
# DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

## CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

**TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y TELECOMUNICACIONES**

**TEMA: "DESARROLLO DE UNA ARQUITECTURA CROSS-PLATFORM PARA EL ANÁLISIS DE TRÁFICO VEHICULAR DE UNA SMART CITY CON HERRAMIENTAS DE MACHINE LEARNING"**

**AUTORAS: CEVALLOS ZAPATA, PRISCILA ESTEFANÍA**
**LOMAS PROAÑO, EVELYN VANESSA**

**DIRECTOR: ING. ALULEMA FLORES, DARWIN OMAR MSc.**

**SANGOLQUÍ**

**2020**

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

## DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

## CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

## CERTIFICACIÓN

Certifico que el presente trabajo de titulación, **"DESARROLLO DE UNA ARQUITECTURA CROSS-PLATFORM PARA EL ANÁLISIS DE TRÁFICO VEHICULAR DE UNA SMART CITY CON HERRAMIENTAS DE MACHINE LEARNING"**, realizado por las señoras **Cevallos Zapata, Priscila Estefanía** y **Lomas Proaño, Evelyn Vanessa** el mismo que ha sido revisado en su totalidad y analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas – ESPE, razón por la cual me permito acreditarlo y autorizar para que lo sustente públicamente.

**Sangolquí, 2019**

Ing. Darwin Omar Alulema Flores MSc.
C. C.: 1002493334
**DIRECTOR**

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

## DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

## CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

## AUTORÍA DE RESPONSABILIDAD

Nosotras, **Cevallos Zapata, Priscila Estefanía** y **Lomas Proaño, Evelyn Vanessa**, declaramos que el contenido, ideas y criterios del trabajo de titulación, **"DESARROLLO DE UNA ARQUITECTURA CROSS-PLATFORM PARA EL ANÁLISIS DE TRÁFICO VEHICULAR DE UNA SMART CITY CON HERRAMIENTAS DE MACHINE LEARNING"**, es de nuestra autoría y responsabilidad, cumpliendo con los métodos de investigación existentes, requisitos teóricos, científicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

**Sangolquí, 17 de enero del 2020**

Priscila Estefanía Cevallos Zapata
C. C.: 171824471-6

Evelyn Vanessa Lomas Proaño
C. C.: 1003559257

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

## DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICIONES

## CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

## AUTORIZACIÓN

Nosotras, **Cevallos Zapata, Priscila Estefanía** y **Lomas Proaño, Evelyn Vanessa,** autorizamos a la Universidad de las Fuerzas Armadas – ESPE publicar el presente trabajo de titulación: **"DESARROLLO DE UNA ARQUITECTURA CROSS-PLATFORM PARA EL ANÁLISIS DE TRÁFICO VEHICULAR DE UNA SMART CITY CON HERRAMIENTAS DE MACHINE LEARNING"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

**Sangolquí, 17 de enero del 2020**

Priscila Estefanía Cevallos Zapata
C. C.: 171824471-6

Evelyn Vanessa Lomas Proaño
C. C.: 1003559257

## DEDICATORIA

*Dedico este trabajo a Dios porque gracias a Él existo y me permite vivir esta maravillosa aventura llamada vida. A mi abuelita Marinita que estoy segura que desde el cielo me manda sus bendiciones como siempre me lo prometió. A mis padres quien ha sido mi ejemplo de superación y son incondicionalmente siempre mi apoyo.*

*A mi esposo e hijos que son mi motivación e inspiración para ser una mejor persona y a quienes entrego mi mayor esfuerzo. Por último, pero no menos importante mi hermana y mi sobrino quienes también son un pilar importante en mi vida. A todos, los amo infinitamente.*

*Priscila Cevallos.*

# DEDICATORIA

*"Sé fuerte y valiente. No tengas miedo ni te desanimes porque el Señor tu Dios estará contigo donde quiera que vayas"* Josué 1:9.

*Este trabajo investigativo lo dedico principalmente a Dios, por la inspiración, fortaleza y guía para la culminación de esta anhelada meta profesional.*

*A mis padres Román y Liliana, que han sido pilar fundamental en este camino educativo que con su amor y paciencia me han llevado a cumplir mis propósitos. Mis hermanas Fer y Gaby por su apoyo incondicional en todas las áreas de mi vida.*

*Una dedicatoria especial a mis pequeños hijos, Alejito e Isabelita que me han mostrado el poder del amor, y lo sencillo y maravilloso de la felicidad.*

*Evelyn Vanessa Lomas Proaño.*

## AGRADECIMIENTO

Agradezco de todo corazón a mis papis Luisa y Patricio y mi ñaña Nathy quienes fueron testigos de todos mis éxitos y fracasos en mi formación estudiantil, que aguantaron mis arrebatos, altos y bajos, momentos en los que ni yo me entendía, pero ustedes siempre supieron cómo darme ánimos y apoyarme en todo momento.

Mi esposo Esteban, gracias mi amor por confiar en mí, que a pesar que culminar mi carrera era incierto tu siempre apoyaste este sueño que con el pasar el tiempo se convirtió en nuestro. Este sueño que mientras formábamos nuestra familia se convirtió en un reto, a ratos inalcanzable.

Gracias a mis hermosos hijos, mi princesa Alejandra y mi príncipe Martín quienes con su carita llena de ilusión y amor me motivan a superarme para darles todo lo mejor. Mis chiquitos y no puede faltar mi sobrino Nicky que están ahí para alegrar mis días en los momentos más difíciles.

También a mis suegros Isabel e Ivan quienes se han convertido en un apoyo importante estos años para que pueda culminar mi carrera, que sin importar cuan ocupados estén siempre me tienden una mano.

Ing. Darwin Alulema gracias por compartirnos su conocimiento, guiarnos y brindarnos el apoyo necesario para culminar este proyecto. A mi amiga Eve con quien realicé este proyecto, gracias por todo lo que hemos logrado juntas, por los años compartidos y porque sin importar cualquier pronóstico estamos cumpliendo este sueño juntas.

A todas las personas que están o estuvieron para mí y también quienes han formado parte de mi formación académica, gracias infinitas.

Priscila Cevallos

## AGRADECIMIENTO

Quiero empezar expresando mi agradecimiento a Dios por su profundo amor y por todo el proceso por el cual me ha llevado para la culminación de este trabajo final en donde ha moldeado mi carácter, mi paciencia, dominio propio y mi corazón, valores que serán puestos al servicio en mi vida profesional.

Agradezco a mis padres, Román y Liliana por su cariño, por su paciencia, apoyo incondicional y guía en cada una de las áreas de mi vida, enseñanzas y disciplina que me han llevado a terminar mi carrera educativa.

Agradezco a mis hermanas Fer y Gaby por sus consejos, apoyo, cariño y ánimo cuando las fuerzas decaen, por confiar, creer y apoyarme en mis metas.

A todos los profesores que tuve la oportunidad de conocer en tan prestigiosa universidad fueron un ejemplo de profesionalismo, rectitud, respeto, responsabilidad, integridad y perseverancia agradezco mucho su labor de enseñanza, De manera especial al Ing. Darwin Alulema por impartirnos su conocimiento, su apoyo y guiarnos hacia el termino exitoso de nuestra profesión.

Además, quiero agradecer a mis compañeros que tuve el agrado de compartir los años de universidad, en especial a mis amigos con los que conocí el valor de la amistad, Fer, Naty, Chalo, Carlitos, Dianita, Dani, David y de manera muy especial para mis amigos Pris y Willy con los cuales siento un gran agradecimiento por brindarme su tiempo, consejos, paciencia y su conocimiento para el avance exitoso del proyecto.

Evelyn Lomas

# TABLA DE CONTENIDOS

# ÍNDÍCE DE TABLAS

# ÍNDICE DE FIGURAS

## RESUMEN

El uso de internet en la actualidad ha crecido a pasos agigantados tanto en teléfonos móviles, electrodomésticos, televisores, computadores, por lo tanto, el enlace entre objeto y persona es un enlace muy cotidiano. Al ser el internet una red que se encuentra al alcance de la mano tiene servicios como cloud computing, por lo tanto, se ha tenido un avance significativo en el área de internet de las cosas (IoT), que a su vez viene de la mano con la tecnología de machine learning para el manejo de predicciones. De esta manera en el presente proyecto de titulación se presenta una arquitectura cross-platform para el análisis de tráfico vehicular con herramientas de machine learnnig. Los datos usados para el entrenamiento del sistema fueron tomados en las calles más concurrentes de la ciudad de Quito. Para el diseño de la arquitectura se empleó técnicas MDA (Model-Driven Architecture) y se usó computación en la nube las plataformas usadas en el presente proyecto es Amazon Web Service (AWS) y Google Cloud (GC). Para validar la propuesta se realizó pruebas de funcionamiento, carga, rendimiento y cualitativas de los aplicativos donde se presenta la información final evidenciando que los resultados de predicción de tráfico vehicular son muy cercanos a la realidad en cuanto más datos se tiene y por un tiempo más extenso aumenta el nivel de confianza.

- LENGUAJE ESPECIFICO DE DOMINIO (DSL)
- APRENDIZAJE DE MÁQUINA
- COMPUTACIÓN EN LA NUBE
- TRÁFICO VEHICULAR

## ABSTRACT

The use of the internet today has grown by leaps and bounds in both mobile phones, appliances, televisions, computers, therefore, the link between object and person is a very daily link. As the internet is a network that is within reach of the hand it has services such as cloud computing, therefore, there has been a significant advance in the area of internet of things (IoT), which in turn comes hand in hand with machine learning technology for prediction management. In this way, in the present titling project a cross-platform architecture is presented for the analysis of vehicular traffic with machine learnnig tools. The data used for the training of the system were taken in the most concurrent streets of the city of Quito. For the design of the architecture, MDA (Model-Driven Architecture) techniques were used and cloud computing used the platforms used in this project is Amazon Web Service (AWS) and Google Cloud (GC). To validate the proposal, performance, load, performance and qualitative tests of the applications were carried out, where the final information is presented, evidencing that the traffic prediction results are very close to reality as more data is available and for a longer period of time. Extensive increases the level of confidence.

- SPECIFIC DOMAIN LANGUAGE (DSL)
- MACHINE LEARNING
- CLOUD COMPUTING
- VEHICULAR TRAFFIC

# CAPÍTULO 1

# PLANTEAMIENTO DEL PROBLEMA DE INVESTIGACIÓN

## Antecedentes

El INEC (Instituto Nacional de Estadísticas y Censos) indica el crecimiento de la población en la provincia de Pichincha entre los años 2015 y 2020. En la Tabla 1 se observa las estadísticas mostradas en donde anualmente hay un aumento de habitantes, por lo cual la movilidad es una dificultad de las ciudades urbanas especialmente de la capital, Quito.

En la Figura 1 se muestra una serie histórica de vehículos matriculados en el período 2008 y 2017 en el país clasificados por provincias. Según INEC en el año 2017, la provincia de Pichincha registro el 22.9% del total de vehículos matriculados a nivel nacional.

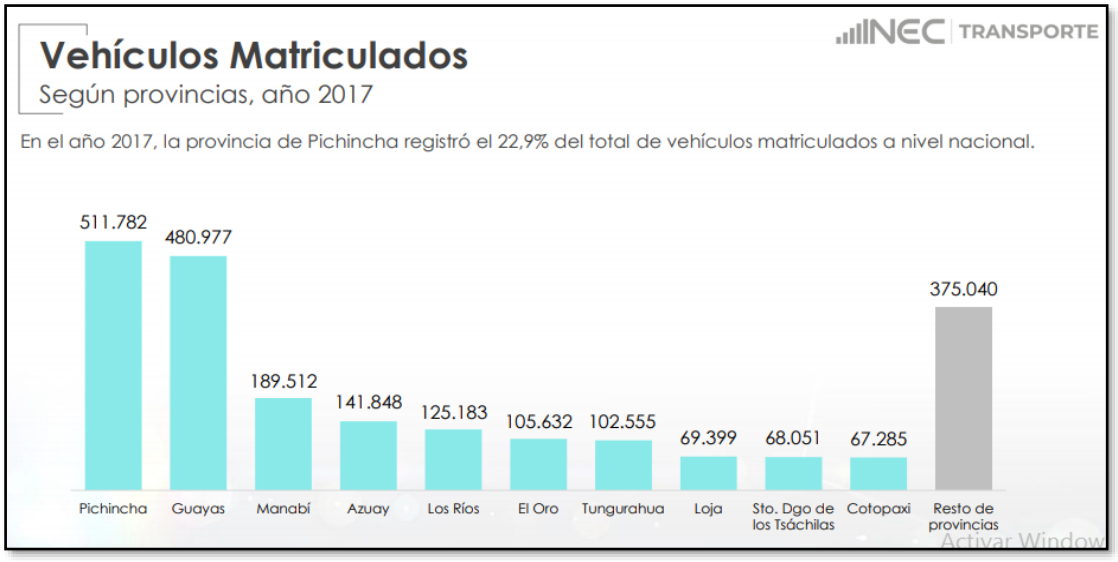



***Figura 1.*** Vehículos matriculados año 2017

Fuente: (INEC, 2018)

Tabla 1.

*Proyección de la población de Pichincha*

| PROYECCIÓN DE LA POBLACIÓN | | | | | | |
|---|---|---|---|---|---|---|
| PERÍODO 2010 - 2020 | | | | | | |
| POBLACIÓN TOTAL | | | | | | |
| **AÑOS CALENDARIO** | | | | | | |
| **PROVINCIAS** | 2,015 | 2,016 | 2,017 | 2,018 | 2,019 | 2,020 |
| **PICHINCHA** | 2,947,627 | 3,003,799 | 3,059,971 | 3,116,111 | 3,172,200 | 3,228,233 |

Fuente: (INEC, 2018)

"En el Distrito Metropolitano de Quito se han aplicado políticas públicas para una movilidad más eficaz tales como: pico y placa (desde el 2010), mejoramiento y ampliación de las vías". (Coyago et al., 2017)

Según (Municipio del DMQ, 2009) el 60% de las vías de la ciudad tenían una relación de volumen de tráfico y capacidad comprendida entre 0 y 0.8, permitiendo una velocidad promedio de desplazamiento de más de 50 km/h. Mientras tanto, el 25% de las vías tenía una relación volumen/capacidad de más de 1, es decir que estaban saturadas con una velocidad promedio de desplazamiento de 0-10 km/h. Para el 2015, dadas las actuales condiciones de crecimiento del parque automotor y de movilización, se prevé que 44% de las vías cuenten con una relación volumen/capacidad de 0-0.8 y que 38% de las vías tenga una relación volumen/capacidad de más de1. Este problema se agravaría aún más para el 2025, cuando se prevé que sólo 27% de las vías de la ciudad tengan una relación volumen/capacidad de 0-0.8 y 55% de las vías estén saturadas con una relación volumen/capacidad superior a 1.

El incremento del parque automotor especialmente en las zonas urbanas ha conllevado a problemas asociados con la congestión, como mayores tiempos de viaje y de emisión de contaminantes atmosféricos.

En la actualidad existen diversas tecnologías que son de gran ayuda para el crecimiento de una industria, ciudad, país. Por ejemplo, PSS (Sistemas de Sensores Participativos), son aplicaciones móviles que permiten compartir datos sobre el entorno de las personas en cualquier momento y lugar. Estos sistemas contribuyen con el proceso de la computación ubicua por lo tanto aporta información potencial para el desarrollo del comportamiento de tráfico vehicular de una ciudad. (Silva et al., 2013)

"Waze es un ejemplo de PSS en donde los usuarios forman una red de sensores con el dispositivo móvil a escala global "crowdsourcing"; fue creado en el 2006 en la actualidad con 115 millones de usuarios en diferentes partes del mundo" (WAZE Mobile, 2006), permite una mayor comprensión de la dinámica de la ciudad y los patrones de comportamiento urbano de sus habitantes, favoreciendo la toma de decisiones inteligentes con eso se evita la zonas peligrosas, rutas con menos tráfico, y se reduce la contaminación ambiental. Un paso fundamental es el análisis de los datos recopilados para lo cual en este proyecto se implementa herramientas de predicción inteligente.

Esta tesis finaliza mostrando la información en tiempo real y predicción de tráfico vehicular, mediante interfaces en TV Digital, Página Web, la arquitectura para el diseño se realizará en modelados con Arquitectura Basada en Modelos (*MDA)* y servicios de Transferencia de Estado Representacional (*REST)*.

**Motivación**

La movilidad es un eje importante para la vida cotidiana del ser humano y gracias a la tecnología con sus avances se han desarrollado diferentes herramientas y soluciones para que faciliten las actividades cotidianas. Existen varios tipos de medios de transporte que en la actualidad circulan

para facilitar la movilidad y las actividades de las personas. Sin embargo, estos medios de transporte pese a que han sido una solución, traen consigo inconvenientes en la circulación y tránsito, por infraestructuras viales no planificadas a futuro.

Un estudio presentado en 2018 por la consultora INRIX Global Traffic Scorecard, analizó datos de 1360 ciudades, repartidas en 38 países en los 5 continentes, y permite comparar el tráfico de la capital ecuatoriana con otras capitales latinoamericanas. Quito se encuentra en la posición 86 con 28 horas al año de atasco en el tráfico, Guayaquil en la posición 68 con 33 horas de atasco, Bogotá con 77 horas y Cuidad de México con 227 horas, siendo la ciudad con más tráfico dentro del análisis de ciudades latinoamericanas (INRIX, 2018a).

En el Distrito Metropolitano de Quito según el último registro del número de vehículos matriculados se ha evidenciado un crecimiento del 8,8% entre el 2016 y 2017 (INEC, 2018). Por lo que indicaría un incremento en problemas de movilidad generando congestión en horas pico. Como solución se han implementado restricciones de circulación en la ciudad, sin embargo, esta normativa no sería tan efectiva a largo plazo. Según las estadísticas del INEC la población se duplicará para el año 2050, lo que conlleva un crecimiento de parque automotor. Además, se comenta que el tráfico de Ecuador ocupa uno de los mejores puestos con referencia a otros países de la región. Por lo que es oportuno y necesario contar con iniciáticas flexibles y adaptables que permitan controlar los problemas de movilidad actuales y mitigar futuros.

En el mundo dichas iniciativas han sido enfocadas al escenario de las tecnologías de la información (Hafedh et al., 2012), que permiten en primer lugar generar datos de forma eficiente, por ejemplo, utilizando dispositivos móviles que automáticamente generen información relevante. Y en segundo lugar, que permita analizar los volúmenes de datos generados.

Por todo lo anteriormente expuesto, es importante iniciarse en la búsqueda de herramientas tecnologías al problema de movilidad, el proyecto de investigación aborda el tema del mejoramiento de la movilidad a través de una arquitectura que involucre el uso de las nuevas tendencias de programación de software y herramientas actuales, apoyada sobre los servicios disponibles de computación en la nube.

**Alcance**

El proyecto se enfoca en la construcción de una arquitectura dirigida por modelos (*Model-Driven Architecture*) para la creación de la solución de software. Además, el proyecto involucra la utilización de aprendizaje automático y computación en la nube. El diseño de la arquitectura pretende servir como base para el análisis de tráfico vehicular (datos previamente recopilados o adquiridos utilizando los servicios de la propia arquitectura en un periodo de alrededor de cuatro meses). El análisis ayudará al usuario (o servicios de transporte) para la toma de decisiones enfocadas en su mejor movilidad. El ambiente en el cual fue desarrollado el sistema es en la ciudad de Quito, en las siguientes calles: Shirys, Eloy Alfaro, Amazonas, 10 de Agosto, Patria, Cristóbal Colón, 12 de Octubre, 6 de Diciembre, Napo, y América.

En primera instancia, se investigó y recolectó información referente a las arquitecturas y desarrollos previos que involucran MDA. Además de tecnologías disponibles para la adquisición de datos, aprendizaje automático y su posibilidad de integración con diferentes servicios, lenguajes de programación y arquitecturas.

Posteriormente, se diseña los microservicios necesarios para recolectar datos y entrenar a la herramienta de predicción y tomar decisiones en función de los datos de tráfico vehicular. Se implementan algoritmos dirigidos a predecir el comportamiento de diferentes parámetros

contenidos en el conjunto de datos de tráfico. Es decir; se obtienen modelos en función de los registros de tráfico, este algoritmo se implementa utilizando aprendizaje automático. El servicio final entrega los resultados mediante servicios REST permitiendo una integración con la interfaz de usuario.

Finalmente, se diseña un conjunto de interfaces en diferentes dispositivos. Por ejemplo, se puede acceder a los servicios y mostrar la información de predicción de tráfico mediante aplicativos webs, de la misma forma mediante la IP del servidor acceder a la interfaz en un dispositivo móvil, y por último en televisión digital. Las interfaces finales tienen dos características, la recopilación de datos de tráfico vehicular y la presentación de información mediante una interfaz que solicita y entrega el análisis del tráfico.

## Objetivos

### General

Desarrollar una arquitectura *cross-platform* para el análisis de tráfico vehicular mediante, herramientas de *machine learning* y computación en la nube

### Específicos

- Investigar el estado del arte sobre *machine learning*, computación en la nube, sensores móviles y análisis de datos.
- Diseñar una arquitectura dirigida por modelos (MDA) para el despliegue de un sistema *cross-platform* de predicción de tráfico vehicular.
- Generar una base de datos de tráfico vehicular en la nube para el entrenamiento del sistema de predicción de tráfico.

- Realizar el entrenamiento del sistema de predicción de tráfico vehicular mediante *machine learning*.
- Aplicar las definiciones de -arquitectura *REST*- para lograr la interoperabilidad de los servicios de la arquitectura *cross-platform* propuesta.
- Desarrollar las aplicaciones web, móvil y para TV Digital que permita la adquisición y presentación de información.

## Estado del Arte

La búsqueda del estado del arte se desarrolla enfocado en el método de Mapeo Sistemático de la Literatura (*Systematic Mapping Study, SMS*) y una Revisión Sistemática de la Literatura (*Systematic Literature Review, SLR*) (Márquez, 2018), con el objetivo de desarrollar la investigación en aspectos más desarrollados para tener un mayor enfoque en el tema.

Tabla 2.

*Preguntas para los Estudios de Mapas Sistemáticos (SMS)*

| Preguntas de investigación | Motivación |
|---|---|
| **SMSP1: ¿Cuáles son las medidas que usa actualmente la ciudad de Quito para contrarrestar el tráfico vehicular?** | Identificar una alternativa tecnológica para solventar problemas de tráfico en la ciudad de Quito. |
| **SMSP2: ¿Cuáles son los métodos de predicción de tráfico vehicular que actualmente se usan en Quito?** | Conocer en base a qué argumentos se tomaron las medidas para contrarrestar el tráfico a futuro. |

**CONTINÚA ⟶**

| | |
|---|---|
| **SMSP3: ¿En qué plataformas actualmente accede el usuario a la información de tráfico vehicular en Quito?** | Identificar de qué manera el usuario se informa de las estrategias para descongestionar la ciudad. |

En la Tabla 2 se definen preguntas de investigación para contribuir con la investigación de metodología SMS, las mismas que nos ayudan a visualizar el entorno actual en el cual se encuentra el campo de la investigación en desarrollo mediante la búsqueda de información en libros, revistas y artículos. En la Tabla 3 se detallan preguntas con la metodología SLR con el objetivo de conocer el panorama de la investigación en aspectos más técnicos.

Tabla 3.

*Preguntas para la Revisión Sistemática de la Literatura (SLR)*

| Preguntas de investigación | Motivación |
|---|---|
| **SLR1: ¿Qué técnicas de ML se usan para predicción de tráfico vehicular?** | Conocer cuáles son los modelos que en la actualidad se usan para predicción de series temporales en el caso de tráfico vehicular. |
| **SLR2: ¿Cómo gestionar el almacenamiento para un big data?** | Indagar y discernir el mejor método de almacenamiento para un *dataset* extenso. |
| **SLR3: ¿Qué plataformas tecnológicas se usa para difundir la información de tráfico vehicular y fácil accesos para los usuarios?** | Conocer el estado actual de la ciudad en cuanto al uso de herramientas tecnológicas para mejorar el fluido de tráfico. |
| **SLR4: ¿Qué tecnologías se implementan en el desarrollo de sistemas y arquitecturas para el análisis de tráfico vehicular?** | Conocer las alternativas de arquitecturas y sistemas que alrededor de mundo se encuentren desarrollando. |

A partir de las preguntas precedentes se realiza la búsqueda de la información en artículos, libros, tesis, proyectos de investigación, que cubren trabajos desde el 2013 hasta el 2019 momento en el que se finaliza la búsqueda de información. Luego de una indagación exhaustiva de documentos con estudios verídicos e investigaciones anteriores nos lleva a clasificar y determinar temas referentes con mayor afín a la investigación presente, se ha encontrado 80 documentos relacionados de los cuales se ha excluido los menos relevantes como presentaciones de power point, prezzi, blogs de la web, video tutoriales y artículos que no tengan relación con lenguajes de aprendizaje automático. Los artículos incluidos son los que tienen información de fuentes confiables como el buscador *Google Scholar*, Dialnet, Scopus, Elsevier, revistas, conferencias y tesis de pregrado o maestrías. A continuación, se muestra algunas de las investigaciones que aportaron para responder las preguntas SMS y SLR.

Según (INRIX, 2018b), la ciudad de Quito se encuentra en el vigésimo sexto lugar en el ranking mundial que más problemas de congestión vehicular presenta además de ser la segunda ciudad en el país con problemas de movilidad esta investigación fue realizada en 200 ciudades de 38 países, a partir de análisis de big data sobre congestión vehicular.

En la información tomada de la investigación realizada por (Ana María Carvajal, 2019) se menciona algunas medidas que se ha tomado en la ciudad de Quito para ayudar con la congestión. El pico y placa saca entre el 20 y el 30% de vehículos privados de las vías, en horas pico. Julio Puga, director de la AMT, indica que hace 10 años había 300000 vehículos matriculados en Quito, pero el año pasado fueron 448000, aunque en la ciudad circulan 650000, añadiendo aquellos que llegan desde otras ciudades.

El consultor en temas de movilidad, Roberto Custode dice que los problemas de tránsito se derivan de un desarrollo desordenado y poco planificado. La EPMMOP informó que en los últimos tres años se han realizado obras como los intercambiadores de la avenida de Los Granados y de Carapungo, la prolongación de la avenida Simón Bolívar y de la Ecovía hacia el sur, la solución vial Charles Darwin, la calle Carapungo, la rehabilitación de la avenida Interoceánica, del corredor exclusivo del Trole y de la Ecovía.

Según Puga, con el inicio de operaciones del Metro de Quito, la carga vehicular se aliviará debido a que transportará a 1500 pasajeros por viaje y 400000 al día. Hoy, en las rutas troncalizadas (Ecovía, Trole y corredores) se movilizan por día 700000. Puga afirma que la AMT hace controles diarios para reducir la congestión, que incluyen 16 contraflujos y personal organizando el flujo vehicular en las vías.

La predicción tráfico vehicular obedece a patrones con características periódicas y con estacionalidad. De acuerdo a lo indicado por (Brownlee, 2018b) SARIMA (Media móvil integrada estacional automática autorregresiva) es un método que permite modelar estas características en un sistema.

En la última década existe una explosión de datos y un cambio de enfoque en el hardware de almacenamiento. En la investigación realizada por (Jose Cavanillas, 2015) indica que las tecnologías NoSQL se han diseñado teniendo en cuenta el objetivo de escalabilidad y presentan una amplia gama de soluciones basadas en modelos de datos, con tolerancia a fallas cuando aumentan el volumen de datos y su complejidad. Adicional en la actualidad se tiene como alternativa computación en la nube, el almacenamiento en la nube proporciona acceso flexible

desde múltiples ubicaciones y capacidad de escalamiento rápido y fácil, así como precios más baratos y un mejor soporte basado en economías de escala con una efectividad de costos.

Los Sistemas de Detección de tráfico (TMS) es un área activa en la investigación, de acuerdo a lo indicado en (Adrián Rodríguez, 2018) actualmente se tiene varios medios para la información al conductor tanto de transporte público como privado como gestión eficaz de semáforos inteligentes, transporte público con sensores y seguidos por aplicaciones móviles, modelos de supermanzanas, aplicaciones móviles para bicicleta compartida, carro compartido.

En las investigaciones estudiadas anteriormente se analiza la información existente para tomar como referencia en el desarrollo del proyecto presente respecto a machine learning, TMS, big data y contribuir con el desarrollo de la ciudad.

# CAPÍTULO 2

# FUNDAMENTO TEÓRICO Y CONCEPTUAL

## Smart Cities

Internet de las Cosas o IoT en un contexto urbano es de particular interés, ya que responde al fuerte impulso de muchos gobiernos nacionales para adoptar soluciones de TIC en la gestión de los asuntos públicos, realizando así la llamada Ciudad Inteligente. Si bien aún no existe una definición formal y ampliamente aceptada de "Ciudad inteligente", el objetivo final es hacer un mejor uso de los recursos públicos, aumentar la calidad de los servicios ofrecidos a los ciudadanos y reducir los costos operativos del público (Zanella et al., 2014).

## Microservicios

Los microservicios son un sistema de desarrollo de software que han alcanzado popularidad actualmente y proponen su propia arquitectura, ya que funciona con un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma. Además, que nos permite contar con una infraestructura IT flexible y moldeable, porque es posible modificar si se requiere a uno de los servicios sin afectar su arquitectura total (Chakray, 2019).

## Niveles de Abstracción

Se puede dividir a la Arquitecturas de software en tres niveles de abstracción bien diferenciados:

## Estilo Arquitectónico

Precisa un nivel general de la estructura del sistema y en base a esto va a comportarse. En "Software Architecture", se muestran los estilos arquitectónicos e incluyen la manera de determinar

los componentes y conectores de un sistema, los mismos que pueden ser utilizados. (Shaw Mary, 1996) En el libro se mencionan estilos arquitectónicos comunes, como:

- Filtros y tuberías (Pipes and Filters)
- Abstracción de datos y organización orientada a objeto (Data Abstraction and Object-Oriented)
- Estilo basado en eventos (Event-based)
- Sistemas en capas (Layered Systems)
- Table Driven Interpreters

**Patrón Arquitectónico**

Un patrón arquitectónico es quien permite definir la plantilla para construir el Software. Aquí se incluye a MVC, el mismo que puede ser enmarcado dentro del estilo arquitectónico orientado a objetos (la programación de este estilo arquitectónico se basa en la programación orientada a objetos).

**Patrón Arquitectónico Modelo Vista Controlador**

Modelo-Vista-Controlador (MVC) conocido como un patrón de diseño y también considerado un patrón de arquitectura de software. La diferencia es que el patrón de diseño tiene un alcance mucho mayor, pues define la arquitectura fundamental de la aplicación o sistema que estemos desarrollando. En la Figura 2 se muestra la interacción entre MVC.

- El componente central y quien representa el conocimiento es el modelo; además contiene y gestiona la lógica de la dependencia, los datos, el estado y reglas fundamentales de la aplicación. Los datos pueden ser almacenados en el modelo o en

una base de datos; si se almacena en una base de datos únicamente el modelo puede tener acceso a esta base de datos. (Bahit Eugenia, 2012).

- La Vista es una representación visual del modelo que el usuario puede ver en pantalla como, por ejemplo: las Interfaces Gráficas de Usuario (GUI), texto en una consola o terminal, los gráficos estadísticos, documentos, etc. La vista únicamente permite al usuario ver los datos contenidos en el Modelo, sin poder manipularlos o modificarlos.
- El tercer componente es el Controlador, que es el enlace o medio de comunicación entre el Modelo y la Vista. Mediante el controlador corre todo tipo de comunicación entre la Vista y Modelo, así evitando la comunicación directa.

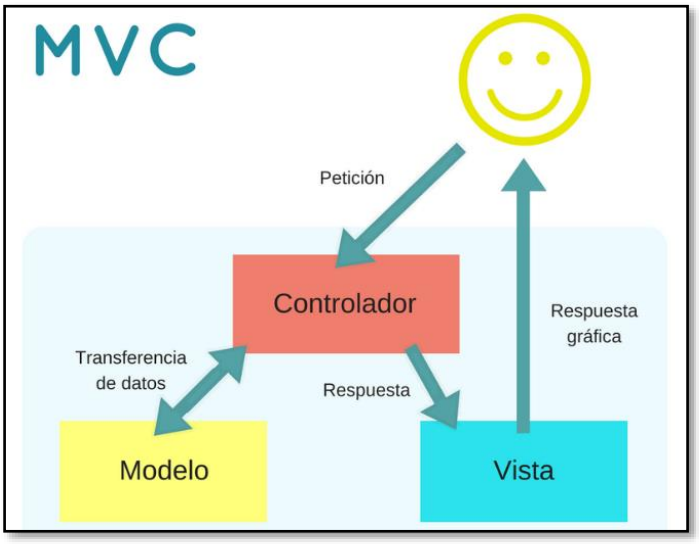



***Figura 2.*** Patrón Modelo, Vista, Controlador
Fuente: (García, 2017)

**Transferencia de Estado Representacional**

La Transferencia de Estado Representacional (*Representation State Transfer* - REST) refiere un estilo arquitectónico de sistemas en red. (Sun Bruce, 2011) REST está comprendida por una serie

de limitaciones y principios arquitectónicos. RESTful es el diseño que cumple con estos límites y principios arquitectónicos.

El principio más importante de REST de mayor importancia para las aplicaciones Web es que la interacción entre el cliente y el servidor. Las solicitudes que realiza el cliente al servidor deben contar con la información necesaria para que sea comprendida. En caso de que el servidor deba reiniciarse, el cliente no debe detectarlo durante sus solicitudes.

Tabla 4.

*Restricciones para API bajo REST*

| RESTRICCIONES | EXPLICACIÓN |
|---|---|
| **Cliente-servidor** | Esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor y existe indiferencia acerca de los datos que son enviados por el cliente. |
| **Sin estado** | Aquí decimos que cada petición que recibe el servidor debería ser independiente, es decir, no es necesario mantener sesiones. |
| **Cacheable** | Debe admitir un sistema de almacenamiento en caché. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso. |
| **Interfaz uniforme** | Define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme. |
| **Sistema de capas** | El servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad. |

Fuente: (Ordóñez, 2018)

Se puede comentar que en la actualidad las empresas utilizan API REST para crear servicios y se debe a que es un estándar lógico y eficiente para la creación de servicios web. Como, por ejemplo, sistemas de identificación de Facebook, autenticación para los servicios de Google utilizan arquitectura REST. Existen algunas restricciones para construir una API bajo REST las mismas que son detalladas en la siguiente Tabla 4.

Algunas características de una API REST son:

- Las operaciones que permiten manipular los recursos son: GET para consultar y leer, POST para crear, PUT para editar y DELETE para eliminar.
- API REST es las respuestas a las peticiones se hagan en XML o JSON, ya que es el lenguaje de intercambio de información más usado.

**Computación en la Nube**

De acuerdo a NIST *National Institute of Standars and Technology* (Mell & Grance, 2011) "La computación en la nube es un modelo para permitir el acceso a la red de forma ubicua, conveniente y bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se puede aprovisionar y lanzar rápidamente con un mínimo esfuerzo de gestión o interacción del proveedor de servicios.

Este modelo de nube se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de implementación."

**Características esenciales**

En la Tabla 5 se detalla las características esenciales que tiene computación en la nube.

Tabla 5.

*Características esenciales de computación en la nube*

| CARACTERÍSTICAS | DESCRIPCIÓN |
|---|---|
| **Self-service a demanda:** | El cliente accede a las capacidades de computo sin necesidad de intervención de recursos humanos. |
| **Amplio acceso a la red:** | Se puede acceder a las capacidades a través de la red desde plataformas disponibles. |
| **Pool de recursos:** | Los recursos del proveedor son variados tanto físicos como virtuales y también son dinámicos es decir se ajustan a las necesidades de los clientes. |
| **Elasticidad rápida:** | Las capacidades pueden aumentar o disminuir rápidamente, de acuerdo a la demanda del usuario. |
| **Servicio medido:** | Controlan y optimizan recursos automáticamente o puede ser supervisado por el usuario. |

## Modelo de servicios

En la Tabla 6 se muestra el modelo de servicios de computación en la nube.

Tabla 6.

*Modelo de servicios de computación en la nube*

| MODELO DE SERVICIO | DESCRIPCIÓN | EJEMPLOS |
|---|---|---|
| **El software como servicio (SaaS):** | El consumidor no gestiona la infraestructura de la nube, incluida la red, los servidores, los sistemas operativos, sino más bien los recursos como aplicaciones. | Gmail<br>Spotify<br>Dropbox<br>Netflix |
| **Plataforma como servicio (PaaS)** | Lugar donde se ejecutan las aplicaciones | Google app engine<br>Heraku |
| **Infraestructura como servicio (IaaS)** | El usuario tiene control sobre los sistemas operativos, el almacenamiento y las aplicaciones. | EC2-AWS<br>Google engine-google<br>Rackspace<br>VMWare |

## Machine Learning

Como se menciona en (Baştanlar & Özuysal, 2014), *machine learning* proporciona técnicas que pueden construir automáticamente un modelo computacional al procesar los datos disponibles y maximizar un criterio de rendimiento dependiente del problema.

Para que se aprenda un modelo puntual, los algoritmos de *machine learning* requieren grandes cantidades de datos para su entrenamiento. La utilidad de *machine learning* es posible debido a las facilidades de uso en herramientas de análisis y la introducción de *cloud computing* considerada una buena opción para la integración de aplicaciones.

## Características

*Machine learning* mediante datos obtenidos o almacenados convertirlos en información. “Existen dos variables que forman parte en el desarrollo de la ejecución de los algoritmos que son las características o atributos (aquellas que se van a medir) y la variable objeto (aquella que se trata de predecir)”. (Morejón et al., 2015).

Dado a que existen varios algoritmos, lo primero que se debe realizar, es escoger el algoritmo de aprendizaje que mejor se acople a nuestras necesidades y base de datos. Para que como segundo paso sea entrenar el algoritmo con el conjunto de datos.

Las técnicas de aprendizaje se clasifican en dos grupos de técnicas de aprendizaje supervisadas y no supervisadas. De acuerdo a (Peter Harrington, 2012) el primer grupo se divide en dos partes que es la clasificación y la regresión (predecir un valor numérico). El segundo grupo no contiene valores objetivos o etiquetas y entre sus tareas es describir un dato.

