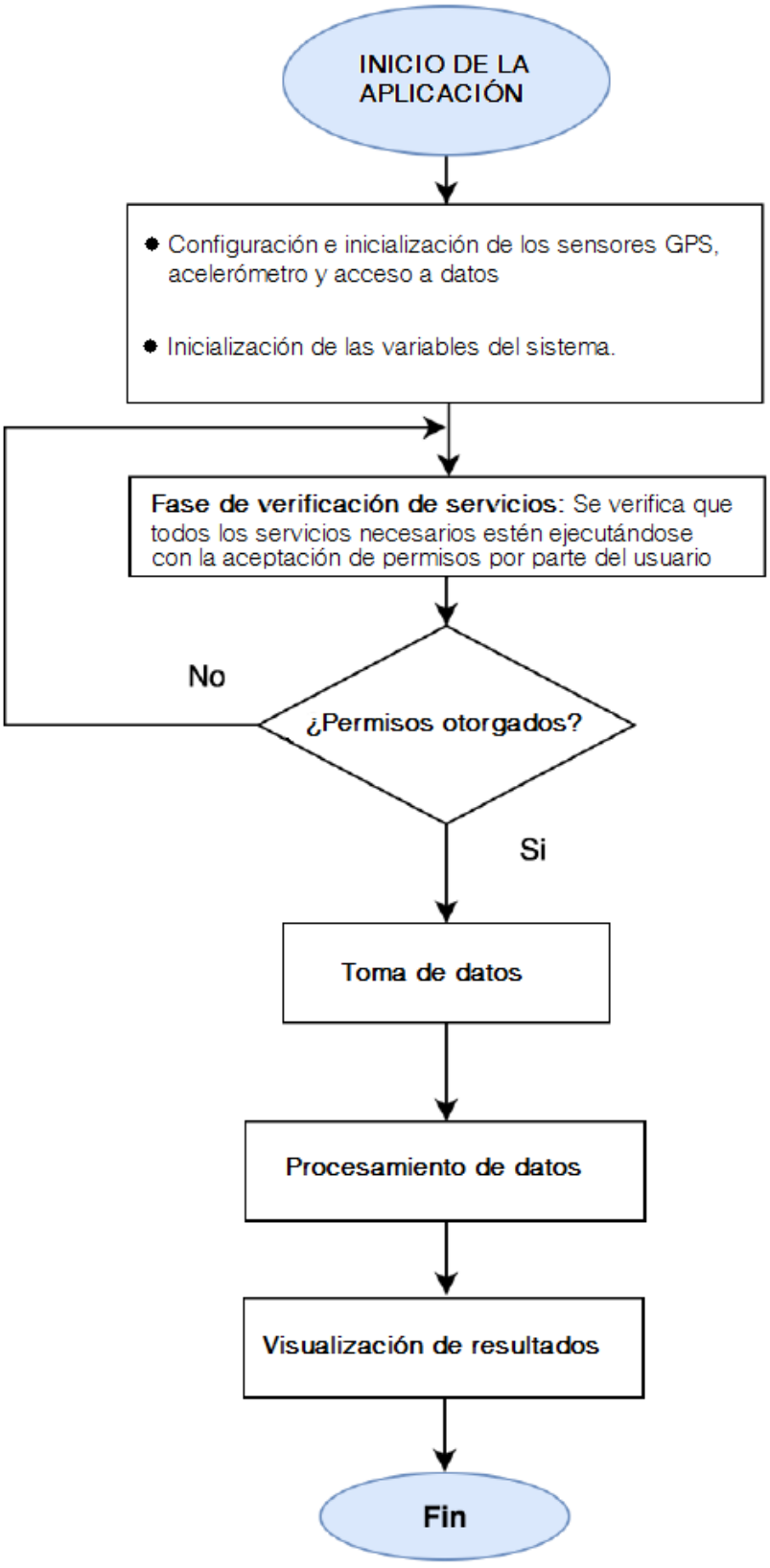


Figura 19. Diagrama de flujo general para la aplicación.

5.3. Aplicación del sistema de aprendizaje automático a la predicción del nivel de contaminación

Para poder elaborar el modelo de predicción lo primero es la obtención de los datos. En el siguiente orden:

- Plantear coordenadas de origen y destino entre dos estaciones de monitoreo dentro del Distrito Metropolitano de Quito.
- Recolectar datos del tiempo del tráfico.
- Concatenación entre los datos de campo y los provistos por la Secretaría del Ambiente.

En el pre procesamiento y exploración de los datos se trataron los mismos eliminando los picos.

En la selección de atributos se separó el análisis:

- Datos meteorológicos: dirección del viento y velocidad del viento.
- Químicos: $PM_{2.5}$.
- Datos de ubicación y tiempo: Marcadores GPS del trayecto y tiempos en cada sección de los mismos.

5.3.1. Recolección de datos y análisis de los resultados de las pruebas de campo (medición con MicroDust)

Las pruebas de campo se la realizaron trazando un trayecto entre dos estaciones de monitoreo implementados por la Secretaría del Ambiente, el trayecto se lo realizó desde la estación de Belisario hacia la estación Centro como muestra en la figura 20.

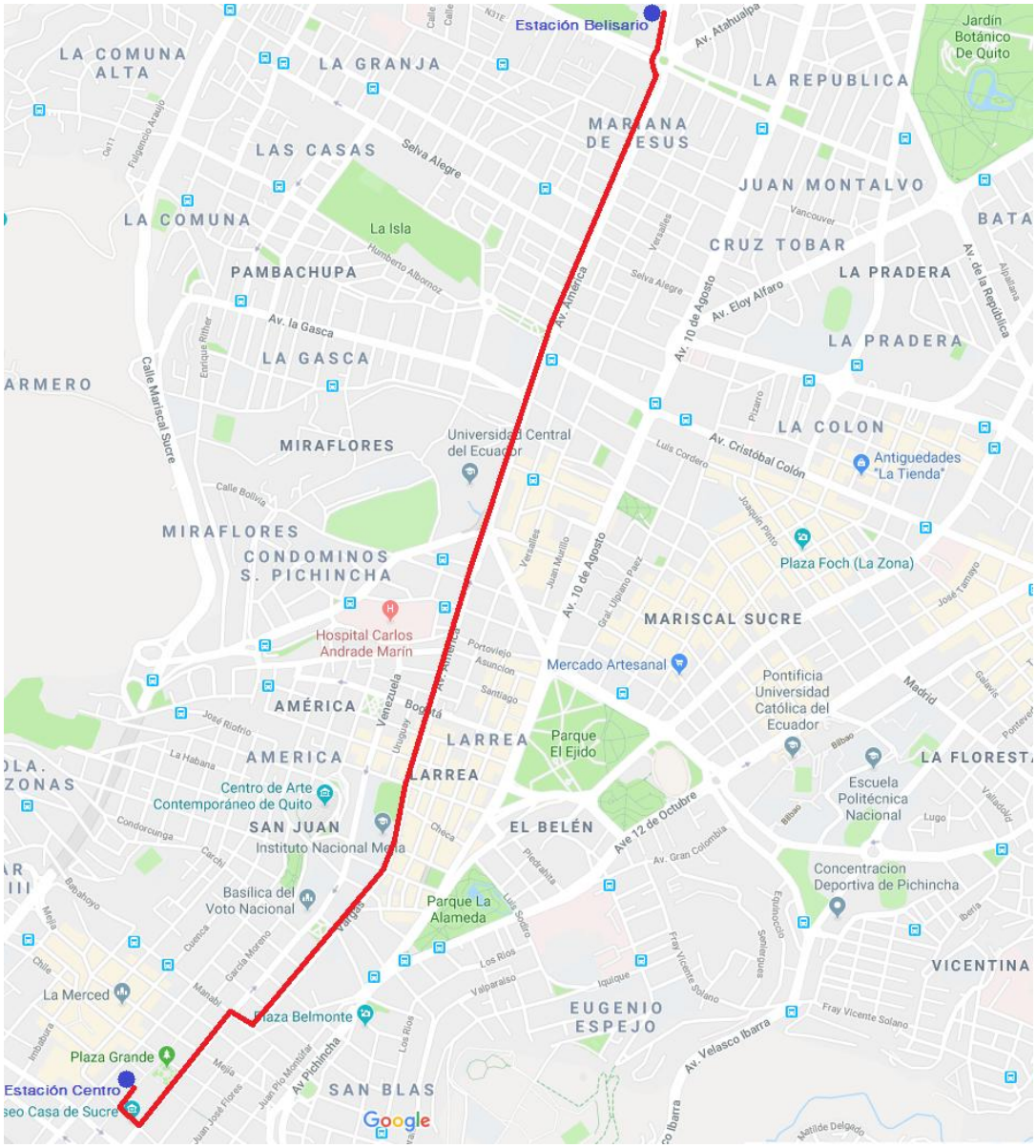


Figura 20. Trayecto realizado para las pruebas de campo. Tomado de Google Maps, s.f.

El trayecto se lo realizó el día 19 de octubre del 2018, empezando a las 10:21:06 y finalizando a las 11:47:26.

Con el equipo de medición de contaminación denominado MicroDust se pudieron tomar 519 mediciones del material particulado $PM_{2.5}$ (ver Anexo 1).

A estos datos se suman los datos obtenidos por el GPS, estos datos comprenden fecha y hora del inicio del trayecto, latitud, longitud y elevación, se registraron 534 datos (ver Anexo 2).

Como primero procesamiento de datos se realizó el emparejamiento de los datos entre el GPS y el MicroDust obteniendo así 535 datos con un margen de error de 4 segundos (ver Anexo 3).

Los resultados obtenidos por el GPS fueron manejados mediante la herramienta GPS Track Editor como se observa en la figura 21, de esta manera se pudo observar todo el trayecto realizado entre las dos estaciones.

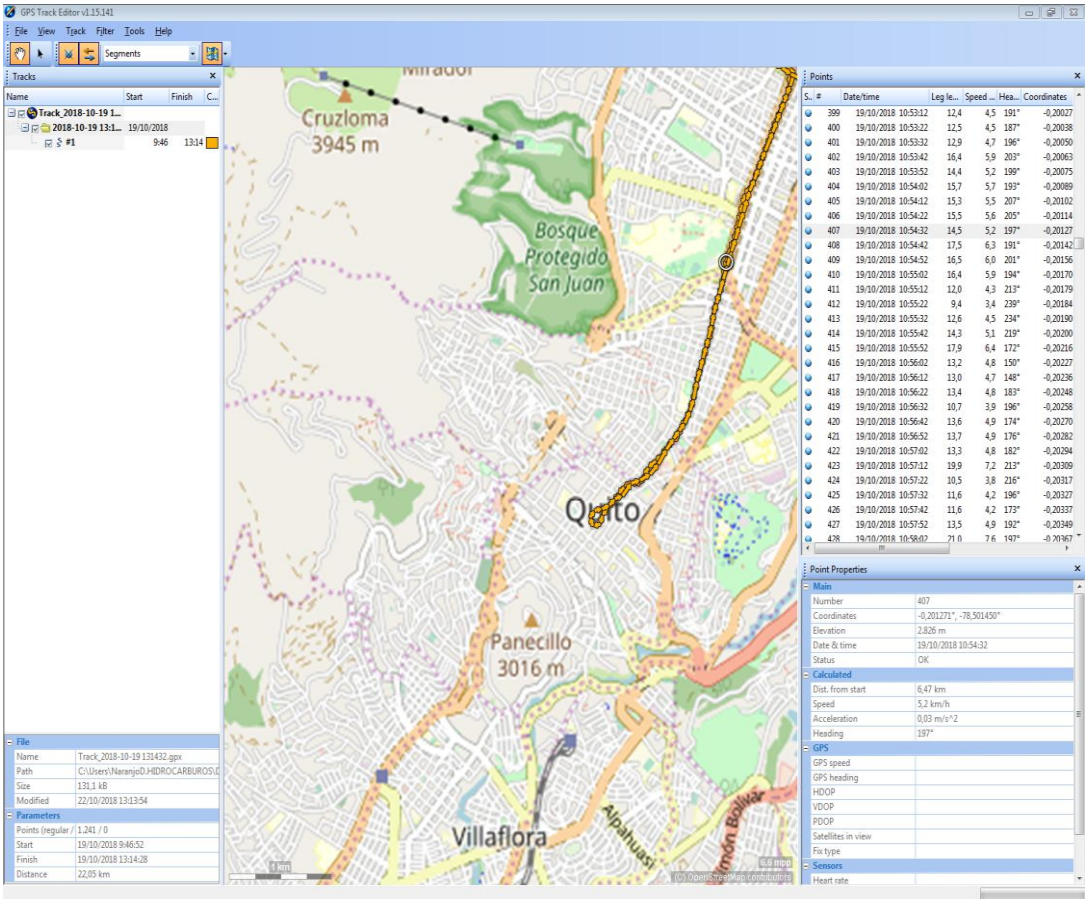


Figura 21. Datos GPS del trayecto realizado.
Tomado de GPS Track Editor, s.f.