**BigML**

Es una herramienta en la nube, enfocada en la modelización de datos y el desarrollo de modelos de inteligencia artificial (aprendizaje automatizo) a partir de los mismos. Los usuarios puedan subir series de datos a un entorno seguro y trabajar en su análisis para el desarrollo de modelos predictivos sobre los mismos. La herramienta es operable mediante una interfaz web creada para ejecutar diferentes algoritmos de aprendizaje automático, además permite la preparación, filtrado y selección de los datos adecuados previo al entrenamiento.

"Una de las ventajas de usar BigML es la integración con diferentes lenguajes de programación como Python, PHP, JavaScript, Swift entre otros" (Dans E., 2012).

**Algoritmos de Aprendizaje**

En este trabajo basado a las características de los algoritmos existentes en *machine learning* comentado en la sección anterior. Se pretende utilizar algoritmos que pertenecen al grupo conocido como supervisados, ya que se verifico las particularidades en los datos y se cuenta con las características y la variable que se quiere encontrar. Las técnicas existentes de estos algoritmos de aprendizaje se describen a continuación.

**Redes Neuronales**

"Una red neuronal se conforma de una serie de unidades de proceso relacionadas mediante conexiones ponderadas. Es decir, que cuenta con un enfoque que trabaja con cualquier número de entradas para producir cualquier número de salidas, haciéndolo muy útil" (McCaffrey, 2012). Como se muestra en la Figura 3 un ejemplo gráfico de cómo son las redes neuronales.

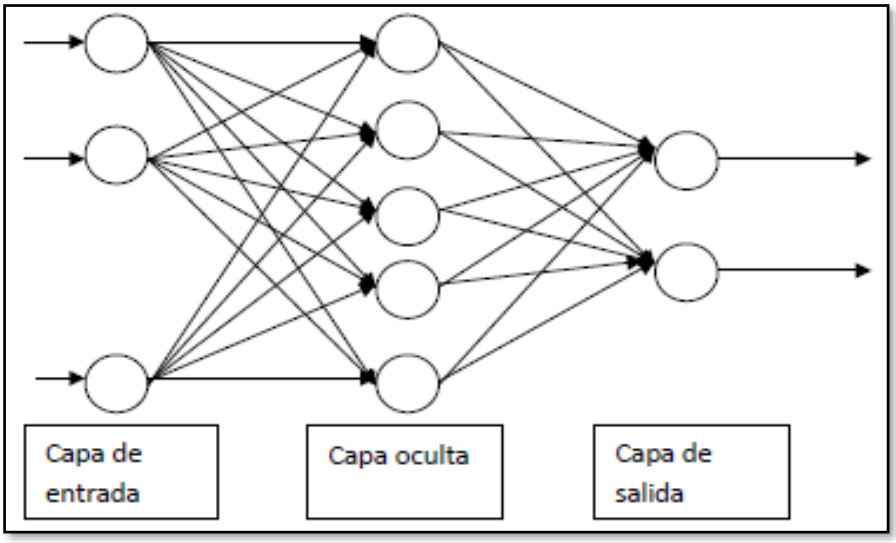



***Figura 3.*** Ejemplo redes neuronales
Fuente: (Morejón et al., 2015)

## Árboles de decisión

Esta técnica es sencilla y con ventajas por su bajo costo computacional, además de su fácil interpretación en los resultados. Presenta también desventajas tales como, inconvenientes cuando se tiene demasiadas ramificaciones y propenso al sobreajuste.

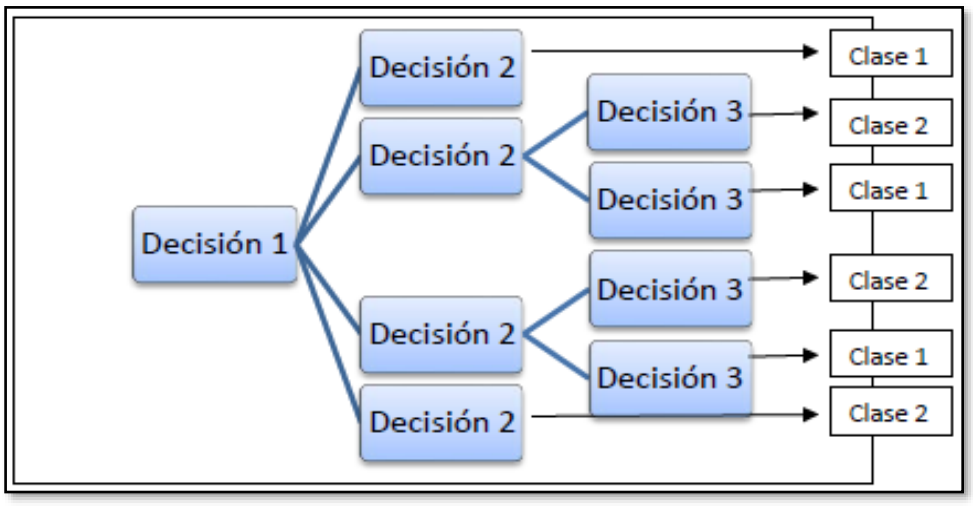



***Figura 4.*** Ejemplo de árbol de decisiones
Fuente: (Morejón et al., 2015)

Para que se obtengan los resultados esperados se requiere dividir el conjunto de datos en función a sus características. Se puede realizar una prueba para saber que esta clasificación se encuentra correcta; mediante el paso de los subconjuntos por la primera rama de decisión y consiguiente ubicarse correctamente, caso contrario realizar un ajuste en el paso anterior. En la Figura 4 se muestra un ejemplo de árbol de decisiones.

A continuación, se detallan los algoritmos a utilizar.

**Series temporales**

Las series de tiempo son datos estadísticos que se adquieren, observan o guardan en intervalos regulares de tiempo. En la Figura 5 se muestra los tipos de datos que se tiene en las cuales se encuentran las series estacionales.

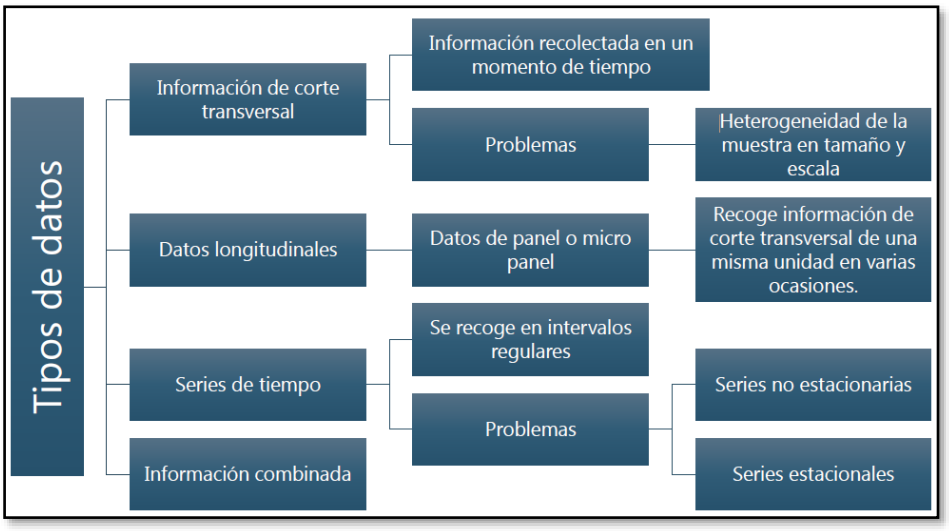



***Figura 5.*** Tipos de datos
Fuente: (Andrade, 2018)

Las series temporales según (Villavicencio, 2010) se puede clasificar en:

- **Estacionarias:** Cuando es estable a lo largo del tiempo, es decir, la media y varianza son constantes en el tiempo. Permanece constante en el tiempo.

- **No estacionarias**: La tendencia y/o variabilidad cambian en el tiempo. Los cambios en la media determinan una tendencia a crecer o decrecer a largo plazo, por lo que la serie no oscila en un valor constante.
- **Variables Endógenas:** Su valor es determinado o influenciado por una o más variables independientes.
- **Variables Exógenas:** Variable que no son afectadas por otras en el sistema.

En la Tabla 7 se muestra los algoritmos de series temporales:

Tabla 7.

*Series temporales de modelos autorregresivos*

| ALGORITMOS DE APRENDIZAJE | Modelo Autorregresivo Integrado de Media Móvil | Modelos Autorregresivos Integrado de Promedio Móviles Estacionales | Modelo Autorregresivo Integrado de Media Móvil con variables exógenas | Modelos Autorregresivos Integrado de Promedio Móviles Estacionales |
|---|---|---|---|---|
| **SIGLAS** | ARIMA | SARIMA | ARIMAX | SARIMAX |
| **Descripción** | Estacional, con variables endógenas | No estacional con variables endógenas | Estacional, con variables exógenas | No estacional con variables exógenas |

A continuación, se ingresa la información de la serie temporal que se utiliza en este proyecto.

**SARIMAX**

Las variables que conforman a la notación (1) son elementos de tendencia que requieren configuración; específicamente:

$$SARIMA(p,d,q)(P,D,Q)m \qquad (1)$$

**p:** orden de tendencia de autorregresión.

**d:** Orden de diferencia de tendencia.

**q:** Tendencia de orden de promedio móvil.

Además, cuenta con cuatro elementos que son estacionales y de igual forma deben configurarse, se definen a continuación:

**P:** orden autorregresivo estacional.

**D:** orden de diferencia estacional.

**Q:** Orden de media móvil estacional.

**m:** El número de pasos de tiempo para un solo período estacional.

Se lo puede utilizar en base a tres pasos (Brownlee, 2018a):

a. Definir modelo: Se puede crear una instancia de la clase SARIMAX proporcionando los datos de entrenamiento y una gran cantidad de parámetros de configuración del modelo.
b. Ajustar el modelo definido: Una vez que se crea el modelo, se puede ajustar a los datos de entrenamiento.
c. Predicción con el modelo de ajuste: Una vez en forma, el modelo se puede utilizar para hacer un pronóstico.

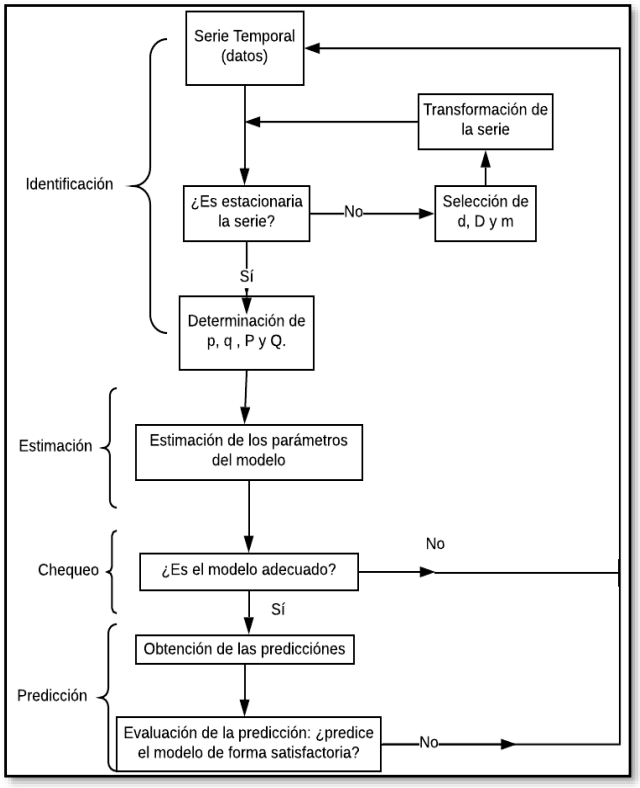



***Figura 6.*** Metodología Box- Jenkins
Fuente: (de la Fuente Fernández, 2017)

**Televisión Digital**

La Televisión Digital Terrestre está conformado por propagación de señales de audio, video y datos, utilizan señales digitalizadas y codificadas permitiendo innovar la cotidianidad del contenido en la televisión.

Características:

- Infraestructura sencilla y no costosa ya que por medio de la red terrestre se transmite sus señales.
- La señal digital presenta robustez frente al ruido, propagación multitrayecto permitiendo una mejor calidad de imagen.

**Ginga**

Ginga es así como recibe por nombre *middleware* del estándar brasileño ISDB-Tb. Posee un base de tecnologías bien constituidas e innovadoras brasileñas; y cuenta con dos subsistemas para el desarrollo de aplicaciones interactivas, los cuales son Ginga-J que usa aplicaciones que controlan y de secuencia de Java y Ginga-NCL utilizado para aplicaciones que utilizan código necesario para ofrecer una solución desarrolladas en NCL con lenguaje *scrpting* LUA.

**Ginga J**

Es un subsistema que utiliza como instrumento principal la máquina virtual de Java. Posee tres API – *Application Programming Interface:* verde, amarillo y azul como se observa en la Figura 7, que cumplen con las necesidades de la norma brasileña y la compatibilidad con la API del *middleware* GEM (Globally Executable MHP[3]).


***Figura 7.*** Componentes de Ginga-J
Fuente: (Torres, 2011)

**Ginga NCL**

Ginga NCL permite presentación de aplicaciones declarativas escritas en lenguaje NCL (*Nested Context Language*); contiene componentes propios y se muestran en la siguiente Figura 8 (Herrera, 2018)


***Figura 8.*** Componentes de Ginga-NCL
Fuente: (Torres, 2011)

a) **Lenguaje NCL: "**Este lenguaje se basa en el modelo conceptual de datos conocido como NCM-*Next Context Model* y se basa de los siguientes fundamentos" (UTFSM, 2013):
   - Nodos: representa partes de la información.
   - Enlaces: relaciones entre los nodos.
   - Para la definición de las relaciones los enlaces no presentan una sola forma-idea disponible.

b) **Lenguaje LUA: "**Este lenguaje cuenta con una programación estructurada y con un conjunto de instrucciones que enseñan de forma ordenada y consecutiva la tarea a realizar. Fue creado en 1993 e introducido como un lenguaje *script* extensible en su semántica" (UTFSM, 2013)
   - LUA se ejecuta en máquina virtual basada en C y en Linux, y siendo por esto multiplataforma permitiendo que el usuario lo utilice en otra plataforma.

Los lenguajes NCL y LUA se relacionan por *scripts*, llamados NCLua que cuentan con herramientas para la relación de eventos causa-efecto (módulo *event*) y configuración de pantalla (módulo *canvas*).

## Android

### Conceptos básicos de las aplicaciones en Android

*a. Servicios en el sistema operativo Android*

Un Servicio es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario. Otro componente de la aplicación puede iniciar un servicio y continuará ejecutándose en segundo plano, aunque el usuario cambie a otra aplicación. Además, un componente puede enlazarse con un servicio para interactuar con él e incluso realizar una comunicación entre diferentes servicios. Por ejemplo, un servicio puede manejar transacciones de red o reproducir música, monitorear la ubicación del dispositivo en segundo sin la necesidad de una interfaz.

*b. Notificaciones para servicios en primer plano*

Un servicio en primer plano es un tipo de servicio que se considera como algo de lo que el usuario está activamente consciente y, por ende, no es candidato a que el sistema lo cierre cuando tiene poca memoria. Un servicio en primer plano debe proporcionar una notificación para la barra de estado, lo que significa que la notificación no se puede descartar, salvo que el servicio se detenga o se quite del primer plano.

Por ejemplo, para la aplicación actual es necesario que se toman datos de localización continuamente, para ello se informa al usuario mediante una notificación en la barra de estado, cuando el usuario termina el servicio la notificación desaparece.

*c. Mensajes de Broadcast*

Un *Broadcast Receiver* es una clase de objeto *Listener* que atiende a eventos del sistema, esto es lanzados por el sistema operativo, o bien a eventos lanzados por aplicaciones. Además, el evento puede incluir información adicional dentro de un objeto *Intent*. Nos encontramos por tanto ante un mecanismo que nos permitirá integrar aplicaciones entre sí o con Android, pero también establecer un canal de comunicación interno entre componentes (actividades, fragmentos) de una misma aplicación.

*d. Permisos de aplicación*

Dentro del sistema operativo Android se manejan procedimientos con privilegios independientes, en el que cada aplicación se ejecuta con una identidad de sistema distinta (ID de usuario de Linux y ID de grupo). También se separan partes del sistema en identidades distintas. Así, Linux aísla las aplicaciones entre sí y del sistema operativo. Se ofrecen funciones de seguridad adicionales más precisas mediante un mecanismo de "permisos" que aplica restricciones en las operaciones específicas para que un proceso en particular puede realizar una tarea o acceder a elementos específicos de datos.

*e. Alarmas de sistema*

Debido a los procedimientos que el sistema operativo Android realiza para mejorar el consumo energético, regularmente configura al procesador en modo de reposo. Debido a esto para realizar tareas como envió de datos en segundo plano es necesario despertar el procesador cada cierto tiempo.

Debido a este inconveniente Android posee una clase llamada *AlarmManager* que permite crear alarmas que despiertan al sistema según se requiera.

*f. Base de Datos SQLite para Android*

*SQLite* es un sistema gestor de base de datos relacional (RDBMS). La mayoría de los sistemas de gestión de bases de datos como *Oracle*, *MySQL,* y *SQL* Server son procesos de servidor autónomos que se ejecutan independientemente, sin embargo, SQLite es que se considera una solución embebida que puede ejecutarse dentro de la cada aplicación de Android. Para el proyecto se utiliza esta base de datos para almacenar muestras de velocidad que no han podido ser guardadas en el servidor debido a la falta de una conexión de datos.

**Base de Datos**

**Definición:** Es un conjunto de datos almacenados en soporte informático sin copias y debe ser accesible por distintos usuarios y aplicaciones. Los datos no pueden aportar conocimiento para que brinden información hay que procesarlos y transformarlos.

**Sistema de gestión de bases de datos (SGBD)**

Es un software o conjunto de programas que permiten crear y proteger la base de datos. Y proporciona un entorno de seguridad y facilidad al momento de almacenar y obtener la información.

- Definir una base de datos consiste en determinar el tipo, estructura y restricciones de los datos.
- Construir una base de datos, es el proceso para almacenar en un medio controlado por SGBD una vez definida la base de datos.
- Manipular la base de datos, refiere a consultar, actualizar los datos de la base de datos y generar informes a partir de los datos reunidos.

**Almacenamiento de datos**

Sin embargo, el crecimiento de los datos en volúmenes en el mundo digital parece acelerar el avance de las muchas infraestructuras informáticas existentes. Las tecnologías de procesamiento de datos establecidas, por ejemplo, la base de datos y el almacén de datos, se están volviendo inadecuadas dada la cantidad de datos que el mundo está generando actualmente. La gran cantidad de datos debe analizarse de forma iterativa y sensible al tiempo. (Abhishek Sharma, Svetlozar Nestorov & Boris Jukić, 2015) Con la disponibilidad de tecnologías avanzadas de análisis de base de datos (por ejemplo, bases de datos NoSQL, BigQuery, MapReduce, Hadoop, WibiData y Skytree), se pueden obtener mejores conocimientos para mejorar las estrategias de negocios y el proceso de toma de decisiones en sectores críticos.

Desde ya hace algún tiempo, las bases de datos relacionales (SQL) son las más utilizadas en el mercado de la tecnología. Están compuestas por tablas con campos estructurados, sin ser un tipo de base de datos flexible, pero si cuenta con un gran soporte y grandes desarrollos en herramientas debido a su larga historia. SQL son muy reconocidas, sucediendo todo lo contrario para las bases de datos no relacionales (NoSQL) y, en definitiva, NoSQL no es un sustituto de SQL sino más bien otra alternativa que ofrece otras opciones de trabajo. Las bases de datos NoSQL no cuentan con una tabla fija permitiendo una escalabilidad alta entre sí, por lo que le vuelve flexible a varios tipos datos y no necesita muchos recursos para ejecutarse. (IMF BUSINESS SCHOOL, 2018).

Las tecnologías tradicionales tienen una capacidad de almacenamiento limitada, herramientas de administración rígidas y son caras. Carecen de escalabilidad, flexibilidad y rendimiento necesarios en el contexto de Big Data. De hecho, la gestión de Big Data requiere recursos significativos, nuevos métodos y tecnologías potentes.

El uso de datos en la toma de decisiones de negocios puede mejorar la competitividad mediante la reducción de costos, el aumento del valor agregado o cualquier otro parámetro que se puede medir contra los criterios de rendimiento existentes.

**Ingeniería Dirigida por Modelos – MDE**

La Ingeniería Dirigida por Modelos pretende reducir la dificultad que tienen los programas de tercera generación, tales como, ADA, C++, Java, etc. MDE es una ingeniería de desarrollo de software basada en modelos que emplea técnicas de reproducción automática de código para que el software final sea obtenido. Stuart Kent (Stuart Kent, 2002) definió por primera vez el término MDE como marco general para la especificación de los modelos y tareas de modelado para realizar un proyecto de desarrollo software de comienzo a fin.

"MDE indica que las soluciones centradas en el código (dominio de la solución) no dan respuesta a las necesidades actuales y plantea centrarse en buscar abstracciones que describan conceptos de dominios" (Sánchez, 2011).

En la Figura 9 se muestra la estructura de la Ingeniería Dirigida por modelos.

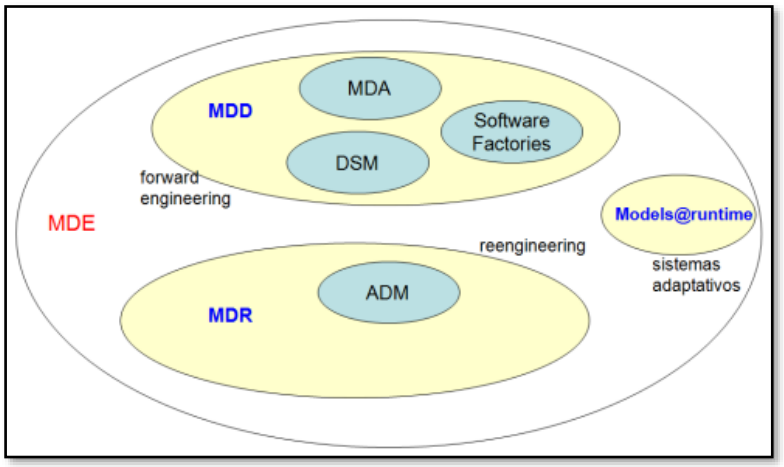



***Figura 9.*** Estructura Ingeniería Dirigida por Modelos – MDE
Fuente: (García Molina & García Rubio, 2013)

**Desarrollo de Software Basado en Modelos – MBD**

El desarrollo basado en modelos refiere al uso de notaciones de modelado específicas del dominio, que pueden analizarse para determinar el comportamiento deseado antes de que se construya un sistema digital. El énfasis en el desarrollo basado en modelos consiste en centrar el esfuerzo de ingeniería en las actividades de modelado, simulación y análisis del ciclo de vida temprano, y en automatizar las actividades de codificación y prueba finales del ciclo de vida del proceso.

El sistema de los modelos es una idealización del ámbito del problema y de su solución. Además, se focaliza en el mundo real: identificando, clasificando y abstrayendo los elementos que constituyen el problema y ordenándolos en una estructura formal. La abstracción muestra a la mente humana eventualmente la complejidad de varios casos; por lo que al ocultar lo irrelevante de un sistema complejo se convierte en algo manejable y comprensible.

"Actualmente casi todos los métodos de desarrollo de software utilizan modelos. El particular que varía entre métodos es la clase de modelos que deben elaborarse, la manera de tratarlos e incorporarlos" (Pons et al., 2010).

**Modelos y Metamodelos**

Son mecanismos de abstracción en desarrollo de software y esto es usada desde tiempos atrás no es nuevo, el ser humano por siglos ha utilizado modelos para realizar una representación resumida de elementos con distintos fines.

El metamodelo (modelo de modelos) es un modelo que define el lenguaje de modelado para formular un modelo. Y a su vez, un lenguaje de modelado es el conjunto de modelos estructurado por todos los modelos que ese lenguaje puede formar pertenecientes a este.

En la siguiente Figura 10 se puede observar la relación que guardan el modelo, metamodelo y lenguaje.

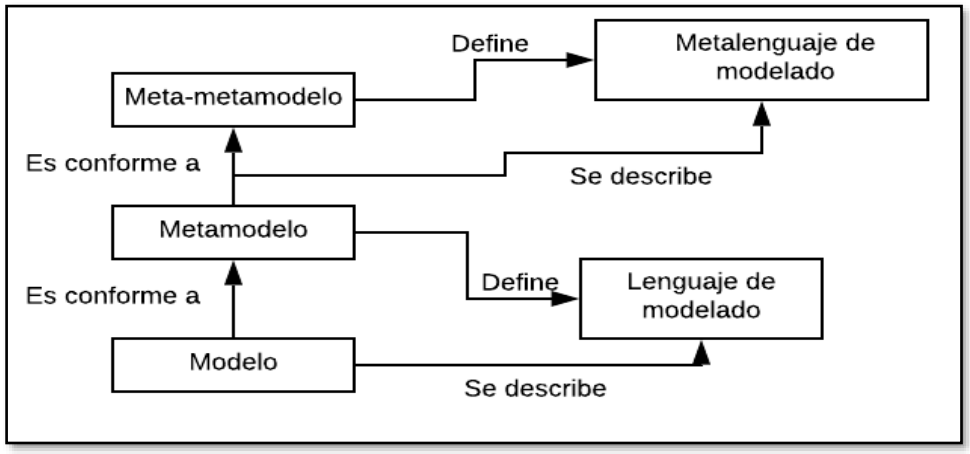



***Figura 10.*** Conexión entre modelos, metamodelos y lenguaje de modelado.
Fuente: (Sánchez, 2011)

La importancia en el desarrollo de software de los metamodelos, denotan en que permiten a los modelos especificarse formalmente. De esta forma los modelos puedan ser interpretados más fácilmente por una máquina, y que además se da paso para automatizar el desarrollo de software y la mejora de herramientas que lo manipulen.

**Desarrollo de Software Dirigido por Modelos – MDD**

MDD ofrece mejorar el proceso de construcción de software basándose en un desarrollo orientados por modelos y como base herramientas de alto rendimiento. Como su nombre lo indica se encuentra ‘dirigido’ a diferencia del ‘basado’ y hace referencia en que los modelos tengan un papel importante, central y dinámico tanto como los códigos fuente.

A continuación, se puede enumerar ciertos beneficios que muestra MDD: (Pons, Giandini, & Pérez, 2010).

- Incremento de productividad: MDD reduce costos de desarrollo de software mediante la generación automática del código a partir de modelos, y de esta forma aumenta la productividad de los desarrolladores.
- Adaptación a los cambios tecnológicos: La tecnología avanza constantemente por lo que MDD ayuda a solucionar este inconveniente por medio de una arquitectura fácil de mantener, adaptable y consistente para una migración de componentes hacia las nuevas tecnologías.
- Adaptación a los cambios en los requisitos: Es importante que el sistema sea adaptable a requerimientos y para MDD esto es una tarea sencilla; debido a que la parte de automatización ya se encuentra realizado.
- Consistencia
- Re-uso: MDD invierte y satisface según el re-uso de los modelos y transformaciones.
- Mejoras en la comunicación con los usuarios
- Mejoras en la comunicación entre los desarrolladores
- Captura de experiencia
- Modelos son productos de larga duración
- Posibilidad de demorar las decisiones tecnológicas

Existen dos propuestas conocidas y usadas para el desarrollo de software dirigido por modelos, se detallan a continuación:

MDA es una arquitectura dirigida por modelos y fue desarrollada por la OMG – Object Managment Group, y el lenguaje que aplica es basada en estándares de la OMG.

DSM es un modelado específico del dominio y utiliza lenguajes específicos del dominio DSLs *Domain Specific Language*.

**Arquitectura Dirigida por Modelos – MDA**

MDA es una iniciativa de Object Management Group (OMG) y es una arquitectura que brinda al usuario una serie de guías para formar especificaciones expresadas como modelos, según el proceso de MDD. También MDA no se limita al desarrollo de otros tipos de sistemas, además que está bien acoplado para el modelado de negocios o empresas.

MDA se clasifica por ser más amplio que el anteriormente expuesto. Está relacionado con múltiples estándares tales como Unified Modeling Language (UML), el Meta-Object Facility (MOF), XML Metadata Interchange (XMI), Enterprise Distributed Object Computing (EDOC), el Software Process Engineering Metamodel (SPEM) y el Common Warehouse Metamodel (CWM).

Consta de 3 etapas que conforman el proceso del desarrollo de software e MDA, se describe brevemente a continuación:

- Computation Independent Model – CIM: “Es el modelo independiente de la computación, surge en la fase inicial del proceso de desarrollo comprendiendo la modelación del negocio en su totalidad.” (Jacho, Martínez, Borrero & Rodríguez, R. M., 2015).
- Platform – Independent Model – PIM: Hace referencia a todos los modelos que definen una solución de software que no tienen información de la plataforma en la cual será implementada la solución.

- Platform Specific Model - PSM: “Modelo de un sistema resultado de refinar un modelo PIM para adaptarlo a los servicios y mecanismos ofrecidos por una plataforma concreta”. (Vergara et al., 2007).
- Code Model – CM: Es el paso final o último nivel de la arquitectura de MDA. CM es el código real generado desde los PSM por medio de la información y es un nivel que no denota alta importancia, ya que PSM es el paso más cercano a la plataforma.

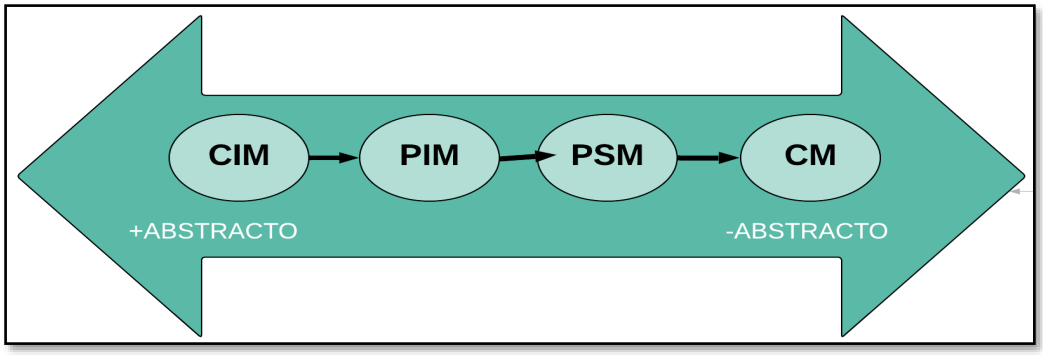



***Figura 11.*** Niveles de Abstracción en MDA

En la Figura 11 se puede observar el proceso que MDA propone para el desarrollo de un sistema de una plataforma. Empezando por la construcción de un CIM para después transformarse en un PIM y para luego PIM se transforma en uno a varios PSM. Finalmente, cada uno de los PSM genera por medio de transformaciones el código de forma automática. Las transformaciones que se establecen idóneamente deben ser automatizadas, pero en ocasiones se recibe instrucciones adicionales para algunas decisiones de diseño.

**Proceso de desarrollo MDA:** En la Figura 12 presenta el proceso de desarrollo de software de MDA. El primer paso en este proceso es la toma de requisitos en forma de texto, el siguiente es sacar el modelo independiente de la plataforma (PIM) expresado en un lenguaje estándar como

UML o MOF. Posterior a esto, mediante transformaciones y herramientas a partir del PIM inicial se genera uno o más modelos PSM dependiendo de las plataformas definidas. Por último, a partir del modelo PSM se obtiene el código expresado en la plataforma específica que haya elegido en el PSM.

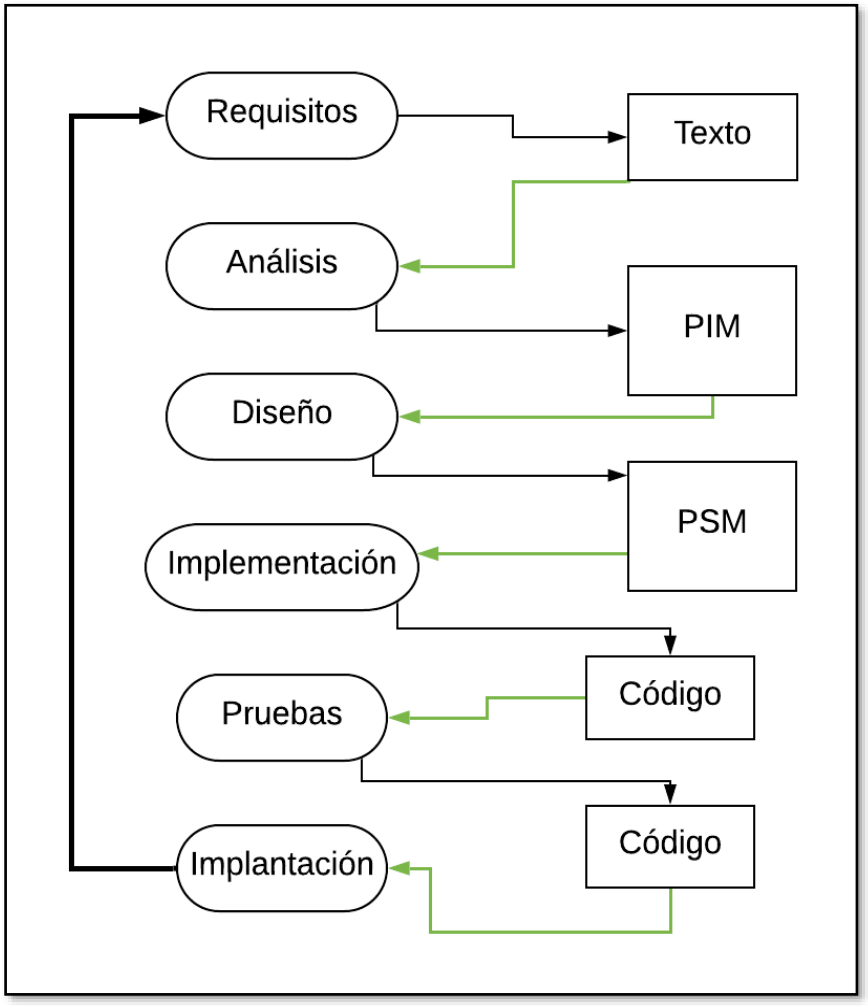



***Figura 12.*** Proceso de Desarrollo MDA

**Estándares en MDE**

En base a lo que indica la OMG se propone el empleo de diferentes estándares, para poner en práctica MDD los estándares se encuentran en la Tabla 8.

Tabla 8.

*Estándares de MDE*

| ESTÁNDARES | DEFINICIÓN |
|---|---|
| **Meta Object Facility – MOF** | Las generalidades de MOF indican una arquitectura de 4 niveles, en la que cada uno de estos elementos del 'nivel x' es instancia del elemento del nivel superior o 'nivel x+1'. Estos niveles se denominan M3, M2, M1 y M0. |
| **Nivel M3** | Este nivel es el más abstracto, el que define un metamodelo y provee un lenguaje indeterminado (MOF) para el metamodelo en el nivel M2. El lenguaje MOF se puede considerar un subconjunto del diagrama de clases UML y presenta la versión actual dos adaptaciones Essential MOF – EMOF y Complete MOF – CMOF. |
| **Nivel M2** | En esta etapa se definen metamodelos, es decir, la estructura y semántica de los modelos en el nivel M1. Aquí es donde figuran conceptos como clases, atributos, relaciones de asociación, etc. |
| **Nivel M1** | En este nivel se definen modelos de sistemas acorde a sus respectivos metamodelos generados en el nivel M2. Por ejemplo, pueden ser entidades como 'persona', atributos como 'nombre' y relaciones entre entidades. |
| **Nivel M0** | Este nivel es de información y ejecución. Define un modelo concreto, es decir un producto software. Los elementos son datos o instancias, por ejemplo 'Pepe' que es una instancia concreta de una clase 'persona'. |

También existen relaciones con otros estándares en la arquitectura MOF, tales como UML, OCL, XMI, QVT, se describen a continuación en la Tabla 9.

Tabla 9.

*Estándares en la arquitectura MOF*

| *ESTÁNDARES MOF* | DEFINICIÓN |
|---|---|
| **Unified Modeling Language - UML** | Es un lenguaje orientado hacia lo visual de modelado estándar para visualizar, especificar y documentar sistemas de software. Este tipo de lenguaje presenta objetivos tales como, ser apto para soportar ordenes de lenguajes de programación independientes y la interoperabilidad entre herramientas. |
| **Object Constraint Language - OCL** | Es un tipo de lenguaje formal y tipado, utilizado para definir restricciones sobre modelos UML. Estas restricciones realizan consultas sobre los objetos definidos en los modelos UML. |
| **XML Metadata Interchange - XMI** | Es un estándar de OMG que define reglas, las mismas que permite expresar en XML cualquier modelo o metamodelo que se formalizó en MOF. XMI define como se debe formar los documentos y esquemas XML. |
| **Query Views Transformations – QVT** | Nos permite consultar modelos, la creación de vistas de modelos y definición de transformaciones de modelos. Además, define el marco de MDA. |

### Modelos Específicos de Dominio – DSM

Son una aproximación al desarrollo de software basados en modelos y utiliza modelos específicos de dominio como elementos principales para el proceso. Los DSM son aplicables a varias plataformas con el mismo dominio y eleva el nivel de abstracción especificando la solución mediante conceptos del dominio.

**Lenguaje Específico de Dominio – DSL:** "Es un lenguaje de programación o de especificación dedicado a un dominio en particular. Son lenguajes que ofrecen un conjunto reducido de notaciones más cercanas al dominio del problema que al dominio de la implementación y ofrecen un alto nivel de abstracción para el desarrollo de aplicaciones" (Sánchez, 2011).

Se fijan ciertas ventajas y desventajas en la utilización de DSL como se muestra en la Tabla 10, las mismas que se describen a continuación:

Tabla 10.