5.3.2. Recolección y análisis de los datos provistos por la Secretaría del Ambiente

La Secretaría del Ambiente para la realización de este proyecto hizo la entrega

de los datos generados por las estaciones de monitoreo Belisario y Centro en el la fecha y hora que se realizó las pruebas de campo.

Los datos tomados por las estaciones son cada 10 minutos con la información de $PM_{2.5}$, dirección del viento y velocidad del viento (ver Anexo 4).

El radio de las estaciones es de 200 metros, esto es un limitante para la concatenación adecuada de los datos ya que aumenta el margen de error en el trayecto en zonas que no están cubiertas por las estaciones.

5.3.3. Pre procesamiento de los datos obtenidos recolectados

Con los datos emparejados obtenidos del MicroDust y GPS se realizó el análisis de los mismos visualizando la tendencia de los datos y los picos de eventos inesperados como la aceleración de un gran automotor junto al MicroDust como se observa en la figura 22.

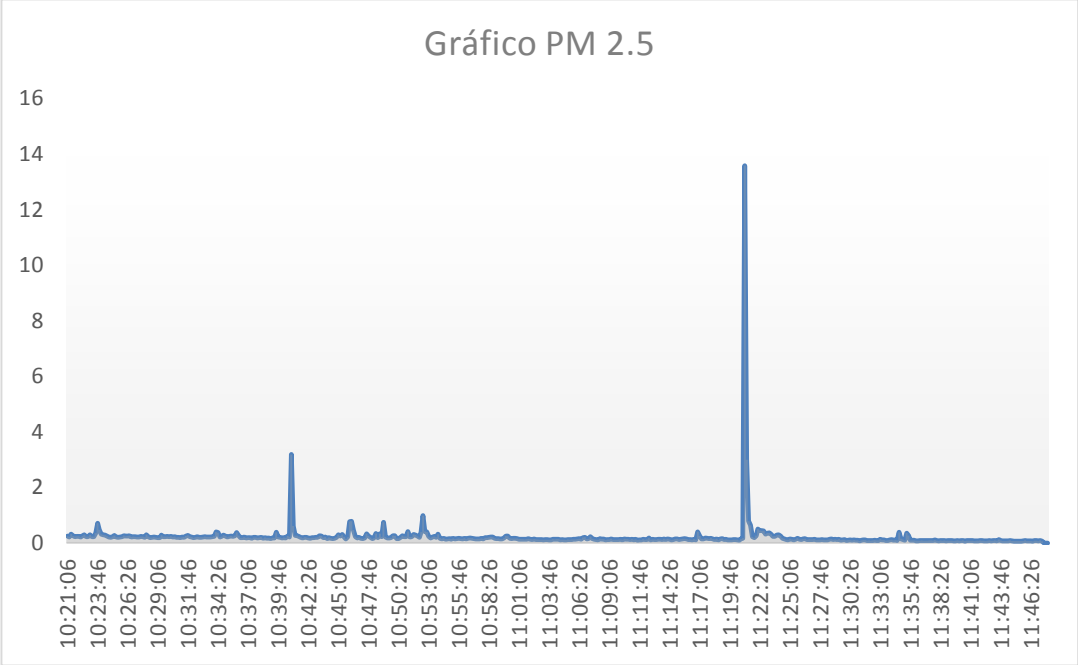


Figura 22. Gráfico del nivel de material particulado del trayecto.

Con esta información se procedió a eliminar los picos y los datos concordantes

para obtener una media del nivel de material particulado registrado en todo el trayecto entre las dos estaciones. La información obtenida posterior a la depuración y los provistos por la Secretaría del Ambiente se realizó la concatenación de los mismos obteniendo 10 mediciones para la construcción del primer modelo (ver Anexo 5).

5.3.4. Diseño del modelo de aprendizaje

5.3.4.1. Selección de atributos

Toda la información recolectada anteriormente forma parte de un grupo de características que nos podrán mostrar los diferentes enfoques para la construcción y selección del mejor modelo para predecir el nivel de $PM_{2.5}$.

En virtud a lo antes mencionado se agregan dos variables adicionales que son la interpolación IDW, dirección del viento, velocidad del viento y el nivel de tráfico, es así que se construyó un archivo XLSX con toda la información recolectada y procesada, esto nos abre las puertas para trabajar con la siguiente consideración:

- Para la elección del mejor modelo se realizará tres métodos de comparación:
    - Primero método MicroDust $PM_{2.5}$ = IDW: Entre los datos recolectados y procesados obtenidos por el MicroDust se realiza el tratamiento mediante la regresión lineal y la ejecución del modelo predictivo obteniendo como resultado el valor del coeficiente de correlación en relación a los datos calculados aplicando la fórmula de IDW.
    - Segundo método MicroDust $PM_{2.5}$ = elevación + meteorología: Entre los datos recolectados y procesados obtenidos por el MicroDust se realiza el tratamiento mediante la regresión lineal y la ejecución del

modelo predictivo obteniendo como resultado el valor del coeficiente de correlación en relación a obtenidos por la Secretaría del Ambiente.

- Tercer método MicroDust $PM_{2.5}$ = tráfico: Entre los datos recolectados y procesados obtenidos por el MicroDust se realiza el tratamiento mediante la regresión lineal y la ejecución del modelo predictivo obteniendo como resultado el valor del coeficiente de correlación en relación a los datos de velocidad del tráfico obtenidos previamente mediante Google Maps.

5.3.4.2. Características de los valores para la creación de los modelos.

En base al análisis de todos los datos obtenidos de las diferentes fuentes, y posterior a la depuración de las misma llegamos a obtener valores promedios de $PM_{2.5,}$ valores específicos de $PM_{2.5}$ y velocidad en un tiempo específico del trayecto de campo. Como se describió en el apartado 2.2.2 del presente documento mediante aplicaciones de terceros podemos visualizar información relevante para la conjetura del mejor modelo para nuestro proyecto.

Es necesario la velocidad del tráfico, es así que hemos utilizado la aplicación Waze para detectar la velocidad del tráfico, hemos realizado la concatenación de los datos de $PM_{2.5}$ y velocidad de tráfico cada 1 minuto del trayecto realizado entre las estaciones de Belisario y Centro.

Este procesamiento de datos fue realizado utilizando el método de minería de datos con cuyo proceso vamos a descubrir patrones en todos los datos recolectados, aplicamos este método porque se especializa en grandes volúmenes de datos como es en nuestro caso, de esta manera se obtuvieron 86 datos para la ejecución de uno de los métodos de regresión (ver Anexo 6). Como fuente para la determinación del nivel de tráfico según la velocidad se ha tomado como referencia los datos provistos por la Agencia Nacional de

Tránsito en cuanto a límites de velocidad y eventos en carretera como muestra la tabla 32.

Tabla 32. Velocidad de tráfico.

Velocidad	Descripción del tráfico
< 15 km/h	Alto
> 15 km/h - < 30 km/h	Medio
> 30 km/h - < 50 km/h	Bajo

Tomado de Agencia Nacional de Transito, s.f.

5.3.4.3. Conjuntos de entrenamiento y validación

Para cada modelo se obtuvo un número de datos para el procesamiento y validación en los tres métodos:

- Para el primero método se utilizó un total de 150 datos.
- Para el segundo método se utilizó un total de 120 datos.
- Para el tercer método se utilizó un total de 180 datos.

Posteriormente con la cantidad de datos antes mencionada en cada uno de los métodos los modelos son entrenados y probados de acuerdo con una validación cruzada de 10 veces denominada Cross-validation la cual trabaja el conjunto de datos obtenido es dividido en subconjuntos k. Uno de los subconjuntos k se utiliza como equipo de pruebas y los otros subconjuntos k-1, como un conjunto de entrenamiento. Después, el error promedio es calculado en todos los ensayos k. Cada punto de datos llega a estar en un conjunto de pruebas exactamente una vez, y llega a estar en un conjunto de entrenamiento k-1 veces. (Witten, Frank, Hall, 2011, p.152-153)

5.3.5. Aplicación de algoritmos de aprendizaje

El rendimiento de los modelos es evaluado por dos métricas: el parámetro coeficiente de correlación, y el error al cuadrado de la media de la raíz. El coeficiente de correlación (r) mide la fuerza de la relación lineal entre dos o más variables. La ventaja de r sobre las otras métricas se basa en una escala con un máximo (± 1) y un mínimo de (0) para cuantificar la fuerza de la relación. Cuanto más cerca de 1 es el valor absoluto mejor es la correlación, el error de la media cuadrática de la raíz (RMSE) es la raíz cuadrada de la variable de error cuadrado promedio por predicción (MSE).

El análisis de regresión entonces determina la intensidad entre las variables a través de coeficientes de correlación y determinación, pero sin lugar a duda el coeficiente de correlación lineal nos permite determinar si, efectivamente, existe relación entre las dos variables.

Una vez que se concluye que sí existe relación, la regresión nos permite definir la recta que mejor se ajusta a esta nube de puntos. En virtud a lo antes mencionado se realizó la normalización de las características obtenidas y se procedió a realizar las regresiones lineales y la aplicación de los métodos antes mencionados para obtener la característica más relevante para la determinación del nivel de contaminación.

5.3.5.1. Primer método: MicroDust $PM_{2.5}$ = IDW

Uno de los principales objetivos de un enfoque de aprendizaje automático, es producir la información más precisa posible y, que la predicción sea lo más simple posible con un modelo. Se construye un modelo basado en los datos obtenidos por el MicroDust y los calculados utilizando la fórmula de IDW para evaluar su fiabilidad. En la figura 23 se puede observar el diagrama de

dispersión cuyo análisis permite estudiar cualitativamente, la relación entre ambas variables.

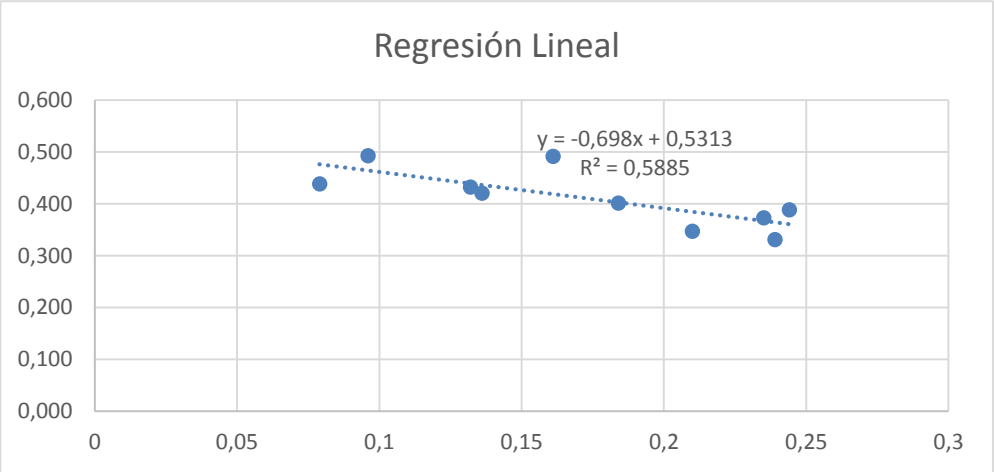


Figura 23. Gráfico regresión lineal IDW y $PM_{2.5}$.

Posteriormente obtenemos el coeficiente de correlación ejecutando el algoritmo regresión lineal como se observa en la figura 24.

```
Linear Regression Model

Microdust_PM =

      -0.1436 * IDW_PM +
       0.2526

Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.5222
Mean absolute error                      0.0552
Root mean squared error                  0.0584
Relative absolute error                 80.5355 %
Root relative squared error             78.7962 %
Total Number of Instances               10
```

Figura 24. Resultado regresión lineal IDW.

Como podemos observar en el resumen de la ejecución de la regresión con la característica elegida obtenemos un coeficiente de correlación de 0.5222.

5.3.5.2. Segundo método: MicroDust $PM_{2.5}$ = elevación + meteorología

Se construye un modelo basado en los datos obtenidos por el MicroDust y los obtenidos por la Secretaría del Ambiente para evaluar su fiabilidad. En la figura 25 se puede observar el diagrama de dispersión cuyo análisis permite estudiar cualitativamente, la relación entre las variables.

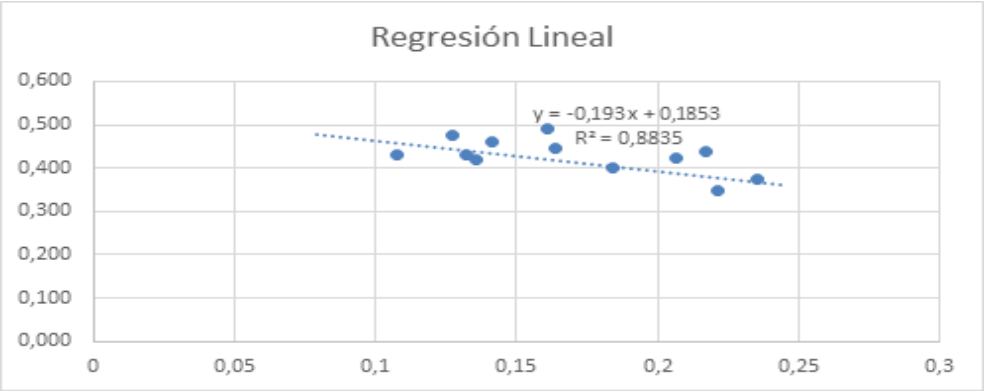


Figura 25. Gráfico regresión lineal MicroDust, elevación y meteorología.

Posteriormente obtenemos el coeficiente de correlación ejecutando el algoritmo regresión lineal como se observa en la figura 26.

```
Linear Regression Model

Microdust_PM =

      0.1747 * Elevation +
     -0.1508 * Xwind_Belisario +
      0.1003 * Xwind_Centro +
     -0.2401 * Ywind_Centro +
      0.2627

Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.6998
Mean absolute error                      0.0549
Root mean squared error                  0.0649
Relative absolute error                 80.0725 %
Root relative squared error             87.5834 %
Total Number of Instances               10
```

Figura 26. Resultado regresión lineal elevación + meteorología.

Como podemos observar en el resumen de la ejecución de la regresión con las características elegidas obtenemos un coeficiente de correlación de 0.6998.

5.3.5.3. Tercer método: MicroDust $PM_{2.5}$ = tráfico

Para finalizar se construye el tercer modelo basado en los datos obtenidos por el MicroDust y los datos de velocidad del tráfico para evaluar su fiabilidad.

En la figura 27 se puede observar el diagrama de dispersión cuyo análisis permite estudiar cualitativamente, la relación entre las variables.

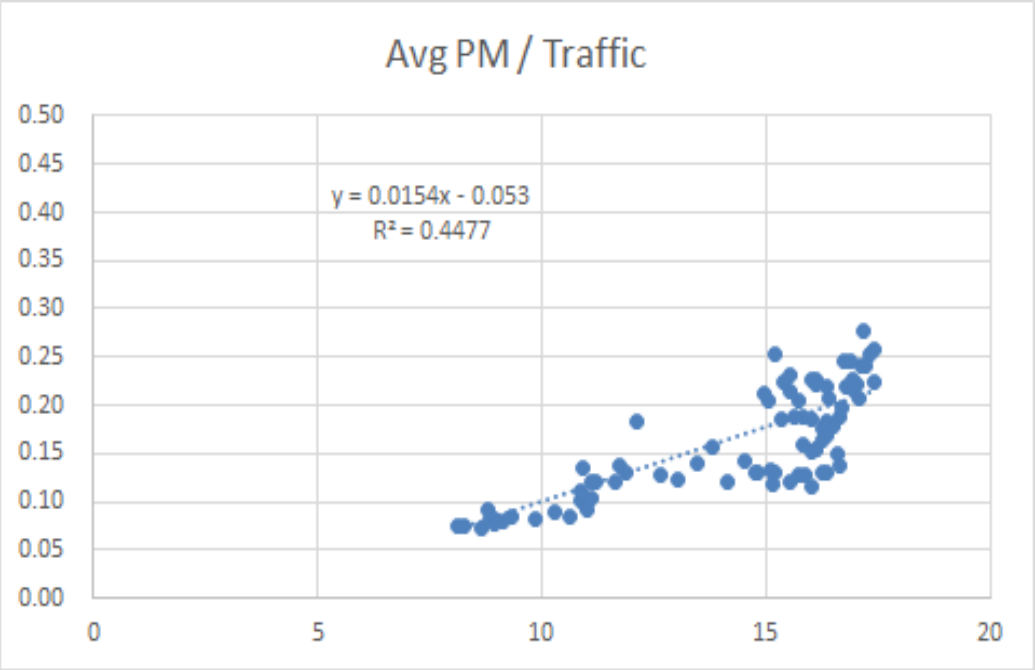


Figura 27. Gráfico regresión lineal Traffic y $PM_{2.5}$.

Posteriormente obtenemos el coeficiente de correlación ejecutando el algoritmo regresión lineal como se observa en la figura 28.

```
Linear Regression Model

microdust_avgPM =

      0.1371 * traffic +
      0.0728

Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.7079
Mean absolute error                      0.03
Root mean squared error                  0.0404
Relative absolute error                 59.5853 %
Root relative squared error             69.9059 %
Total Number of Instances               85
```

Figura 28. Resultado regresión lineal velocidad del tráfico.

Como podemos observar en el resumen de la ejecución de la regresión con las características elegidas obtenemos un coeficiente de correlación de 0.7079 siendo el más opcional de todos.

5.4. Aspectos generales e interfaz de usuario

Como una aplicación creada para el desarrollo y prueba, de un modelo predictivo de detección de niveles de contaminación, su estructura provee los mecanismos necesarios para poder expandirse e integrarse a cualquier arquitectura, en un entorno de servicios de navegación, esto conlleva que la interfaz de usuario sea simple y amigable.

Posterior a la instalación del aplicativo el usuario reconocerá la aplicación mediante el siguiente ícono que aparecerá en la lista de sus aplicaciones instaladas como se observa en la figura 29.

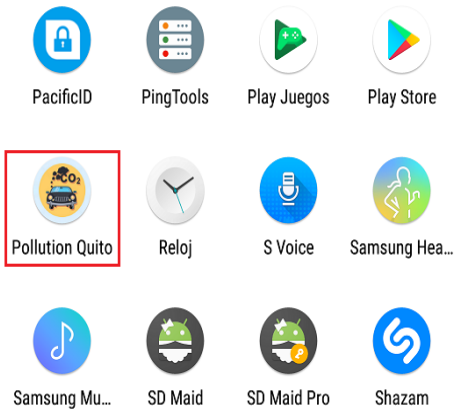


Figura 29. Ícono de acceso a la aplicación.

Al abrir la aplicación esta consta de una interfaz de inicio con una vista de bienvenida con los logos de la Universidad de las Américas y la Secretaría del Ambiente indicando la colaboración entre estas dos instituciones como se observa en la figura 30.

Figura 30. Interfaz inicial del aplicativo.

Para el acceso a la interfaz principal se ha implementado un menú desplegable para comodidad de los usuarios como se observa en la figura 31.

Figura 31. Menú desplegable del aplicativo.

El usuario ingresará a la interfaz de monitoreo mediante la opción MONITOREO, la cual desplegará una vista en un mapa generado por la utilización de la Api de Google Maps.

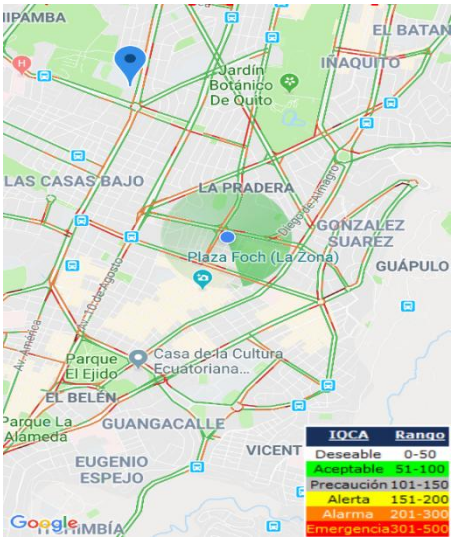


Figura 32. Interfaz de visualización del nivel de contaminación.

El la figura 33 podemos observar mediante código de colores el nivel de

contaminación, esto mediante el estudio realizado por la Secretaría del Ambiente se determina seis rangos del nivel de gases contaminantes según el impacto a la salud de las personas:

IQCA	Rango
Deseable	0-50
Aceptable	51-100
Precaución	101-150
Alerta	151-200
Alarma	201-300
Emergencia	301-500

Figura 33. Código de colores nivel de contaminación.

Esta interfaz muestra al usuario en forma de radar el nivel de contaminación a su alrededor en un radio de 500m, podemos observar un cuadro inferior con la descripción del código de colores correspondientes a los niveles de contaminación.

5.5. Aspectos generales del desarrollo de la aplicación

La APP desarrollada cuenta con dos Activities principales para el manejo de los diferentes métodos se han implementado tres clases diferentes, una Interface y seis clases POJO.

5.5.1. MainActivity

Es la actividad principal, donde se inicia la aplicación. Contiene la inicialización de algunas variables, de distintos objetos. Se creará la interfaz gráfica inicial e inicializará sus partes, así como sus escuchadores, una parte principal de este componente es el control del manejo de permisos que deben ser otorgados por el usuario para el correcto funcionamiento del aplicativo.

Esto se lo controla en base a la versión del sistema Android donde se ejecute la aplicación, ya que a partir de la versión 6.0 los permisos se manejan dentro de la aplicación y en los próximos años con las actualizaciones que saque al mercado Android pueden variar, pero la aplicación está desarrollada para adaptarse a cualquier cambio, para ello se implementó el siguiente método como se muestra en la figura a continuación:

```
public void onSectionAttached(int number) {
    switch (number) {
        case 1:
            mTitle = "Inicio";
            Intent intent2 = new Intent(this, MapsActivity.class);
            if (ActivityCompat.checkSelfPermission(MainActivity.this, android.Manifest.permission.ACCESS_FINE_LOCATION)
                    != PackageManager.PERMISSION_GRANTED) {

                if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this, android.Manifest.permission.ACCESS_FINE_LOCATION)) {
                    new SweetAlertDialog(MainActivity.this, SweetAlertDialog.WARNING_TYPE)
                            .setTitleText("Atencion")
                            .setContentText("Debes otorgar permisos")
                            .setConfirmText("Solicitar Permiso")
                            .setCancelText("Cancelar")
                            .setCancelClickListener((sweetAlertDialog) -> {
                                    sweetAlertDialog.cancel();
                            })
                            .setCancelClickListener((sweetAlertDialog) -> {
                                    sweetAlertDialog.cancel();
                                    ActivityCompat.requestPermissions(MainActivity.this, new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION}, PERMISO_MAPA)
                            })
                            .show();
                } else {
                    ActivityCompat.requestPermissions(MainActivity.this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, PERMISO_MAPA);


                }
            }else{

                this.startActivity(intent2);

            }

            break;

        case 2:
            mTitle = "Inicio";
            Intent intent5 = new Intent(this, SalirActivity.class);
            this.startActivity(intent5);
            break;

    }
}
```

Figura 34. Método que maneja la solicitud permisos para la aplicación.

5.5.2. MapsActivity

Es la actividad que se encarga de mostrar el componente principal de la aplicación que es la visualización de los niveles de contaminación alrededor del usuario en un mapa con la capa de tráfico proveniente de la Api de Google Maps, este componente inicializa los servicios de georeferenciación así como

también los diferentes métodos para la visualización de los marcadores de las estaciones de monitoreo, el radar relativo a los niveles de contaminación, el cuadro de información del código de colores.