*Ventajas y desventajas de DSL*

| | |
|---|---|
| **VENTAJAS** | Eleva el nivel de abstracción hasta el dominio del problema, por lo que el entendido del dominio puede comprender, validar y desarrollar programas haciendo uso del lenguaje. |
| | Las especificaciones creadas por DSL son breves, autodocumentadas y pueden ser reutilizables. |
| | Mejora la productividad, portabilidad y mantenibilidad de las aplicaciones para proyectos grandes. |
| | DSL se usan para la generación automática de código. |
| **DESVENTAJAS** | Uno de los mayores inconvenientes es la inversión significativa que se realizaría inicialmente para el coste del diseño. |

Los pasos que son principales para el desarrollo de un DSL son:

1. Análisis del dominio específico: Tener claro el objetivo de DSL y el dominio para definir el lenguaje.

2. Definición del metamodelo: Definir el metamodelo tomando en cuenta los elementos del lenguaje que se quiere implementar, y verificando también las relaciones o restricciones de algún elemento.
3. Representación visual: Encontrar una representación adecuada de cada elemento y conexión del lenguaje.
4. Desarrollar generadores: Los generadores permiten crear al código del nuevo lenguaje generado. Además, también producen la documentación, métricas, información de configuración, etc.

**Técnicas de Testeo**

El testeo de los aplicativos se realiza para verificar calidad, eficacia y garantizar éxito en su desempeño. Esta prueba o evaluación forma parte del desarrollo de los aplicativos y permite reducir riesgos en el proceso y corregirlos en el menor tiempo posible. A continuación, se describen las técnicas que se consideran las más importantes al momento de evaluar el desempeño de aplicativos o software (Apiumhub, 2017).

**Pruebas funcionales**

Este tipo de pruebas se basan en asegurarse de que las características funcionen de principio a fin. Además, que sirven para verificar que las aplicaciones y funcionalidades actúen correctamente acorde a su diseño. Los test se utilizan para verificar que la aplicación, página web ejecute su funcionalidad a través de una respuesta adecuada a los controles de usuario, una interfaz de usuario consistente, integración con otros sistemas y procesos de negocios, y manejo adecuado de información y búsqueda. Herramientas de pruebas funcionales se enlistan a continuación:

- Jmeter
- Gatling
- Supertest
- SoapUI
- Cucumber

**Pruebas de rendimiento**

La prueba de rendimiento es una práctica de test que determina la actuación de un sistema en términos de respuesta y estabilidad en una carga de trabajo en particular.

También puede servir para investigar, medir, validar o verificar otros atributos de calidad del sistema, como la escalabilidad, seguridad y uso de recursos. Herramienta de pruebas de rendimiento se enlistan a continuación:

- Jmeter y Gatling.

**JMeter**

Es una herramienta de testing cuyas funcionalidades se pueden resumir bajo los siguientes ítems:

- Diseñar un *testplan* o generar el fichero *.jmx*.
- Ejecutar el *testplan*.
- Entender los resultados de distintas formas después de la ejecución del testplan.

"JMeter es una herramienta ideal para realizar pruebas de rendimiento de aplicaciones web. Se puede utilizar para simular una carga en un servidor, grupo de servidores o red, y así probar su eficiencia y el rendimiento en general bajo diferentes cargas" (Apache JMeter, 2019).

**Escala de Usabilidad de Sistemas (S. U. S.)**

Es un sistema fue creado por John Brooke hace 31 años aproximadamente para medir la usabilidad de sistemas computacionales (MONT, 2017). Con SUS se nos permite medir los siguientes parámetros que se detallan en la Tabla 11.

Tabla 11.

*Parámetros medidos por SUS.*

| PARÁMETROS | DETALLE |
|---|---|
| **Efectividad** | Si las personas pueden alcanzar sus objetivos y cumplir las tareas. |
| **Eficiencia** | Cuantos recursos se utilizan para cumplir los objetivos. |
| **Satisfacción** | El nivel de confort que experimentaron para alcanzar los objetivos. |

Este sistema ya cuenta con un formato, el cual puede ser descargado y ser modificado para el sistema que se quiere evaluar. Se solicita al usuario que se evalúe del 1 al 5 (siendo 1 totalmente desacuerdo y 5 totalmente de acuerdo) de cómo percibe los aspectos del sistema debe ser manipulado previamente (ejemplo app o página web).

El usuario define si está de acuerdo o no con los aspectos consultados para que se puede realizar un cálculo y promedio de la puntuación. Con este resultado se pueda validar el nivel de usabilidad de lo que se ha puesto a prueba, y estas escalas de valores se fijan y detallan en la Figura 13.

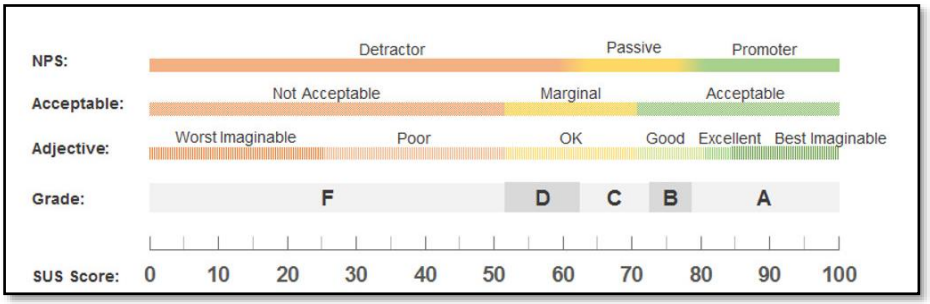



***Figura 13.*** Escala de valores SUS
FUENTE: (MeasuringU, 2019)

Resumiendo, que la escala se termina definiendo como se detalla a continuación:

Tabla 12.

*Escala de valoración SUS*

| Rango Puntuación | Significado |
|---|---|
| **25 puntos** | Usabilidad lo peor imaginable. |
| **25 hasta 38 puntos** | Usabilidad es pobre. |
| **38 hasta 52 puntos** | Es un mínimo aceptable bajo. |
| **52 hasta 73 puntos** | La usabilidad está considerada como buena. |
| **85 puntos** | Excelente |
| **85 a 100 puntos** | es lo mejor posible, a esto es lo que debería aspirar cualquier plataforma digital |

## Amazon Web Service

Conocido también con sus siglas AWS, es una plataforma en la nube más completa y acogida en el mundo debido a que ofrece 165 servicios de centro de datos. Grandes compañías reconocidas u organismos gubernamentales líderes confían en su estructura y prestaciones de AWS. A continuación, se enlistas las características que hacen AWS plataforma líder en la nube:

- Mayor funcionalidad.
- La comunidad más grande de socios y clientes.
- Ritmo de innovación más rápido.
- Mayor seguridad.
- Experiencia operacional más comprobada.

En la Tabla 13 se enlista algunos de los servicios que presta AWS siendo los más destacados o más utilizados por los usuarios que acceden a esta plataforma.

Tabla 13.

*Servicios más destacados de AWS*

| Servicios de Amazon Web Service | | Definiciones |
|---|---|---|
| Informática | EC2 | Permite seleccionar una configuración de memoria, CPU y almacenamiento de la instancia. |
| | Lightsail | Configura de forma automática redes, accesos y entornos de seguridad. |
| | ECR | Es un registro de contenedores de Docker administrado que facilita a los desarrolladores en el almacenamiento, administración e implementación |
| | EKS | Facilita la implementación, la administración y el escalado de aplicaciones en contenedores. |
| | ECS | Permite crear fácilmente todo tipo de aplicaciones en contenedores, desde microservicios y aplicaciones de ejecución prolongada a trabajos en lote y aplicaciones de aprendizaje automático. |
| | Elastic Beanstalk | Es un servicio fácil de utilizar y escalar servicios y aplicaciones web desarrollados con Java, .NET, PHP, Node.js, Python, Ruby, Go y Docker en servidores familiares como Apache, Nginx, Passenger e IIS. |
| | Batch | Permite a los desarrolladores, científicos e ingenieros ejecutar de manera fácil y eficiente miles de trabajos informáticos por lotes en AWS. |
| | Serverless | Se refiere a la arquitectura nativa de la nube, con la cual se le permite trasladar más responsabilidades operativas a AWS. |

**CONTINÚA ⟶**

| Servicios de Amazon | Web Service | Definiciones |
|---|---|---|
| Almacenamiento | S3 | Es un servicio de almacenamiento de objetos creado para almacenar y recuperar cualquier volumen de datos desde cualquier ubicación de Internet. |
| | EFS | Es un servicio completamente administrado que facilita la configuración, el escalado y la rentabilización del almacenamiento de archivos en la nube de Amazon. |
| | FSx | Integra los sistemas de archivos de terceros con servicios de AWS nativos de la nube |
| | Storage Gateway | Es un servicio de almacenamiento en la nube híbrida que le brinda acceso local a almacenamiento en la nube prácticamente ilimitado. |
| | AWS Backup | Es un servicio de copias de seguridad completamente administrado que facilita la tarea de centralizar y automatizar el respaldo de los datos en los servicios de AWS en la nube |
| Base de datos | RDS | Es un servicio administrado de bases de datos relacional. |
| | DynamoDB | Es una base de datos no relacional para aplicaciones que necesitan rendimiento alto a cualquier escala. |
| | ElastiCache | Es un servicio de almacenamiento de datos en memoria compatible con Redis. |
| | Neptune | Es un servicio completamente administrado de bases de datos de gráficos que funciona con conjuntos de datos altamente conectados. |
| | Redshift | Es un servicio de almacenamiento de datos en la nube rentable, sencillo y rápido. |
| | Amazon QLDB | Es una base de datos de contabilidad completamente administrada en la que se proporciona un registro de transacciones transparente e inmutable. |
| | DocumentDB | Es un servicio de base de datos de documentos completamente administrado que es compatible con cargas de trabajo de MongoDB. |
| Internet de las Cosas | IoT Core | Es una plataforma que permite conectar dispositivos a servicios de AWS y a otros dispositivos |
| | FreeRTOS | Es un sistema operativo de código abierto para microcontroladores que facilita la programación, la implementación, la protección, la conexión y la administración de los dispositivos de borde pequeños y de bajo consumo. |
| | IoT 1-Click | Es un servicio para facilitar que dispositivos simples activen las funciones de AWS Lambda que ejecutan una acción específica. |
| | IoT Analytics | Es un servicio completamente administrado que facilita la ejecución e instrumentación de análisis sofisticados de enormes volúmenes de datos de IoT. |
| | IoT SiteWise | Facilita la recopilación, la estructuración y la búsqueda de datos de IoT desde equipos industriales a escala. |
| | IoT Things Graph | Permite crear aplicaciones compatibles con IoT mediante la conexión de dispositivos, como sensores y actuadores, y servicios web con poco código o sin él. |

**Google Cloud Platform**

Es la primera plataforma integral, híbrida y multinube que nos permite desarrollar, construir, probar e implementar varias aplicaciones, un mismo código en cualquier lugar. Está dirigida para quienes requieren una gran infraestructura para sus proyectos; cuenta con herramientas para almacenamiento, computación y servicios para aplicaciones web o móviles. Además, se puede acceder a una serie de servicios de la nube, para crear sitios web desde sencillas hasta las más complejas.


***Figura 14.*** Estructura Google Cloud Platform

FUENTE: (Google Cloud, 2019)

A continuación, en la siguiente Tabla 14 se presenta las ventajas *Google Cloud* que permitirán a organizaciones, empresas u otras entidades generar soluciones innovadoras y sumergirse en una transformación digital.

Tabla 14.

*Ventajas de Google Cloud*

| VENTAJAS | |
|---|---|
| Confianza y Seguridad | Mantiene los datos protegidos y conformes con las normativas pertinentes. |
| Nube abierta | Escala la infraestructura con una tecnología abierta y flexible. |
| Experiencias de éxito Clientes | Se puede conocer como usan Google Cloud las empresas. |

**CONTINÚA →**

| | VENTAJAS |
|---|---|
| Centros de datos y red | Desarrollar sobre la misma infraestructura que usa Google. |
| Partners | Aprovechar el ecosistema mundial de expertos en la nube. |
| Blog de Google Cloud | Informarse de las noticias y novedades sobre productos. |

## Alternativas de Servicios de computación en la nube

Para el presente proyecto se pretende crear una instancia que se ejecute sobre uno de los proveedores de servicios de computación en la nube que se presentan en la Tabla 14.

Tabla 15.

*Opciones de Servicios de computación en la nube*

| SERVICIO | DETALLE |
|---|---|
| *IaaS de Amazon Web Services* | *Amazon Elastic Compute Cloud* (Amazon EC2) es el servicio de cómputo de AWS. Permite a los usuarios alquilar computadores virtuales y poder ejecutar propias aplicaciones.<br>EC2 de Amazon se apoya en las tecnologías de virtualización, permitiendo utilizar gran variedad de sistemas operativos a través de sus interfaces de servicios web, personalizarlos, gestionar permisos de acceso a la red y ejecutar tantos sistemas como desee. |
| *IaaS de Google Cloud Platform* | *Google Compute Engine* (Google EC) suministra máquinas virtuales que se ejecutan sobre los centros de datos de Google y su red mundial de fibra óptica.<br>Las herramientas y el soporte para flujos de trabajo de *Compute Engine* hacen posible el escalamiento de instancias mediante una red global de procesamiento de nube con balanceo de cargas.<br>Los servidores virtuales de Google EC están disponibles en muchas configuraciones, tales como, la opción de crear tipos personalizados de máquinas optimizadas a necesidad. |
| *IaaS de Microsoft Azure* | *Azure VM* es el servicio de IaaS de Microsoft Azure, ofrece una amplia flexibilidad de virtualización para una variada gama de soluciones informáticas: desarrollo y pruebas. Permite desplegar máquinas virtuales compatibles con Linux, Windows Server, SQL Server, Oracle, entre otros. |

# CAPÍTULO 3

# DISEÑO DE LA ARQUITECTURA

## Requisitos de diseño

Los requisitos de diseño de arquitectura del sistema es el medio de comunicación entre el cliente y desarrollador**.** Para iniciar con el diseño de la arquitectura es necesario traer a colación los requerimientos que tiene el proyecto, y cumplir con todos los objetivos detallados en el capítulo 1 para tener una implementación exitosa en los resultados esperados; y de esta forma ir plasmando los propósitos con los que fue planteado el proyecto inicialmente. En la Tabla 16 y Tabla 17 se encuentra la información de los requisitos funcionales y no funcionales del proyecto respectivamente.

Tabla 16.

*Requisitos de diseño funcionales*

| REQUISITOS FUNCIONALES | |
|---|---|
| Requerimientos del cliente | Al ser una arquitectura *Cross-Platform* se debe buscar varias alternativas de plataformas accesibles para el usuario con interfaces GUI que deben contener información útil, además de ser amigable para el usuario. |
| | Información entendible y fácil de manejar. |
| | El cliente debe conocer la confiabilidad de la predicción que presenta el software. |
| | El cliente debe poder cerrar la aplicación móvil de recolección de datos cuando desee. |

**CONTINÚA →**

REQUISITOS FUNCIONALES

| | |
|---|---|
| Necesidades y requisitos del software que debe cumplir de manera satisfactoria | El software de recolección de datos, cliente móvil, debe tener funcionar en segundo plano.<br>La recopilación de datos debe ser en tiempo real.<br>La velocidad que adquiere para la base de datos debe ser instantánea.<br>El administrador debe tener acceso a los servidores desde cualquier dispositivo y en cualquier lugar.<br>La base de datos debe ser escalable debido a que los datos son acumulables en el tiempo.<br>Se debe crear metamodelos para el desarrollo de la implementación para tener una plantilla del programa |
| Funciones que él software será capaz de realizar | El software principalmente debe ser capaz de predecir el tráfico de varias calles en la ciudad de Quito.<br>La información debe ser presentada en varias plataformas.<br>La recolección de datos debe ser mínimo de 2 fuentes. |
| Detalles técnicos | El parámetro a predecir es la velocidad y se debe mostrar en la interfaz conjuntamente con la intersección, es la información mínima a mostrar en cualquier GUI.<br>La herramienta de recolección de datos funciona bajo un sistema operativo android y hasta la versión 8.<br>Es obligatorio tener acceso a la red de internet en todas las plataformas.<br>El cliente móvil debe contener GPS.<br>Los servicios de los aplicativos webs deben presentarse en arquitectura REST.<br>La app de adquisición de datos debe funcionar en el sistema operativo android 8.0 |
| Manipulación de datos de entrada y salida | En el orquestador los datos deben ser preparados antes de entrar a la base de datos, el envío de información a los diferentes aplicativos será en JSON.<br>Los valores a predecir es la velocidad en una calle. |

Tabla 17.

*Requisitos de diseño no funcionales*

| REQUISITOS NO FUNCIONALES | |
|---|---|
| Propiedades o cualidades que el software debe tener | La herramienta de adquisición de datos en el dispositivo móvil, debe tener una interfaz sencilla y con la capacidad de seguir funcionando sin la intervención del usuario y parar la aplicación en caso que el usuario no desee enviar datos de ubicación.<br>La recolección de datos móvil debe iniciar la sesión con ID de usuario.<br>La aplicación de TVD debe ser muy simple de manejar para el usuario y el fin de la aplicación es mostrar la velocidad de una intersección en la fecha que el usuario desee predecir.<br>La aplicación web debe mostrar varias páginas para lo cual se necesita un menú, en donde la página principal muestre un mapa con el tráfico del momento actual por default, y la opción de cambiar la fecha y hora a la que desean ver la predicción teniendo como referencia la gama de colores para indicar la cantidad de tráfico siendo verde cuando no hay tráfico y rojo cuando el tráfico es intenso.<br>La siguiente página web debe mostrar un gráfico del tráfico vehicular en un tramo de tiempo de una sección de la calle.<br>Por último, la siguiente opción de página web debe predecir cualquier sección de una calle sin previo entrenamiento.<br>El administrador de la plataforma tendrá servicios mediante la página para agregar información de elementos en la base de datos. |
| Restricción del sistema operativo | Debe funcionar en los dispositivos móviles android hasta la versión 8. |
| Restricción de ambiente | El aplicativo Web debe utilizar la dirección IP para el ingreso a la interfaz.<br>Los datos ingresados a la base de datos los debe manejar el administrador. |

**CONTINÚA →**

| | |
|---|---|
| Restricciones o condiciones del producto | El perímetro a predecir es algunas calles del Distrito metropolitano de Quito.<br>La base de datos no necesita ser *BigData* debido a que los datos aún no llegan a ese punto, pero en el futuro se debería pensar en almacenamiento continuo y en tener una base de datos escalable. |
| Restricción de rapidez | La conexión a internet es indispensable en todos los dispositivos.<br>Los datos tomados deben ser en tiempo real.<br>El aplicativo Web y el aplicativo de TVD deben reflejar la información de predicción de tal manera que el usuario no pueda percibir la demora en la predicción. |
| Restricción de seguridad | Los datos son ingresados solo por los administradores, los datos no son guardados por el cliente.<br>Todos los datos de la base de datos no son manipulables por el usuario. |
| Restricción de usabilidad | El sistema se diseñará bajo una arquitectura REST, es decir tendrá microservicios que integrados formarán la plataforma final. |

## Estructura del diseño del sistema

Parte fundamental del presente proyecto de titulación es poder realizar predicciones de tráfico vehicular de algunas calles de la ciudad de Quito. Para ello se debe contar con datos previos, que permitan obtener un modelo mediante la ejecución de un algoritmo de entrenamiento; entonces como primer punto a tener en cuenta es la adquisición de datos. El tipo de datos necesario para realizar un entrenamiento y posterior predicción comúnmente está relacionado con las series de datos temporales, es decir que los datos muestreados son tomados periódicamente cada cierto intervalo de tiempo. Al estar desarrollándose una plataforma de predicción, se deben crear una instancia que permita adquirir, filtrar y procesar los datos. Para ello se ha planteado la creación de

un conjunto herramientas que permiten al administrador de la plataforma cumplir con estos estas tareas.

A partir de contar con datos ya filtrados y procesados a las necesidades del proyecto se procede a buscar una herramienta de almacenamiento de datos, por el tiempo de ejecución de la investigación se usará *datasets*, pero en trabajos futuros se propone trabajar con *"big data*".

A continuación, se encuentra la predicción de tráfico vehicular para este módulo se utiliza el conocimiento de aprendizaje automático "*Machine Learning*".

Para finalizar se encuentra la interfaz de usuario, en donde el usuario va a solicitar y recibir las peticiones de predicción, para lo cual se tiene varias alternativas TVDigital, smartphone, web.

En la Figura 15 se indica en general el comportamiento de la estructura del sistema el diseño arquitectónico de software tiene un patrón MVC, en donde el Modelo es la base de datos, Vista son las interfaces de usuario y el orquestador es el servidor web y la instancia de predicción.

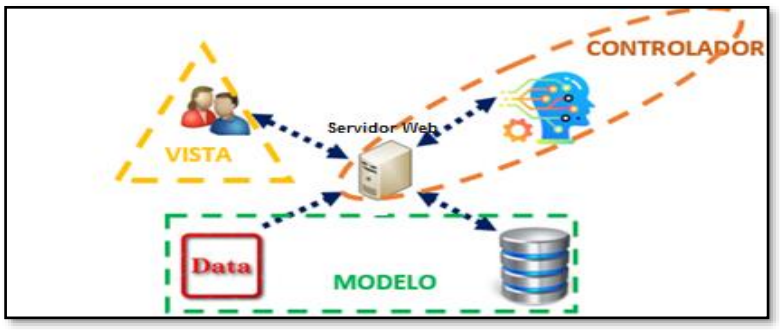



***Figura 15.*** Estructura del diseño del Sistema

En la Tabla 18 se muestra los elementos de la arquitectura MVC con el proceso de cada elemento.

Tabla 18.

*Elementos y procesos del diseño de la arquitectura*

| Lugar en donde se ejecutan | | Proceso |
|---|---|---|
| **FRONTEND** | Cliente móvil | Adquisición de Datos |
| | Cliente TVD | Interfaz gráfica de usuario |
| | Cliente Web | Interfaz gráfica de usuario |
| **BACKEND** | Servicio Web | Controla que todos los procesos se encuentren sincronizados. |
| | Machine Learning | Predicción |
| | Base de Datos | Almacenamiento de datos |

En base a los elementos del sistema se ha creado un diagrama general como se muestra en la Figura 16 y un diagrama de flujo como se muestra en la Figura 17.

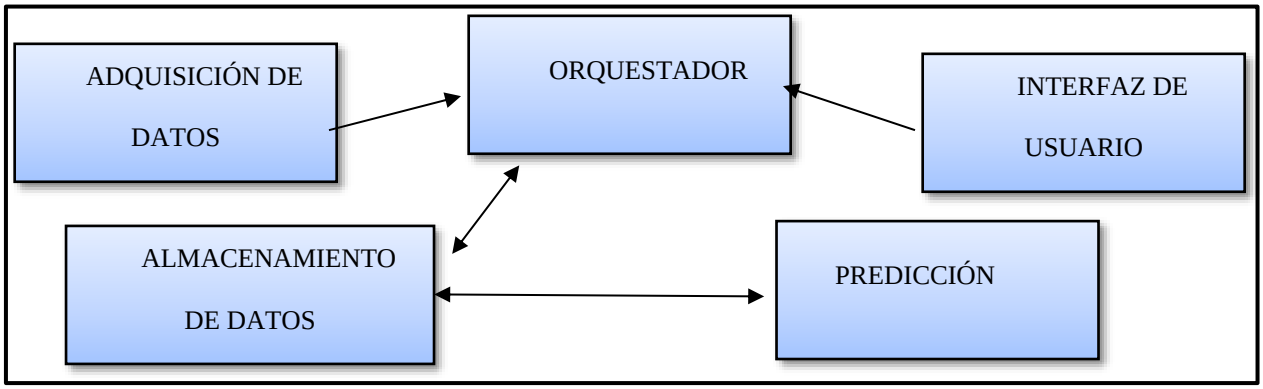



***Figura 16.*** Arquitectura del sistema

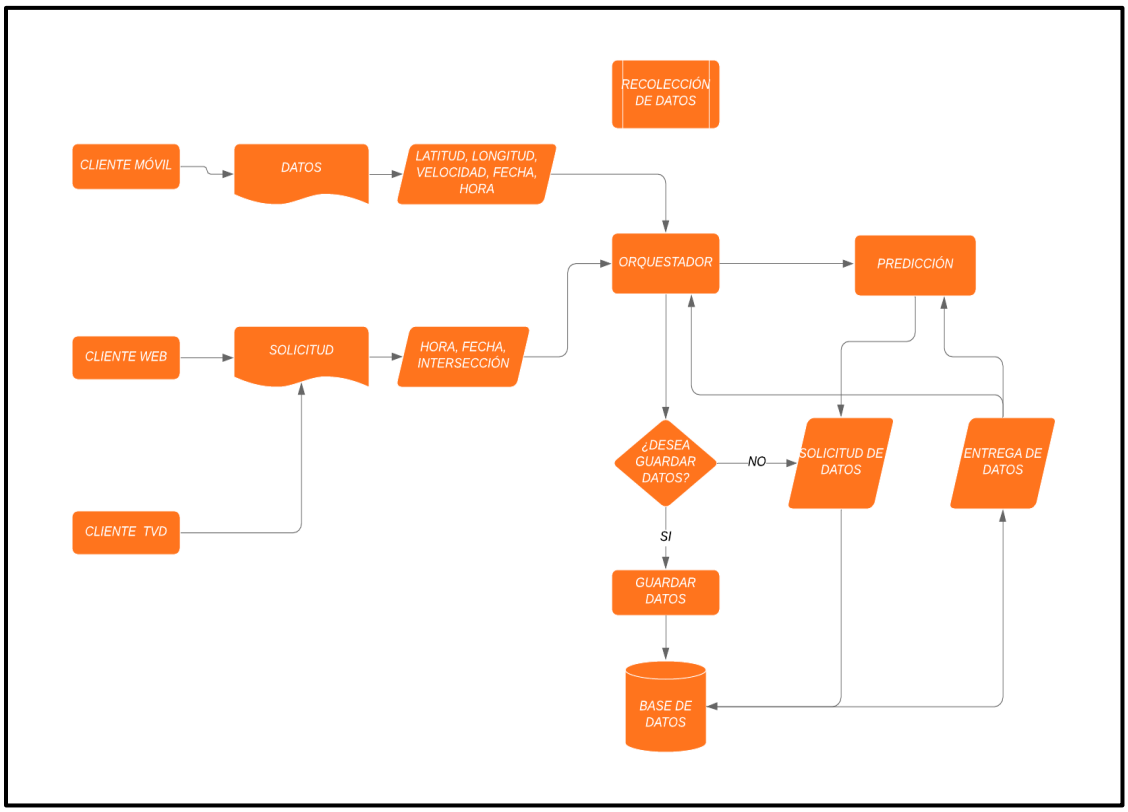



***Figura 17.*** Diagrama de flujo de la estructura del sistema

## Backend

### Diseño para la adquisición de datos de tráfico vehicular

En grandes ciudades poseen sistemas de monitoreo mediante sensores de tráfico vehicular, dichos elementos se encuentran posicionados estratégicamente para permitir obtener una medición fiable del tráfico. En el contexto del proyecto no se tienen instalados sensores físicos y por tanto se ha tenido que pensar en alternativas que permitan el monitoreo de los datos de tráfico. Sin embargo, ha sido una tarea prioritaria la definición de un esquema de datos que permita obtener información apropiada para la predicción. Debido a esto se ha utilizado un flujo de trabajo que permite obtener datos adecuados para efectuar un entrenamiento.


***Figura 18.*** Flujo de trabajo para la adquisición de datos

**Identificación de los datos relevantes para el entrenamiento**

Tomando como referencia la Tabla 19 que proporciona el municipio de la ciudad de Madrid (Departamento Tecnologías de Tráfico Madrid, 2013), se ha encontrado que ellos guardan los siguientes datos por cada sensor:

Tabla 19.

*Campo de datos de la ciudad*

| Nombre | Tipo | Descripción |
|---|---|---|
| **Id** | Entero | Identificación única del Punto de Medida en los sistemas de control del tráfico del ayuntamiento de Madrid. |
| **Fecha** | Fecha | Fecha y hora oficiales, de Madrid con formato yyy-mm-dd hh:mi:ss |
| **Tipo_elem** | Texto | Nombre del Tipo de Punto de Medida: Urbano o M30 |
| **Intensidad** | Entero | Intensidad del Punto de Medida en el periodo de 15 minutos (vehículos/hora). |
| **Ocupación** | Entero | Tiempo de ocupación del Punto de Medida en el período de 15 minutos (%) |
| **Carga** | Entero | Carga de vehículos en el período de 15 minutos. Parámetro que tiene en cuenta intensidad, ocupación, y capacidad de la vía y establece el grado de uso de la vía de 0 a 100. |
| **Vmed** | Entero | Velocidad Media de los vehículos en el período de 15 minutos (Km/h) Solo para puntos de medida interurbanos M30. |
| **Error** | Texto | Indicación de si ha habido al menos una muestra errónea o sustituida en el periodo de 15 minutos.<br>N: no ha habido errores ni sustituciones.<br>E: los parámetros de calidad de alguna de las muestras integradas no son óptimos.<br>S: alguna de las muestras recibidas era totalmente errónea y no se ha integrado. |
| **periodo_integracion** | Entero | Número de muestras recibidas y consideradas para el periodo de integración. |

Fuente: (Departamento Tecnologías de Tráfico Madrid, 2013)

Se tiene que, en el caso particular de Madrid, se toma como medida de tráfico el número de vehículos que han cruzado delante del sensor fijo en un intervalo de tiempo. Adicional se tiene que el intervalo entre muestras usado en la base de datos de Madrid es 15 minutos, un total de 96 muestras al día. Otros datos incluyen información de identificación del sensor, fecha y tipos de sensor; variables que para el efecto del proyecto podrían descartarse.

La ubicación de algunos de los sensores se muestra a continuación y se utiliza como criterio de referencia para la localización de sensores dependiendo de la factibilidad ya que debido diferencias en geografía formas de la vía podría en algunos casos no aplicarse.

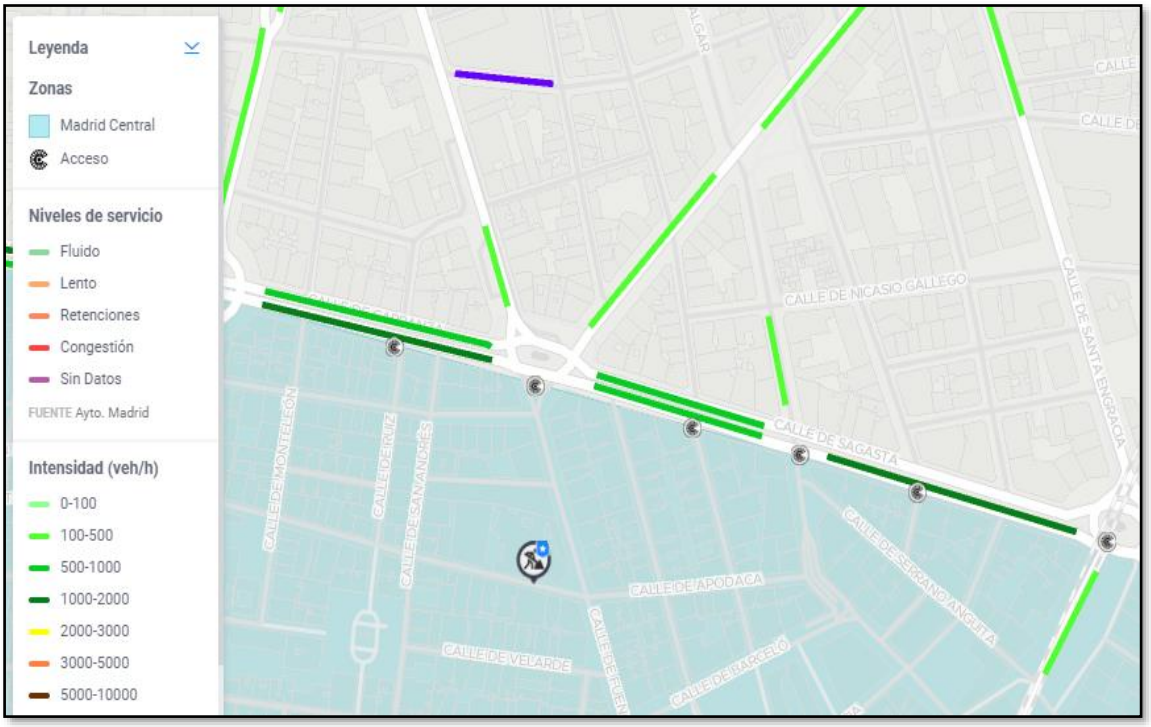



***Figura 19.*** Mapa de referencia de sensores de una calle de Madrid

*Fuente: (Departamento Tecnologías de Tráfico Madrid, 2013)*

Se puede apreciar que en el caso de la *Calle de Sagasta* se han instalado sensores físicos en cada una de las intersecciones con dicha calle, en el proyecto se intenta replicar ese comportamiento posicionando sensores de manera similar.

**Alternativas de datos a recolectar para el entrenamiento**

Para la toma de datos se usará los sensores del dispositivo móvil como se indica en los requisitos del proyecto, además en la sección 0 se tiene un ejemplo de los campos que deben ser adquiridos por las herramientas de recolección de datos y luego almacenar la información en una tabla llamada '*Datos de tráfico*'. En la Tabla 20 se muestra los campos que contendrán esta información.

Al tener un dispositivo móvil como herramienta de adquisición de datos se buscará como alternativa crear puntos en lugares estratégicos para filtrar los datos en estas coordenadas y tener

información por secciones de la calle, en la Tabla 21 se muestra los datos necesarios para determinar el área en la cual los datos serán filtrados es decir serán ‘Sensores virtuales de tráfico’ *SVT* en los cuales la información de tráfico será filtrada, esta tabla será guardada en la base de datos con el nombre ‘sensores’.

Los datos a tomar en cuanta serían los siguientes:

Tabla 20.

*Estructura de los datos necesarios para el aprendizaje automático del proyecto*

| VARIABLE | IMPORTANCIA |
|---|---|
| **Hora de la muestra** | Los datos tomados cambian en el tiempo por lo tanto es fundamental la hora en la que los datos son tomados con la hora local Quito-Ecuador, |
| **Fecha de la muestra** | Los datos a tomar son temporales y cambiantes en el tiempo, por tal motivo es necesario tomar la fecha. |
| **Latitud y longitud de la muestra** | Sistema de coordenadas, para definir la ubicación exacta |
| **Velocidad** | Conocer la velocidad en la que los vehículos recorren ayuda a determinar la cantidad de tráfico que existe en la zona |
| **Dirección Cardinal** | El Tráfico vehicular no es igual en sentido SN que NS por tal motivo es necesario conocer la posición el momento de tomar los datos. |

Tabla 21.

*Tabla sensores*

| Variable | Importancia |
|---|---|
| **Id** | Número de sensor |
| **Nombre** | Nombre del sensor |
| **Lt1- Lg1, Lt2- Lg2, Lt3- Lg3, Lt4- Lg4** | Los sensores abarcan un área rectangular por lo tanto necesitan 4 puntos |

## Identificación de las herramientas para recolectar

Previamente en la planificación del proyecto se ha establecido la utilización de los sensores que poseen los teléfonos inteligentes para tomar los datos requeridos en la sección 0.

En los requerimientos se indica que mínimo debe existir 2 herramientas de recolección de datos, debido al tiempo de ejecución del proyecto y a la necesidad de la gran cantidad de datos para efectuar una predicción óptima.

En esta sección se realizará una comparación de los servidores de aplicaciones de mapas en la web que ofrecen información de tráfico en tiempo real y luego posteriormente tomar una decisión de la herramienta a usar.

Tabla 22.

*Elección de la API*

| | DISTANCE MATRIX API | DIRECTIONS API | GEOLOCALIZACIÓN API |
|---|---|---|---|
| **Costo** | 0.04cents por petición | 0.04cents por petición | 0.04cents por petición |

**CONTINÚA →**

| Velocidad | No | No | No |
|---|---|---|---|
| **Distancia** | Si | 2 o más | No |
| **Tiempo** | Si | Si | No |
| **Datos que necesita** | Punto origen, punto destino | Tipo de vehículo, punto de origen, punto de destino | Coordenadas |

De acuerdo a la Tabla 22 indica que la opción más valedera es el uso de la API DISTANCE MATRIX a la cual se le asignaran tramos de la calle con punto origen y punto final para tomar la distancia y el tiempo que tarda en recorrer y con esta información tener la velocidad del auto.


***Figura 20.*** Definición de puntos para la API Distance Matrix

Los requerimientos de la API DISTANCE MATRIX son punto de origen y punto final y entre estos puntos calcula el tiempo que se demora el vehículo y la distancia por tanto la velocidad es fácilmente calculable, entonces a partir de esta necesidad se crea la tabla puntos con los parámetros que muestra la Figura 20 en donde se secciona la calle con punto origen y punto final como se muestra en la Tabla 23 por cada dirección cardinal es decir NS y SN.

Tabla 23.

*Tabla para el ingreso de puntos para filtrar datos*

| Parámetros | Descripción |
|---|---|
| **ID** | Dirección cardinal de la calle:<br>Impar: NS<br>Par: SN |
| **Sub_ID** | Es un número secuencial que inicia en 1 en el punto inicial y termina en el último punto en donde termina la calle en la misma |
| **Ruta** | Nombre de la calle |
| **Nombre** | Nombre de la intersección |
| **Latitud-Longitud** | Coordenadas del punto inicial-final |

## Orquestador

El orquestador tiene 3 funciones principales:

1) **Función de toma de datos con API DISTANCE MATRIX:** De acuerdo a (Departamento Tecnologías de Tráfico Madrid, 2013) se toma datos cada 15 minutos en todos los sensores, en el proyecto se ha decidido ampliar el tiempo sin afectar la recolección de datos para el entrenamiento a cada 20 minutos, debido a que se está usando una API que incluyen costos

por cada solicitud, además el orquestador será el encargado de estructurar y hacer un casting de los datos antes de ser guardados en la base de datos.

2) **Función de preparación de datos para predicción**: Cada 24 horas se filtra los datos por cada sensor los cuales serán guardados en una *tabla tránsito* en la base de datos la cual tendrá los mismos parámetros de la Tabla 23. Y posteriormente se realiza el encendido de la instancia que tiene la función de predicción.
3) **Función de Standby:** El orquestador se encuentra esperando las peticiones de las interfaces del usuario o del cliente móvil para enviar las respuestas.

**Instancia para el orquestador**

Para el presente proyecto se pretende crear una instancia que se ejecute sobre uno de los proveedores de servicios de computación en la nube que se presentan a continuación:

Tabla 24.