En este componente se obtienen y se ejecutan los principales métodos para el procesamiento de los datos, uno de los principales es el método build_retrofit_and_get_response con el cual se hace uso del Web Service de Google Maps para determinar el nivel de tráfico alrededor del usuario, esto tomando en cuenta la posición actual del usuario se traza varios destinos en un radio de 500m y se calcula el valor promedio de distancia y tiempo para obtener la velocidad promedio del tráfico como se observa en la figura 35.

```
private void build_retrofit_and_get_response(String type) {

    String url = "https://maps.googleapis.com/maps/";

    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(url)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    RetrofitMaps service = retrofit.create(RetrofitMaps.class);

    Call<Example> call = service.getDistanceDuration("metric", origin.latitude + "," + origin.longitude, dest.latitude + "," + dest.longitude, type);


    call.enqueue(new Callback<Example>() {
        @Override
        public void onResponse(Response<Example> response, Retrofit retrofit) {

            try {
                //Remove previous line from map
                if (line != null) {
                    line.remove();
                }
                // This loop will go through all the results and add marker on each location.

                for (int i = 0; i < response.body().getRoutes().size(); i++) {
                    String distance = response.body().getRoutes().get(i).getLegs().get(i).getDistance().getText();
                    String time = response.body().getRoutes().get(i).getLegs().get(i).getDuration().getText();

                    String distancia = distance;
                    String tiempo = time;

                    String[] valor_distancia = distancia.split("\\s+");
                    String[] valor_tiempo = tiempo.split("\\s+");

                    float auxdistancia = Float.parseFloat(valor_distancia[0]);
                    float auxtiempo = Float.parseFloat(valor_tiempo[0]);

                    float auxdist = auxdistancia;
                    float auxtie = auxtiempo / 60;
                    velocidad = auxdist / auxtie;
                    String resvelocidad = String.format("%.02f", velocidad);
```

Figura 35. Método que maneja la obtención de la velocidad promedio.

Otro punto importante que se ejecuta en este componente es el cálculo del IDW mediante el método *onLocationChanged*, considerando la posición actual del usuario y la ubicación de las distintas estaciones de monitoreo se aplica la fórmula para el cálculo de la interpolación como se observa en la figura 36.

```
public void onLocationChanged(Location location) {
    //mapRipple.withNumberOfRipples(3);
    this.location = location;

    if (mapRadar.isAnimationRunning())
        mapRadar.withLatLng(new LatLng(location.getLatitude(), location.getLongitude()));


    //CALCULOS DE LA INTERPOLACION (IDW)
    //OBTENCION DE LA UBICACION ACTUAL

    Location ubicacion = new Location("UBICACION");
    ubicacion.setLatitude(location.getLatitude());
    ubicacion.setLongitude(location.getLongitude());

    //UBICACION DE LOS RADARES

    Location location1 = new Location("LOS_CHILLOS");
    location1.setLatitude(-0.2987465);  //latitud
    location1.setLongitude(-78.456); //longitud

    Location location2 = new Location("TUMBACO");
    location2.setLatitude(-0.215192);  //latitud
    location2.setLongitude(-78.4); //longitud

    Location location3 = new Location("GUAMANI");
    location3.setLatitude(-0.334);  //latitud
    location3.setLongitude(-78.5534); //longitud

    Location location4 = new Location("EL_CAMAL");
    location4.setLatitude(-0.2515083);  //latitud
    location4.setLongitude(-78.5172); //longitud

    Location location5 = new Location("CENTRO");
    location5.setLatitude(-0.2206528);  //latitud
    location5.setLongitude(-78.5134); //longitud

    Location location6 = new Location("BELISARIO");
    location6.setLatitude(-0.1848222);  //latitud
    location6.setLongitude(-78.496); //longitud

    Location location7 = new Location("COTOCOLLAO");
    location7.setLatitude(-0.1114389);  //latitud
    location7.setLongitude(-78.5); //longitud

    Location location8 = new Location("CARAPUNGO");
    location8.setLatitude(-0.095686);  //latitud
    location8.setLongitude(-78.4497); //longitud

    Location location9 = new Location("SAN_ANTONIO");
    location9.setLatitude(-0.0121152);  //latitud
    location9.setLongitude(-78.45); //longitud

    //OBTENCIÓN DE LA DISTANCIA

    float distance1 = ubicacion.distanceTo(location1);
    float distance2 = ubicacion.distanceTo(location2);
    float distance3 = ubicacion.distanceTo(location3);
    float distance4 = ubicacion.distanceTo(location4);
    float distance5 = ubicacion.distanceTo(location5);
    float distance6 = ubicacion.distanceTo(location6);
    float distance7 = ubicacion.distanceTo(location7);
    float distance8 = ubicacion.distanceTo(location8);
    float distance9 = ubicacion.distanceTo(location9);

    //MOSTRAR RESULTADOS

    res = ((valor_est1 / distance1)+(valor_est2 / distance2)+(valor_est3 / distance3)+(valor_est4 / distance4)+(valor_est5 / distance5)+(valor_est6 / distance6)+(valor_est7 / distance7)+(valor_est8 /...
    residw = String.format("%.02f", res);

}
```

Figura 36. Método que maneja el cálculo de la interpolación.

Como último apartado de este componente tenemos la gestión del resultado final por medio de la coloración del radar, de esta manera el usuario podrá visualizar de una manera amigable el nivel de material contaminante a su alrededor, esto lo logramos en el método denominado *setUpMapIfNeeded* como se observa en la figura 37.

```
private void setUpMapIfNeeded() {
    // Do a null check to confirm that we have not already instantiated the map.

    float resultado_final = velocidad + res;

    if (resultado_final <= 100)
    {
        color_radar_secundario = "#0014c221";
        color_radar_primario ="#14c221";

    } else {


        color_radar_secundario = "#00e87e2c";
        color_radar_primario ="#e87e2c";

    }

    if (mMap == null) {
        // Try to obtain the map from the SupportMapFragment.
        mMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();

        // Check if we were successful in obtaining the map.*/

        if (mMap != null) {


            mMap.getUiSettings().setScrollGesturesEnabled(true);
            mMap.getUiSettings().setAllGesturesEnabled(true);
            mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
            mMap.setTrafficEnabled(true);
            mMap.setMyLocationEnabled(true);
            mMap.getUiSettings().setMyLocationButtonEnabled(true);
            //mMap.getUiSettings().setCompassEnabled(false);
            mMap.getUiSettings().setRotateGesturesEnabled(true);
            //mMap.getUiSettings().setMapToolbarEnabled(false);
            Location location = mMap.getMyLocation();

            if (location == null)
                location = locationTrackObj.getLocation();
            try {
                mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(new
                        LatLng(location.getLatitude(), location.getLongitude()), 14));
            } catch (Exception e) {
                e.printStackTrace();
            }
            if (location != null) {
                latLng = new LatLng(location.getLatitude() , location.getLongitude());


            }else {
                latLng = new LatLng(0.0, 0.0);


            }
            // mMap is GoogleMap object, latLng is the location on map from which ripple should start
            mapRadar = new MapRadar(mMap, latLng, context);

            //mapRadar.withClockWiseAnticlockwise(true);
            mapRadar.withDistance(1000);
            mapRadar.withClockwiseAnticlockwiseDuration(2);
            mapRadar.withOuterCircleFillColor(Color.parseColor("#12000000"));
            mapRadar.withOuterCircleStrokeColor(Color.parseColor(color_radar_secundario));
            mapRadar.withRadarColors(Color.parseColor("#00000000"), Color.parseColor("#ff000000"));  //starts from transparent to fuly black

            //CAMBIAR DE COLOR AL RADAR SEGUN ID#############################
            //PRIMER PARSE COLOR ES DEGRADADO, SEGUNDO PARSE COLOR ES COLOR PRINCIPAL

            mapRadar.withRadarColors(Color.parseColor(color_radar_secundario), Color.parseColor(color_radar_primario));  //starts from transparent
            mapRadar.withOuterCircleStrokewidth(7);
            mapRadar.withRadarSpeed(5);
            mapRadar.withOuterCircleTransparency(0.5f);
            mapRadar.withRadarTransparency(0.5f);
            mapRadar.startRadarAnimation();      //in onMapReadyCallBack

        }

    }
}
```

Figura 37. Método que maneja la visualización del resultado final.

5.5.3. MapRadar

Es la clase encargada de construir sobre a capa de tráfico de Google Maps un radar cuyo centro se moverá según el desplazamiento del usuario, además podemos customizar el radar mediante diferentes parametrizaciones como se observa en la figura 38.

```
public class MapRadar {
    private GoogleMap mGoogleMap;
    private LatLng mLatLng, mPrevLatLng;
    private BitmapDescriptor mBackgroundImageDescriptor, mBackgroundImageSweepDescriptor;          //ripple image.
    private float mTransparency = 0.5f;                                 //transparency of image.
    private volatile int mDistance = 1000;                              //distance to which ripple should be shown in metres
    private int mFillColor = Color.TRANSPARENT;                         //fill color of circle
    private int mStrokeColor = 0x38728f;                                //border color of circle
    private int mStrokeWidth = 4;                                       //border width of circle
    private GradientDrawable mOuterDrawable;
    private boolean isAnimationRunning = false;
    private Handler mSweepHandler, mOuterHandler;
    private int mRotationAngle = 0;
    private boolean isThreeSixty = false;
    private GroundOverlay mGroundOverlaySweep, mGroundOverlay;
    private ValueAnimator mAnimatorSweep;
    private float mSweepTransparency = 0.5f;                            //transparency of image.
    private boolean mSweepAnimationClockwiseAnticlockwise = false;
    private int mSweepAnimationClockwiseAnticlockwiseDuration = 1;
    private int mSweepSpeed = 2;                                        //increase this to increase speed
    private int mCurrentAngle = 0;
    private int mColors[] = {0x0038728f, 0xff38728f};                    //sweep animation colors

    public MapRadar(GoogleMap googleMap, LatLng latLng, Context context) {
        mGoogleMap = googleMap;
        mLatLng = latLng;
        mPrevLatLng = latLng;
        mOuterDrawable = (GradientDrawable) ContextCompat.getDrawable(context, R.drawable.background);
    }
```

Figura 38. Método que maneja el despliegue del radar sobre el mapa.

5.5.4. NavigationDrawerFragment

Clase relacionada directamente con el MainActivity, la cual maneja las diferentes parametrizaciones de la interfaz principal, es decir, tiene todos los métodos de escucha para direccionar las peticiones del usuario dentro de la interfaz principal así como la customización del menú desplegable.

Este menú puede cambiar la forma de desplazamiento ya sea horizontal o vertical según sea el requerimiento funcional.

5.5.5. UiUtil

Es la clase encargada de utilizar las diferentes dependencias para poder incorporar gráficos y otras utilidades sobre el Api de Google Maps. A continuación se muestra la versión de SDK y las dependencias utilizadas como se observa en la figura 39.

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

android {
    compileSdkVersion 23
    buildToolsVersion "21.1.2"
    defaultConfig {
        applicationId 'com.android.naranjod.viajaremos'
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
        multiDexEnabled true
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }

}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile 'com.android.support:appcompat-v7:23.2.0'
    compile 'com.google.android.gms:play-services:8.4.0'
    compile 'org.jbundle.util.osgi.wrapped:org.jbundle.util.osgi.wrapped.org.apache.http.client:4.1.2'
    compile 'cn.pedant.sweetalert:library:1.3'
    compile 'com.google.code.gson:gson:2.6.1'
    compile 'com.mcxiaoke.volley:library:1.0.19'
    compile 'com.akexorcist:googledirectionlibrary:1.0.4' // Custom Google Direction API \\
    compile 'com.squareup.retrofit:retrofit:2.0.0-beta2'
    compile 'com.google.code.gson:gson:1.7.2'
    compile 'com.squareup.retrofit:converter-gson:2.0.0-beta2'
    compile 'com.squareup.okhttp:okhttp:2.4.0'
}
```

Figura 39. Componentes base utilizados en la aplicación.

5.5.6. RetrofitMaps

Es la interfaz encargada del envío de la ApiKey creada para el uso del Api de Google Maps, además se encarga de también enviar los parámetros necesarios como origen, destino y modo para la obtención de la velocidad promedio como se observa en la figura 40.

```
package com.android.naranjod.viajaremos;
import  com.android.naranjod.viajaremos.POJO.Example;
import retrofit.Call;
import retrofit.http.GET;
import retrofit.http.Query;

/**
 * Created by NaranjoD on 23/10/2018.
 */
public interface RetrofitMaps {
    /*
     * Retrofit get annotation with our URL
     * And our method that will return us details of student.
     */
    @GET("api/directions/json?key=AIzaSyAVHhfyr_qfr_nxx3EZz2YosMhhM49xxxx")
    Call<Example> getDistanceDuration(@Query("units") String units, @Query("origin") String origin, @Query("destination") String destination, @Query("mode") String mode);

}
```

Figura 40. Utilización de la ApiKey.

5.5.7. Clases POJO

Para el proyecto se implementaron seis clases POJO ya que se tratan de objetos Java común, no vinculado por ninguna restricción especial que no sea forzada por la especificación de lenguaje Java y que no requieren ninguna ruta de clase, además son fáciles de escribir y de entender. Las clases POJO en el proyecto se los utilizan para aumentar la legibilidad y la reutilización de un programa, específicamente para la obtención de los datos de distancia, tiempo, rutas y modo para el cálculo de la velocidad promedio.

5.5.8. AndroidManifest

Un elemento muy importante en el desarrollo e implementación de cualquier

aplicativo móvil dirigido para dispositivos Android es el componente AndroidManifest.xml, este archivo de configuración es donde vamos a poder aplicar las configuraciones básicas de nuestra aplicación. Su configuración puede realizarse a través de una interfaz gráfica, pero es recomendable conocer la sintaxis ya que en muchas ocasiones será más fácil y rápido hacerlo desde el propio XML. El AndroidManifest está situado en la raíz de cada aplicación, para el correcto funcionamiento de la aplicación necesitamos tener acceso a componentes tales como el acceso al almacenamiento interno, acceso a la localización, acceso a internet, permiso para recibir mapas y acceso al estado de la red como se observa en la figura 41.

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

    <permission
        android:name="com.android.naranjod.viajaremos.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission android:name="com.android.naranjod.appturismor.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <protected-broadcast android:name="android.intent.action.MEDIA_MOUNTED" />

    <!--
The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but are recommended.
    -->

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="Pollution in Quito"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="8487000" />
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AIzaSyAVHhfyr_qfr_nxx3EZz2YosMhhM4 " />

        <activity
            android:name=".MainActivity"
            android:label="Pollution Quito" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MapsActivity"
            android:label="MONITOREO"
            android:parentActivityName=".MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity" />
        </activity>
        <activity
            android:name=".SalirActivity"
            android:label="SALIR"
            android:parentActivityName=".MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".MainActivity" />
        </activity>

    </application>

</manifest>
```

Figura 41. AndroidManifest.xml del aplicativo.

6. Capítulo VI: Evaluación y resultados

En este capítulo se pondrá a prueba la aplicación desarrollada en base al mejor modelo de regresión, es así que se comprobará el correcto funcionamiento de cada una de sus partes. Se analizarán los resultados obtenidos, y se concluirá si es correcta o no.

6.1. Evaluación de las regresiones

En base a la ejecución de las tres regresiones se puede constatar que la característica más relevante que influye en el nivel de contaminación alrededor del usuario es el tráfico con un coeficiente de correlación de 0.7079, tal como se puede observar en el resumen de la tabla 33.

Es así que en base al aprendizaje automático podemos concluir que para el desarrollo del aplicativo y su fiabilidad se debe considerar la velocidad del tráfico como eje principal para el cálculo del nivel de contaminación, características como la elevación, dirección del viento, velocidad del viento son irrelevantes para el modelo inclusive IDW pero para una resolución mejor también se lo implementará.

Tabla 33. Resumen coeficiente de correlación.

Métodos	Valor coeficiente de correlación
Método 1: MicroDust = $PM_{2.5}$	0.5222
Método 2: MicroDust = elevación + meteorología	0.6998
Método 3: MicroDust = tráfico	0.7079

6.2. Evaluación del aplicativo (entorno de pruebas)

Para la realización de pruebas en la aplicación, tanto en su desarrollo como en su finalización se han utilizado dos tipos de entornos distintos. Por un lado un entorno virtual y por otro lado un entorno físico.

– Entorno Virtual: Android Studio provee una herramienta llamada Android Virtual Device (AVD). Durante las pruebas se ha utilizado este entorno con una simulación del dispositivo Nexus 5 en la versión de API número 23.

– Entorno Físico: El dispositivo físico utilizado para la realización de pruebas de la aplicación han sido:

- Samsung S5
- HTC One
- Sony Xperia Z5

6.2.1. Pruebas

Para la descripción de las diferentes pruebas a realizar, se ha construido una tabla con la siguiente información:

Tabla 34. Ejemplo tabla pruebas del aplicativo (PS).

Identificador:	PS-XX
Nombre:	
Descripción:	
Pasos:	
Resultado:	

Tomado de Instituto de Ingeniería Eléctrica y Electrónica, s.f.

– Identificador: Para la identificación de cada prueba del sistema se le asignará un número de identificador único. Este número empezará por PS (prueba del sistema) seguido de un número.

– Nombre: Titular que resume la prueba.

– Descripción: Breve descripción sobre el objetivo de la prueba.

– Pasos: Pasos a realizar para verificar y reproducir la prueba.

– Resultado: Resultado de la prueba, puede ser Correcta o Incorrecta.

Se ha decidido realizar una clasificación en distintos tipos de prueba para facilitar su identificación y lectura.

6.2.2. Pruebas de Interfaz

En esta sección se incluirán todas las pruebas sobre la interfaz. Estas pruebas indicarán la aparición o no de los elementos a probar como se observa en las tablas 35 y 36.

Tabla 35. PS-01: Mensaje de permisos (GPS).

Identificador:	PS- 01
Nombre:	Mensaje de permisos (GPS)
Descripción:	Aparición del mensaje de solicitud de permiso al entrar por primera vez a la aplicación.
Pasos:	1. Instalar la aplicación 2. Abrir la aplicación 3. Comprobar que el mensaje aparece.
Resultado:	✓ Correcto

Tabla 36. PS-02: Despliegue de la interfaz principal.

Identificador:	PS-02
Nombre:	Despligue de la interfaz principal
Descripción:	Se mostrará el despliegue de la interfaz principal con sus respectivas opciones
Pasos:	1. Abrir la aplicación 2. Otorgar los permisos 3. Desplegar menú principal
Resultado:	✓ Correcto

6.2.3. Pruebas de funcionamiento

En esta sección se incluirán todas las pruebas sobre el correcto funcionamiento de todos aquellos elementos que aparecen en la interfaz. Estas opciones serán evaluadas por el usuario verificando que cumplan con su función para la cual fueron creadas como se observa en las tablas 37 y 38.

Tabla 37. PS-03: Funcionamiento de la interfaz de monitoreo.

Identificador:	PS-03
Nombre:	Interfaz MONITOREO
Descripción:	Al seleccionar la opción MONITOREO del menú principal se debe desplegar la interfaz del mapa con la capa de tráfico y el radar de nivel de contaminación alrededor de la ubicación del usuario
Pasos:	1. Abrir la aplicación 2. Otorgar permiso de acceso a localización 3. Desplegar menú principal 4. Seleccionar opción MONITOREO 5. Se mostrará interfaz de MONITOREO
Resultado:	✓ Correcto

Tabla 38. PS-04: Funcionamiento de la opción salir.

Identificador:	PS-04
Nombre:	Salir de la aplicación
Descripción:	Al seleccionar la opción SALIR del menú la aplicación debe cerrarse
Pasos:	1. Abrir la aplicación 2. Otorgar permisos de acceso a ubicación 3. Desplegar menú principal 4. Seleccionar la opción SALIR 5. La aplicación debe cerrarse
Resultado:	✓ Correcto

6.2.4. Pruebas de fiabilidad

En esta sección se definirán las pruebas creadas para comprobar la fiabilidad de la aplicación en su objetivo principal: monitorear el nivel de contaminación alrededor del usuario. Esta prueba se la realizó en base al sistema de monitoreo implementado vía web por la Secretaría del Ambiente pero interpolando los resultados ya que en base al modelo escogido para el desarrollo de la aplicación la velocidad del tráfico alrededor del usuario es la característica más importante para la determinación del nivel de contaminación alrededor de los usuarios como se observa en la tabla 39.

Tabla 39. PS-05: Fiabilidad de la detección del nivel de contaminación

Identificador:	PS-05
Nombre:	Detección del nivel de contaminación
Descripción:	En esta prueba se comprobará si el nivel de contaminación mostrado mediante el radar en la aplicación es correcta
Pasos:	1. Abrir aplicación 2. Otorgar permisos para acceso a localización 3. Seleccionar en el ménu la opción MONITOREO 4. Comprobar que el radar muestre el nivel de contaminación correcta
Resultado:	✓ Correcto

En base a los datos obtenidos por la página de la Secretaría del Ambiente como se observa en la figura 43 y en la aplicación como se observa en la figura 44 se ha realizado la prueba de concatenación entre las dos aplicaciones a la misma hora, siendo las 10:22 del 03 de Enero del 2019 se constata que las dos aplicaciones muestran que el nivel de contaminación es ACEPTABLE. Es así que podemos determinar la fiabilidad del aplicativo con datos del tráfico mediante la comparativa de la solución implementada por la Secretaría del Ambiente y el aplicativo móvil desarrollado, tomamos para la prueba la estación Belisario.

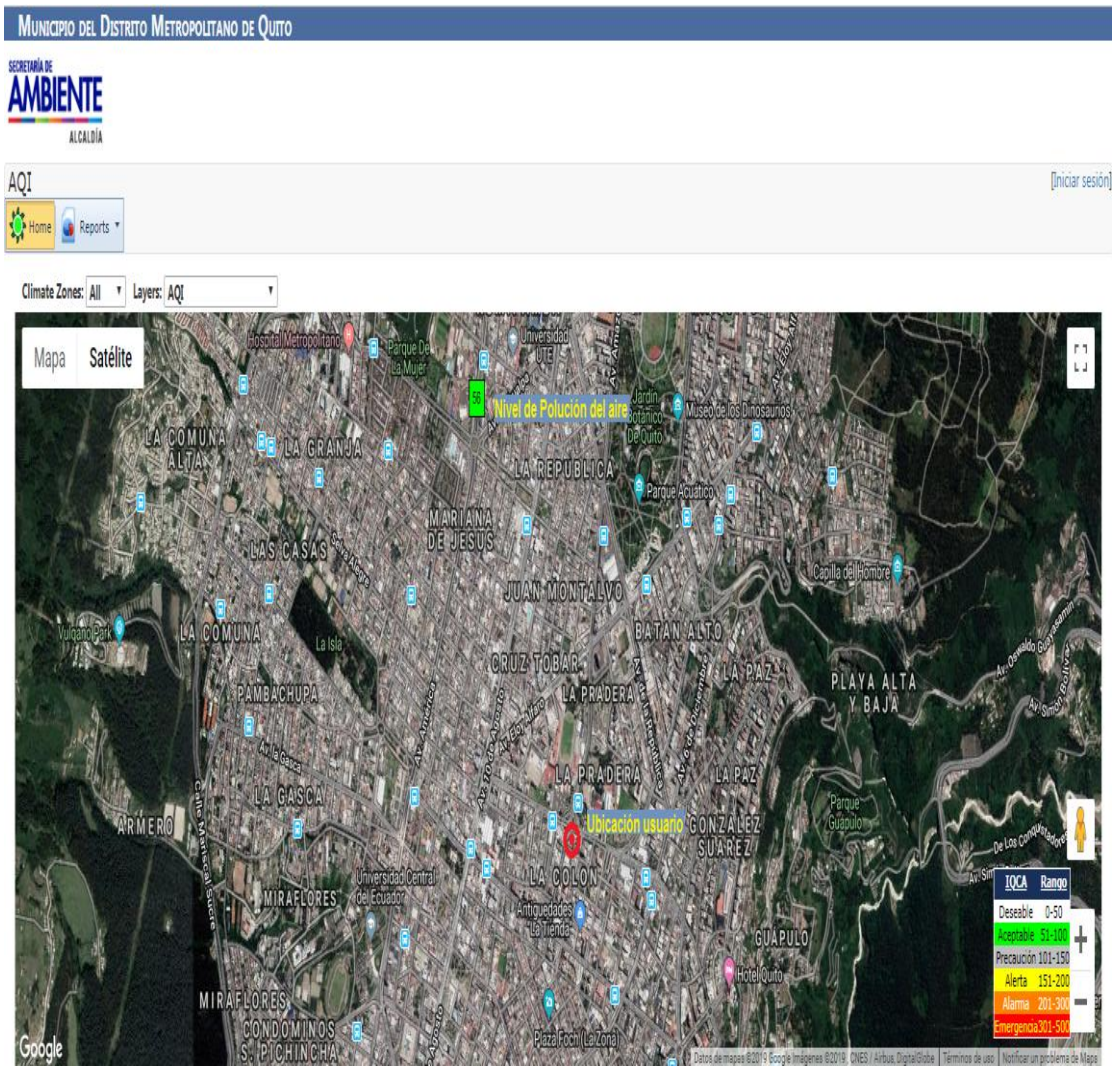


Figura 42. Calidad del aire
Tomado de Secretaría del Ambiente, s.f.