*Comparación de instancias para el orquestador*

| Elementos | Google Cloud Platform | Amazon Web Service | Microsoft Azure |
|---|---|---|---|
| **Costo**<br>***1 núcleo de CPU***<br>***1 GB de RAM***<br>***50 GB de almacenamiento*** | Aproximadamente 30 USD mensuales.<br>Entrega un crédito de 300$ | 0 USD sobre su capa de gratuidad. 5 USD mensuales fuera de gratuidad. 1 año de gratuidad | 0 USD durante 30 días. |
| **Licencias de software**<br>***Windows Server 2016**** | Aproximadamente 50 USD mensuales. | 0 USD sobre su capa de gratuidad. | 0 USD durante 30 días. |
| **Administración de la instancia** | Mediante clientes SSL o RDP | Mediante clientes SSL o RDP | Mediante clientes SSL o RDP |
| **Disponibilidad de instancias** | Ilimitadas sobre pago mensual. | Prácticamente ilimitada en capa gratuita | 750 Horas gratuitas, ilimitada sobre pago. |

**CONTINÚA →**

| Elementos | Google Cloud Platform | Amazon Web Service | Microsoft Azure |
|---|---|---|---|
| **Conectividad desde y hacia la instancia.** | Configurable mediante consola de GCP.<br>Acceso a internet y posibilidad de crear VLAN entre instancias dentro de GCP. | Configurable mediante consola de AWS.<br>Acceso a internet y posibilidad de crear VLAN entre instancias dentro de AWS. | Configurable mediante la consola de Azure |
| **Facturación de los servicios utilizados** | Se cobra mensualmente mediante una tarjeta de crédito. Durante el periodo de prueba Google brinda 300 USD de crédito por 12 meses.<br>Es posible crear presupuestos mensuales y alertas de facturación. | Se cobra mensualmente mediante una tarjeta de crédito. Si se exceden los términos de uso (tráfico de datos, almacenamiento o uso de CPU) de la capa gratuita se comienzan a facturar los servicios.<br>Es posible crear presupuestos mensuales y alertas de facturación. | Se cobra mensualmente mediante una tarjeta de crédito. Durante el periodo de prueba Microsoft brinda 200 USD de crédito por 30 días. |

** Se escogió Windows Server 2016 como SO de comparación ya que se encuentra disponible bajo la capa de gratuidad en las tres plataformas.*

**Alternativa recomendada como servicio de computación en la nube para el proyecto**

Amazon Web Services, Google Cloud Platform y Microsoft Azure poseen servicios y funcionalidades muy similares, tanto en posibilidades de escalamiento en almacenamiento, procesamiento y conectividad. Los tres servicios ofrecen integración con bases de datos, almacenamiento en la nube y herramientas de desarrollo para aplicaciones web. Las tres plataformas permiten crear instancias casi idénticas mediante su consola de configuración.

Sin embargo, tomando en cuenta que tiempo de desarrollo se podría optar por *Amazon Web Services* o *Google Cloud Platform* ya que ambos servicios permiten uso de sus plataformas de forma casi gratuita por 12 meses. Azure no es recomendable ya que su periodo de uso gratuito se limita a 30 días, su uso incurría en costos adicionales que no serían necesarios utilizando las otras dos alternativas encontradas. *Google Cloud Platform* entrega un crédito de 300$ vs. *AWS* brinda

12 meses de crédito gratuito, por lo tanto, la alternativa más eficiente económicamente con prestaciones similares a las otras plataformas es *AWS.*

**Diseño de Predicción**

Los datos son adquiridos del cliente móvil o de la API Distance Matrix, luego guardados en la base de datos, posteriormente se filtran los datos por sensor para iniciar con la predicción de tráfico vehicular.

**Procedimiento de predicción**

Los requerimientos del proyecto indica que se debe mostrar la predicción de varias calles de la ciudad de Quito, y en vista de los costos que representa tener el entrenamiento por cada calle se ha decidido usar el siguiente flujo de trabajo para cumplir con los objetivos planteados y brindar resultados óptimos y eficientes.

- Determinar las calles a las que se brindará el servicio de predicción en el Distrito Municipal de Quito.
- Dividir las calles por intersecciones.
- Asignar características a cada intersección en función de lo determinado.
- Asignar información característica a cada una de las intersecciones de las calles sin entrenamiento previo de tráfico vehicular.
- Crear una tabla de entrenamiento y otra tabla para predicción en función de las características de cada intersección.
- Realizar la predicción del sistema dividida en dos categorías:
    a) Modelo temporal de predicción para la intersección con información de entrenamiento de tráfico vehicular.

b) Árbol de decisiones para la clasificación de las intersecciones sin información de tráfico vehicular.

**Proceso de determinación de las calles de entrenamiento y predicción**

De forma cualitativa se definió características que definen el tránsito en una avenida, las características son las siguientes:

- Carriles
- Restaurantes
- Semáforos
- Escuelas
- Centros Comerciales
- Parques
- Transportes
- Redondel
- Área ejecutiva

De acuerdo a las características nombradas en la parte superior se eligió a la calle Shyris, que cuenta con la mayoría de los parámetros en la ciudad de Quito, y su vez se eligió calles determinadas al azar para la predicción de tráfico las cuales son: 10 de agosto, Amazonas, Eloy Alfaro, Patria, 12 de Octubre, 6 de Diciembre, Napo y América.

**Parámetros de las tablas para entrenar y aprender de las calles sin previa información de tráfico**

Tabla 25.

*Características de las calles a predecir*

| VARIABLE | DESCRIPCIÓN |
|---|---|
| **CALLES** | Nombre de la intersección de la que se toman los datos característicos. |
| **ID** | Es un identificador de la dirección cardinal NS o SN<br>ID_PAR: NS ID_IMPAR: SN |
| **SUB_ID** | Una secuencia de números que inicia en 1 y finaliza cuando terminan las intersecciones de una dirección (SN) de la calle, inicia nuevamente para la siguiente dirección de la misma calle (NS). |
| **ETIQUETA** | Es la variable para predecir, es decir va a identificar la intersección a la que pertenece la sección de calle. |
| **CARRILES** | El número de calle a la que pertenece. |
| **RESTAURANTS** | Identifica una cantidad abundante de restaurants |
| **SEMÁFOROS** | Identifica si en la intersección existe semáforos. |
| **CENTROS EDUCATIVOS** | Identifica si existe centros educativos cercanos a la intersección. |
| **PARQUES** | Identifica si en la intersección existe parques. |
| **TRANSPORTE METROPOLITANO** | Identifica si existen buses que transitan por la zona. |
| **HOSPITALES** | Identifica si en la zona existen hospitales. |
| **REDONDEL** | Identifica si existe un redondel en la intersección. |
| **ÁREA EJECUTIVA** | Identifica si existen abundantes oficinas o empresas en la zona. |
| **PARADA DE BUS** | Identifica si existen cerca paradas de buses. |

En la Tabla 25 se muestra las características que se tomarán en cuenta para realizar la predicción de calles sin información de entrenamiento.

**Como configurar SARIMAX**

a) Elementos de tendencia

p : orden de tendencia de autorregresión.

d : Orden de diferencia de tendencia.

q : Tendencia de orden de promedio móvil.

b) Elementos estacionales

**P** : orden autorregresivo estacional.

**D** : orden de diferencia estacional.

**Q** : Orden de media móvil estacional.

**m** : El número de pasos de tiempo para un solo período estacional.

Es importante destacar que, el parámetro *m* influye en los parámetros *P*, *D*, y *Q*. Por ejemplo, una *m* de 12 para datos mensuales sugiere un ciclo estacional anual.

Es importante destacar que, el *m* parámetro influye en las *P*, *D*, y *Q* parámetros. Por ejemplo, una m de 12 para datos mensuales sugiere un ciclo estacional anual. De acuerdo a recomendaciones de la API se debería usar los valores de 0 en todas las variables por ser un modelo estacional excepto *m* ya que en este caso sería 72 que son el valor de muestras totales de todos los puntos diarios.

**Frontend**

El *Frontend* se enfoca en el usuario, en todo con lo que se interactúa y lo que se ve mientras el usuario navega.

El sistema presenta tres interfaces con el usuario, en la sección presente vamos a detallar cada una de ellas:

**Cliente Móvil**

Como ya se ha mencionado en capítulos anteriores el cliente móvil es el encargado de recolectar datos.

La interfaz del usuario debe iniciar la aplicación con el mapa y tres botones: iniciar, detener, login.

En la Figura 21 se muestra el mockup de la pantalla de inicio de sesión con usuario.


***Figura 21.*** Mockup de inicio de sesión de cliente móvil.

No debe ser necesario iniciar sesión con usuario, al presionar en iniciar se debe iniciar la recolección de datos y al seleccionar detener se debe pausar la toma de datos para respetar la privacidad de información en caso que el usuario no desee enviar la información de su ubicación.

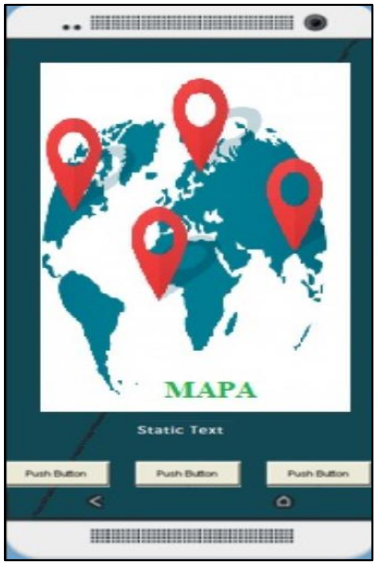



***Figura 22.*** Mockup de inicio de recolección de datos

**Cliente Web**

El cliente web pretende mostrar los resultados obtenidos de predicción de manera amigable al usuario, y también facilitar al administrador el ingreso de información para robustecer más la base de datos.

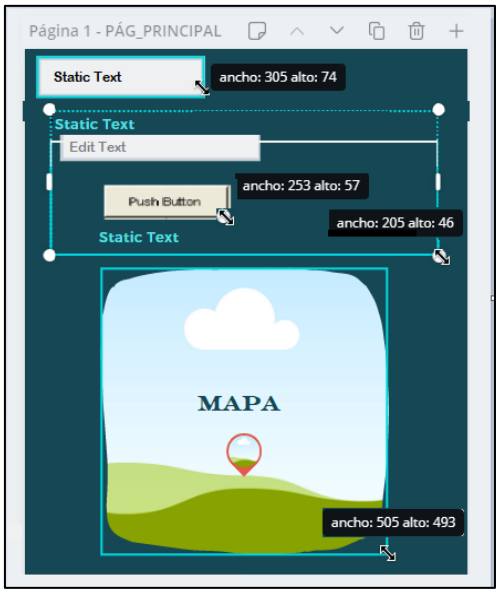



***Figura 23.*** Página principal del cliente web

En la Figura 23 se muestra la página principal que se presentará al cliente web en donde se debe mostrar un menú desplegable y el mapa con las calles del Distrito Metropolitano de Quito con el tráfico vehicular de la fecha y hora actual, además la opción de ingresar la hora y fecha a la que se desee predecir.

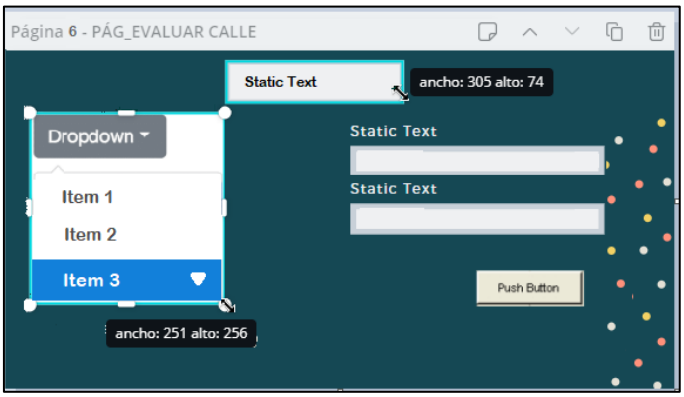



***Figura 24.*** Mockup de gráfico por calle

La Figura 24 muestra un mockup de la página de gráficos por calle, los elementos que debe ingresar el usuario es el intervalo que desea predecir, fecha inicial y fecha final.


***Figura 25.*** Mockup de Evaluar calle

La Figura 25 muestra el mockup de evaluar calle en donde se ingresan varias características de una calle y la web me presenta la gráfica de un intervalo de tiempo si y solo si existe la información en la base de datos.

**Cliente TVD**

Para el diseño de la interfaz de televisión digital se quiere mostrar la velocidad, la fecha y hora de predicción solicitada, a continuación, las plantillas para TVD.

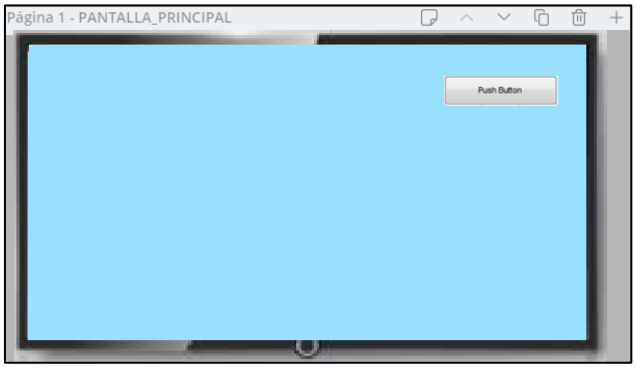



***Figura 26.*** Botón de inicio de predicción

En la Figura 26 se muestra la plantilla principal con el botón inicia la predicción. El cual deberá permitir el despliegue del menú con las calles que se puede escoger.

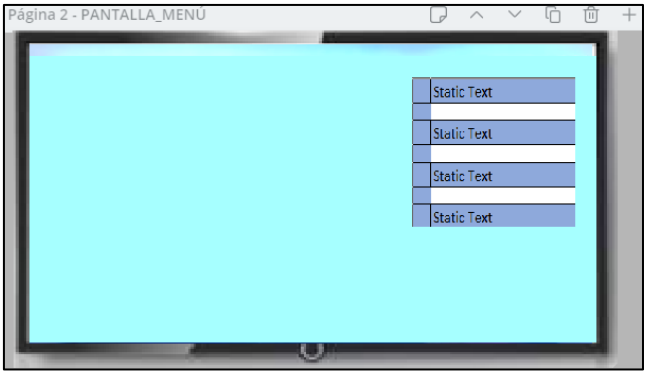



***Figura 27.*** Lista de calles para elegir predicción

En la Figura 27 se debe mostrar un menú de las calles que se puede elegir para la predicción, esto debe aparecer una vez que se seleccione el botón principal que se describió en la Figura 28.

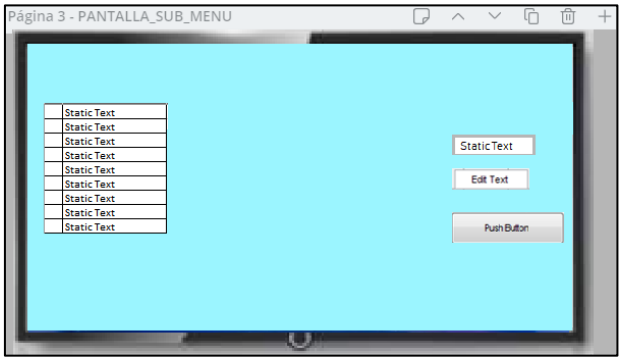



***Figura 28.*** Leyenda con las intersecciones e ingreso del texto

En la Figura 28 se puede observar cómo se requiere que aparezca el menú de las calles con sus intersecciones. Aquí también debe contener un campo de texto donde se pueda ingresar la fecha, hora e intersección y además el botón que debe ejecutar la acción para predecir con la información ingresada.

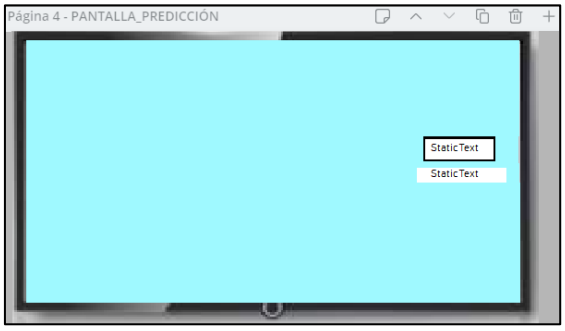



***Figura 29.*** Muestra la respuesta de predicción solicitada.

Por último, en la Figura 29 se muestra la pantalla de cómo debe ir la presentación de la información con la predicción en TVD. Se observa que únicamente debe tener dos campos estáticos, el primero será para describir la información que se va a presentar y el segundo el que contendrá los resultados de la predicción que devolverá el servidor.

## Diagrama BPMN

## Herramienta de adquisición de datos

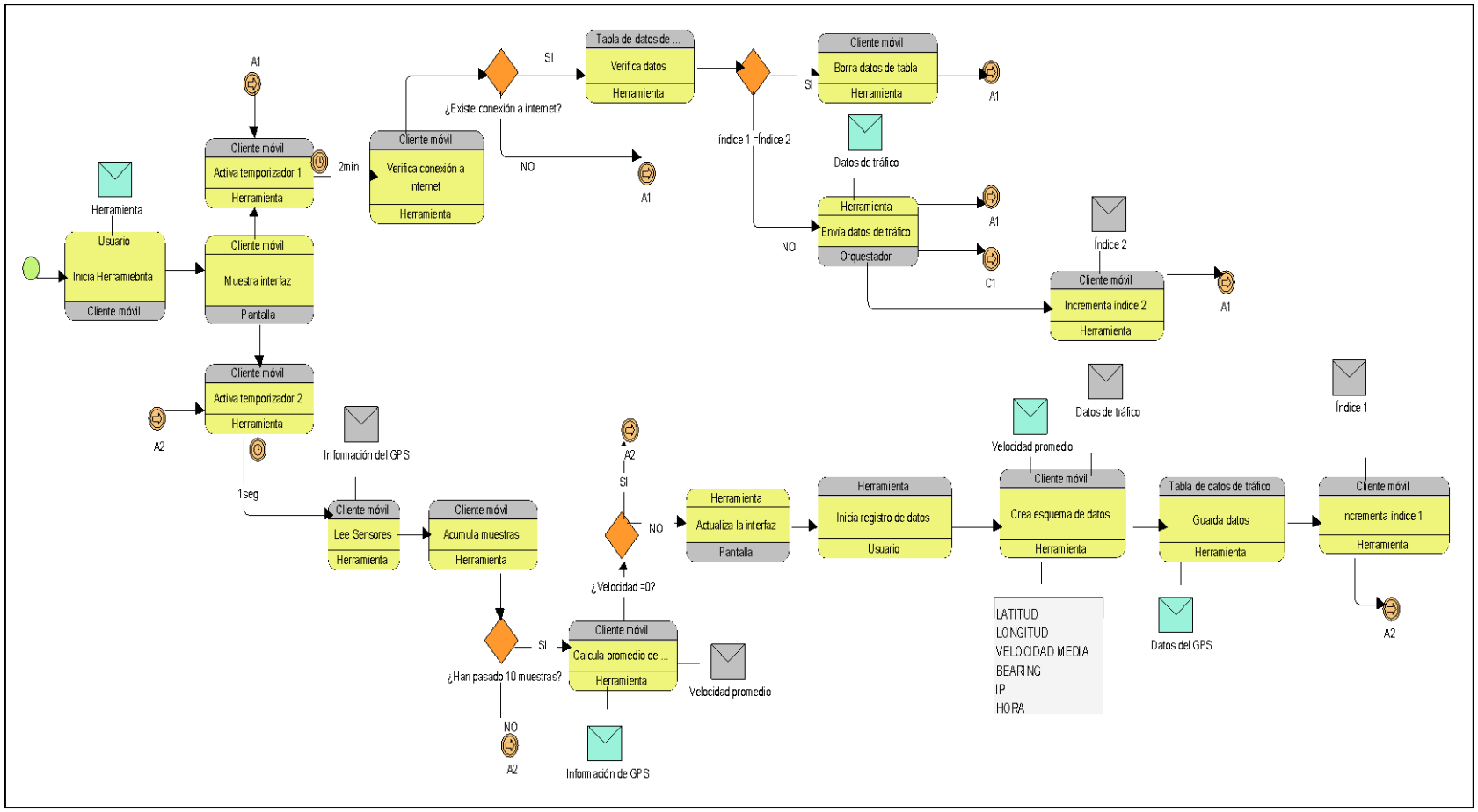



***Figura 30.*** Mapa BPMN de la adquisición de datos por medio de herramientas

En la Figura 30 se muestra un diagrama que presenta el flujo de acciones que tiene la herramienta de adquisición de datos, el diagrama describe lo siguiente; el usuario inicia la herramienta en el dispositivo, a continuación, se inicia tres acciones paralelamente:

- Mostrar interfaz: Dispositivo con pantalla inicial, mapa y botones inicio, login, detener.
- Activa temporizador 1: La herramienta activa temporizador 1 que se activa cada dos minutos en el dispositivo, la herramienta verifica conexión a internet en el dispositivo, si no existe conexión a internet se queda en un ciclo de verificación de internet cada 2 minutos. En cambio sí existe conexión a internet la herramienta verifica si existe datos en la tabla de datos de tráfico interna mediante la comprobación de un índice 1(índice en la tabla del orquestador) y un índice 2 (índice que se incrementa cuando envía datos al orquestador), si él índice 1= al índice 2 la herramienta borra datos de la tabla interna del dispositivo y regresa al proceso del temporizador 1, en caso que los índices sean diferentes la herramienta envía datos de tráfico al orquestador incrementa la herramienta al índice 2 y regresa al temporizador 1.
- Activa temporizador 2: La herramienta activa temporizador 2 cada 1 segundo, la herramienta lee los sensores del GPS en el dispositivo, posteriormente la herramienta acumula 10 muestras en el dispositivo si aún no pasa las diez muestras regresa al temporizador hasta acumular 10 muestras, a partir de las 10 muestras acumuladas la herramienta calcula un promedio de velocidad en el dispositivo si la velocidad promedio es igual a 0 regresa al temporizador 2 caso contrario, la herramienta actualiza la interfaz en la pantalla, el usuario inicia el registro de datos en la herramienta, la herramienta crea un esquema de datos en el dispositivo con la velocidad promedio, coordenadas

fecha, hora, *bearing*, ip, más adelante la herramienta guarda los datos en la tabla de datos de tráfico interna y la herramienta incrementa el índice 1 en el dispositivo por último el ciclo regresa al temporizador 2.

## Orquestador

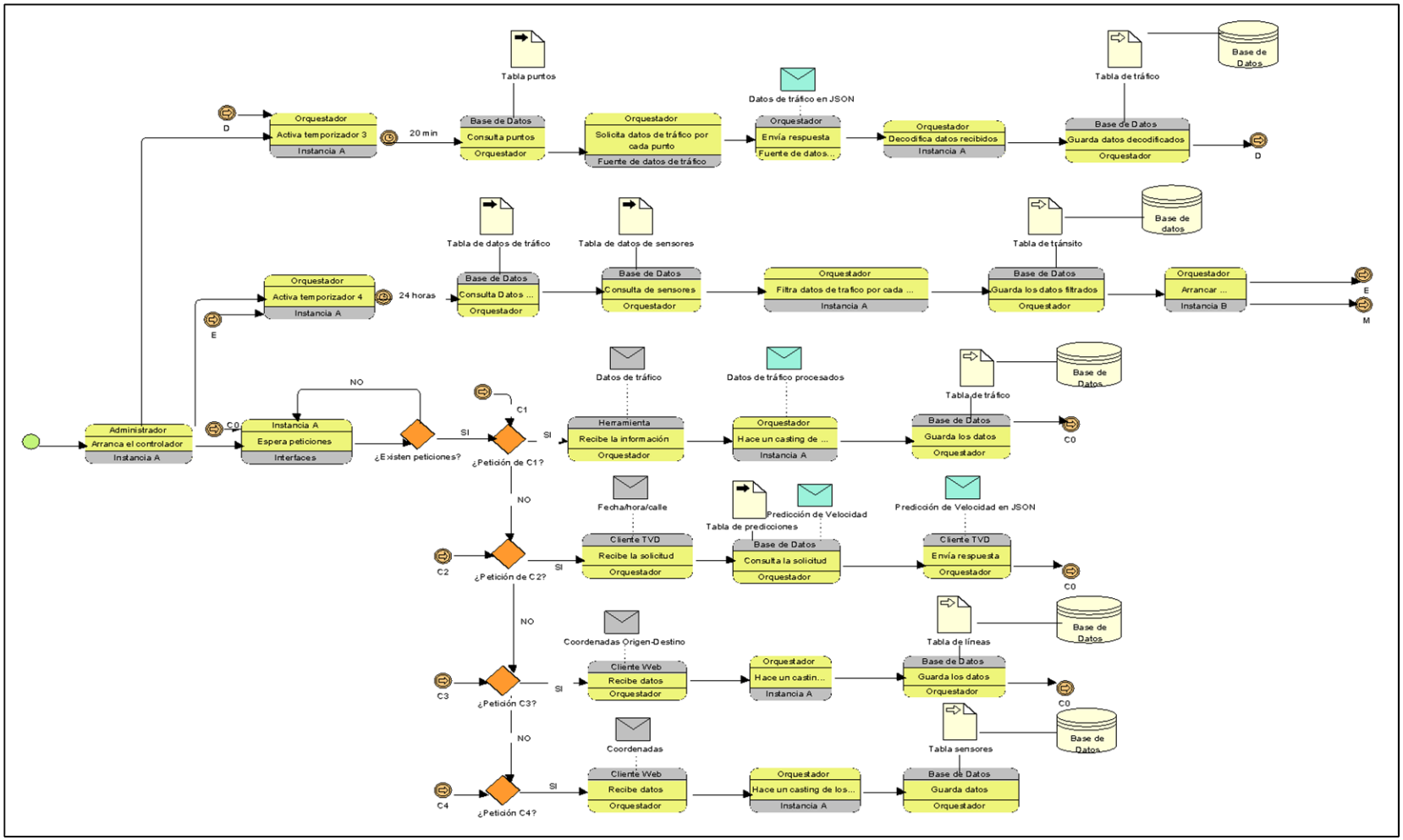



***Figura 31.*** Diagrama BPMN orquestador 1

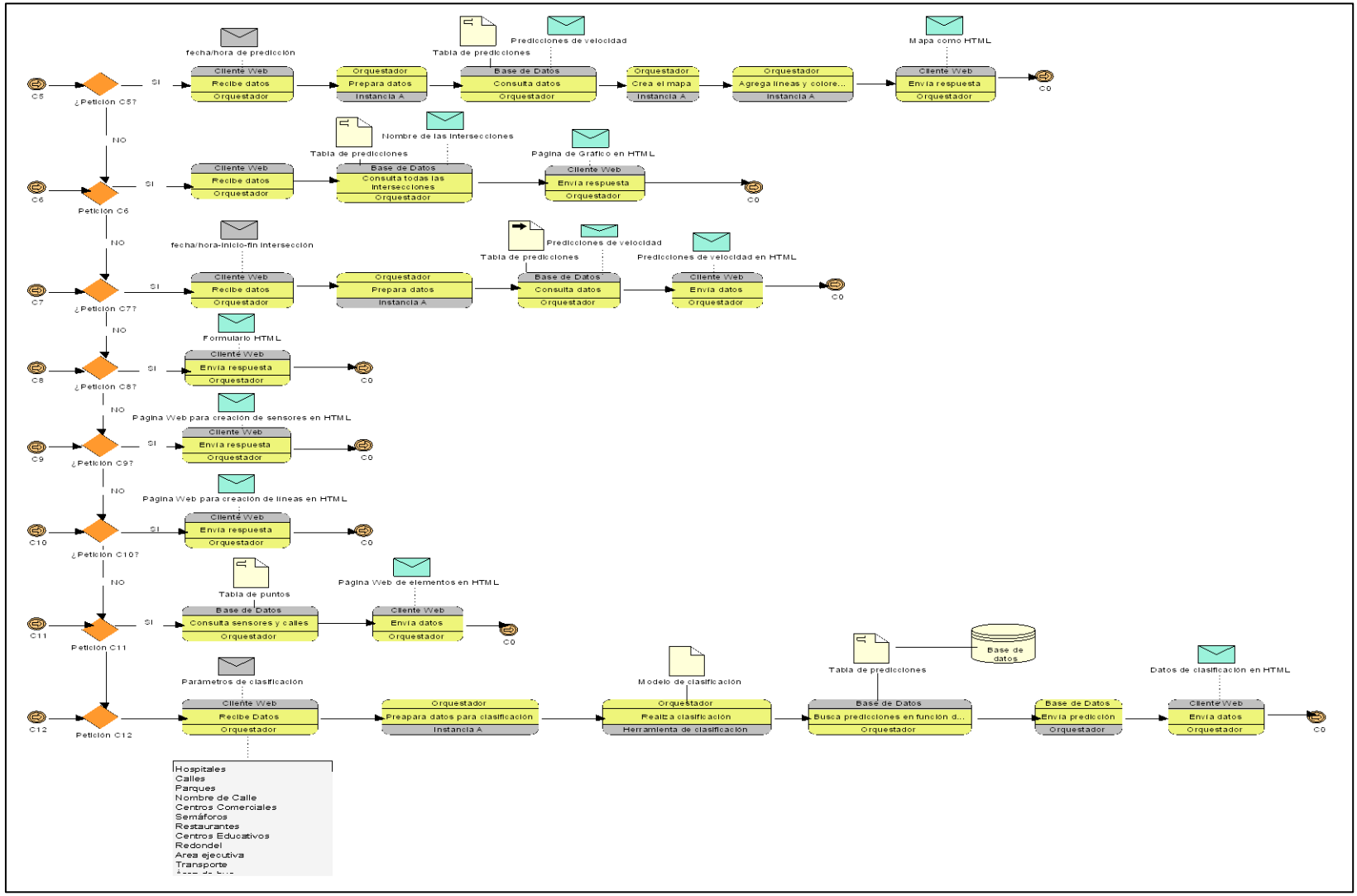



***Figura 32.*** Diagrama orquestador 2

En la Figura 31 y Figura 32 se tiene el proceso del orquestador a continuación el administrador arranca el proceso dele orquestador en la instancia ‘A’ (hardware en donde el orquestador realiza las funciones), posteriormente se inician 3 tareas paralelas:

El orquestador activa temporizador 3 en la instancia ‘A’ cada 20 min posteriormente el orquestador consulta puntos en la base de datos en la tabla puntos, luego el orquestador solicita datos de tráfico por cada punto en la fuente de datos de tráfico(servidor web de GPS), siguiente paso la fuente de datos envía respuesta como datos de tráfico al orquestador en formato JSON, el orquestador decodifica los datos de tráfico en la instancia ‘A’, por último el orquestador guarda datos decodificados en la base de datos en la tabla de tráfico y el ciclo se repite.

El orquestador activa el temporizador 4 en la instancia ‘A’ cada 24 horas, posteriormente el orquestador consulta datos en la tabla de datos de tráfico de la base de datos, también el orquestador consulta la tabla de sensores en la base de datos luego el orquestador filtra datos de tráfico por cada sensor en la instancia ‘A’, en el siguiente paso el orquestador guarda datos filtrados en la tabla de tránsito en la base de datos, por último el orquestador arranca la instancia ‘B’(hardware en donde se realizan las funciones de predicción ) el ciclo se repite y a la vez inicia el proceso de predicción.

La instancia ‘A’ espera peticiones de las interfaces, Existen 12 peticiones diferentes de las interfaces a continuación se enumera cada una de las peticiones.

- C1(Interfaz de herramienta de adquisición de datos): el orquestador recibe los datos de tráfico de la herramienta, el orquestador hace un casting de los datos en la instancia ‘A’ por último el orquestador guarda los datos en la tabla de tráfico en la base de datos y regresa a esperar otra petición de las interfaces.

- C2(Interfaz cliente TVD): el orquestador recibe la solicitud de fecha/hora/calle del cliente TVD, el orquestador consulta la tabla de predicciones en la base de datos y envía respuesta de predicción de velocidad en JSON al cliente TVD y regresa nuevamente a esperar peticiones de las interfaces.
- C3(Interfaz cliente Web): el orquestador recibe datos de coordenadas origen-destino del cliente web, el orquestador hace un casting de los datos en la instancia 'A', por último, el orquestador guarda los datos en la tabla de líneas en la base de datos y regresa a esperar peticiones de las interfaces.
- C4(Interfaz cliente Web): el orquestador recibe las coordenadas del aplicativo web, posteriormente el orquestador hace un casting en la instancia 'A', po ultimo el orquestador guarda los datos en la tabla de sensores en la base de datos.
- C5(Interfaz cliente Web): el orquestador recibe datos fecha/hora de predicción del cliente Web, luego el orquestador prepara los datos en la instancia 'A', como siguiente paso el orquestador consulta los datos en la tabla de predicciones de la base de datos y la base de datos devuelve las predicciones de velocidad al orquestador el cual crea el mapa en la instancia 'A', adicional el orquestador se agrega líneas y colores al mapa en la instancia 'A', por último el orquestador envía el mapa como HTML siendo esta la página principal de la interfaz del aplicativo web, a continuación nuevamente espera peticiones de las interfaces.
- C6(Interfaz cliente Web): el orquestador recibe datos del cliente web, luego el orquestador consulta todas las intersecciones de las cuales hay información en la tabla de predicciones de la base de datos y devuelve el nombre de la intersecciones al

orquestador, por último en este proceso el orquestador dibuja la página de gráfico en HTML en el cliente web, es decir se muestra la página de gráficos con las intersecciones de las calles después de o sigue a C7 o espera una nueva petición de alguna interfaz.

- C7(Interfaz cliente web): el orquestador recibe la fecha y hora de inicio/fecha y hora final e intersecciones enviados por el cliente web, a continuación el orquestador prepara los datos en la instancia ‘A’, el orquestador consulta los datos en la tabla de predicciones en la base de datos, y devuelve la base de datos las predicciones de velocidad por último el orquestador envía predicciones de velocidad en HTML al cliente web y el ciclo se repite el orquestador espera nuevamente peticiones de las interfaces.
- C8(Interfaz del aplicativo web): Si existe una petición C8 el orquestador envía al cliente web el Formulario HTML para el ingreso de datos de C12 y regresa a esperar una nueva solicitud de las interfaces.
- C9(Interfaz cliente web) Si existe una petición de C9 el orquestador envía página web para creación de sensores en HTML al cliente web y regresa a esperar solicitudes de las interfaces C0.
- C10(Interfaz cliente web): Si existe petición de C10 el orquestador envía la página web para la creación de líneas en HTML luego regresa a esperar solicitudes de las interfaces C0.
- C11(Interfaz cliente web): el orquestador consulta sensores y líneas en la tabla de puntos de la base de datos y el orquestador envía la página web de elementos en HTML al cliente web luego regresa a esperar solicitudes de las interfaces C0.

- C12(Interfaz cliente web) : el orquestador recibe los parámetros de clasificación del cliente web(hospitales, calles, parques, nombre de calle, centros comerciales, semáforos, restaurantes, centros educativos, redondel, área ejecutiva, transporte, área de bus.), luego el orquestador prepara los datos en la instancia ‘A’, posteriormente el orquestador realiza el modelo de clasificación en la herramienta para la clasificación, el orquestador busca predicciones en función de la clasificación en la tabla de predicciones en la base de datos,, a continuación el orquestador envía los datos de clasificación al cliente web y regresa a esperar solicitudes de las interfaces C0.

**Machine Learning**

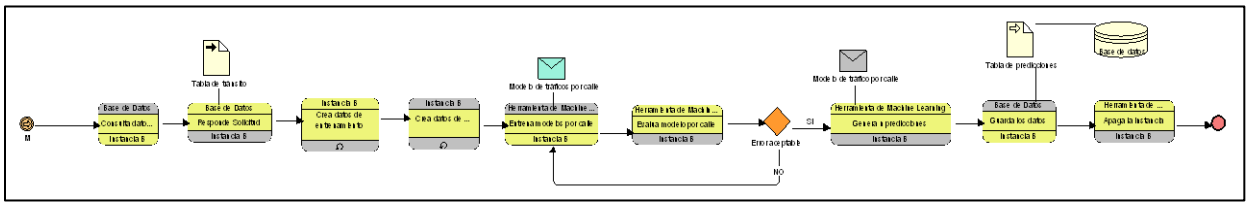



***Figura 33.*** Mapa BPMN de la herramienta de predicción

En la Figura 33, se muestra el proceso de la instancia ‘B’ (máquina de predicción). La instancia ‘B’ consulta los datos por calle en la tabla de tránsito de la base de datos, la instancia ‘B’ crea datos de entrenamiento en la misma instancia ‘B’, la misma instancia ‘B’ crea datos de evaluación, La instancia ‘B’ entrega el modelo de tráfico por calle, la herramienta de *machine learning* evalúa el modelo por calle en la instancia ‘B’ si no existe un error aceptable regresa nuevamente a entrenar el modelo por calle caso contrario la herramienta de *machine learning* toma el modelo y realiza las predicciones en la instancia ‘B’, por último la instancia ‘B’ envía los datos a la tabla de predicciones en la base de datos y apaga la instancia ‘B’.