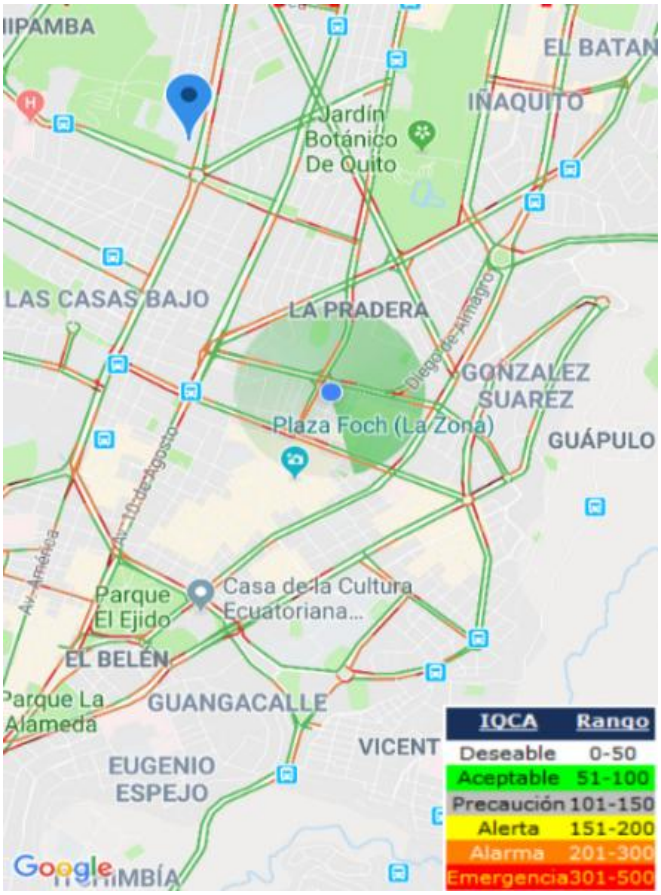


Figura 43. Nivel de contaminación.

7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

La oportunidad de realizar este proyecto, ha supuesto en reto tanto personal como académico. El empeño y dedicación durante su desarrollo, ha supuesto la adquisición de una nueva perspectiva para la resolución de problemas, que será de gran utilidad en proyectos futuros.

Como hemos observado durante el desarrollo de este proyecto, se han llegado a varias conclusiones:

– La elaboración de varios modelos nos llevó como conclusión a determinar la mejor característica para determinar el nivel de tráfico alrededor del usuario.

– La característica de IDW al inicio era la más opcional para la determinación del nivel de contaminación pero al final se observó que el nivel de tráfico fue más efectiva, llegando a la conclusión de que IDW puede ser descartable.

– El análisis de diferentes APIs para la obtención de información de tráfico, concluyendo que Google Maps fue la mejor opción para el proyecto.

7.2. Recomendaciones

En base al estudio realizado referente a la aplicación de modelos predictivos, se puede recomendar el uso de otros métodos existentes para el procesamiento de los datos.

Uno de los puntos más sencillos y obvios al que todo desarrollador de APPs debería prestar atención es estudiar bien la documentación que ofrece la empresa responsable de cada sistema operativo.

En este caso como desarrollo del aplicativo fue dirigido al sistema operativo Android, se tuvo que estudiar y analizar toda la información que Google puso al alcance, en especial el uso de sus diferentes APIs, así como el estudio de los diferentes SDK que están disponibles. Se recomienda el estudio de estos componentes para seleccionar la mejor arquitectura y componentes para el desarrollo de cualquier aplicativo dirigido a Android.

REFERENCIAS

Android Dev, (2015). “Apis para desarrolladores de Android”. Recuperado el 18 de noviembre de 2018 de https://developer.android.com/apisandroid

A. Katar, (2015). “Estudio sobre ventas en el mercado móvil”. Recuperado el 12 de octubre de 2018 de http://www.kantarworldpanel.com/global/smartphone-os-market-share/

C.J. Bernardos Cano, (2015). “Material asignatura Redes Inalámbricas y móviles”. Recuperado el 28 de noviembre de 2018 de http://bibing.com/models/apis/1948/fichero/info-networking.pdf

D. Barth, Barth, (2009). “*The bright side of sitting in traffic: Crowdsourcing road congestion data*”. Recuperado el 12 de noviembre del 2018 de https://googleblog.blogspot.com.es/2009/08/bright-side-of-sitting-in-traffic.html

F. Rose, (2009). “*Pocket Monster: How DoCoMo’s wireless Internet service went from fad to phenom – and turned Japan into the first post-PC nation*”. Recuperado el 12 de enero de 2019 de https://www.wired.com/2001/09/docomo/

Google Analíticas, (2017). “*Google Traffic*”. Recuperado el 12 de diciembre de 2018 de https://en.googletraffic.org/wiki/Google_Traffic

Google Analíticas, (2017). “Información sobre mapas”. Recuperado el 22 de diciembre de 2018 de https://support.google.com/maps/answer/3092439?hl=en&rd=2

Google Analíticas, (2015). “API de mapas de servicios Google para desarrolladores”. Recuperado el 18 de noviembre de 2018 de https://developers.google.com/maps/?hl=es

J. Brain, (2018). “Barómetro”. Recuperado el 10 de enero de 2019 de https://es.techbrain.org/Barómetro

K. Kroening, (2018). “Magnetómetro”. Recuperado el 13 de enero de 2018 de https://es.wikipedia.org/wiki/Magnetómetro

M. Arenas, (2008). “Diseño e implementación de un sistema de adquisición de aceleraciones con procesamiento mediante microprocesador”. Recuperado el 21 de diciembre de 2018 de http://bibing.us.es/proyectos/abreproy/11638/fichero/Capitulo+4.pdf

M. Bazeley, (2005). “*Google acquires traffic info start-up Zipdash*”. Recuperado el 13 de diciembre de 2018 de http://www.siliconbeat.com/entries/2005/03/30/google_acquires_traffic_info_startup_zipdash.html

N. Chintalacheruvu, V. Muthukumar, (2012). “*Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems*”. Recuperado el 13 de noviembre de 2018 de http://file.scirp.org/pdf/JTTs20120400004_70073830.pdf

Nokia Enterprise, (2015). “API de tráfico de servicios HereWeGo para desarrolladores”. Recuperado el 26 de noviembre de 2018 de https://developer.here.com/api_traffic/android

O. Lambda, (2011). “Casos de Uso”. Recuperado el 14 de enero de 2019 de https://es.lambda.org/structure/Caso_de_uso

P. Mohan, (2008). “*TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones*”. Recuperado el 12 de diciembre de 2018 de https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2008-59.pdf

R. Stoichkov, (2013). “*Android Smartphone Application for Driving Style Recognition*”. Recuperado el 13 de diciembre de 2018 de http://www.eislab.fim.unipassau.de/files/publications/students/Stoichkov-Projektarbeit.pdf

S. Ayub, A. Bahraminisaab, (2012). “*A sensor Fusion Method for Smartphone Orientation Stimation*”. Recuperado el 17 de noviembre de 2018 de http://www.cms.livjm.ac.uk/pgnet2012/Proceedings/Papers/1569603133.pdf

Scrum S.A, (2015). “*Scrum of Scrum Alliance*”. Recuperado el 28 de noviembre de 2018 de https://www.scrumalliance.org

S. Morrissey, (2010). “*iOS Forensic Analysis for iPhone, iPad and iPod touch*”. Recuperado el 12 de noviembre de 2018 de http://refer.com/ficheros/ /1948/fichero/ios.pdf

Tecnología móvil, (2018). “GPS y sus diversas aplicaciones en los dispositivos móviles”. Recuperado el 10 de diciembre de 2018 de https://es.technomov.org/wiki/Sistema_de_posicionamiento_global

UDLA, (2018). “*Quantifying decade-long effects of fuel and traffic regulations on urban ambient PM2.5 pollution in a mid-size South American city. Atmospheric Pollution Research*”. Recuperado el 23 de diciembre de 2018 de http://biblioteca.udla.edu.ec/client/air_pollution.pdf

V. Arias, L. Lugo, (2014). “Informe calida del aire de Quito”. Recuperado el 12 de enero de 2019 de http://www.quitoambiente.gob.ec/ambiente/images/Secretaria_Ambiente/red_monitoreo/informacion/ iqca_2014.pdf

Waze Enterprise, (2017). “Información sobre navegador”. Recuperado el 14 de noviembre de 2018 de https://www.waze.com/es/about

Witten, I. Frank, E, Hall, M. (2011). “*Data Mining Practical Machine Learning Tools and Techniques*”. Recuperado el 10 de octubre de 2018 de https://www.wired.com/218/10/tecniques

ANEXOS

Anexo 1. Datos obtenidos por MicroDust (extracto de los datos)

	A	B	C	D	E
1	DATE	TIME	PM2.5	COUNT	
2	19/10/18	10:21:06	0.244	977	
3	19/10/18	10:21:16	0.22	976	
4	19/10/18	10:21:26	0.323	975	
5	19/10/18	10:21:36	0.256	974	
6	19/10/18	10:21:46	0.231	973	
7	19/10/18	10:21:56	0.242	972	
8	19/10/18	10:22:06	0.241	971	
9	19/10/18	10:22:16	0.21	970	
10	19/10/18	10:22:26	0.258	969	
11	19/10/18	10:22:36	0.298	968	
12	19/10/18	10:22:46	0.231	967	
13	19/10/18	10:22:56	0.231	966	
14	19/10/18	10:23:06	0.294	965	
15	19/10/18	10:23:16	0.229	964	
16	19/10/18	10:23:26	0.229	963	
17	19/10/18	10:23:36	0.355	962	
18	19/10/18	10:23:46	0.712	961	
19	19/10/18	10:23:56	0.478	960	
20	19/10/18	10:24:06	0.317	959	
21	19/10/18	10:24:16	0.293	958	
22	19/10/18	10:24:26	0.281	957	
23	19/10/18	10:24:36	0.251	956	
24	19/10/18	10:24:46	0.211	955	
25	19/10/18	10:24:56	0.198	954	
26	19/10/18	10:25:06	0.226	953	
27	19/10/18	10:25:16	0.271	952	
28	19/10/18	10:25:26	0.213	951	
29	19/10/18	10:25:36	0.207	950	
30	19/10/18	10:25:46	0.211	949	
31	19/10/18	10:25:56	0.221	948	
32	19/10/18	10:26:06	0.263	947	
33	19/10/18	10:26:16	0.247	946	
34	19/10/18	10:26:26	0.251	945	
35	19/10/18	10:26:36	0.262	944	
36	19/10/18	10:26:46	0.219	943	
37	19/10/18	10:26:56	0.231	942	
38	19/10/18	10:27:06	0.221	941	
39	19/10/18	10:27:16	0.214	940	
40	19/10/18	10:27:26	0.214	939	
41	19/10/18	10:27:36	0.234	938	
42	19/10/18	10:27:46	0.226	937	
43	19/10/18	10:27:56	0.208	936	
44	19/10/18	10:28:06	0.296	935	
45	19/10/18	10:28:16	0.229	934	