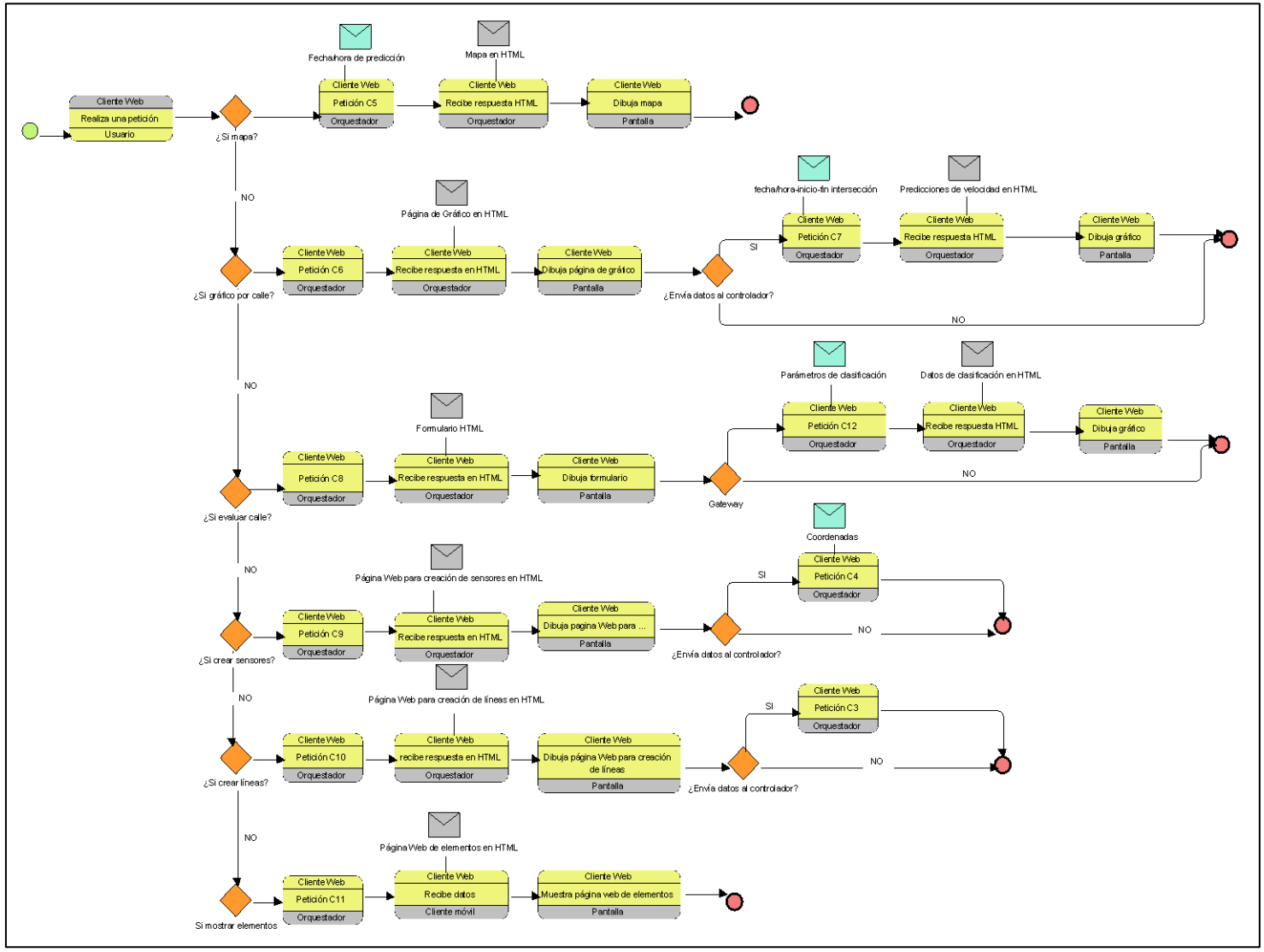



***Figura 34.*** Mapa BPMN del cliente Web

**Cliente Web**

En la Figura 34 indica el proceso del aplicativo Web, el usuario realiza una petición del aplicativo web, existen 6 casos diferentes:

- Si la petición es visualizar el mapa o la página principal el aplicativo Web envía una petición C5 con la fecha/hora de predicción al orquestador en caso que el usuario no haya enviado fecha la página actualizará con la fecha y hora actual, el aplicativo web recibe el mapa HTML del orquestador y el aplicativo web dibuja el mapa en la pantalla, se finaliza el proceso.
- Si el usuario desea la opción gráfico por calle: El aplicativo Web hace una petición C6 al orquestador luego el aplicativo web recibe página de grafico en HTML del orquestador, posteriormente el aplicativo web dibuja la página en la pantalla, posteriormente si la aplicación no envía datos al orquestador se finaliza el proceso, caso contrario el aplicativo web envía hace una petición C7 al orquestador con los datos de fecha/hora inicio-fin e intersección y el orquestador envía predicciones de velocidad en HTML al aplicativo web por último el aplicativo web dibuja gráfico en el aplicativo web y el proceso finaliza.
- Si el usuario desea evaluar calle el aplicativo web hace una petición C8 al orquestador, el orquestador envía la página formulario al aplicativo web, por último el aplicativo web dibuja la página de formulario en la pantalla, si el aplicativo web no envía datos al orquestador el proceso finaliza, caso contrario el aplicativo web envía petición C12 con los parámetros de clasificación al orquestador y el aplicativo web recibe respuesta en HTML de predicciones de velocidad del orquestador por ultimo dibuja la gráfica y el proceso finaliza.

- Si el administrador desea crear sensores el aplicativo web realiza una petición C9 al orquestador y el aplicativo web recibe la página web para creación de sensores en HTML y el aplicativo web dibuja la página de sensores en la pantalla, si no se envía datos al orquestador se finaliza el proceso, caso contrario el aplicativo web envía una petición C4 al orquestador con las coordenadas de los sensores y el proceso finaliza.
- Si el administrador desea crear líneas, el aplicativo web realiza una petición C10 al orquestador, el aplicativo web recibe la página para la creación de líneas en HTML y dibuja en la pantalla la página de creación de líneas si no envían datos al orquestador finaliza el proceso caso contrario el aplicativo web hace una petición C3 al orquestador con las coordenadas de las líneas y el proceso finaliza.
- Si el administrador desea mostrar elementos el aplicativo web hace una petición C11 al orquestador, el aplicativo web recibe la página de los elementos que envía el orquestador, por último, el aplicativo web muestra la página con los elementos y finaliza el proceso.

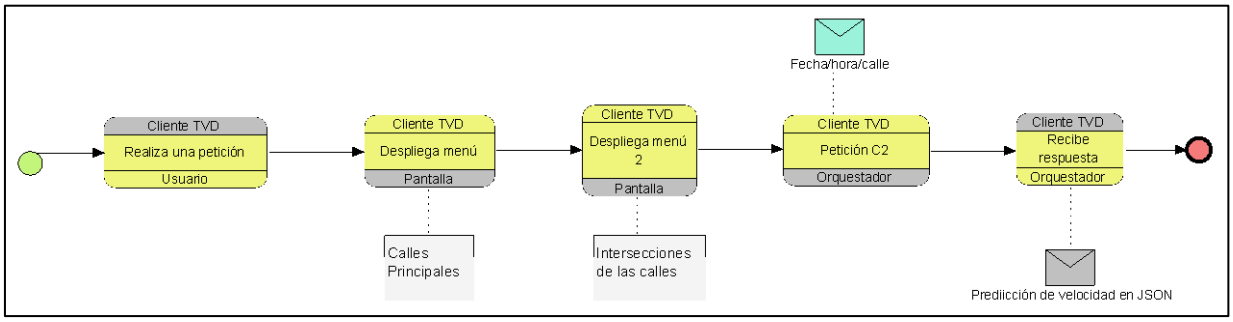



***Figura 35.*** Mapa BPMN del cliente TVD

**Cliente TVD**

La Figura 35 muestra el proceso que se ha diseñado para la parte de TVD, como se puede observar al cliente TVD se le va a permitir realizar una petición y cuando la realice se debe

desplegar un menú con las opciones de las calles que podría elegir. Una vez que se escoja la calle el siguiente paso es que le aparezca las intersecciones propias que conforman a la misma, además, debe contar con un campo de introducción de texto con la información que se quiere consultar al servidor para la predicción en este caso fecha, hora y número de intersección. Es por consiguiente que se recibe la respuesta del servidor de la velocidad en JSON y esta información es la que se debe mostrar en pantalla al cliente de TVD.

**Metamodelo**

A continuación, se procede con el diseño del metamodelo luego del análisis de cada uno de los componentes y definiciones en la Figura 36 se puede ver el resultado final.

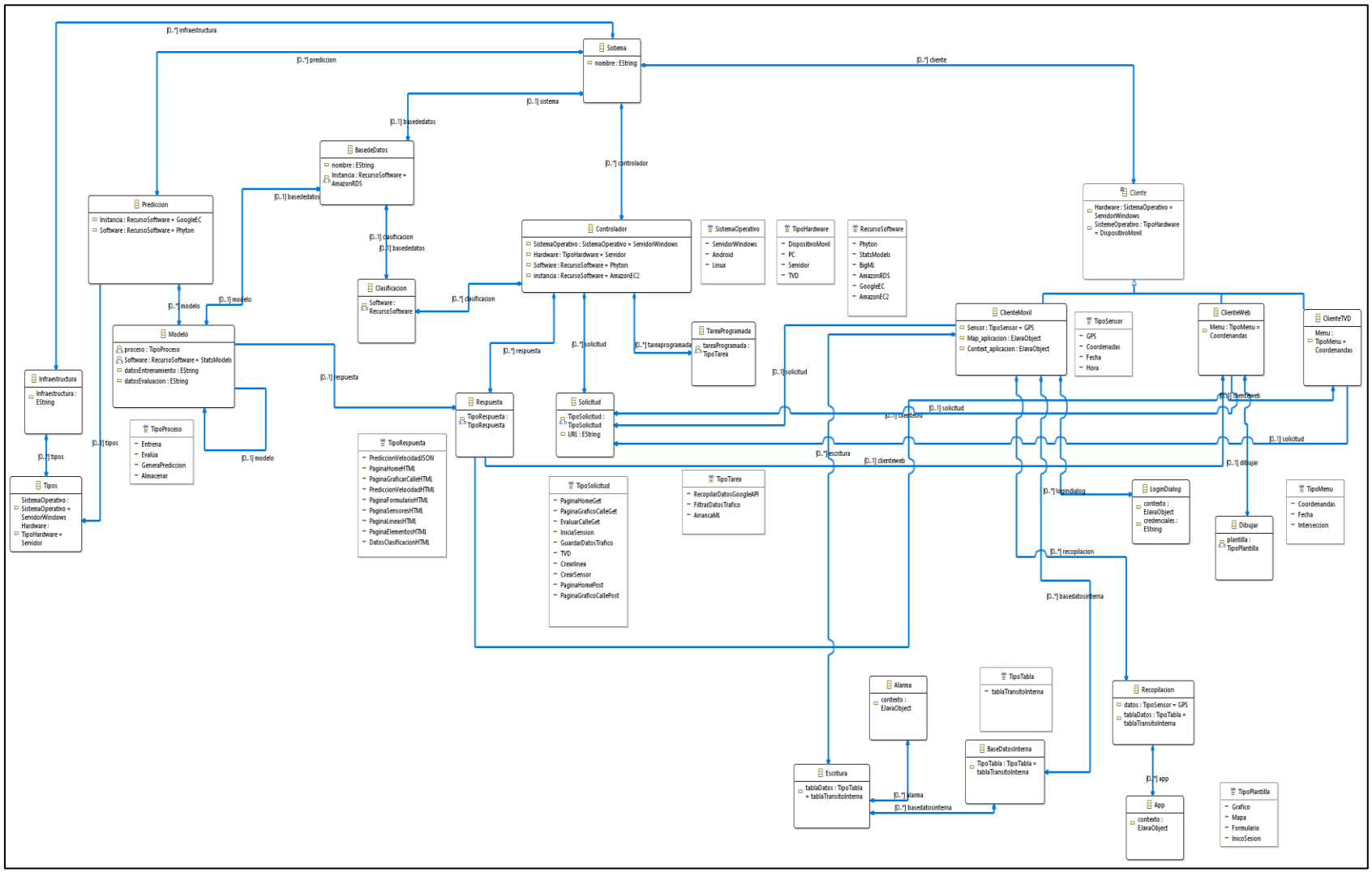



***Figura 36.*** Metamodelo

# CAPÍTULO 4

# IMPLEMENTACIÓN DE LA ARQUITECTURA

Como se ha explicado en capítulos anteriores, la creación de una plataforma que permita ejecutar tareas de aprendizaje automático conlleva una seria de tareas que son detallas en el presente capítulo. El componente principal en cualquier sistema que implemente tareas de aprendizaje automático son los datos**.** Por lo tanto, el primer paso es obtener datos que permitan entrenar un modelo. Para recolectar dichos datos se necesita espacio de alojamiento y herramientas que nos permitan obtenerlos. Estas herramientas son la aplicación de Android y el servicio de adquisición de datos de tráfico desde GDM-API. Ambas soluciones utilizan los microservicios de almacenamiento de la API de la arquitectura para almacenar la información de tráfico en una base de datos alojada en Amazon RDS.

Una vez obtenidos los datos suficientes para cumplir con los requisitos de diseño, es necesario filtrar y preparar los datos para el entrenamiento. Esta preparación dependerá de la herramienta que se haya escogido para el *machine learning*, en el caso particular de la plataforma la preparación de datos verificara tipo de datos y estructura de los mismos, ya que las tareas de adquisición fueron previas al desarrollo del código para entrenamiento, es necesario adaptar la estructura de los datos almacenados en la base de datos en la nube con los requisitos de la herramienta *Statsmodels* utilizada para el entrenamiento de los modelos.

Con datos preparados para el aprendizaje automático se procede con el entrenamiento, dicho entrenamiento entrega un modelo que es usado para predecir el comportamiento de los datos en el futuro. Con las predicciones obtenidas (que deben cumplir con un criterio de precisión) se procede

a almacenar las predicciones en una tabla de la base de datos de la nube para su consulta y posterior presentación mediante interfaces de usuario.

El usuario posee dos interfaces disponibles para aprovechar los servicios de predicción de la plataforma. En primer lugar, se ha construido una interfaz web compatible con navegadores de dispositivos móviles y de escritorio. Esta interfaz permite al usuario utilizar todos los servicios de la plataforma: revisión de mapa en tiempo real con el tráfico de la ciudad, predicción de tráfico por cada calle registrada en el sistema, clasificación y predicción de una calle según sus características. Además, como administrador es posible crear nuevos sensores virtuales donde se requiera conocer el tráfico y su predicción.

En segundo lugar, se ha creado una aplicación de televisión digital que permite al cliente ver siempre desplegado un botón de ayuda para cuando desee accionarlo y a continuación, aparezca un menú con las calles que se puede consultar y realizar una predicción con la fecha y hora que se requiera. También debe contener una opción en la cual se pueda llenar con la información que se quiera consultar y que pueda ser ingresada y modificable con las teclas que cuenta el control remoto. La información que se presente al usuario en pantalla debe ser resumida y entendible.

**Tareas realizadas mediante el cliente de Android**

Como se ha mencionado previamente para la obtención de un modelo mediante aprendizaje automático, es necesario haber recolectado una cantidad de datos suficientes. Para este propósito la plataforma posee dos soluciones de software que permiten cumplir con la tarea de recolección de datos, la aplicación de Android *Datos de Trafico* y un servicio de recolección de datos desde GDM – API.

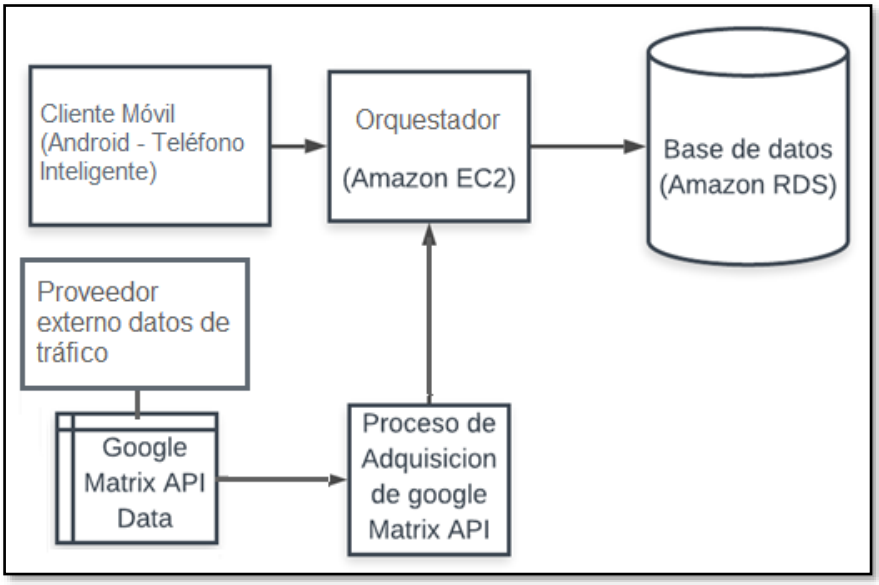



***Figura 37.*** Procesos para la adquisición de datos de tráfico vehicular

En capítulos anteriores se ha mencionado que se aprovechan dos fuentes de datos, los servidores de Google que contienen datos de tráfico que son accesibles mediante la *Distance Matrix API* de Google Maps y la aplicación de Android que recopila datos de geolocalización utilizando el GPS de un teléfono inteligente. Todos los datos son recibidos y procesados por el servidor principal que posteriormente guarda los datos en una base de datos persistente.

**Adquisición mediante la aplicación para Android.**

Como se ha descrito previamente una de las herramientas que permiten la obtención de datos de tráfico es la aplicación para Android *Datos de Trafico*. Dicha aplicación ha sido construida utilizando el entorno de desarrollo Android Studio IDE, el mismo permite agregar librerías y complementos de Google (y de terceros) fácilmente, ya la herramienta está enfocada en la creación de aplicaciones para el sistema operativo Android, el lenguaje de programación utilizado es Java. Sin embargo, también es posible utilizar el lenguaje de programación *Kotlin*.

**Diagrama de flujo de la aplicación de Datos de Tráfico**

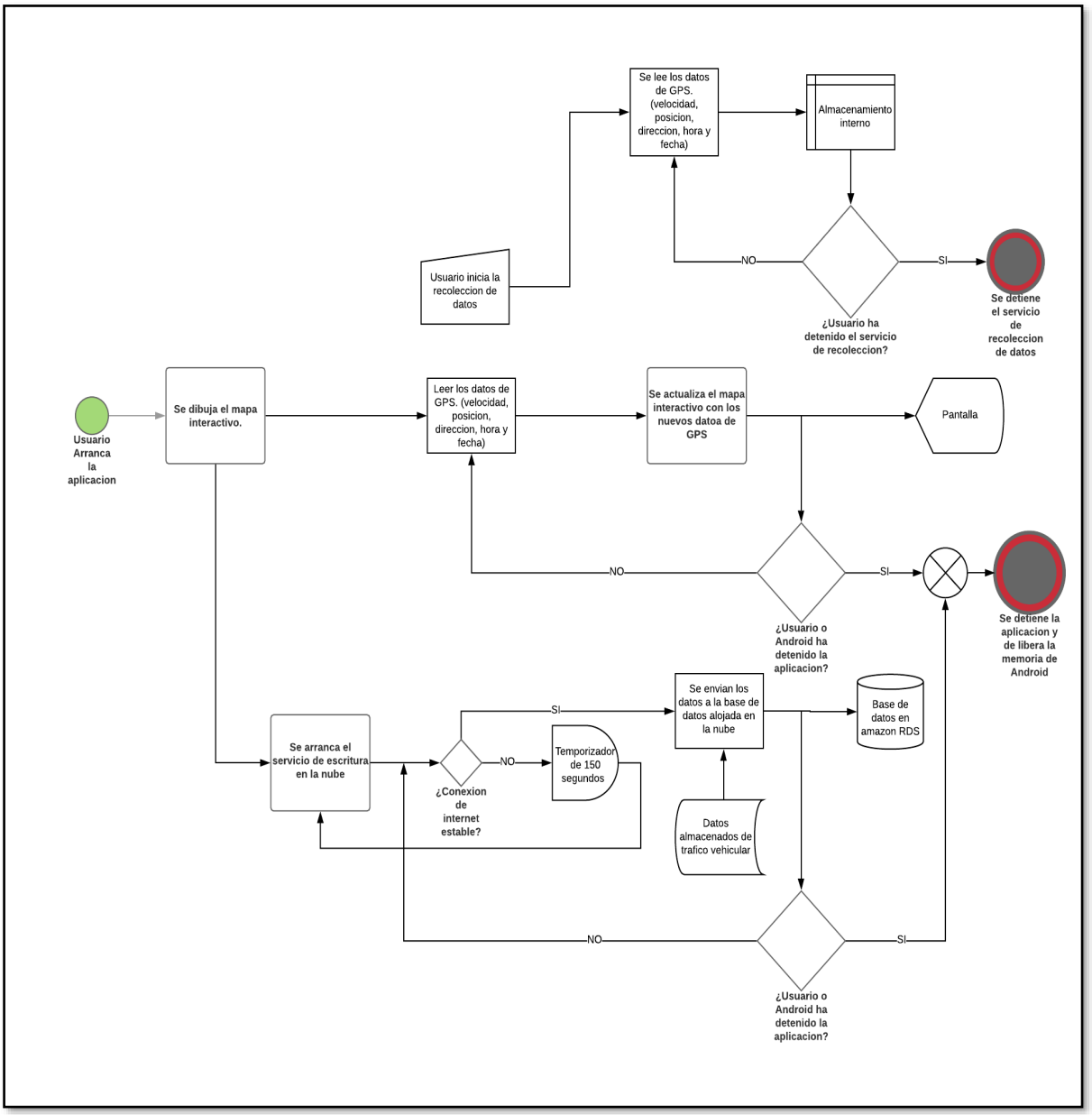



***Figura 38.*** Diagrama de flujo del cliente móvil “Tráfico Vehicular”

El diagrama de flujo de la Figura 38 describe cómo funciona la aplicación Datos de tráfico. Al arrancar la aplicación el sistema operativo renderiza una plantilla principal que contiene una pantalla con un mapa interactivo, varios espacios de texto y botones que permiten la interacción

con los servicios de la aplicación. Paralelamente al renderizado de pantallas, se inicia un servicio que verifica si existen conexión a internet y además comprueba si todas las muestras de tráfico almacenadas en la base de datos interna hayan sido escritas en la base de datos de la nube. De esta forma se asegura que todos los datos albergados internamente en la aplicación sean guardados en la nube.

Al momento de iniciar la aplicación se inicializan los servicios de localización del dispositivo, es decir se configura con qué frecuencia el GPS reporta la ubicación (latitud y longitud), velocidad, altitud, dirección, etc. Utilizando esta información es posible actualizar el mapa que se muestra en la pantalla del dispositivo y se puede mostrar información relevante al usuario. En la pantalla principal de la aplicación se tiene dos botones que permiten iniciar o detener el registro de los datos de tráfico.

Al iniciar el servicio de recopilación de datos, un segundo servicio que registra la información de tráfico vehicular y la almacena en la base de datos interna de la aplicación es lanzado en el dispositivo. Este servicio solo puede ser detenido por el usuario mediante el botón detener de la aplicación.

En cada arranque o parada de la aplicación se almacenan datos, credenciales o información relevante para el correcto funcionamiento de la aplicación mediante la utilización del fichero de referencias compartidas.

**Servicios y procesos de la aplicación**

Debido a la necesidad de tomar datos de velocidad utilizando los sensores de un teléfono inteligente, se ha creado un servicio dentro de la aplicación que permite acceder a la información

de ubicación y otros recursos del sistema. A continuación, se describen los servicios y procesos que la aplicación ejecuta para cumplir con la recopilación de datos de tráfico.

a. Servicio para la recolección de datos de geolocalización

La aplicación para Android posee una clase llamada *LocationService.java* que extiende los métodos de la clase *android.app.Service*. Además utiliza las funcionalidades de la clase *android.location.LocationManager* que permite recuperar los datos de ubicación GPS de un dispositivo mediante los servicios de localización de Android. Para utilizar esta clase es necesario configurar ciertos parámetros como, por ejemplo: el tiempo mínimo o la distancia entre muestras de una nueva posición.

Según los parámetros configurados el sistema Android genera un evento cada cierto tiempo o cada cierta distancia, para acceder a esta información es necesario crear un *android.location.LocationListener* que tiene como parámetro de inicialización un método *OnLocationChanged* en donde se reciben todos los parámetros de localización del dispositivo. Al recibir este evento con una nueva posición, se procede a guardar la información relevante en la base de datos embebida SQLite mediante la utilización de la clase *SQLiteDatabase.* Si la información fue correctamente guardada el sistema devuelve un valor entero que representa el número de filas que están utilizándose en la base de datos. Esta información es importante ya que mediante este valor se puede conocer cuántos datos necesitan aún ser guardados en el servidor.

La clase también posee un método *OnStartCommand* que se ejecuta al iniciar el servicio, este método es el encargado de crear una notificación persistente para informar al usuario que un servicio que recopila información de ubicación está activo. La creación de la notificación persistente es posible al utilizar la clase *android.location.Notification.*

b. Servicio para el almacenamiento de datos de geolocalización en la nube

Este servicio permite extraer los datos guardados en la base de datos *android.database.sqlite.SQLiteDatabase* que se guardan en mediante el servicio *LocationService.* La clase *WriteToCloudService.java* extiende los métodos de la clase *android.app.Service* que permite crear procesos en primer o segundo plano que se ejecutan sin necesidad de una interfaz gráfica de usuario, sin embargo para respetar la decisión de un usuario al cerrar una aplicación, se ha decidido que si el usuario no tiene la aplicación abierto el este servicio no se ejecutara.

Esta clase utiliza también las funcionalidades de la clase *android.net.ConnectivityManager* para determinar si el dispositivo Android posee una conexión de internet. Finalmente se utilizan los servicios de las clases *java.net.HttpURLConnection* y *java.net.URL* para escribir los datos utilizando los endpoints creadas para la *API-REST.*

Esta clase posee un método *OnCreate()* que se encarga de recuperar los valores de la base de datos del usuario que usa la aplicación, así como también el número de filas datos escritos en el servidor y el número de filas que aún quedan por escribirse.

Finalmente, debido a la necesidad de verificar constantemente por nuevos datos de ubicación guardados. Se ha creado un método *onConnectionAvailable()* que permite ejecutar una verificación de dicha información si existe una conexión estable de internet.

a. Proceso principal para la renderización de la aplicación

Esta clase es la encargada de dibujar la interfaz de usuario mediante la renderización de fragmentos utilizando para ello las librerías *android.support.v4.app.ActivityCompat* y *android.support.v4.app.FragmentActivity.* Además debido a que parte de la interfaz de usuario

posee un mapa interactivo se ha agregado también la librería *com.google.android.gms.maps* que cuenta con todas las herramientas para renderizar y manipular mapas en aplicaciones Android.

Esta clase además es encargada de iniciar el servicio de escritura de información de ubicación en la base de datos alojada en la nube, además crea los botones para el inicio del servicio de localización. Finalmente, al arrancar la aplicación o cerrarla se ejecutan los métodos *onCreate()* y *onDestroy()* para recuperar y guardas respectivamente datos de usuario.

b. Proceso para el despliegue de ventanas de dialogo en la aplicación

La clase LoginDialog.java permite la renderización de un *fragment* que contiene una ventana con varios campos para que el usuario pueda ingresar sus datos personales. Esto permite que de ser necesario se pueda encontrar las muestras que contiene la ubicación de un usuario en específico.

Al momento de presionar en aceptar la clase llama a un método *onClick()* que envía los datos actualizados para que dicha información sea almacenada en el fichero de *SharedPreferences*. Dichos datos son leídos cuando se arranca el proceso principal y son guardados cuando el usuario Cierra la aplicación.

c. Componente para crear la estructura de la base de datos

Ya que los datos de ubicación son guardados continuamente cuando el servicio *LocationService* se encuentre activo, puede ser el caso que no exista una conexión estable de internet, es por ello por lo que antes de enviar los datos a la base de datos estos son almacenados temporalmente. Para poder manejar el esquema de la base de datos se ha desarrollado una clase que aprovecha las funcionalidades de la clase TrafficDB.java. Con ello se logra crear un objeto tipo SQLite y acceder y manejar los datos eficientemente.

Los datos, sus unidades y tipos corresponden con el esquema de datos de tráfico presentado anteriormente en el capítulo de diseño en la arquitectura.

d. Componente para la creación de notificaciones permanentes

De acuerdo con lo señalado previamente, el uso de servicios en primer plano conlleva la visualización de una notificación permanente para el usuario. Dicha notificación es lanzada al momento de la activación del servicio *LocationService*, sin embargo, para que esto puede ser realizado es necesaria la creación de una clase base que permite recibir un objeto tipo *Context* para al momento de lanzar la notificación se entienda con que proceso está relacionada.

e. Componente para salir del modo reposo periódicamente

El Sistema operativo Android posee un conjunto de herramientas y procesos encaminados en aumentar la autonomía del dispositivo. Es debido a esto que, algunos procesos son detenidos automáticamente según el criterio del Sistema operativo. Para solventar este inconveniente es necesario despertar periódicamente ciertas tareas. En el caso de la aplicación, existe un temporizador que levanta los procesos de escritura y comprobación de la base de datos interna cada dos minutos, si existe una conexión activa estable de internet y de datos que no hayan sido enviados a la *nube* se procede a escribirlos uno por uno.

La comprobación periódica se logra creando un reloj que levanta una alarma en el procesador permitiendo la ejecución de la tarea. Una vez que la tarea está completa, el CPU del dispositivo regresa al estado de reposo.

f. Componente para almacenar información de geolocalización dentro de la base interna

Para poder manejar el almacenamiento de datos de velocidad se ha creado una clase *Sample.java* que facilita la tarea de manejo de datos mediante constructores y del *Context* de la aplicación.

Además, se han creado métodos que permiten a la clase manejar eficiente los datos de tráfico según necesidad de la aplicación y de la plataforma.

**Clases creadas para la aplicación**

La aplicación posee diferentes librerías y complementos para permiten que los algoritmos desarrollados funcionen adecuadamente, a continuación, se presenta una tabla con la descripción de las clases que se ha desarrollado para la aplicación de Android.

Tabla 26.

*Clases creadas para el cliente móvil.*

| Librería | Descripción |
|---|---|
| *Alarm.java* | Permite a la aplicación despertar al dispositivo cada cierto tiempo para verificar si existen nuevos datos almacenados en la base de datos interna de la aplicación. |
| *App.java* | Permite a la aplicación crear notificaciones persistentes, los métodos de esta clase sirven para notificar al usuario si el servicio *LocationService* se encuentra activo. |
| *LocationService.java* | Permite recibir los datos de geolocalización del GPS del dispositivo y almacenarlos en la base de datos interna. |
| *LoginDialog.java* | Permite crear notificaciones emergentes para el configurar los datos de la persona que utiliza la aplicación. |
| *MapsActivity.java* | Es la clase principal que dibuja la aplicación y actualiza el mapa interactivo con los datos provenientes del GPS. Permite renderizar los menús botones y mapas que permiten la interacción con el usuario. |
| *Sample.java* | Consiste en una estructura de datos para el manejo de los datos de tráfico vehicular, es necesaria además para la escritura y consulta de la base de datos interna. |
| *WriteCloudService.java* | Permite recuperar la información de la base de datos interna y enviarla al servido de Base de datos alojado en la nube. |

A continuación, se muestra el diagrama UML y la interacción entre las clases de la aplicación de datos de tráfico.


***Figura 39.*** Diagrama UML de clases de la aplicación Datos de Trafico

**Mockups creados para la aplicación de Android**

A continuación, se presenta una serie de plantillas que muestran los diferentes elementos que el sistema operativo renderiza en la pantalla del dispositivo para que el usuario pueda interactuar con los servicios de la aplicación.

a. Presentación de las pantallas de la aplicación *Datos de tráfico*

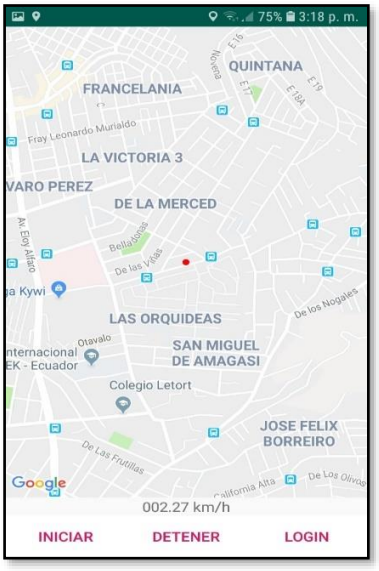



***Figura 40.*** Mockups que Android renderiza como pantalla principal en la aplicación Datos de Trafico

Se muestra una pantalla que contiene un mapa interactivo encapsulado en un objeto tipo fragmento de pantalla, además en la parte inferior contiene varios botones que permiten iniciar la toma de datos y detenerla. En el lado inferior derecho se muestra además un botón que permite incluir datos de la persona que está utilizando la aplicación.


***Figura 41.*** Presentación del fragmento de dialogo para ingresar datos

b. Presentación de las pantallas de la aplicación de *Sensores de Tráfico*

Las siguientes plantillas se utilizan para la aplicación de creación de sensores virtuales, su utilidad es similar a la interfaz del cliente web que se describe en la sección 0.a más adelante.


***Figura 42.*** Interfaces creadas para la aplicación de Sensores de Tráfico

**Tareas realizadas por el orquestador principal alojado en Amazon AWS**

El orquestador principal de la plataforma se describe como una instancia de computación en la nube que se ejecuta en servidores de Amazon AWS y realiza las funciones programadas en el código fuente principal.

Las tareas de las que se encarga el servidor son almacenar datos provenientes de los clientes, ejecutar y enviar respuestas a los diferentes clientes, crear tablas de datos de tráfico y preparar los datos previos a los procesos de entrenamiento automático. Además, se encarga periódicamente de extraer datos de trafico de los servidores de Google y de iniciar la tarea del entrenamiento

automático de los modelos de intersección (no se encarga de la ejecución del entrenamiento ya que esto se realiza sobre los servidores de Google Cloud Platform).

Las interacciones entre los orquestador, instancias y bases de datos, así como la interacción con los clientes y servicios adicionales se describen en el siguiente diagrama.

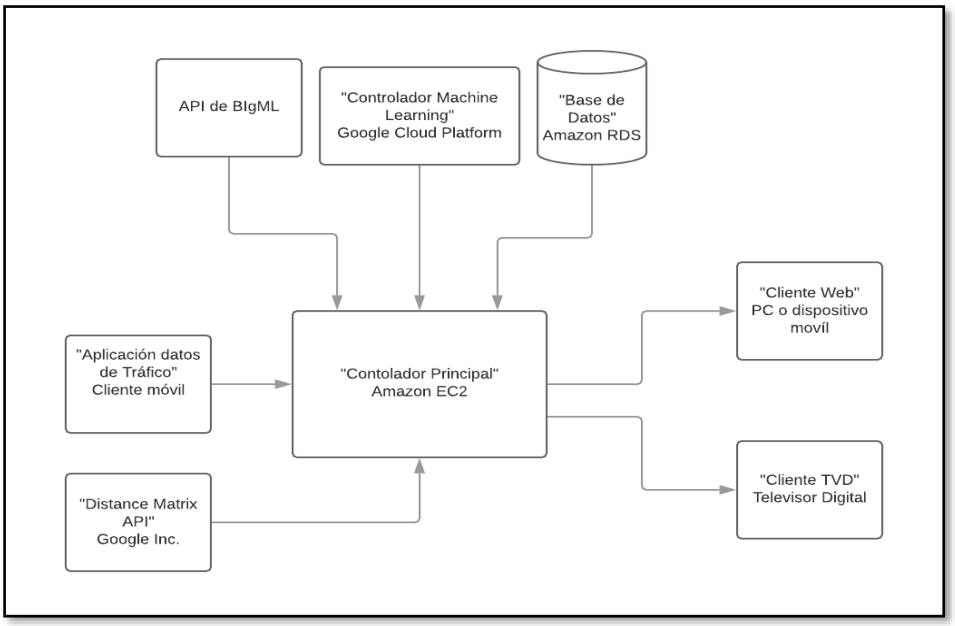



***Figura 43.*** Interacción entre entidades de la plataforma

El orquestador alimenta la base de datos con información obtenida mediante la aplicación de Android y los datos obtenidos mediante la API Distance Matrix de Google.

Los clientes emiten peticiones y el orquestador responde con la información solicitada a modo de archivos HTML o respuestas en formato JSON.

La instancia de Google Cloud Platform es utilizada para entrenar modelos de intersecciones y provee de datos de predicción al orquestador para que este los almacene en la base de datos.

El recurso de BigML es utilizado para clasificación de calles, el orquestador utiliza la API de BigML para generar solicitudes que son manejadas y presentadas al usuario por medio del orquestador.

Todos los datos de tráfico, predicciones, intersecciones, configuraciones y demás son almacenados en la instancia de Amazon RDS.

La lógica de programación del orquestador ha sido pensada para que pueda ejecutarse independientemente del sistema operativo o hardware sobre el lenguaje de programación Python. Esto permite que el orquestador puede ser migrado con facilidad a diferentes plataformas y mejora la escalabilidad y repetibilidad de sus funciones.

La plataforma ha sido pensada como un conjunto de entidades distribuidas, donde cada instancia o recurso realiza una tarea en un momento determinado. Esto permite que el orquestador no se sature con tareas, además permite que no se desperdicien recursos innecesarios cuando no se está llevando a cabo una tarea que demande grandes capacidades de procesamiento.

El código principal que ejecuta el orquestador ha sido creado tomando en cuenta las practicas comunes en el mundo de la producción de software. Ha sido modificado y adaptado para que puede ser distribuido fácilmente y su implementación en otros servidores sea más entendible para los desarrolladores. Además, en toda la plataforma se ha implementado el uso de un repositorio que permite el control y corrección del código principal de todas las aplicaciones que interactúan con el orquestador principal.

A continuación, se presenta el diagrama de relación de clases del código principal luego de la adaptación que se realiza para distribución de aplicación escritas en Python.