Anexo 2. Datos obtenidos por el GPS (extracto de los datos)

	A	B	C	D	E
1	Date/time	Latitude	Longitude	Elevation (m)	
2	19/10/2018 10:21	-0.1861352	-78.4957524	2842.8	
3	19/10/2018 10:21	-0.1861127	-78.4957391	2842.8	
4	19/10/2018 10:21	-0.1860894	-78.4957454	2842.8	
5	19/10/2018 10:21	-0.1860696	-78.4957386	2842.8	
6	19/10/2018 10:21	-0.1860632	-78.4957288	2842.8	
7	19/10/2018 10:21	-0.186057	-78.4957259	2842.3	
8	19/10/2018 10:22	-0.186056	-78.4957254	2842.3	
9	19/10/2018 10:22	-0.1860545	-78.4957275	2842.3	
10	19/10/2018 10:22	-0.1860596	-78.4957345	2842.3	
11	19/10/2018 10:22	-0.1860737	-78.4957397	2842.3	
12	19/10/2018 10:22	-0.1860834	-78.4957242	2842.3	
13	19/10/2018 10:22	-0.1860939	-78.4957078	2842.3	
14	19/10/2018 10:23	-0.1860909	-78.4957098	2842.3	
15	19/10/2018 10:23	-0.1860822	-78.4957211	2842.3	
16	19/10/2018 10:23	-0.186098	-78.4957227	2842.3	
17	19/10/2018 10:23	-0.1861085	-78.4957309	2843.7	
18	19/10/2018 10:23	-0.1861184	-78.4957249	2843.7	
19	19/10/2018 10:23	-0.1861028	-78.495733	2843.7	
20	19/10/2018 10:24	-0.1860988	-78.4957408	2843.7	
21	19/10/2018 10:24	-0.1861004	-78.4957237	2843.7	
22	19/10/2018 10:24	-0.1861005	-78.4957177	2843.7	
23	19/10/2018 10:24	-0.1861041	-78.4957119	2843.7	
24	19/10/2018 10:24	-0.1860891	-78.4957099	2843.7	
25	19/10/2018 10:24	-0.1860872	-78.495696	2843.7	
26	19/10/2018 10:25	-0.1860855	-78.4957014	2843.7	
27	19/10/2018 10:25	-0.186087	-78.4957084	2844.2	
28	19/10/2018 10:25	-0.1860918	-78.4957243	2844.2	
29	19/10/2018 10:25	-0.1861017	-78.4957135	2844.2	
30	19/10/2018 10:25	-0.1861104	-78.495703	2844.2	
31	19/10/2018 10:25	-0.186118	-78.4956967	2844.2	
32	19/10/2018 10:26	-0.1861206	-78.4957059	2844.2	
33	19/10/2018 10:26	-0.1861373	-78.4957083	2844.2	
34	19/10/2018 10:26	-0.186157	-78.4957149	2844.2	
35	19/10/2018 10:26	-0.1861364	-78.4957397	2844.2	
36	19/10/2018 10:26	-0.1861348	-78.4957464	2844.2	
37	19/10/2018 10:26	-0.186126	-78.4957469	2844.2	
38	19/10/2018 10:27	-0.1861236	-78.4957553	2844.2	
39	19/10/2018 10:27	-0.1861043	-78.4957497	2844.2	
40	19/10/2018 10:27	-0.1861003	-78.4957336	2844.2	
41	19/10/2018 10:27	-0.186098	-78.4957299	2844.2	
42	19/10/2018 10:27	-0.1860963	-78.4957397	2844.2	
43	19/10/2018 10:27	-0.1860792	-78.4957326	2844.2	
44	19/10/2018 10:28	-0.186072	-78.495734	2844.2	
45	19/10/2018 10:28	-0.1860583	-78.4957288	2843.7	

Anexo 3. Concatenación de los datos del MicroDust y los de GPS (extracto de los datos)

	A	B	C	D	E	F	G	H	I	J	K
1	Date MIcroDust	Time MicroDust	PM 2.5	BELISARIO PM 2.5	CENTRO PM 2.5	Date GPS	Time GPS	Latitude	Longitude	Elevation (m)	
2	19/10/18	10:21:06	0.244	0.378	0.697	19/10/2018	10:21:02	-0.1861352	-78.4957524	2842.8	
3	19/10/18	10:21:16	0.22			19/10/2018	10:21:12	-0.1861127	-78.4957391	2842.8	
4	19/10/18	10:21:26	0.323			19/10/2018	10:21:22	-0.1860894	-78.4957454	2842.8	
5	19/10/18	10:21:36	0.256			19/10/2018	10:21:32	-0.1860696	-78.4957386	2842.8	
6	19/10/18	10:21:46	0.231			19/10/2018	10:21:42	-0.1860632	-78.4957288	2842.8	
7	19/10/18	10:21:56	0.242			19/10/2018	10:21:52	-0.186057	-78.4957259	2842.3	
8	19/10/18	10:22:06	0.241			19/10/2018	10:22:02	-0.186056	-78.4957254	2842.3	
9	19/10/18	10:22:16	0.21			19/10/2018	10:22:12	-0.1860545	-78.4957275	2842.3	
10	19/10/18	10:22:26	0.258			19/10/2018	10:22:22	-0.1860596	-78.4957345	2842.3	
11	19/10/18	10:22:36	0.298			19/10/2018	10:22:32	-0.1860737	-78.4957397	2842.3	
12	19/10/18	10:22:46	0.231			19/10/2018	10:22:42	-0.1860834	-78.4957242	2842.3	
13	19/10/18	10:22:56	0.231			19/10/2018	10:22:52	-0.1860939	-78.4957078	2842.3	
14	19/10/18	10:23:06	0.294			19/10/2018	10:23:02	-0.1860909	-78.4957098	2842.3	
15	19/10/18	10:23:16	0.229			19/10/2018	10:23:12	-0.1860822	-78.4957211	2842.3	
16	19/10/18	10:23:26	0.229			19/10/2018	10:23:22	-0.186098	-78.4957227	2842.3	
17	19/10/18	10:23:36	0.355			19/10/2018	10:23:32	-0.1861085	-78.4957309	2843.7	
18	19/10/18	10:23:46	0.712			19/10/2018	10:23:42	-0.1861184	-78.4957249	2843.7	
19	19/10/18	10:23:56	0.478			19/10/2018	10:23:52	-0.1861028	-78.495733	2843.7	
20	19/10/18	10:24:06	0.317			19/10/2018	10:24:02	-0.1860988	-78.4957408	2843.7	
21	19/10/18	10:24:16	0.293			19/10/2018	10:24:12	-0.1861004	-78.4957237	2843.7	
22	19/10/18	10:24:26	0.281			19/10/2018	10:24:22	-0.1861005	-78.4957177	2843.7	
23	19/10/18	10:24:36	0.251			19/10/2018	10:24:32	-0.1861041	-78.4957119	2843.7	
24	19/10/18	10:24:46	0.211			19/10/2018	10:24:42	-0.1860891	-78.4957099	2843.7	
25	19/10/18	10:24:56	0.198			19/10/2018	10:24:52	-0.1860872	-78.495696	2843.7	
26	19/10/18	10:25:06	0.226			19/10/2018	10:25:02	-0.1860855	-78.4957014	2843.7	
27	19/10/18	10:25:16	0.271			19/10/2018	10:25:12	-0.186087	-78.4957084	2844.2	
28	19/10/18	10:25:26	0.213			19/10/2018	10:25:22	-0.1860918	-78.4957243	2844.2	
29	19/10/18	10:25:36	0.207			19/10/2018	10:25:32	-0.1861017	-78.4957135	2844.2	
30	19/10/18	10:25:46	0.211			19/10/2018	10:25:42	-0.1861104	-78.495703	2844.2	
31	19/10/18	10:25:56	0.221			19/10/2018	10:25:52	-0.186118	-78.4956967	2844.2	
32	19/10/18	10:26:06	0.263			19/10/2018	10:26:02	-0.1861206	-78.4957059	2844.2	
33	19/10/18	10:26:16	0.247			19/10/2018	10:26:12	-0.1861373	-78.4957083	2844.2	
34	19/10/18	10:26:26	0.251			19/10/2018	10:26:22	-0.186157	-78.4957149	2844.2	
35	19/10/18	10:26:36	0.262			19/10/2018	10:26:32	-0.1861364	-78.4957397	2844.2	
36	19/10/18	10:26:46	0.219			19/10/2018	10:26:42	-0.1861348	-78.4957464	2844.2	
37	19/10/18	10:26:56	0.231			19/10/2018	10:26:52	-0.186126	-78.4957469	2844.2	
38	19/10/18	10:27:06	0.221			19/10/2018	10:27:02	-0.1861236	-78.4957553	2844.2	
39	19/10/18	10:27:16	0.214			19/10/2018	10:27:12	-0.1861043	-78.4957497	2844.2	
40	19/10/18	10:27:26	0.214			19/10/2018	10:27:22	-0.1861003	-78.4957336	2844.2	
41	19/10/18	10:27:36	0.234			19/10/2018	10:27:32	-0.186098	-78.4957299	2844.2	
42	19/10/18	10:27:46	0.226			19/10/2018	10:27:42	-0.1860963	-78.4957397	2844.2	
43	19/10/18	10:27:56	0.208			19/10/2018	10:27:52	-0.1860792	-78.4957326	2844.2	
44	19/10/18	10:28:06	0.296			19/10/2018	10:28:02	-0.186072	-78.495734	2844.2	
45	19/10/18	10:28:16	0.229			19/10/2018	10:28:12	-0.1860583	-78.4957288	2843.7	

Anexo 4. Datos provistos por la Secretaría del Ambiente (extracto de los datos)

	A	B	C	D
1		Belisario	Belisario	Belisario
2		DIR_VEC	PM2.5_ug	RAP_VEC
3	Date			
4	19-oct-2018 00:00	98.79	25.37	1.19
5	19-oct-2018 00:10	109.78	20	1.67
6	19-oct-2018 00:20	92.06	22	1.6
7	19-oct-2018 00:30	110.7	17.37	1.47
8	19-oct-2018 00:40	145.36	23.26	0.92
9	19-oct-2018 00:50	178.45	28.39	1.11
10	19-oct-2018 01:00	157.67	34.69	0.78
11	19-oct-2018 01:10	141.13	24.42	0.73
12	19-oct-2018 01:20	175.57	23.31	0.65
13	19-oct-2018 01:30	79.4	19.82	0.47
14	19-oct-2018 01:40	310.52	23.33	0.18
15	19-oct-2018 01:50	359.14	21.39	0.43
16	19-oct-2018 02:00	172.57	27.73	0.95
17	19-oct-2018 02:10	193.59	16.93	0.58
18	19-oct-2018 02:20	191.64	14.68	0.94
19	19-oct-2018 02:30	193	23.56	0.91
20	19-oct-2018 02:40	161.87	31.11	0.96
21	19-oct-2018 02:50	167.7	39.84	1.42
22	19-oct-2018 03:00	155.43	46.18	1.59
23	19-oct-2018 03:10	165.35	38.81	1.25
24	19-oct-2018 03:20	172.04	35.01	0.85
25	19-oct-2018 03:30	172.7	37.59	0.81
26	19-oct-2018 03:40	162.75	34.84	0.46
27	19-oct-2018 03:50	100.99	30.39	0.5
28	19-oct-2018 04:00	318.38	23.93	0.86
29	19-oct-2018 04:10	301.85	22.38	0.54
30	19-oct-2018 04:20	310.51	26.43	0.39
31	19-oct-2018 04:30	77.3	28.9	0.63
32	19-oct-2018 04:40	130.31	31.39	0.61
33	19-oct-2018 04:50	97.55	33.19	0.85
34	19-oct-2018 05:00	236.81	28.12	0.17
35	19-oct-2018 05:10	170.95	35.4	0.88
36	19-oct-2018 05:20	206.74	42.38	0.94
37	19-oct-2018 05:30	196.91	46.54	1.7
38	19-oct-2018 05:40	186.47	41.71	1.29
39	19-oct-2018 05:50	181.01	38.72	1.01
40	19-oct-2018 06:00	251.48	32.21	0.87
41	19-oct-2018 06:10	205.52	32.83	0.58
42	19-oct-2018 06:20	173.71	36.28	0.83
43	19-oct-2018 06:30	325.84	39.72	0.59
44	19-oct-2018 06:40	316	39.26	0.9
45	19-oct-2018 06:50	331.93	42.93	1.52
46	19-oct-2018 07:00	351.64	43.85	1.19
47	19-oct-2018 07:10	25.03	42.86	1.01
48	19-oct-2018 07:20	6.44	38.82	0.94
49	19-oct-2018 07:30	14.59	40.14	1.01
50	19-oct-2018 07:40	1.32	41.59	1.12
51	19-oct-2018 07:50	4.85	46.48	1.32
52	19-oct-2018 08:00	356.96	50.1	1.11
53	19-oct-2018 08:10	336.37	58.09	1.21

	A	B	C	D
1		Centro	Centro	Centro
2		DIR_VEC	PM2.5_ug	RAP_VEC
3	Date			
4	19-oct-2018 00:00	31.3360258	16.21	1.27567242
5	19-oct-2018 00:10	43.1381455	17.25	2.08358947
6	19-oct-2018 00:20	40.9107661	20.7	1.98099114
7	19-oct-2018 00:30	40.602985	23.69	2.31450413
8	19-oct-2018 00:40	36.744827	22.21	2.01202292
9	19-oct-2018 00:50	50.3385346	24.88	1.48175955
10	19-oct-2018 01:00	43.4938045	26.43	0.890498022
11	19-oct-2018 01:10	17.7855372	29.79	0.398643191
12	19-oct-2018 01:20	67.2850562	30.66	0.29634768
13	19-oct-2018 01:30	42.5277193	33.22	0.530502542
14	19-oct-2018 01:40	37.2343839	29.39	0.89134612
15	19-oct-2018 01:50	41.4978493	27.19	0.747344431
16	19-oct-2018 02:00	55.0627194	27.58	1.20621386
17	19-oct-2018 02:10	64.0756634	25.37	1.28412863
18	19-oct-2018 02:20	69.3617092	23.15	1.35185094
19	19-oct-2018 02:30	77.2806401	25.88	1.30875584
20	19-oct-2018 02:40	83.6382521	27.47	1.11541192
21	19-oct-2018 02:50	93.818946	30.15	0.784833142
22	19-oct-2018 03:00	77.4034011	32.69	1.0023983
23	19-oct-2018 03:10	72.2896713	32.02	0.940903627
24	19-oct-2018 03:20	127.784809	27.98	1.18385984
25	19-oct-2018 03:30	119.134111	24.13	1.0455709
26	19-oct-2018 03:40	113.960771	21.97	0.756339168
27	19-oct-2018 03:50	115.379123	22.63	0.607764
28	19-oct-2018 04:00	87.514603	27.84	0.582041546
29	19-oct-2018 04:10	129.244273	34.18	0.622169322
30	19-oct-2018 04:20	140.043001	33.94	0.917539514
31	19-oct-2018 04:30	62.0863945	32.47	0.602783401
32	19-oct-2018 04:40	21.4203423	27.1	1.25561963
33	19-oct-2018 04:50	67.7235313	24.69	1.00274259
34	19-oct-2018 05:00	79.3331546	26.45	1.39158075
35	19-oct-2018 05:10	84.7590563	28.3	1.13201334
36	19-oct-2018 05:20	91.8483385	32.56	0.944619536
37	19-oct-2018 05:30	103.360141	38.31	0.943340566
38	19-oct-2018 05:40	105.047041	44.75	1.18146985
39	19-oct-2018 05:50	89.2144232	48	1.30410849
40	19-oct-2018 06:00	122.075386	46.71	1.21753083
41	19-oct-2018 06:10	122.191894	44.02	0.977039655
42	19-oct-2018 06:20	132.66303	41.99	0.263496906
43	19-oct-2018 06:30	58.7713983	41.5	0.652571544
44	19-oct-2018 06:40	67.4711256	44.55	1.34424482
45	19-oct-2018 06:50	57.0006817	48.82	1.14604697
46	19-oct-2018 07:00	52.2452703	52.32	0.960432276
47	19-oct-2018 07:10	38.439968	58.18	1.06738933
48	19-oct-2018 07:20	23.10616	63.88	0.718535907
49	19-oct-2018 07:30	35.0133841	68.02	1.48470913
50	19-oct-2018 07:40	34.2165447	61.8	1.83275136
51	19-oct-2018 07:50	43.3152376	52.52	1.98679966
52	19-oct-2018 08:00	47.3630928	54.43	2.01758108
53	19-oct-2018 08:10	45.604689	57.41	1.33634403

Anexo 5. Concatenación de los datos del MicroDust, GPS y Secretaría (extracto de los datos)

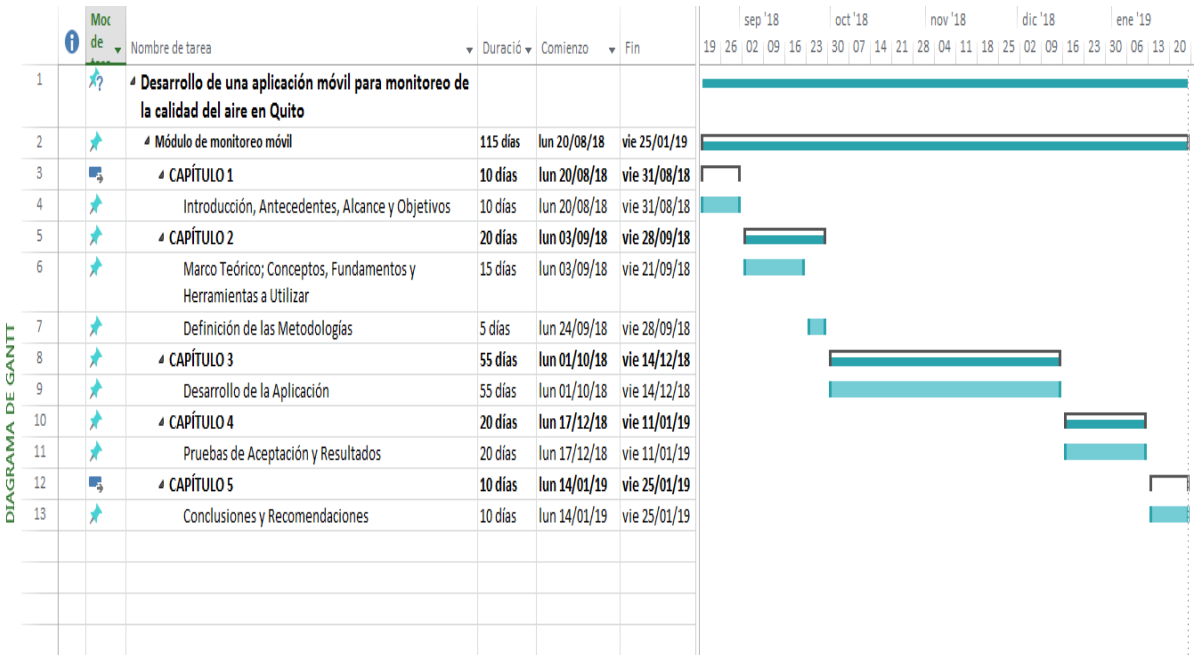
BELISARIO	PM 2.5	DIR. VIENTO	VEL. VIENTO
19-oct-2018 10:20	0.378	24.50	1.38
19-oct-2018 10:30	0.321	37.52	1.60
19-oct-2018 10:40	0.297	35.87	1.85
19-oct-2018 10:50	0.316	44.71	1.47
19-oct-2018 11:00	0.413	45.02	1.01
19-oct-2018 11:10	0.406	12.31	1.86
19-oct-2018 11:20	0.306	18.56	1.87
19-oct-2018 11:30	0.223	26.39	2.25
19-oct-2018 11:40	0.176	358.73	1.72
19-oct-2018 11:50	0.185	12.01	1.68

CENTRO	PM 2.5	DIR. VIENTO	VEL. VIENTO
19-oct-2018 10:20	0.697	37.03	3.07
19-oct-2018 10:30	0.622	40.38	2.89
19-oct-2018 10:40	0.584	33.53	2.69
19-oct-2018 10:50	0.480	37.83	2.73
19-oct-2018 11:00	0.390	37.06	2.49
19-oct-2018 11:10	0.428	45.76	2.66
19-oct-2018 11:20	0.459	47.48	3.35
19-oct-2018 11:30	0.498	41.00	4.25
19-oct-2018 11:40	0.496	52.93	3.49
19-oct-2018 11:50	0.447	42.57	3.44

Anexo 6. Construcción de la base de datos final para ejecución de regresiones (extracto de los datos)

	A	B	C	D	E
1	TIME	PM2.5 (PROMEDIO)	PM2.5 (ESPECIFICO)	VELOCIDAD (KM/H)	
2	10:21:56	0.25	0.24	17.32	
3	10:22:56	0.24	0.23	16.86	
4	10:23:36	0.28	0.31	17.15	
5	10:24:56	0.26	0.20	17.42	
6	10:25:56	0.22	0.22	17.39	
7	10:26:56	0.25	0.23	16.74	
8	10:27:56	0.22	0.21	16.81	
9	10:28:56	0.22	0.20	17.05	
10	10:29:56	0.23	0.24	16.91	
11	10:30:56	0.22	0.21	16.35	
12	10:31:56	0.23	0.23	16.12	
13	10:32:56	0.21	0.21	16.42	
14	10:33:56	0.22	0.22	16.93	
15	10:34:56	0.24	0.28	17.2	
16	10:35:56	0.24	0.26	17.12	
17	10:36:56	0.21	0.18	17.08	
18	10:37:56	0.19	0.18	16.62	
19	10:38:56	0.18	0.18	16.34	
20	10:39:56	0.19	0.20	16.04	
21	10:40:46	0.20	0.22	15.74	
22	10:41:56	0.23	0.18	15.52	
23	10:42:56	0.19	0.19	15.84	
24	10:43:56	0.22	0.22	16.11	
25	10:44:56	0.18	0.21	16.28	
26	10:45:56	0.23	0.21	16.02	
27	10:46:56	0.43	0.20	15.94	
28	10:47:56	0.19	0.20	15.61	
29	10:48:56	0.19	0.22	15.32	
30	10:49:56	0.20	0.26	15.06	
31	10:50:56	0.21	0.23	14.96	
32	10:51:56	0.25	0.29	15.22	
33	10:52:16	0.22	0.20	15.37	
34	10:53:46	0.21	0.21	15.53	
35	10:54:56	0.15	0.15	16.13	
36	10:55:56	0.15	0.16	16.02	
37	10:56:56	0.16	0.17	16.25	
38	10:57:56	0.15	0.15	16.58	
39	10:58:56	0.20	0.17	16.67	
40	10:59:56	0.17	0.25	16.36	
41	11:00:56	0.18	0.16	16.48	
42	11:01:56	0.14	0.16	16.62	
43	11:02:56	0.13	0.12	16.34	
44	11:03:56	0.12	0.11	16.04	
45	11:04:56	0.13	0.12	15.74	

Anexo 7. Cronograma de trabajo



Anexo 8. Abreviaturas

DMQ Distrito Metropolitano de Quito

OMS Organización Mundial de la Salud

IDW Inverse Distance Weighting

REMMAQ Red Metropolitana de Monitoreo Atmosférico de Quito

PM2.5 Material Particulado 2.5

HR Humedad Relativa

USEPA Agencia de Protección Ambiental de los Estados Unidos

Zn Zinc

Ni Níquel

V Vanadio

Ar Arsénico

Pb Plomo

CO2 Dióxido de Carbono

SO2 Dióxido de Azufre

O3 Ozono

NO2 Dióxido Nitroso

RMSE Error de la media cuadrática de la raíz

MSE Error cuadrado promedio por predicción

GPS Sistema de Posicionamiento Global

S.O Sistema Operativo

ART Android Runtime

APIs Interfaz de programación de aplicaciones

ML Machine Learning

CRM Sistemas de gestión de la relación con el cliente

BI Inteligencia de Negocios

ERS Requisitos de software