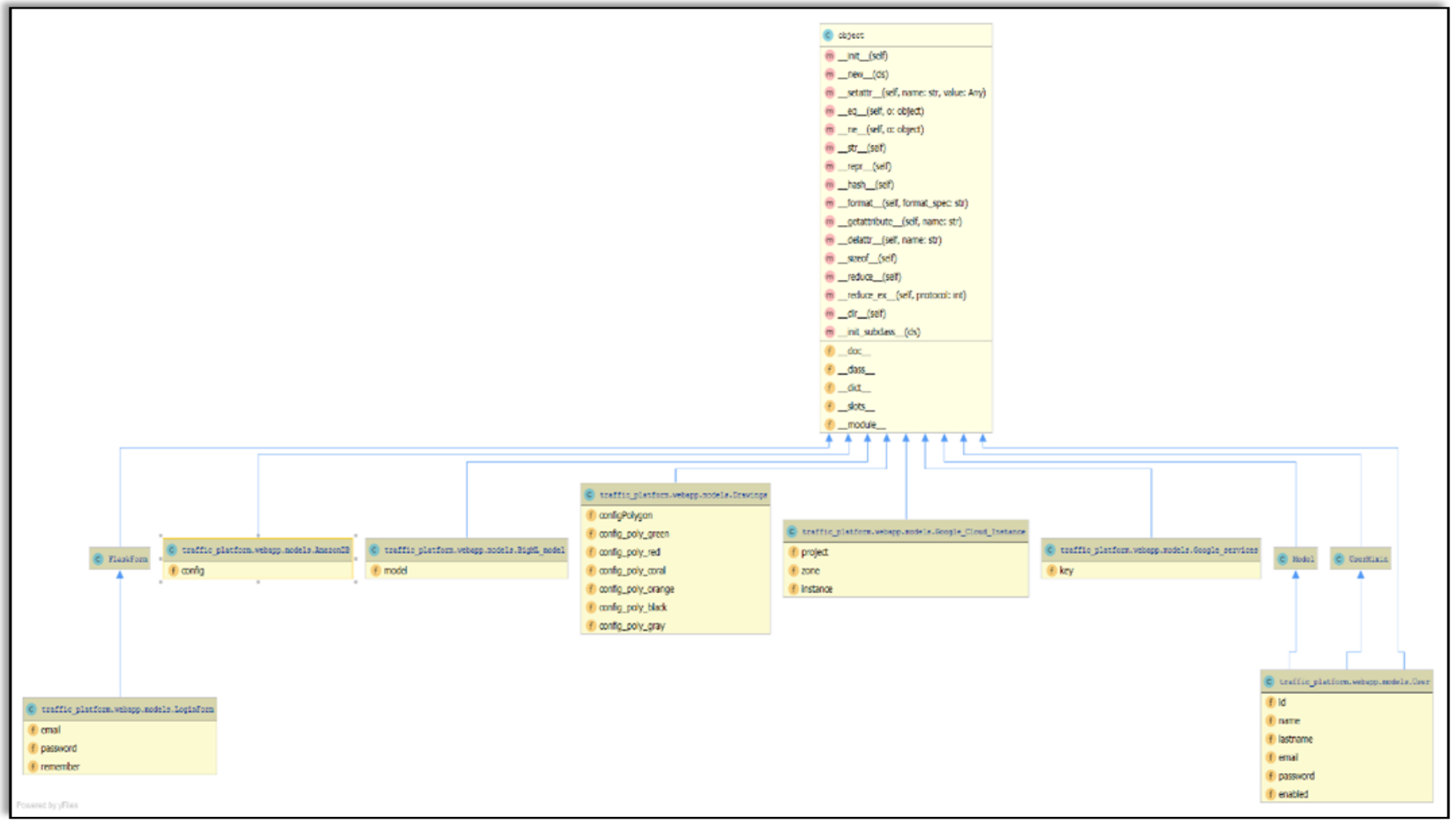



***Figura 44.*** Diagrama de Clases del orquestador de la plataforma

**Adquisición de datos utilizando el orquestador y los servicios de Distance Matrix API de Google**

La plataforma de Google Maps posee varios servicios relacionados con tráfico vehicular que Google ha recopilado durante varios años.

Entre aquellos servicios, debido a las necesidades de la plataforma destaca la Google Matrix Distance API (GDM-API) que permite solicitar información relacionada con el tráfico de una ciudad. Entre los datos de interese se encuentran: el tiempo de viaje, distancia, velocidad media.

Estos datos son provistos como una respuesto en JSON o XML mediante una URL donde se debe proporcionar datos de origen-destino, tipo de unidades a devolver, tipo de recorrido (tren, vehículo, bicicleta) y llave de API proporcionada por Google.

Es decir, que para nuestro propósito se utiliza un *endpoint* similar al siguiente:

```
request="https://maps.googleapis.com/maps/api/distancematrix/json?origins="+origin+"&destinations="+destination+"&mode=driving&departure_time=now&traffic_model=best_guess&key="+Google_services.key
```

***Figura 45.*** Código fuente de solicitud de información de los puntos de origen y destino.

Donde los parámetros <*destination*>, <*origin*> y <*Google_service.key*> se configuran dinámicamente mediante un algoritmo. Dicho algoritmo crea pares origen - destino utilizando una sentencia *For Loop* como se muestra en el siguiente fragmento de código.

```
for id_index in range(0, len(ID)):
    is_ID = puntos['ID'] == ID[id_index]
    puntos_ID = puntos[is_ID]
    puntos_ID = puntos_ID.reset_index(drop = True)
    sub_ID = puntos_ID['sub_ID']
    latitude = puntos_ID['latitude']
    longitude = puntos_ID['longitude']
    print("index: "+str(ID[id_index]))
    #print(puntos_ID)

    for sub_index in range(0,  len(puntos_ID)-1):
        origin = str(latitude[sub_index])+","+str(longitude[sub_index])
        destination = str(latitude[sub_index+1])+","+str(longitude[sub_index+1])
        print("origen: "+origin+" destino:"+destination)
```

***Figura 46.*** Código fuente para la definición de puntos de origen y destino utilizando los índices de calle

Los parámetros <id_index> y <sub_index> corresponden a parámetros de una tabla contenida en la Base de datos *Traffic.DB* que posee un esquema que contiene el nombre de la calle para motivos de identificación además de puntos de longitud y latitud para crear una secuencia de puntos que generan una línea que se asemeja a la ruta de un vehículo sobre una calle específica.

| ID | sub_ID | ruta | nombre | latitude | longitude |
|---|---|---|---|---|---|
| 1 | 1 | Shyris NS | Shyris - 6 de Diciembre | -0.159203000000000000 | -78.477438000000000000 |
| 1 | 2 | Shyris NS | Shyris - Rio Coca | -0.163057000000000000 | -78.478678000000000000 |
| 1 | 3 | Shyris NS | Shyris - Tomas de Berlanga | -0.165361000000000000 | -78.479092000000000000 |
| 1 | 4 | Shyris NS | Shyris - Isla Floreana | -0.168172000000000000 | -78.479564000000000000 |
| 1 | 5 | Shyris NS | Shyris - Gaspas de Villaroel | -0.170565000000000000 | -78.479983000000000000 |
| 1 | 6 | Shyris NS | Shyris - El Telegrafo | -0.173363000000000000 | -78.480486000000000000 |
| 1 | 7 | Shyris NS | Shyis - Naciones Unidad | -0.177574000000000000 | -78.481231000000000000 |
| 1 | 8 | Shyris NS | Shyris - Portugal | -0.181722000000000000 | -78.482023000000000000 |
| 1 | 9 | Shyris NS | Shyris - Rep. del Salvador | -0.186069000000000000 | -78.482788000000000000 |
| 1 | 10 | Shyris NS | Shyris - Belgica | -0.187416000000000000 | -78.483031000000000000 |
| 1 | 11 | Shyris NS | Shyris - Eloy Alfaro | -0.189152000000000000 | -78.482649000000000000 |

***Figura 47.*** Entradas en la tabla de calles, donde se aprecian los índices y subíndices de la Avenida de los Shyris NS

Para el caso de la “avenida de los Shyris” la tabla contiene un <id_index> igual a 1 en el sentido norte -sur (Shyris NS), el parámetro <sub_ID> de la tabla corresponde al parámetro <sub_index> en el código de la Figura 45. Código fuente para la definición de puntos de origen y destino utilizando los índices de calle Figura 46 este parámetro permite crear una secuencia de pares longitud - latitud que crean una ruta que se ajusta a la Avenida de los Shyris en el sentido norte-sur entre las Avenidas 6 de diciembre y Eloy Alfaro.

Mediante esta estructura de lazo iterativo (*for loop*) podemos solicitar datos de cada tramo de la avenida de los Shyris en un momento específico o en intervalos consecutivos de tiempo.

Para el propósito de nuestra plataforma se han recibido datos cada 20 minutos creando un volumen de 720 muestras por intersección diariamente. Al final del mes se toman 43200 muestras ya que se tiene 20 sensores de tráfico virtuales (puntos donde se solicita la información de tráfico)

**Trigger en el orquestador principal**

Se ha utilizado la herramienta de *Task Scheduler* de Microsoft Windows para crear un servicio que ejecute el script de adquisición de datos desde los servidores de Google. Para ello es necesario realizar unas comprobaciones básicas y ajustes previos a la ejecución del script. El código fuente del servicio de adquisición de datos desde Google “*dataRetrieve.py*” se encuentra en repositorio de la aplicación accesible mediante el enlace:

https://github.com/TesisAWSML/API/tree/master/traffic_platform/scheduled_tasks

Tabla 27.

*Trigger usados en el orquestador princincipal*

| Tarea | Importancia | Sugerencia |
|---|---|---|
| Verificación de instalación de Python | Es necesario comprobar que se ha instalado correctamente el intérprete de Python antes de correr la tarea en Windows | Si no se ha instalado el intérprete, es necesario descargar la versión adecuada para el sistema operativo e instalar. |
| Verificar que todos los módulos necesarios para la ejecución del script han sido instalados | se debe comprobar que los siguientes módulos están instalados para poder ejecutar el script de adquisición de google: urllib.request, ast, mysql.connector, pandas, datetime | si no se han instalado, es necesario instalarlos mediante *pip install.* |
| Comprobar el directorio de instalación de Python | Se debe conocer la ubicación de la instalación del intérprete de Python | Utilizando la herramienta IDE PyCharm se puede obtener la ruta donde se ejecutan los scripts además brinda información respecto del intérprete de Python. |
| Comprobar donde se encuentra alojado el script | Se debe conocer la ubicación del script a ser ejecutado | Utilizando la herramienta IDE PyCharm se puede obtener la ruta donde se ejecutan los scripts además brinda información respecto del intérprete de Python. |

Una vez se ha comprobado los puntos anteriores es necesario crear una tarea que se ejecute cada 20 minutos utilizando la herramienta de *Task Scheduler* con ello se logra recopilar los datos de tráfico según el intervalo definido creando una serie de datos equidistantes en el tiempo (requisito fundamental para poder realizar un entrenamiento de aprendizaje automático.

Con los datos recopilados por intersección se ha conseguido monitorear el tráfico por aproximadamente 4 meses, además se han obtenido modelos de tráfico para los 20 principales puntos principales de la "Avenida de los Shyris" como se describe en la siguiente sección.

**Almacenamiento De Datos provenientes de clientes y otras instancias**

Existen algunos procesos de almacenamiento que se ejecutan en el servidor de la plataforma, los procesos de almacenamiento de datos desde las aplicaciones se realizan según demanda de los clientes. Mientras que los procesos de almacenamiento de datos de datos obtenidos de Google y de predicciones de tráfico se realizan con una frecuencia establecida.

En total el cont principal procesa tres diferentes tipos de almacenamientos, según los lineamientos que se describen en las siguientes tres secciones.

**Proceso de almacenamiento de datos desde la aplicación**

Los datos enviados desde el cliente móvil y los datos recopilados desde los servidores de Google se almacenan en la misma base de datos como se muestra en la Procesos para la adquisición de datos de tráfico vehicular. Sin embargo, los datos se almacenan separadamente en tablas diferentes dependiendo de la fuente de los datos.

Adicionalmente los datos se dividen por día, es decir que se tiene dos tablas (una con datos de Google y una con los datos de la aplicación) por cada día desde el inicio de la toma de datos.

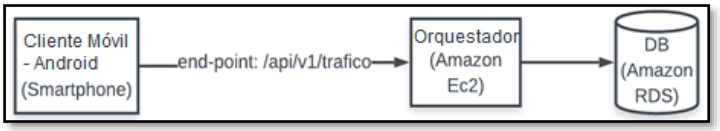



***Figura 48.*** Proceso de almacenamiento del cliente móvil

Además, los datos enviados desde la aplicación se registran mediante un end-point creado específicamente para este propósito. El dispositivo inteligente (u otro dispositivo inteligente) pueden acceder al end-point:

```
/api/v1/trafico?date=<fecha>&time=<hora>&longitude=<longitud>&latitud
e=<latitud>&speed=<velocidad>&bearing=<dirección>&aux=<dato_auxiliar>
```

***Figura 49.*** End-point para registrar los datos enviados a la base de datos.

Dicha URL debe contener todos los parámetros para que puede ser procesada y guardada en la base de datos. Caso contrario la API devolverá un código de error debido a que no se puede procesar una entrada en la tabla de velocidades de tráfico sin todos los parámetros El proceso de almacenamiento se describe en el siguiente diagrama de flujo:

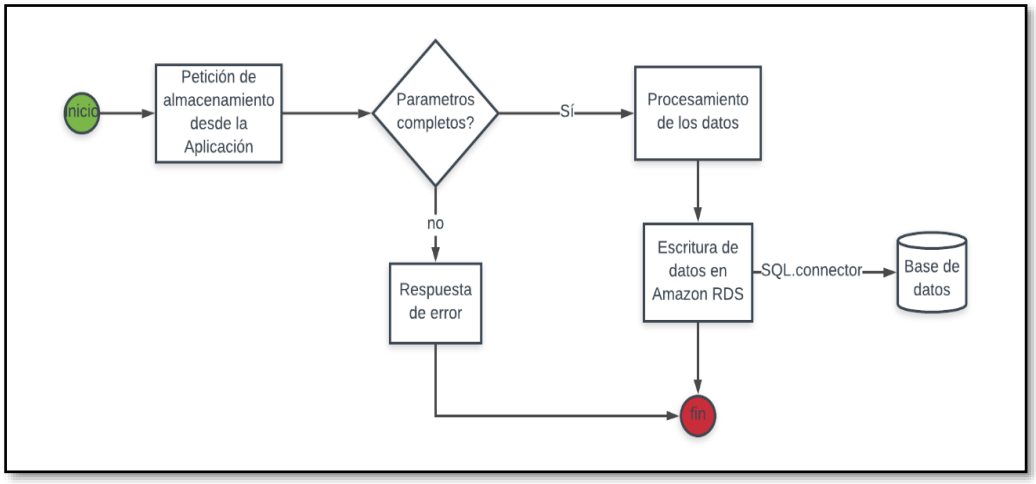



***Figura 50.*** Proceso de almacenamiento de datos desde la aplicación móvil

La aplicación de Android accede al end-point */api/v1/trafico?* para que el servidor a modo de interfaz procese los datos y los escriba en la base de datos de la plataforma.

**Proceso de almacenamiento de datos de datos obtenido a través de GDM-API**

El proceso de almacenamiento se realiza con una periodicidad de 20 minutos, según la información obtenida desde el servidor de Google mediante la Distance Matrix API. El proceso se describe mediante el siguiente diagrama de flujo:

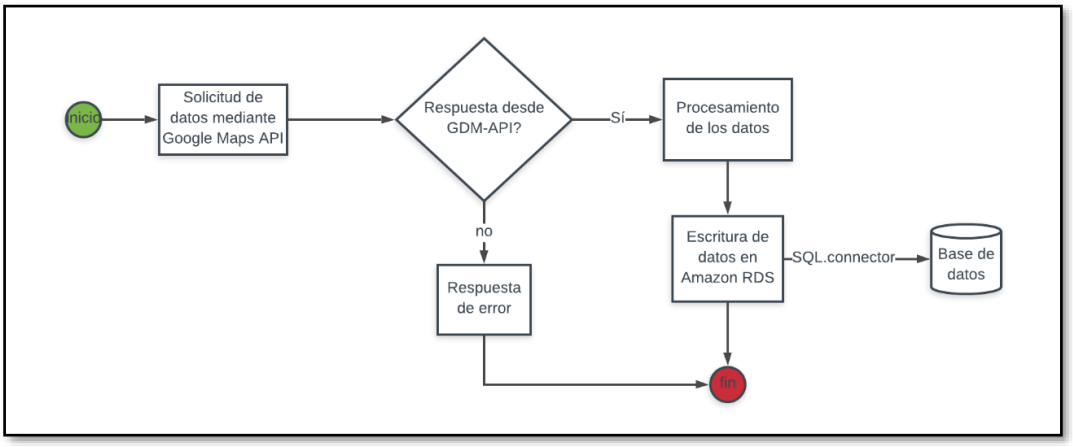



***Figura 51.*** Proceso de almacenamiento de datos desde Google Distance Matrix API

Cuando el sistema operativo determina que han pasado 20 minutos desde la última ejecución de la tarea de recopilación, se inicia una nueva tarea que realiza varias peticiones a los servidores de Google.

Estas peticiones solicitan duración distancia y tiempo estimado de viaje entre dos puntos. Los datos son devueltos como un fichero codificado en JSON.

Estos datos son decodificados en el orquestador, procesados y adaptados según el esquema de la tabla de tráfico. Finalmente son guardados utilizando la librería mySQL.connector.

**Procesamiento y almacenamiento de predicciones**

El proceso de almacenamiento se realiza de forma similar al proceso de almacenamiento de datos desde Google. Existe una tarea que se ejecuta periódicamente cada 24 horas. Dicha tarea inicialmente recopila los datos de tráfico de las últimas 24 horas y crea un conjunto de datos que son utilizados para entrenar los modelos de las calles de la plataforma como se describe en el capítulo 3.

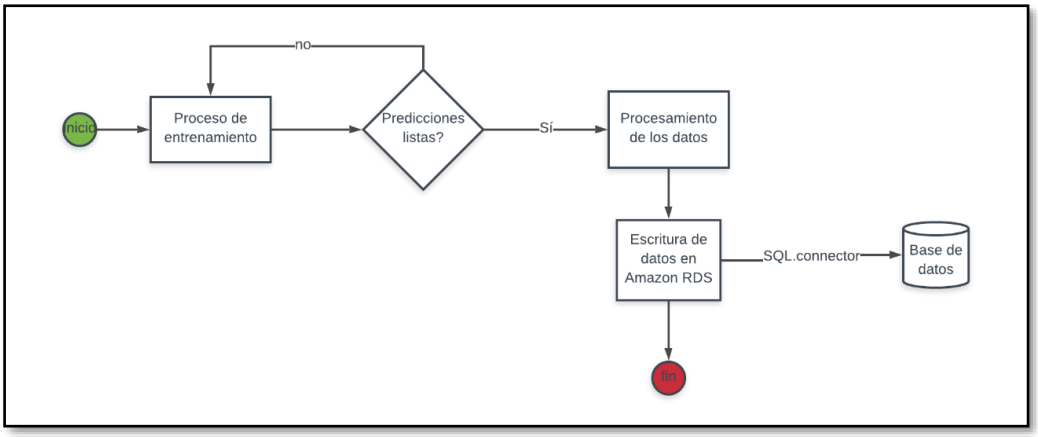



***Figura 52.*** Proceso de almacenamiento de datos de predicción desde la instancia de Google CE

Luego de ejecutarse el entrenamiento y haberse obtenido las predicciones de velocidad se procede a crear conjuntos de datos separados según el índice de la intersección. Estos datos son guardados en la base de datos alojada en Amazon RDS utilizando la librería MySQL.connector de Python.

**Procesamiento De Datos (Preparación De Datos Previo Al Aprendizaje Automático)**

Al obtener la cantidad suficiente de datos, según los requisitos de diseño establecidos para el proyecto, se procede a realizar una depuración y adaptación de datos, específicamente para el proceso de predicción de tráfico vehicular.

Esto debido a que no todos los datos almacenados pueden ser idóneos para el aprendizaje automático. Adicionalmente, dependiendo del tipo de herramienta que se escoja utilizar para el *machine learning* se deberán crear (o no) diferentes estructuras de datos.

Este paso es imprescindible con cualquier librería o herramienta de machine learning que se escoja utilizar, debido a que los repositorios de datos a veces utilizan datos genéricos o sin unidades reales que deben ser convertidas previamente o acondicionadas, o en casos como variables de tipo cadenas de caracteres deban ser eliminadas debido a que no proveen información relevante al modelo.

En el caso de predicción de datos dependientes del tiempo, se relacionan fechas como índices de un conjunto de datos, por ejemplo, si las muestras comienzan en el 1 enero de 2000 a las 00:00 esa muestra corresponde a un índice de <0>. Suponiendo el caso que la variable fue muestreada cada 20 minutos entonces; la muestra <1> corresponderá al valor medido el 1 de enero del 2000 a las 00:20, y así sucesivamente.

Debido a esto y considerando por ejemplo un conjunto de 100 datos diarios, durante un periodo de 30 días se tiene un *dataset* de 30000 muestras, si se pretende predecir el comportamiento del sistema 10 días en adelanto, se debe solicitar las muestras con índices entre 30000 y 30099. De esta forma se obtiene 100 valores en adelanto que contienen el comportamiento de la variable seleccionada.

Para el presente proyecto se ha seleccionado a la velocidad en un trayecto de calle, dicho trayecto de calle está delimitada por dos intersecciones semaforizadas como se ha detallado en el capítulo 3.

Entonces el modelo nos devolverá un modelo por cada calle según la siguiente definición:

$$velocidad_i = modelo_i(k);\ 0 < k < n\ muestras + m\ predicciones \qquad (2)$$

Donde entonces se obtiene una velocidad por cada modelo de intersección obtenido, la variable de entrada del modelo es por lo tanto un índice <k> que puede tener valores desde 0 hasta el número de muestras más el número de predicciones deseadas.

**Presentación de interfaces para el cliente web**

Debido a que la plataforma realiza interacciones entre varios clientes (PC, aplicaciones móviles, aplicativos de TVD) el orquestador se encarga de recibir solicitudes y procesarlas de acuerdo con el tipo de URL (endpoint) a la que la que accede el dispositivo cliente.

Los *endpoints* que presentan la sintaxis `http://<ip_base>/<recurso>` reciben parámetros de entrada (no en todas las ocasiones) y devuelven una respuesta HTML que es interpretada por el navegador del dispositivo cliente y mostrada al usuario como una página web.

En esta sección se describe el proceso de recepción de solicitudes desde los clientes, el procesamiento de datos para crear las interfaces y la respuesta en formato HTML hacia el cliente. Las interfaces graficas de la plataforma que el controlador utiliza para su presentación al usuario cliente son descritas en la sección 4.4.1 más adelante.

Como estado inicial, el servidor se encuentra en reposo, a la espera de una solicitud entrante.

Al momento que alguno de los clientes accede a los recursos del servidor por medio de una URL.

El controlador comprobara si la URL corresponde a alguna de las tareas programadas en la API, posteriormente comprobara si los parámetros de entrada se ajustan a los necesarios para la ejecución de la tarea.

Una vez comprobados los datos de entrada el controlador ejecuta el código asignado para dicha tarea. De existir alguna inconsistencia el controlador emite una excepción, procede a cancelar la tarea, devuelve un código de error (error 400) vuelve al estado de reposo.

Sin embargo, si la tarea se ejecutó con normalidad sin ninguna excepción critica, el controlador devuelve un fichero en HTML que contiene la página web que es mostrada al usuario, emite un código de éxito (código 200) y regresa al estado de reposo.

**Respuesta de solicitudes a clientes**

Los *endpoints* que contienen la sintaxis `http://<ip_base>/api/v1/<recurso>` no devuelven interfaces HTML, en cambio proveen una respuesta que contiene información encapsulada como un fichero con codificación JSON. Estos *endpoints* sirven para la interacción donde el usuario no necesita una interfaz gráfica, por ejemplo: para el almacenamiento de datos de tráfico o simplemente devolver los resultados de una consulta en la base de datos.

**Tareas de Aprendizaje Automático implementadas en los servidores de Google Cloud Platform**

Uno de los objetivos principales del proyecto es el desarrollo de una solución que utilice una herramienta de aprendizaje automático. En el entorno de la plataforma se ha logrado implementar un sistema de predicción y un sistema de clasificación mediante *machine learning.*

En primer lugar, se ha logrado construir un sistema de predicción de velocidad utilizando un algoritmo recursivo de aprendizaje automático. Este proceso se ejecuta utilizando varios recursos de lenguaje de programación Python y se realiza sobre una instancia de computación en la nube que posee un hardware virtualizado que se explica con más profundamente en el capítulo *Implementación en la nube*.

En segundo lugar, se tiene el proceso de clasificación que se realiza sobre la plataforma BigML, mediante la utilización de la API de Python. La API permite crear *datasets* y entrenar modelos de clasificación (también conocidos como arboles de decisión). Además, permite la evaluación del modelo entrenado para crear predicciones. Estas funcionalidades se han aprovechado para que, mediante un conjunto de datos clasificados manualmente, se puede entrenar un modelo de clasificación que permite ingresar parámetros y obtener clasificaciones automáticas. La implementación de esta funcionalidad de describe en la sección 4.3.2.1.

**Predicción de tráfico vehicular realizada fuera del controlador**

Para la presente plataforma se ha escogido utilizar el módulo de *Statsmodels* compatible con Python (2.7 y 3.7). Sin embargo, la librería *mysql.connector* necesaria para conectar el servidor de *machine learning* con la base de datos disponibles (al momento de la elaboración de esta documentación) presenta un problema de compatibilidad con Python 3.7, por lo tanto el entorno de ejecución para el servidor de entrenamiento del modelo de trafico de tránsito vehicular es Python 2.7.

**Algoritmo de predicción SARIMAX**

De acuerdo con el análisis realizado en el Capítulo 3 se ha determinado que para el entrenamiento de los modelos (intersecciones) de la Avenida de los Shyris es recomendable utilizar un algoritmo recursivo estadístico. Por lo tanto, para el entrenamiento se utiliza la librería SARIMAX del módulo de Python *Statsmodels*. Este módulo contiene un conjunto de algoritmos recursivos estadísticos para el entrenamiento, evaluación y manipulación de modelos matemáticos.

El algoritmo de media móvil integrada autorregresiva estacional con soporte de variables exógenas, SARIMAX por sus siglas en inglés (*Seasonal Autoregressive Integrated Moving*

*Average with exogenous variables support*) se utiliza iterativamente para obtener un modelo para cada una de las intersecciones de la Avenida de los Shyris. En total se generan 20 modelos diferentes tomando el conjunto de datos consolidados previamente. El flujo de trabajo se muestra en la siguiente Figura 53.

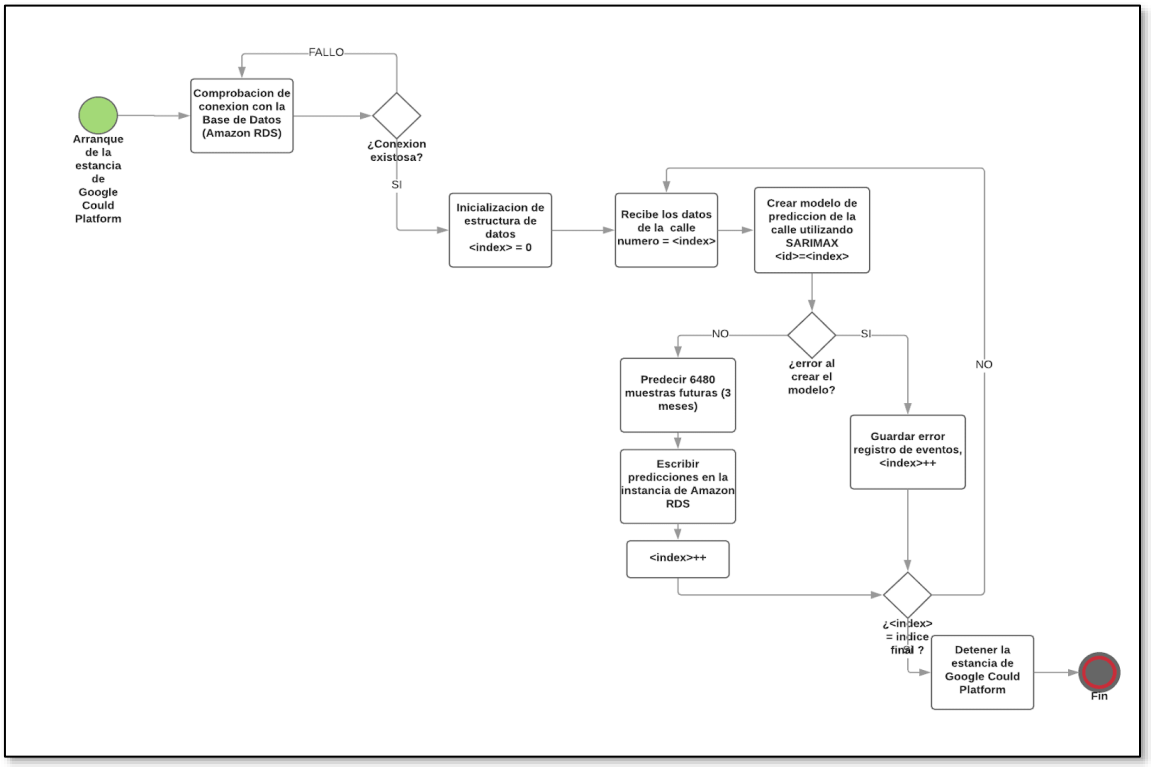



***Figura 53.*** Diagrama de Flujo del entrenamiento de modelos de intersecciones

El algoritmo recibe los datos previamente preparados, los procesa para crear un conjunto de datos denominado *DataFrame* (que consiste en una estructura de datos compatible con Python). Adicionalmente, el algoritmo SARIMAX explicado en la Sección 0, se necesita recibir un conjunto de parámetros según la siguiente notación (1). Estos parámetros se detallan en la siguiente Tabla 28 y permiten al algoritmo arrancar con una aproximación inicial, posteriormente estos parámetros

podrían ser modificados si el algoritmo recursivo lo considerara necesario para la convergencia del modelo.

Tabla 28.

*Parámetros iniciales del algoritmo SARIMAX para el entrenamiento de intersecciones vehiculares*

| Parámetro | Valores posibles | | Valores escogidos |
|---|---|---|---|
| **Orden de la serie** | p | [0, 1, 2] | 0 |
| | d | [0, 1] | 0 |
| | q | [0, 1, 2] | 0 |
| **Orden estacional** | P | [0, 1, 2] | 1 |
| | D | [0, 1] | 0 |
| | Q | [0, 1, 2] | 1 |
| **Periodicidad de las muestras** | m | Conjunto de los enteros positivos | 72 |
| **Tendencia** | t | 'n', 'c', 't', 'ct' | 'n' |

A continuación, se muestra un fragmento del código que se utiliza para la creación y obtención de predicciones:

```
1 history = data[:n]
2 my_order = (p, d, q)
3 my_seasonal_order = (P, D, Q, m)
4 model = SARIMAX(history, order=my_order, seasonal_order=my_seasonal_order, trend='n')
5 model_fit = model.fit()
6 yhat = model_fit.predict(final = n)
8 db = 'mysql+mysqldb://tesis_aws_ml:201809Admini$trad0r@trafficdb.cb7fxcpttava.sa-east-1.rds.amazonaws.com/TrafficDB'
9 datos = pd.DataFrame(yhat)
10 engine =sqlalchemy.create_engine(db)
11 datos.to_sql(name='predicciones', con=engine, if_exists='replace')
```

***Figura 54.*** Algoritmo para la creación y obtención de predicciones

Inicialmente, en la variable <history> se guardan la serie de datos temporales una intersección (es decir los datos de velocidad correspondientes). Posteriormente, es necesario configurar los parámetros con los valores mostrados en la Tabla 28.

Una vez programados todos los parámetros de entrada, es posible iniciar el entrenamiento. En la variable <model> se guarda el modelo con todos los parámetros iniciales según los datos de entrada. En la variable <model_fit> se crea un modelo entrenado utilizando el método *fit()* de la variable <model>. Una vez terminado el entrenamiento del modelos se procede a generar predicciones, para ello se configura un parámetro de numero de muestras en el futuro utilizando el método *predict()* del modelo guardado en la variable <model.fit>. Finalmente, las n muestras futuras se guardan como un vector en la variable <yhat>.

Finalmente el error de las predicciones se calcula utilizando la utilidad `mean_squared_error` de la librería `sklearn.metrics;` esta librería permite el cálculo del error medio cuadrático

basándose en los datos de prueba y una porción de las predicciones. Los resultados de los errores medios cuadráticos se asignan a cada uno de los modelos de intersección creados y se presentan en la interfaz de *Gráficos por calle* como se describe en la sección 0.

**Hardware utilizado para predicción de tráfico vehicular**

La selección de hardware adecuado se realizó mediante un método empírico, a partir de las características dispuestas de los servidores virtuales ofrecidos por *google cloud*, en función de lo indicado se contrató un servidor con el siguiente conjunto de características de datos de una intersección durante un día entero (72 muestras) y se procedió a realizar el entrenamiento en una instancia con un hardware de 1 CPU, 1 GB de RAM con sistema operativo Windows Server 2008 R2 de 64 bits. El entrenamiento duro aproximadamente 3 minutos. Luego de una semana se realizó un entrenamiento con 504 muestras que duro aproximadamente 15 minutos. Posteriormente se incrementó paulatinamente el número de muestras y se midió el tiempo de entrenamiento.

Al alcanzar el conjunto de 1500 muestras (aproximadamente 3 semanas de mediciones) se presentaron problemas en la convergencia debido a la falta de memoria RAM. Se decidió incrementar la memoria RAM hasta 4 GB, lo que evito el problema de convergencia sin embargo incremento el tiempo de entrenamiento a aproximadamente 25 minutos, lo que resulta en aproximadamente 8 horas de ejecución de la instancia para lograr entrenar los 20 modelos necesarios para la plataforma.

Siguiendo este procedimiento, manipulando el número de muestras, modificando la RAM, ajustando el número de CPUs y midiendo el tiempo de entrenamiento se llegó a la siguiente tabla que muestra la relación entre hardware y conjunto de datos.

Tabla 29.

*Costos y viabilidad de hardware según el conjunto de datos usados para el entrenamiento.*

| Conjunto de datos por intersección | Memoria RAM | Numero de CPUs | Tiempo de entrenamiento | Costo en GCP | Viabilidad del hardware utilizado |
|---|---|---|---|---|---|
| **72 muestras (1 día)** | 1 GB | 1 | 3 minutos | 0.07 USD/hora | No, el conjunto de datos no es demasiado pequeño para el propósito de la plataforma. |
| **504 muestras (1 semana)** | 1 GB | 1 | 25 minutos | 0.07 USD/hora | No, el conjunto de datos no es demasiado pequeño para el propósito de la plataforma. |
| **1512 muestras (3 semanas)** | 1 GB | 1 | Errores de convergencia (RAM insuficiente) | 0.07 USD/hora | No, el hardware no es suficiente para lograr convergencia. |
| **1512 muestras (3 semanas)** | 4 GB | 1 | 5 horas | 0.08 USD/hora | No, el conjunto de datos procesados no es el adecuado para la plataforma. |
| **2160 muestras (1 mes)** | 4 GB | 1 | Error de convergencia (RAM insuficiente) | 0.27 USD/hora | No, el hardware no es suficiente para lograr convergencia. |
| **2160 muestras (1 mes)** | 16 GB | 4 | 8 horas | 0.31 USD/hora | No, el tiempo de entrenamiento es demasiado largo. El costo mensual es muy elevado. |
| **2160 muestras (1 mes)** | 10 | 8 | 2 horas | 0.54 USD/hora | No, el conjunto de datos procesados no es el adecuado para la plataforma. |

**CONTINÚA →**

| Conjunto de datos por intersección | Memoria RAM | Numero de CPUs | Tiempo de entrenamiento | Costo en GCP | Viabilidad del hardware utilizado |
|---|---|---|---|---|---|
| **4230 muestras (2 meses)** | 10 | 8 | 2 horas | 0.54 USD/hora | No, el conjunto de datos procesados no es el adecuado para la plataforma. |
| **7830 muestras (3 meses)** | 10 | 6 | Error de convergencia (RAM insuficiente) | 0.41 USD/hora | No, el hardware no es suficiente para lograr convergencia. |
| **7830 muestras (3 meses)** | 12 | 6 | 2 horas | 0.42 USD/hora | Si, el hardware permite manejar el conjunto de datos. |

Como se puede apreciar en los datos de la Tabla 29, la cantidad de memoria RAM es crucial para la convergencia de los modelos de la plataforma. Adicionalmente se muestra la relación entre número de núcleos y tiempo de entrenamiento, donde claramente se puede apreciar que la relación en inversa. Tomando en consideración que especificaciones altas de hardware conllevan altos costos económicos se ha considerado adecuado llegar a un equilibrio entre tiempos de entrenamiento (mayor tiempo igual a mayor costo) y el número de entrenamiento que se pueden ejecutar gratuitamente. Se estableció un numero de entrenamientos equivalente a 6 meses realizando un entrenamiento diario, es decir 180 entrenamientos de aproximadamente 2 horas lo que equivale a 360 horas. Teniendo en cuenta que con el hardware de 8 CPUs y 16GB de RAM se puede obtener 550 horas de ejecución se estableció que 12GB de RAM y 6 CPUs cumplen con el criterio de 6 meses de entrenamientos gratuitos diarios.

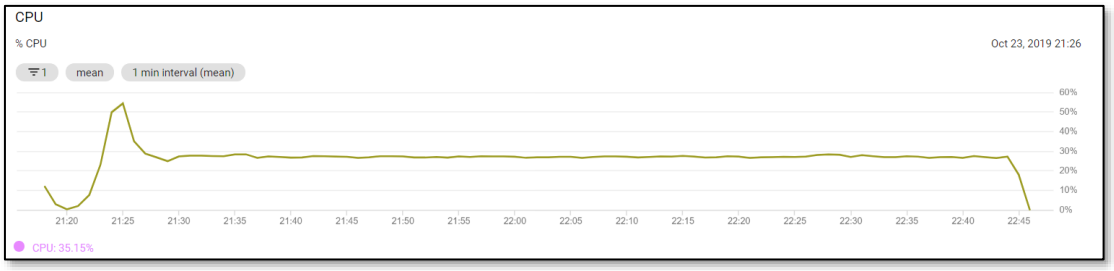



***Figura 55.*** Gráfico de recursos de CPU utilizados durante un entrenamiento

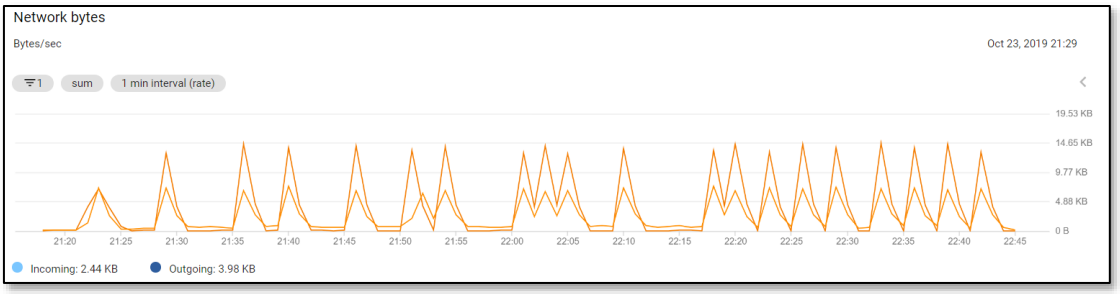



***Figura 56.*** Gráficos de utilización de recursos de red durante un entrenamiento

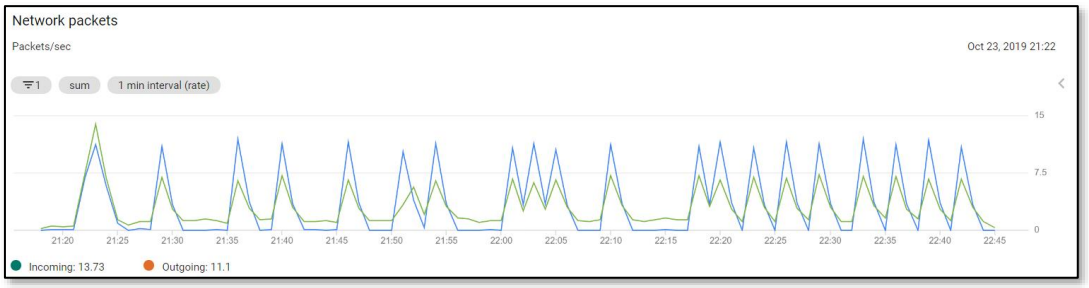



***Figura 57.*** Gráficos de utilización de recursos relacionados con datos

En las figuras anteriores se puede apreciar que la duración del entrenamiento de las 20 intersecciones demora aproximadamente 3 horas, en ensayos previos se verifica que por cada modelamiento la instancia necesita alrededor de 15 minutos

Adicionalmente se verifico que, con el conjunto de datos actuales equivalentes a 3 meses de muestras, aumentar el número de CPUs por encima de 6 no refleja una reducción significativa en

el tiempo de entrenamiento resultando en mayor costo de ejecución por hora sin mayor variación en el tiempo total de entrenamiento.

**Implementación en los servicios de Google Cloud Platform**

Uno de los objetivos principales de la plataforma es lograr que su ejecución sea realizada sobre alguna de las plataformas disponibles de computación en la nube. Para ello, se procedió a realizar una comparativo entre funcionalidades, costos, ventajas y beneficios en el capítulo 3. El análisis arrojó que el servicio *Compute Engine* de la plataforma *Google Cloud Computing* es el servicio adecuado para el entrenamiento de los modelos de predicción de la plataforma. Ya que *Compute Engine* permite la modificación de especificaciones de instancias sin una restricción de hardware (como sucede en Amazon y otros servicios). Es decir, que es posible mejorar gratuitamente especificaciones de hardware en la instancia disminuyendo el número de horas de ejecución gratuitas.

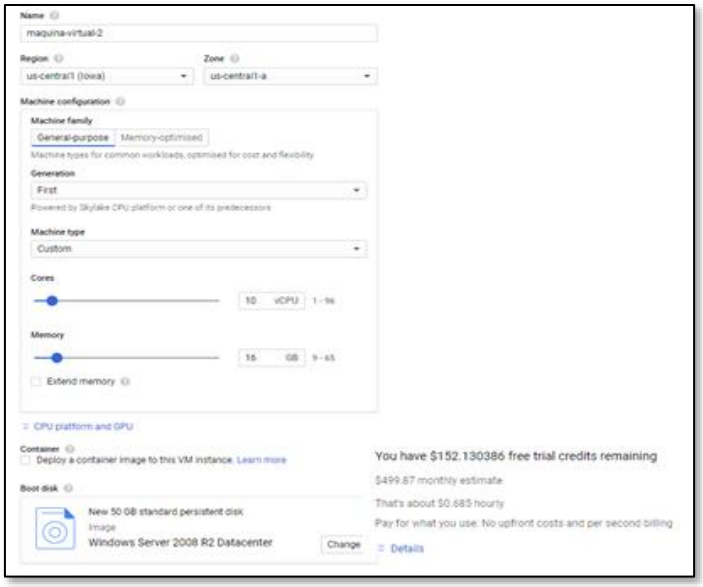



***Figura 58.*** Interfaz de configuración de Google Compute Engine y costo estimado por horas y meses

Para la creación de una instancia en Google Cloud Computing, es necesario crear una cuenta y contar con una tarjeta de crédito o débito aceptada por Google. Una vez creada la cuenta y comprobado que el método de pago. Se procede a acceder al menú de *VM Instances* (instancias de máquina virtual) donde el botón *Create Instance* nos permite configurar una máquina virtual según nuestras necesidades, dicho botón mostrara le interfaz mostrada en la Interfaz de configuración de Google Compute Engine.

El asistente de GCP permite seleccionar el nombre, el número de núcleos, la cantidad de RAM y el sistema operativo base como se muestra en la siguiente figura, además el asistente muestra el costo de ejecución por hora.

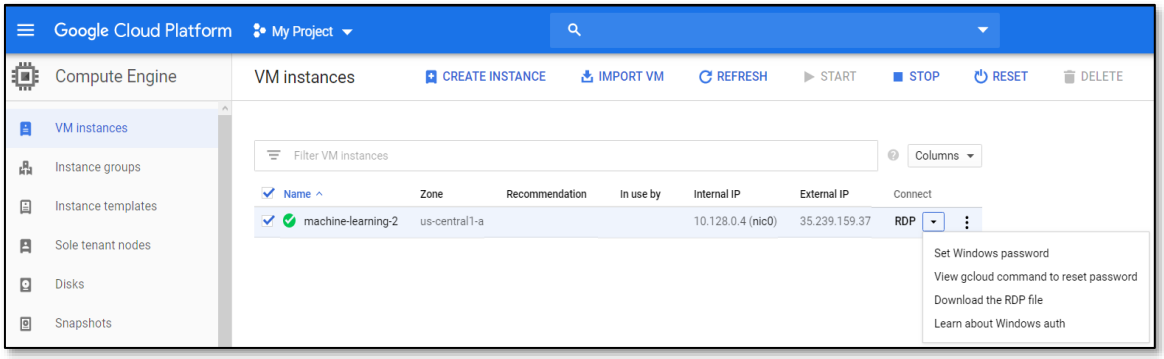



***Figura 59.*** Instancia en ejecución en Compute Engine de Google Cloud Platform

Una vez creada la instancia de *Compute Engine*, es posible arrancarla y comenzar a utilizar la instancia como si se tratara de un servidor remoto, mediante una conexión de escritorio remoto. Google asignará una dirección IP externa como se muestra en la Instancia en ejecución en Compute Engine de Google Cloud Platform que permitirá acceder a la instancia mediante el software *Remote Desktop Connection* utilizando la dirección IP que Google configuro en la máquina y las credenciales configuradas en la plataforma (dichas credenciales corresponden a nombre de usuario

y contraseña, que pueden ser modificadas directamente en la consola de GCP utilizando la opción "*set Windows password"*).

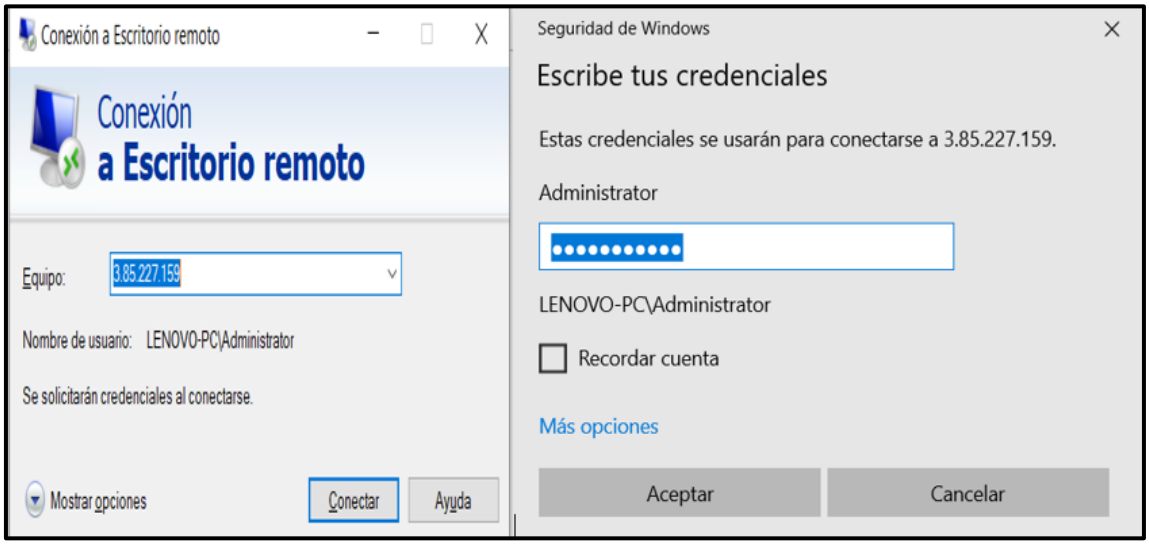



***Figura 60.*** Software RDC con IP configurada y credenciales de usuario

Una vez se ha accedido a la instancia de máquina virtual mediante RDP (acrónimo en inglés de protocolo de escritorio remoto) se puede utilizar como una PC cualquiera con Windows. Procedemos a comprobar las características de la instancia configurada.

En la Escritorio remoto de la Instancia creada en Google Compute Engine se puede apreciar que la PC posee un procesador con 6 CPUs además de 12 GB de memoria RAM. Una vez completada la verificación se procede a instalar Python 3.7 (dicho instalador ha sido previamente descargado desde la página oficial del proyecto *Python.org*) que es motor para poder obtener modelos de predicción basados en datos de tipo serie temporal.

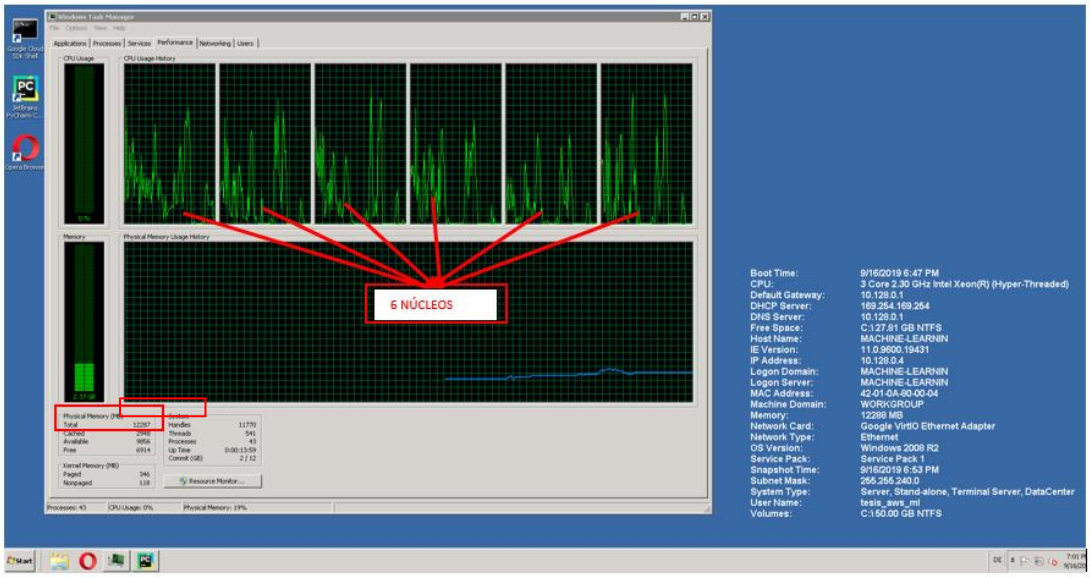



***Figura 61.*** Escritorio remoto de la Instancia creada en Google Compute Engine para predicción.

**Procedimiento de clasificación de calles según características cualitativas**

Para la implementación de la clasificación de calles se ha realizado un análisis de las calles principales de la Avenida de los Shyris, este análisis considera factores como el número de carriles, el número de paradas de autobús, escuelas, hospitales, etc. como se muestra en el siguiente fragmento de la tabla para entrenamiento.

| CALLES | ID | SUB_ID | ETIQUETA | CARRILES | RESTAURANTS | SEMAFOROS | ESCUELAS | CENTROS_COMERCIALES | PARQUES | TRANSPORTES | HOSPITALES | REDONDEL | AREA_EJECUTIVA | PARADA_BUS | LATITUD | LONGITUD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 de Agosto-Santa Prisca NS | 3 | 23 | 10 | 2 | False | False | False | False | True | True | False | False | True | False | -0.214738 | -78.503380 |
| 10 de Agosto-Gral Briceno NS | 3 | 24 | 10 | 2 | False | False | False | False | True | True | False | False | True | False | -0.216665 | -78.504906 |
| 10 de Agosto-Francisco de Caldas NS | 3 | 25 | 7 | 2 | True | True | False | True | True | True | False | False | True | True | -0.217559 | -78.505747 |
| 10 de Agosto-Avenida Gran Colombia SN | 4 | 1 | 16 | 2 | True | True | False | True | True | True | False | False | True | True | -0.215867 | -78.504159 |
| 10 de Agosto-Santa Prisca SN | 4 | 2 | 0 | 3 | False | False | False | False | True | True | False | False | True | True | -0.214487 | -78.503116 |
| 10 de Agosto-Luis Sodiro SN | 4 | 3 | 0 | 2 | False | False | True | False | True | True | True | False | True | True | -0.212860 | -78.501826 |
| 10 de Agosto-Tarqui SN | 4 | 4 | 0 | 2 | True | True | True | False | True | False | False | False | True | True | -0.211477 | -78.501312 |
| 10 de Agosto-Patria SN | 4 | 5 | 14 | 2 | True | True | False | False | True | True | False | False | True | True | -0.206579 | -78.501312 |
| 10 de Agosto-18 de Septiembre SN | 4 | 6 | 21 | 2 | True | True | False | False | False | True | False | False | True | True | -0.205744 | -78.499234 |
| 10 de Agosto-Colon SN | 4 | 7 | 21 | 2 | True | True | False | False | False | True | False | False | True | True | -0.198495 | -78.496105 |
| 10 de Agosto-Francisco de Orellana SN | 4 | 8 | 0 | 2 | True | False | False | False | False | False | False | False | True | True | -0.196122 | -78.495073 |
| 10 de Agosto-Cuero y Caicedo SN | 4 | 9 | 21 | 3 | True | True | False | False | False | True | False | False | True | True | -0.191794 | -78.493293 |
| 10 de Agosto-Marina de Jesus SN | 4 | 10 | 14 | 3 | True | True | False | False | True | True | False | False | True | True | -0.187778 | -78.491646 |
| 10 de Agosto-Rumipamba SN | 4 | 11 | 21 | 3 | True | True | False | False | False | True | False | False | True | True | -0.185528 | -78.490897 |
| 10 de Agosto-Republica SN | 4 | 12 | 21 | 3 | True | True | False | False | False | True | False | False | True | True | -0.179898 | -78.489234 |
| 10 de Agosto-Naciones Unidas SN | 4 | 13 | 18 | 3 | True | True | False | True | False | True | False | False | True | True | -0.175746 | -78.488318 |
| 10 de Agosto-Joaquin Auz SN | 4 | 14 | 17 | 2 | False | True | False | False | False | True | False | False | True | True | -0.174582 | -78.488052 |
| 10 de Agosto-Av Juan Jose de Villalengua SN | 4 | 15 | 20 | 3 | False | True | False | True | False | True | False | False | True | True | -0.172461 | -78.487657 |
| 10 de Agosto-Gaspar de Villarroel SN | 4 | 16 | 12 | 4 | False | True | False | False | False | True | False | True | True | True | -0.16861 | -78.486922 |

***Figura 62.*** Fragmento de la tabla para entrenamiento del modelo de clasificación

La columna <etiqueta> es el parámetro que el modelo de clasificación devolverá según los parámetros que el usuario ingrese. Los parámetros de entrada son las columnas restantes con excepción de las columnas <CALLES>, <ID>, <SUB_ID>, <LATITUD> y <LONGITUD> que no servirían para clasificar una calle ya que todas son entradas únicas y no poseen una relación con los demás parámetros.

**Tareas de clasificación realizadas sobre la Plataforma BigML**

Actualmente existen muchas herramientas que permiten realizar una clasificación de un conjunto de datos según asociación. Sin embargo, de acuerdo con el análisis realizado en el capítulo 3, se ha determinado que los servicios de la plataforma BigML permiten cumplir con los objetivos de clasificación de calles de la ciudad de Quito.

El archivo con conjunto de datos para entrenamiento *entrenamiento.xlsx* (incluido en los entregables de este proyecto y accesible mediante el enlace al repositorio https://github.com/TesisAWSML/API) se procede a ingresar manualmente en los servidores de BigML mediante la interfaz web, en dicha interfaz se puede configurar las entradas y salidas del modelo de clasificación.

Como se ha establecido, el parámetro <ETIQUETA> identifica las intersecciones de la avenida de los Shyris con un número único. Cada una de estas calles posee un conjunto de parámetros que la diferencia de otra calle. Por lo tanto, cada <ETIQUETA> corresponde a un conjunto de parámetros únicos. Por ejemplo, el tramo de la avenida de los Shyris, entre las calles 6 de diciembre y Rio Coca que ha sido evaluado con un valor <ETIQUETA> igual a 1 posee los siguientes parámetros:

Tabla 30.

*Parámetros de clasificación de calles*

| CALLES | ID | SUB_ID | ETIQUETA | CARRILES | RESTAURANTS | SEMAFOROS | ESCUELAS | CENTROS_COMERCIALES | PARQUES | TRANSPORTES | HOSPITALES | REDONDEL | AREA_EJECUTIVA | PARADA_BUS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Shyris-Rio Coca NS | 1 | 1 | 1 | 3 | True | True | False | False | False | False | False | False | True | True |

Es decir que, se clasifica cada calle con un número <ETIQUETA> diferente según las características (parámetros) de la calle. La salida del entrenamiento es un modelo que recibe 12 parámetros (11 datos boléanos y uno tipo categórico) y como respuesta devuelve un numero entero que corresponde a la variable <ETIQUETA>.

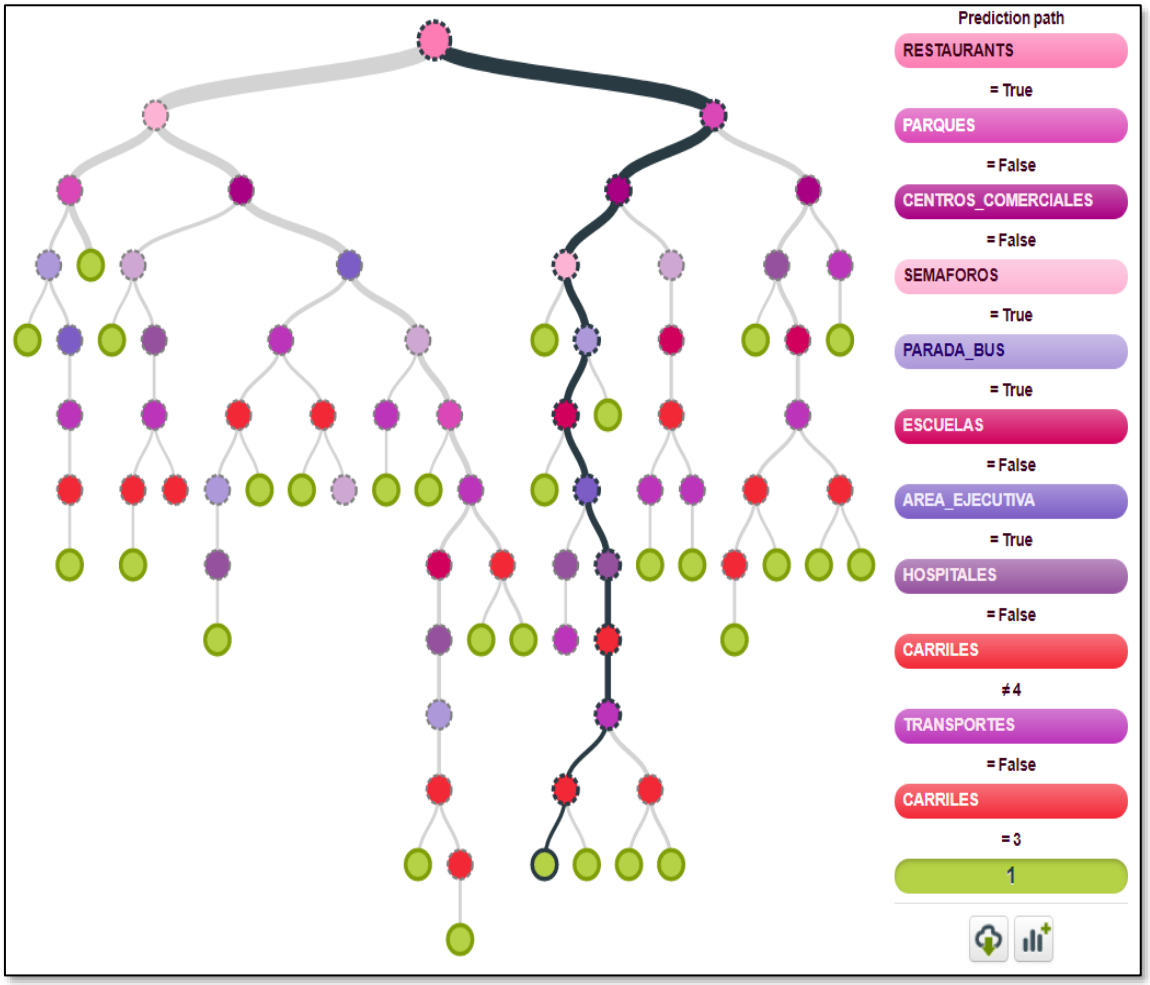



***Figura 63.*** Árbol de decisión para la clasificación de calles de Quito en BigML

Para la creación del conjunto de datos para el entrenamiento, se ha analizado un conjunto de 284 intersecciones de calles en la ciudad de Quito. Una vez se ha completada a tabla, se ha procedido a acondicionar los datos según los requerimientos de la plataforma BigML. Posterior al entrenamiento, la plataforma *BigML* devuelve un modelo que es mostrado gráficamente como un árbol de decisión que, representa a un código fuente que puede ser evaluado local o remotamente utilizando los servidores de *BigML*.

En la Árbol de decisión para la clasificación de calles de Quito se aprecia en el lado derecho el conjunto de condiciones que una intersección debe cumplir para que tenga un valor <ETIQUETA> igual a 1. Sin embargo, existen también otras condiciones para que el valor sea igual a uno como se muestra en la siguiente Figura 64.


***Figura 64.*** Captura de pantalla de BigML para las condiciones en el árbol de decisión para que <ETIQUETA> tenga valor 1

Se puede apreciar que dependiendo del parámetro <TRANSPORTES> y el número de <CARRILES> el tipo de calle puede también ser clasificada como <ETIQUETA> igual a 1.

Esto nos indica que el algoritmo iterativo encuentra diferentes combinaciones de parámetros que correspondan a un tipo de calle dependiendo de sus parámetros de entrada.

**Evaluación de calles según el modelo entrenado en la plataforma BigML**

Una vez se ha obtenido el modelo, es posible evaluar sus resultados ingresando programáticamente los parámetros de entrada. El siguiente fragmento de código utiliza la clase *BigML_model* alojada en código *traffic_platform.webapp.models* que contiene el objeto <model>, este posee las credenciales necesarias para evaluar un conjunto de datos de entrada en los servidores de *BigML*.

```
class BigML_model:
    model = Model(  'model/5d2a632edf6bdb089b000365',
                    api=BigML(  "tesis_aws_ml",
                                "eb8a9c2e91a8d102a7d9c9e921ec99dfde75098f",
                                domain="bigml.io"
                    )
                )
```

***Figura 65.*** Código de BigML

Dentro del código *traffic_platform.webapp.views* existe un método *classify()* que se recibir un conjunto de parámetros enviados mediante el end-point `/api/v1/classify`. Este end-point recibe los 12 parámetros de la calle a evaluar como se muestra en la siguiente URL.

http://3.85.227.159/api/v1/classify?paradaBus=SI&areaEjecutiva=SI&redondel=SI&parques=SI&centrosEducativos=SI&hospitales=SI&restaurantes=SI&carriles=1&centrosComerciales=SI&semaforos=SI&nombreCalle=&transporte=SI

A través de esta URL, es posible enviar los parámetros de tráfico hacia el servidor principal. En dicha URL se tiene claramente establecidos los parámetros boléanos representados mediante las palabras SI y NO, además del parámetro categórico representado como un numero entre 1 y 4.

El servidor recibe dichos parámetros y los convierte en un diccionario que será enviado como el parámetro <input_data> del método model.preditc(). Este método devuelve un diccionario con información referente a la clasificación realizada, entre todos los datos recibidos se encuentra la variable <ETIQUETA> que es accedida mediante el índice [‘prediction’] del diccionario <classification>. Este valor es guardado en una variable auxiliar <id> como se muestra en el fragmento de cogido siguiente.

```
model = BigML_model.model
# To make predictions fill the desired input_data in next line.

input_data = {"CARRILES": str(carriles), "RESTAURANTS": change_string(restaurants),
              "SEMAFOROS": change_string(semaforos), \
              "ESCUELAS": change_string(escuelas), "CENTROS_COMERCIALES": change_string(CCs),
              "PARQUES": change_string(parques), \
              "HOSPITALES":change_string(hospitales), "TRANSPORTE": change_string(transporte),
              "REDONDEL": change_string(redondel), \
              "AREA_EJECUTIVA": change_string(area_ejecutiva),
              "PARADA DE BUS": change_string(paradas)}  # REDONDEL,AREA_EJECUTIVA,PARADA_BUS
print(input_data)
classification = model.predict(input_data, full=True)
print(classification)
time_i = datetime.datetime.now()
time_f = time_i + datetime.timedelta(minutes=1440)
print(classification['prediction'])
id=str(classification['prediction'])
```

***Figura 66.*** Fragmento de código utilizado para evaluar un conjunto de datos utilizando el modelo de BigML

De forma similar a como el algoritmo de predicción provee datos de error en la predicción, los árboles de decisión proveen datos de confianza. Esta medida permite decidir si una clasificación

es suficiente confiable o no. al momento de clasificar las calles el sistema devuelve la confiabilidad de las clasificaciones dentro de los datos que el servidor de BigML provee como respuesta a la solicitad de clasificación.

**FRONT-END**

De acuerdo con los mockups diseñados en el *Capítulo 3.* se ha construido un conjunto de plantillas en HTML5 que representan las páginas web a las que el usuario puede acceder para aprovechar los servicios de la plataforma.

Se han creado un total de 8 plantillas que se describen en la siguiente tabla donde cada página web se encarga de una tarea de presentación de información, los códigos fuente de estos archivos son accesible mediante el enlace:

https://github.com/TesisAWSML/API/tree/master/traffic_platform/webapp/templates.

Tabla 31.

*Servicios REST del proyecto*

| Archivo | Descripción | Tecnologías utilizadas |
|---|---|---|
| **charts.html** | Contiene los controles para seleccionar las fechas de inicio y de final de una búsqueda por calle. A | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Libreria Chart.min.js |
| **class_form.html** | Contiene los diferentes elementos para poder evaluar una calle según parámetros. Contiene 2 menús desplegables donde se configuran los parámetros de evaluación. | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Libreria Chart.min.js |

**CONTINÚA →**

| Archivo | Descripción | Tecnologías utilizadas |
|---|---|---|
| **elements.html** | Contiene un mapa donde se dibujan los diferentes elementos del mapa que se muestran en la página de inicio. Su propósito es de evaluación y no está disponible para los usuarios sin autenticación. | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Librería Flask-GoogleMaps<br>API Google Maps |
| **home.html** | Contiene el mapa interactivo donde se dibuja el mapa y las líneas de tráfico. Además, se presentan dos contenedores donde se puede especificar la fecha del mapa interactivo con un límite de 90 días en el futuro desde la fecha actual. | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Librería Flask-GoogleMaps<br>API Google Maps |
| **polylines.html** | Contiene un mapa interactivo donde es posible asignar puntos para crear líneas dentro de la plataforma, para su presentación en el mapa principal. | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Librería Flask-GoogleMaps<br>API Google Maps |
| **sensor.html** | Contiene un mapa interactivo que permite crear un polígono a modo de sensor virtual, dicho polígono servirá para buscar muestras de velocidad dentro del sensor y obtener un valor por cada sensor. | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Librería Flask-GoogleMaps<br>API Google Maps |
| **_chart.html** | Contiene un gráfico dinámico que permite mostrar diferentes tipos de gráficos en función del tiempo, se utiliza como un DOM para la página de gráficos por calle. | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Libreria Chart.min.js |
| **_eval_chart.html** | Contiene un gráfico dinámico que permite la presentación de la variable de velocidad en función del tiempo. Se utiliza como un DOM para la página de clasificación. | Lenguaje HTML versión 5<br>Librería Bootstrap<br>Jinja Templates (Flask Framework)<br>Libreria Chart.min.js |

## Interfaces para el cliente Web

### Mapa de tráfico vehicular

Es la página principal del conjunto de interfaces creadas para el cliente web. Esta página ha sido creada utilizando el lenguaje HTML en su versión 5, además utiliza algunas tecnologías propias del *framework* Flask para Python. Por ejemplo, permite la creación de funciones y algoritmos dentro de la misma página web, o la presentación de textos, tablas y figuras dinámicas gracias a la compatibilidad con las plantillas de Jinja2.

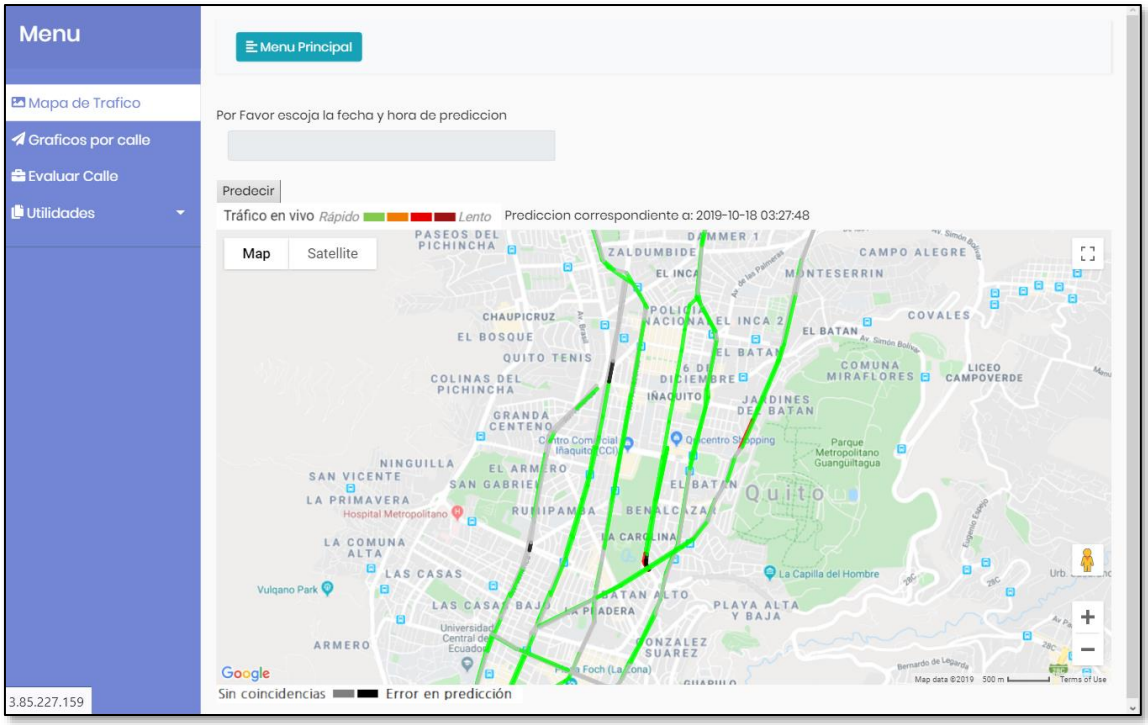



***Figura 67.*** Página principal de las pantallas de la interfaz web

Se puede apreciar que la página contiene un menú lateral con todos los enlaces a las páginas secundarias de la interfaz web. En la parte central posee un objeto *DatePicker* que permite seleccionar la hora y fecha de la predicción hasta con 3 meses en el futuro. El mapa que se muestra

en el interior contiene diferentes polilíneas que se dibujan sobre la calle de esta forma se muestran los diferentes estados de las calles gráficamente.

Para la renderización y despliegue del mapa se utiliza la librería *flask_googlemaps* que permite manipular el mapa y crear formas básicas en el mismo. Sin embargo, es necesario contar con una clave de API para que los mapas se rendericen adecuadamente.

**Clasificación de calle según parámetros**

Esta pantalla posee al igual que la pantalla principal un menú de navegación lateral, además es adaptable según en tamaño de pantalla debido a la utilización de la librería de Bootstrap. Poseo un conjunto de elementos de entrada que permiten definir un valor para cada parámetro de clasificación. Una vez que el usuario ha realizado la petición de clasificación se procede a solicitar que el proceso se realice en los servidores de la plataforma BigML debido a que de esta forma el proceso se realiza con un menor tiempo.

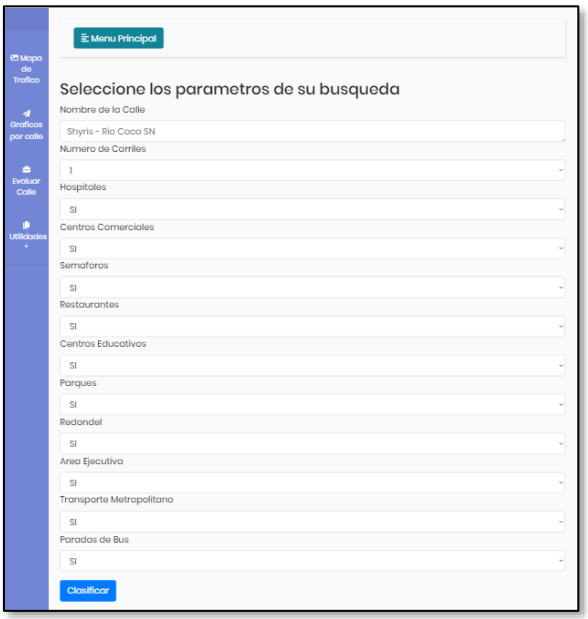



***Figura 68.*** Presentación de la interfaz de clasificación para el cliente web

Una vez la clasificación esta lista, el servidor devuelve la información de predicción al servidor. Mismo que la proceso y comprime toda la respuesta como un archivo HMTL que es devuelto al navegador para que sea renderizado.

La interfaz de respuesta posee la curva de velocidad a la que responden los datos proporcionados, es decir el servidor recibe los parámetros de la calle y el servidor devuelve el comportamiento en el tiempo de la intersección. Además, se proporciona una calle a la que se parece y el nivel de confianza de la clasificación.

**Gráficos de velocidad según calle**

Esta interfaz permite obtener graficas de velocidad de una intersección según demanda. La pantalla contiene el menú lateral similar a las otras páginas web además es redimensionable según el tamaño de pantalla del dispositivo que este mostrando la interfaz.

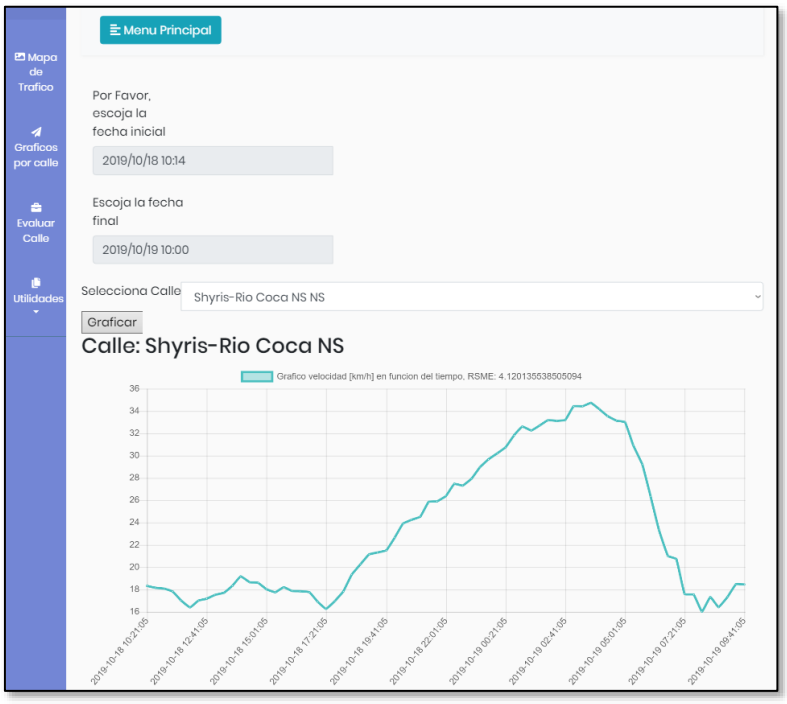



***Figura 69.*** Pantalla de gráficos por calle

Posee dos objetos tipo *DatePicker* que son utilizados para definir fechas de inicio y final del gráfico, además la interfaz posee un menú desplegable que se usa para seleccionar el grafico de la intersección que se va a mostrar. Al obtener la curva de velocidad la interfaz además proporciona el error medio cuadrático que fue calculado al momento de la creación del modelo de la intersección.

**Herramientas adicionales**

El sistema posee tres herramientas que permiten control de tres funciones no pensadas para un usuario estándar, han sido específicamente pensadas para administradores y desarrolladores.

El sistema permite crear sensores y calles en el sistema. Siempre y cuando se tengan las credenciales del sistema, debido a que estas herramientas están protegidas mediante credenciales.

a. Interfaz para la creación de sensores virtuales

La interfaz de creación permite seleccionar 4 puntos vértices, estos puntos se utilizan para definir un área rectangular dentro de la cual la plataforma buscara por muestras de velocidad. De esta forma es posible verificar las velocidades promedio de los vehículos en áreas específicas. Este sensor virtual (polígono rectangular en el mapa) posteriormente define una intersección o área de modelamiento.

Si el número de muestras dentro del polígono tiende al infinito, es posible obtener un conjunto de datos que se separen a intervalos definidos de tiempo.

Es decir, podemos obtener una curva definida para cada intervalo de tiempo especificado. La presentación de la página y el almacenamiento del sensor en la base de datos son realizados en el orquestador de la plataforma. Inicialmente el usuario solicita la página de creación de sensores, el

orquestador verifica la solicitud y devuelve una página con el mapa y los elementos de entrada. El usuario debe definir 4 puntos (área de verificación) que representan un polígono.

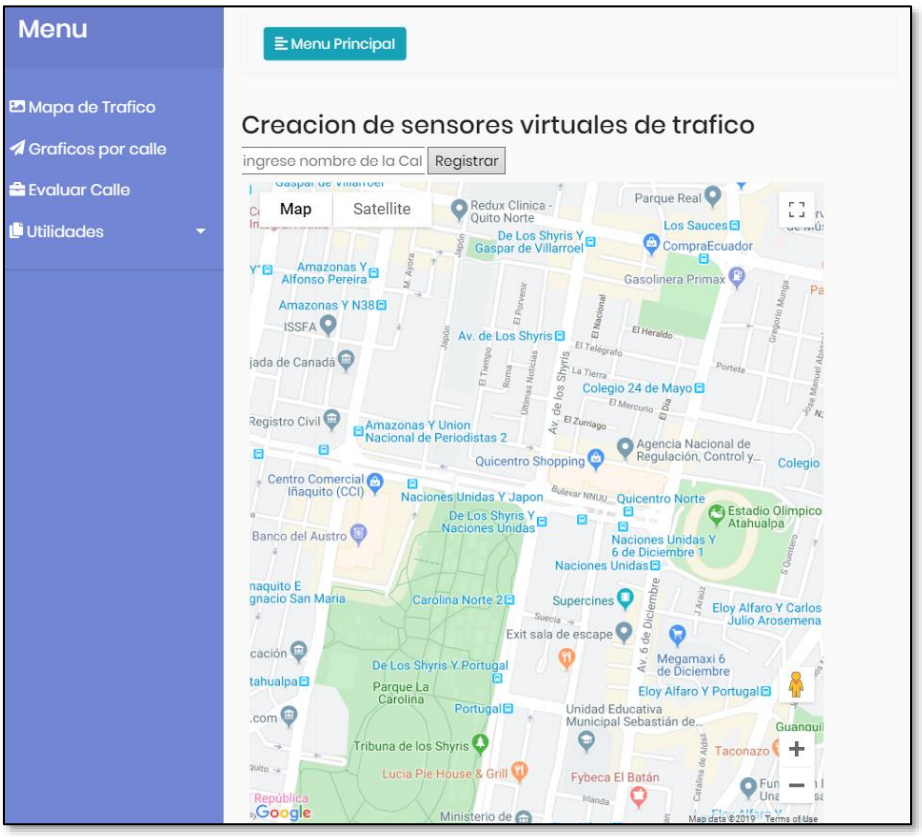



***Figura 70.***. Presentación de la interfaz de creación de sensors

Una vez definido el polígono, el usuario debe proporcionar un nombre de calle. Al presionar en registrar el sistema guarda los datos en la instancia de Amazon RDS utilizando la librería mysql.connector en la tabla de sensores.

b. Interfaz para la creación de calles en el mapa

Desafortunadamente el sistema no permite crear dinámicamente las rutas de una calle que posee muchas curvas, debido a estos es necesario definir algunos puntos que ayuden a la plataforma a mostrar la información gráficamente. Es por ello que es posible crear dinámicamente calles y rutas

con colores dependiendo de la necesidad. Como se ha mencionado es una herramienta de depuración y desarrollo.

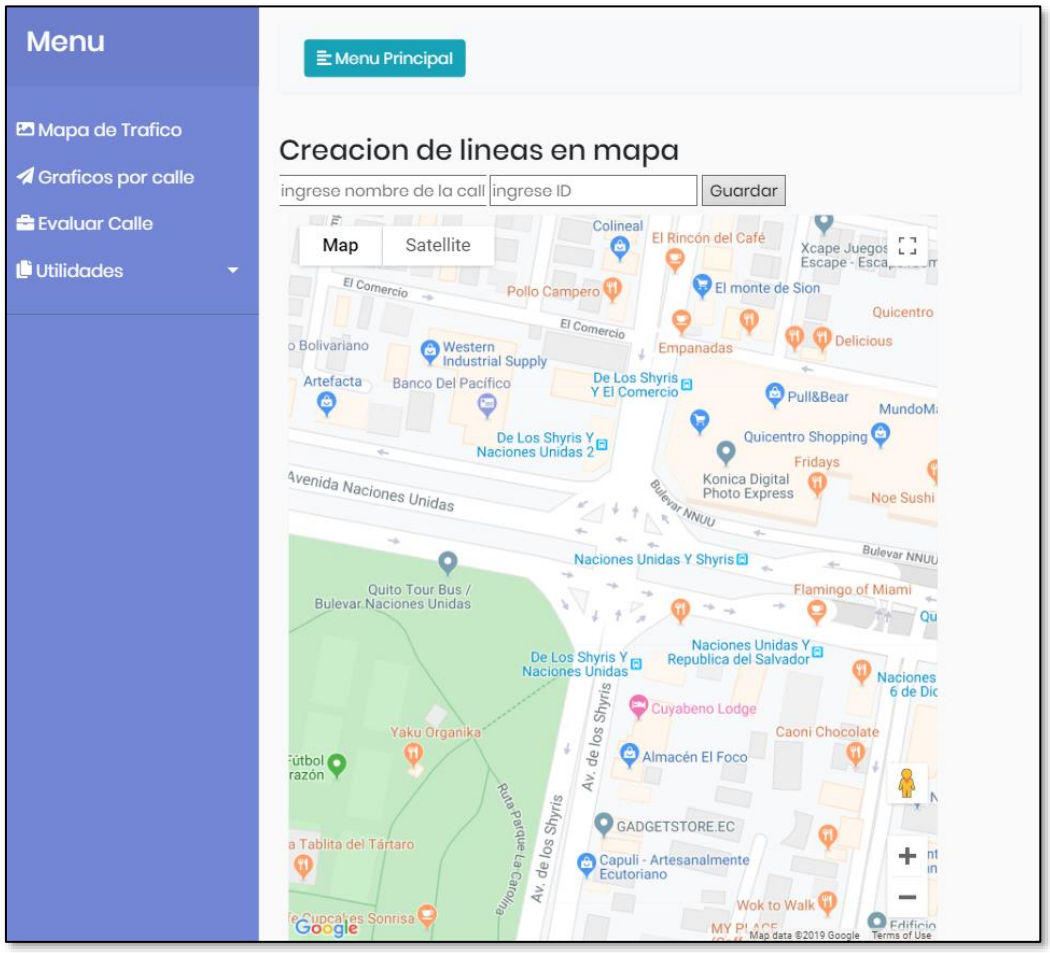



***Figura 71.*** Interfaz de la plataforma para crear calles dentro de la plataforma

La presentación y almacenamiento de la interfaz de creación de líneas está a cargo del orquestador. Al igual que en otros casos, si el usuario solicita la página de creación de sensores, el orquestador procesa la solicitud y devuelve un archivo HTML que contiene el mapa y los controles de la interfaz. El usuario debe definir dos puntos (inicio y fin) de la línea. Posterior es necesario asignar un número de índice según corresponda. Al presionar registrar el sistema almacena la

información de la línea utilizando la librería *mysql.connector* en la tabla de líneas de la instancia de Amazon RDS.

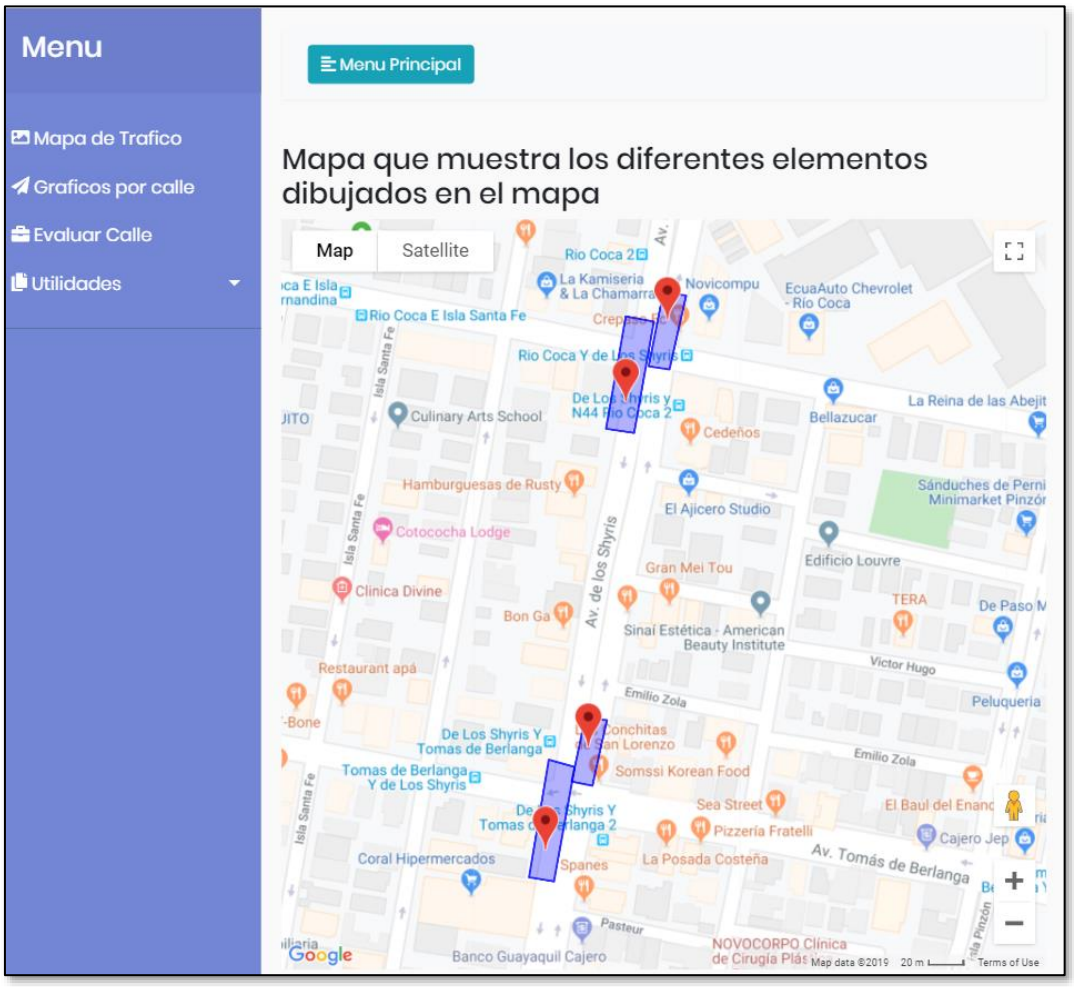



***Figura 72.*** Pantalla que permite mostrar los sensores (polígonos azules) y los puntos de toma de datos

**Integración con servicio de Televisión Digital**

Se realiza por medio de Eclipse con lenguaje NCL y LUA el programa que va a permitir ejecutar el despliegue del menú, botones y así como también el envío de la información al servidor para que nos devuelva el resultado de la predicción. A continuación Tabla 32 se describe los archivos que conforman al programa:

Tabla 32.

*Archivos – módulos utilizados en el programa para TVD*

| ARCHIVOS – MÓDULOS | DESCRIPCIÓN |
|---|---|
| **ConnectorBase.ncl** | Este archivo o base destaca también como uno de los más importantes y el cual no debe faltar en ningún programa; ya que se encarga de establecer las relaciones genéricas que serán utilizados en un documento NCL. |
| **Interfaz.ncl** | Es la estructura principal del programa en donde se fijan las imágenes, videos que aparecerán y se hace el llamado de los demás archivos ncl y lua. Aquí los parámetros más importantes o nodos son los de entrada y salida. El nodo de entrada tiene propiedad text en donde se ingresa bajo el formato (AA-MM-DD/HH:MM/#intersección) lo que se quiere predecir y el ancla *select* se inicia cuando el usuario pulsa *ENTER* (CONSULTAR). El nodo de salida de igual forma tiene propiedad *text*, el cual su contenido o valor es el que se mostrará en pantalla. |
| **Input.lua** | En este módulo se fija la escritura del texto que por medio de las teclas del computador que simulan los botones del control remoto para TV, se pueda ingresar los caracteres tales como letras, símbolos y números, es decir el teclado numérico del '0-9'. Se puede asignar las teclas de UPPER, DOWN para movimiento del usuario en el menú y las teclas LEFT para borrar y ENTER para seleccionar. |
| **Output.lua** | Se encarga de lo que recibe mediante tcp.lua se muestre en pantalla bajo las configuraciones de tamaño y tipo de letra que se asignó. |
| **tcp.lua** | La comunicación se realiza a través de TCP es la más importante; mediante el uso clase de eventos 'tcp'. El módulo tcp.lua admite el control de llamadas asíncronas, permitiendo la comunicación sea programada secuencialmente.<br>Es para recibir una función que puede contener los siguientes comandos:<br>Tcp.connect: recibe la dirección y el puerto de destino.<br>Tcp.send: recibe una cadena con el valor que se transmite.<br>Tcp.receive: devuelve una cadena con el valor que se recibe.<br>Tcp.disconnect: sierra la conexión. |
| **consulta.lua** | Permite la conexión hacia el host del servidor y además permite el envío de datos que complementan la información para que el servidor nos responda con la información que requiere el cliente TVD. Tales como en nuestro caso el envío de fecha, hora e ID de las intersecciones a consultar el estado del tráfico. |

Es importante realizar una revisión del tráfico que se genera en una consulta al servidor mediante la herramienta Wireshark; con esto podremos validar que es lo que se requiere colocar en GET para el envío y conexión en el archivo consulta.lua. Como se observa Figura 73, se captura el tráfico en una consulta al servidor y se analiza el resulta en *Hypertext Transfer Protocol* - HTTP.


***Figura 73.*** Captura tráfico Wireshark – HTTP

Con esta información de cabecera se coloca las líneas de código en consulta.lua lo cual permitirá una conexión exitosa al servidor, el envío de información y recepción de datos por parte del servidor. Dicha información que contiene la cabecera que permitirá conectarse al servidor con nuestro programa de Ginga se muestra en la Figura 73; se copia la misma información que nos proporciona Wireshark. En la Figura 73 también se puede observar que se recibe una respuesta por parte del servidor con ccódigo 200 (significado conexión exitosa) y además muestra los campos necesarios para la presentación de la información en la interfaz de TVD (date, id, name, speed).

```
hostIP = '3.85.227.159'
tcp.connect(hostIP,80)

local url = 'GET /api/v1/consulta/'..evt.value..' HTTP/1.1\r\n'
tcp.send(url)
tcp.send('Host:3.85.227.159\r\n')
tcp.send('Connection: keep-alive\r\n')
tcp.send('Cache-Control: max-age=0\r\n')
tcp.send('Upgrade-Insecure-Requests: 1\r\n')
tcp.send('User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.79 Safari/537.36\r\n')
tcp.send('Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
tcp.send('Accept-Encoding: gzip, deflate\r\n')
tcp.send('Accept-Language: es-ES,es;q=0.9,en;q=0.8\r\n')
tcp.send('\r\n')
```

***Figura 74.*** Código enviado para la conexión exitosa con el servidor

Ahora para receptar la información que nos da como respuesta el servidor se genera un archivo en blanco .txt, el mismo que se guarda en la carpeta media de nuestro programa y es aquí donde se copiará la información que nos envía el servidor con los campos de date, id, name, speed como se muestra en la Figura 74. Para que se pueda leer (r), abrir (a) y escribir (w) se ocupa la librería 'io' en LUA; esto se realiza con las siguientes líneas de código que se observa en la siguiente Figura 75. Para presentar los resultados que se han almacenado únicamente se realizará el llamado de este archivo .txt en el módulo principal del programa.

```
result=tcp.receive()
print('Dato recibido')

file = io.open("media/respuesta.txt","w")
            file:write(result)
            file:close()
```

***Figura 75.*** Código para escribir resultados receptados en un archivo .txt

## Interfaz de Televisión Digital

A continuación, se muestra la implementación en televisión digital.


***Figura 76.*** Pantalla inicial implementado


***Figura 77.*** Despliegue menú calles

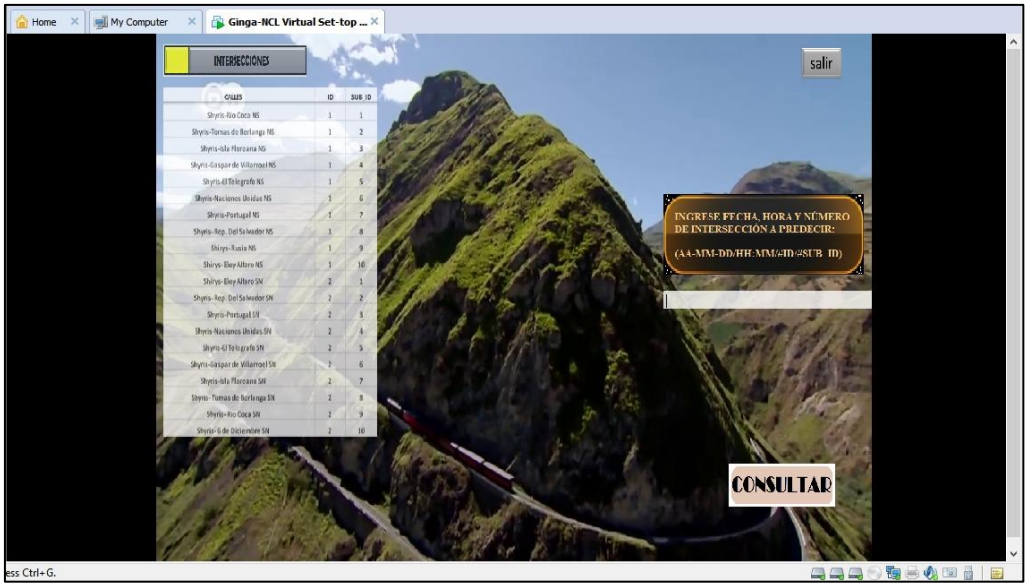



***Figura 78.*** Intersecciones y campo a consultar

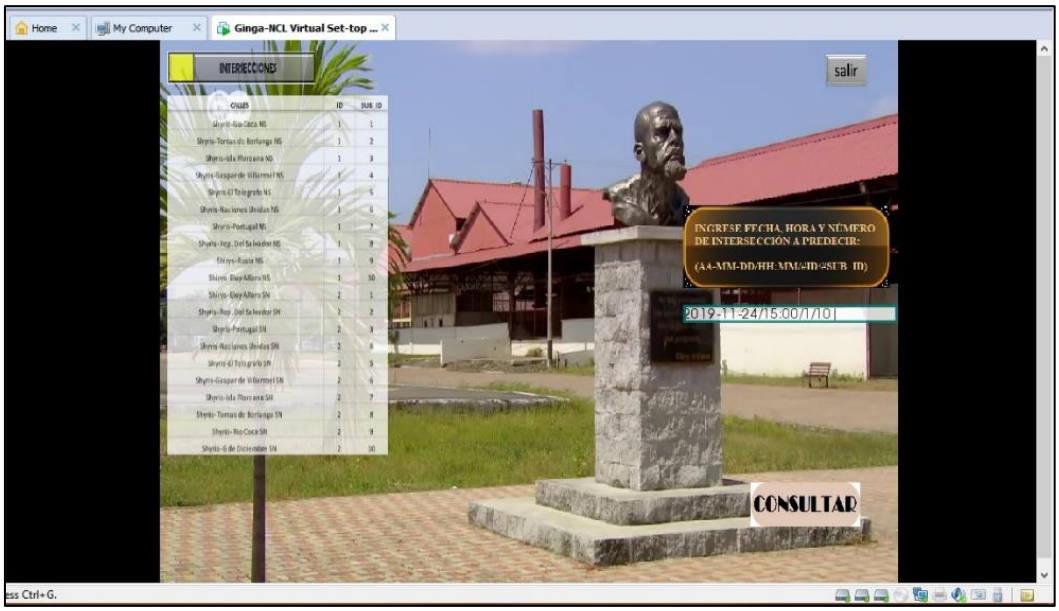



***Figura 79.*** Campo bajo el formato solicitado

En la Figura 76 se puede observar el botón que siempre aparecerá, estará ahí desplegado en la pantalla para cuando se necesite se lo pueda seleccionar en cualquier momento por el usuario.

El menú que aparecerá para predecir es de las cuatro siguientes calles: Av. Shyris, La Patria, La Colón y 12 de octubre como se muestra en la Figura 77. Como siguiente se puede observar en la Figura 78 en esta pantalla ya muestra las intersecciones que se puede escoger y adicional el campo de texto a llenar con la fecha, hora e intersección para que se realice la consulta de predicción al servidor.

El ingreso del formato el cual se va a enviar a la consulta al servidor debe ser bajo las siguientes especificaciones: Año/Mes/Día/Hora/#Intersección. El número de intersección se lo puede colocar de acuerdo al número que va acompañado con la leyenda de las intersecciones en el menú desplegado en la pantalla. Como se puede observar en la Figura 79 se ingresa el formato solicitado y por siguiente se coloca consultar para que enseguida aparezca la información de predicción solicitada.

# CAPÍTULO 5

# PRUEBAS Y RESULTADOS

## Pruebas de carga y rendimiento con JMeter

Para visualizar la página web de predicción de tráfico se puede acceder con la IP proporcionada por la plataforma AWS colocándola como URL desde cualquier navegador de cualquier dispositivo (celular o PC) como se muestra en la siguiente Figura 80.

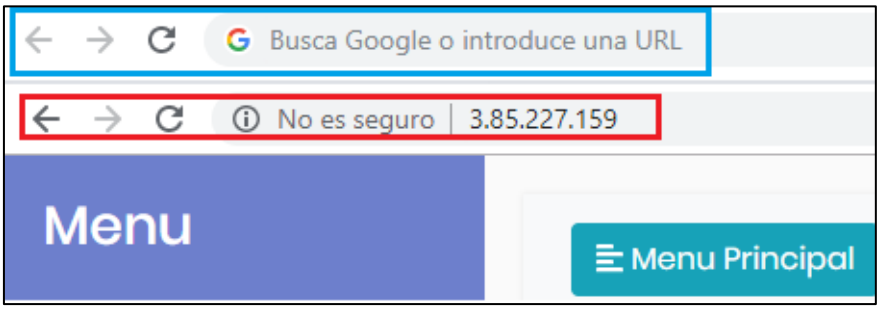



***Figura 80.*** Ingreso de IP página web en navegador

## Pruebas del Cliente Web

Como se observa en la figura anterior por medio de la IP (http://3.85.227.159) se puede realizar pruebas de testeo de la arquitectura elaborada. Para las pruebas de rendimiento y carga se ocupa el software JMeter; se realiza para el caso de seis escenarios simulando 50, 150, 250, 350, 500 y 550 usuarios concurrentes (llamado hilos en JMeter) realizando peticiones durante un periodo de 60 segundos al servidor (HTTP – GET). Las configuraciones se realizan como se muestra en la Figura 81, de acuerdo a lo que se requiere poner a prueba en nuestro aplicativo Web.

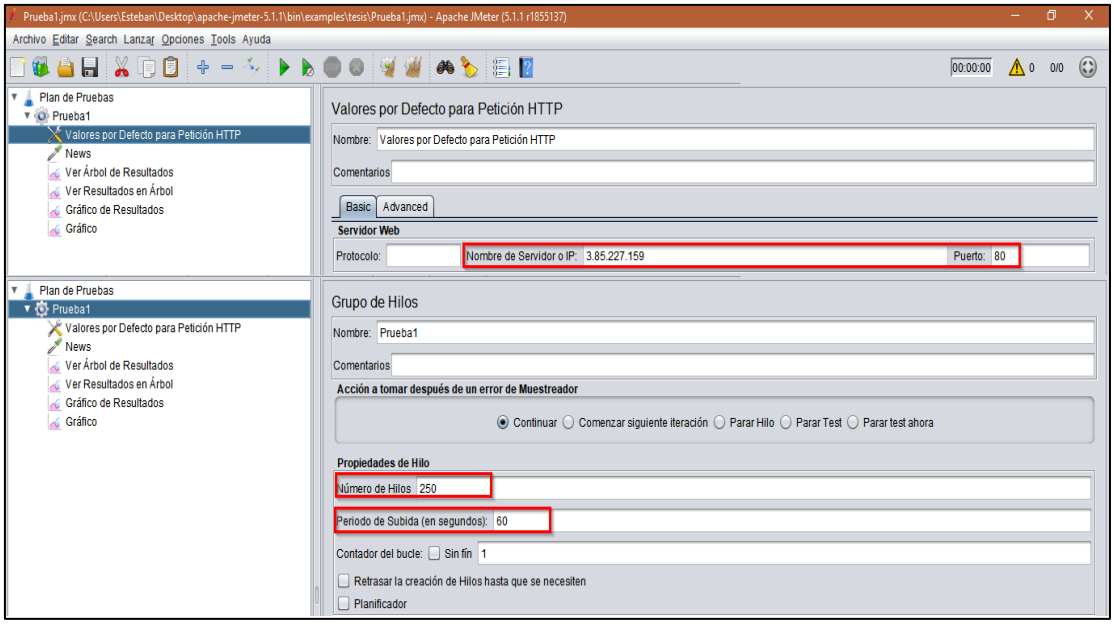
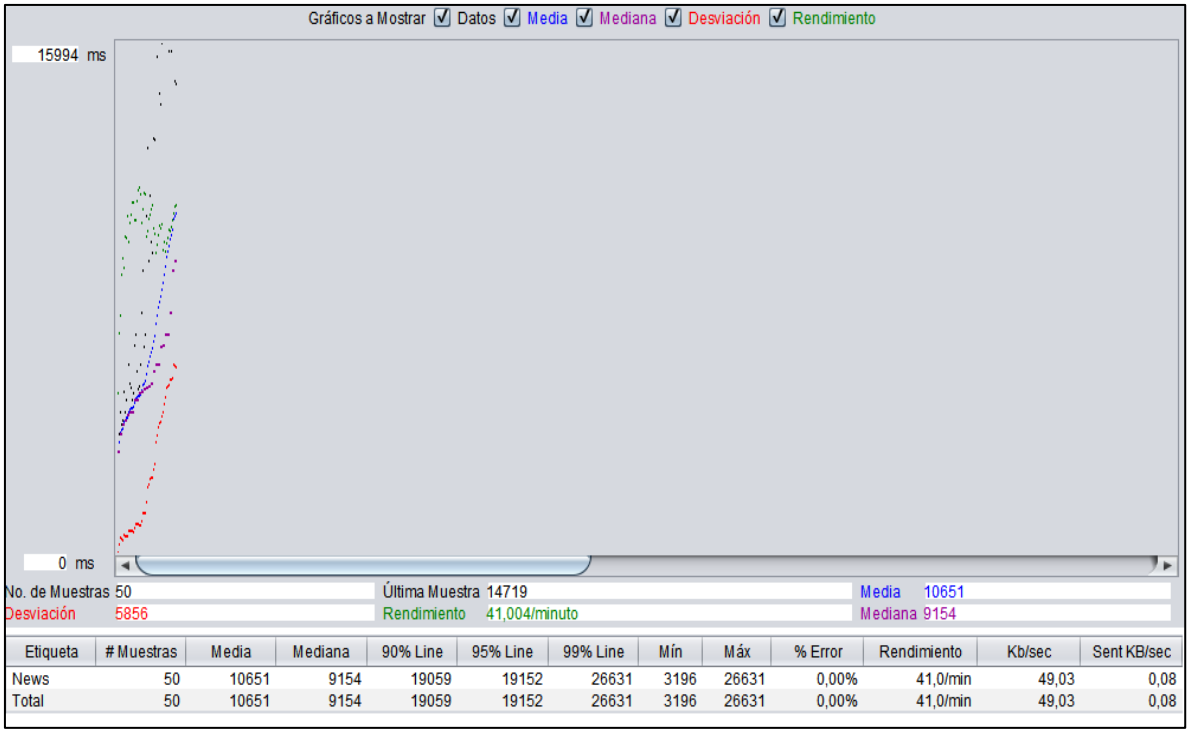


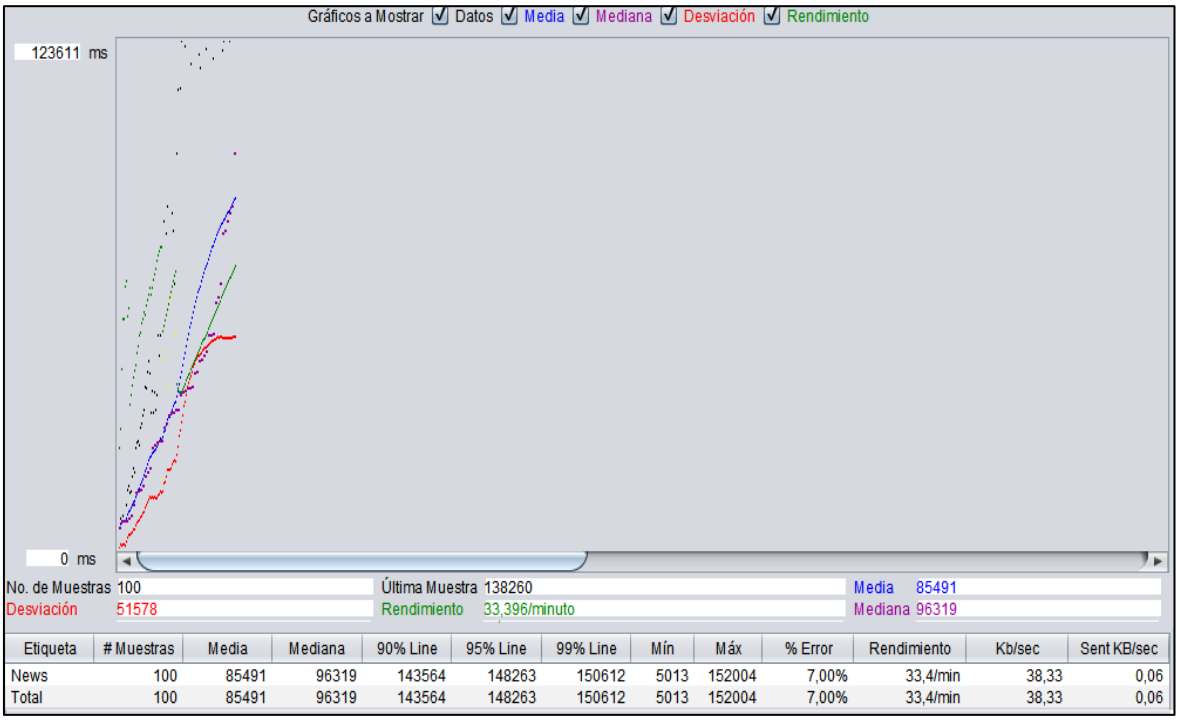

***Figura 81.*** Configuración JMeter




| Etiqueta | # Muestras | Media | Mediana | 90% Line | 95% Line | 99% Line | Mín | Máx | % Error | Rendimiento | Kb/sec | Sent KB/sec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| News | 50 | 10651 | 9154 | 19059 | 19152 | 26631 | 3196 | 26631 | 0,00% | 41,0/min | 49,03 | 0,08 |
| Total | 50 | 10651 | 9154 | 19059 | 19152 | 26631 | 3196 | 26631 | 0,00% | 41,0/min | 49,03 | 0,08 |

***Figura 82.*** Escenario con 50 usuarios

Como se puede observar en la Figura 82 existe un error de 0% indicando que las peticiones realizadas al servidor fueron exitosas durante las 50 muestras ejecutadas; y existe en media 10651ms (equivale 10 s) de tiempo para cada solicitud. En cambio, para la Figura 83 se puede observar que ya existe un error de 7% y el tiempo medio de solicitud es alta de 85491ms (equivale 85 s).




| Etiqueta | # Muestras | Media | Mediana | 90% Line | 95% Line | 99% Line | Mín | Máx | % Error | Rendimiento | Kb/sec | Sent KB/sec |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| News | 100 | 85491 | 96319 | 143564 | 148263 | 150612 | 5013 | 152004 | 7,00% | 33,4/min | 38,33 | 0,06 |
| Total | 100 | 85491 | 96319 | 143564 | 148263 | 150612 | 5013 | 152004 | 7,00% | 33,4/min | 38,33 | 0,06 |

***Figura 83.*** Escenario con 100 usuarios

En la siguiente Figura 84, con la prueba de 100 usuarios el resultado del muestreador para una muestra se puede apreciar que el código de respuesta es 200 indicando que la conexión fue exitosa y, además un mensaje de respuesta OK. Se realiza una comparación de la fecha y hora (2019-10-31 00:46) en la que fue realizado el muestreo exitoso con el resultado de la Figura 85 que muestra la respuesta del orquestador a las peticiones de URL siendo las mismas y exitosas en ambos casos.

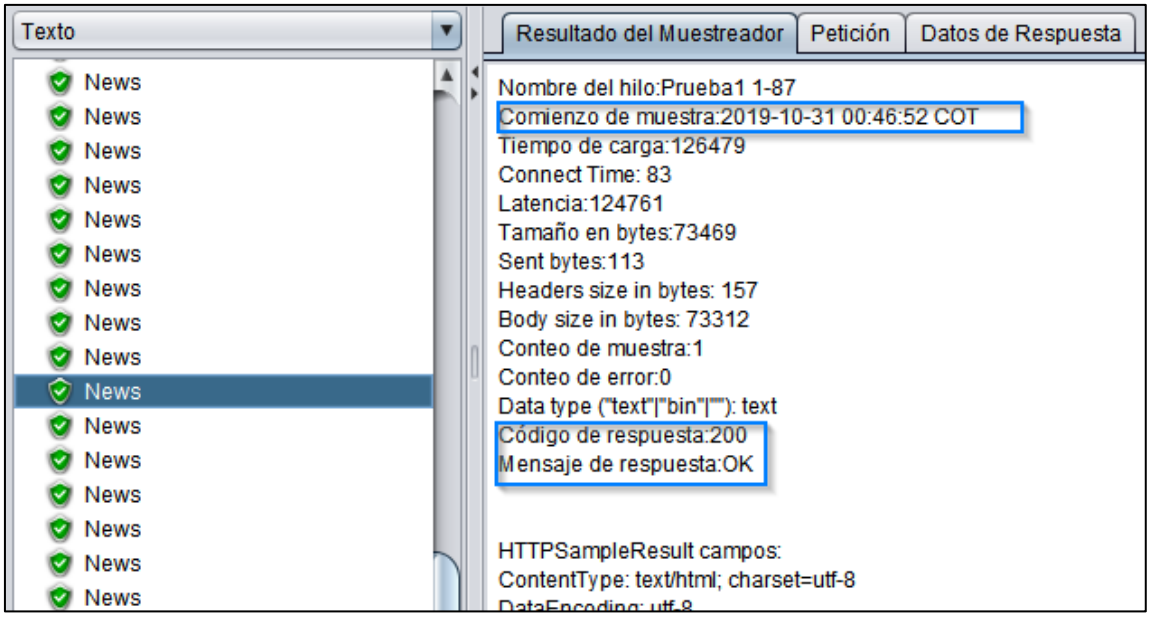



***Figura 84.*** Resultado del Muestreador (100 usuarios)

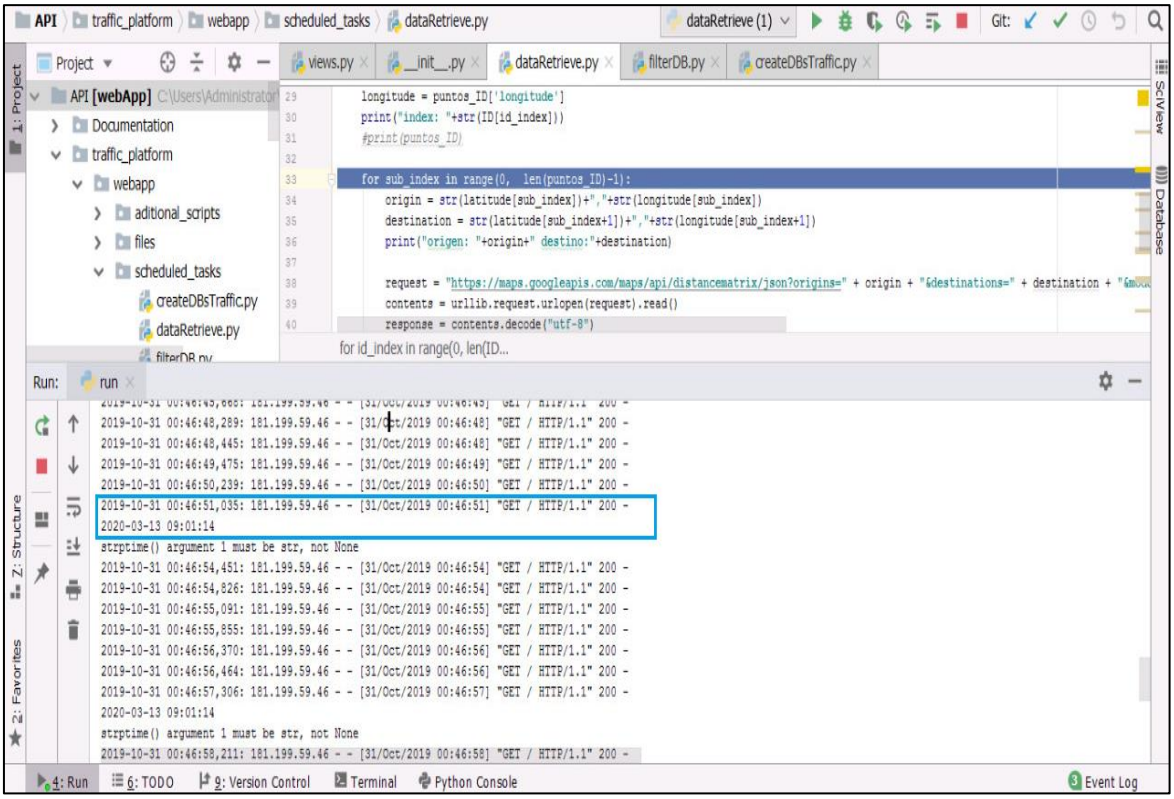



***Figura 85.*** Respuesta del orquestador a peticiones de URL

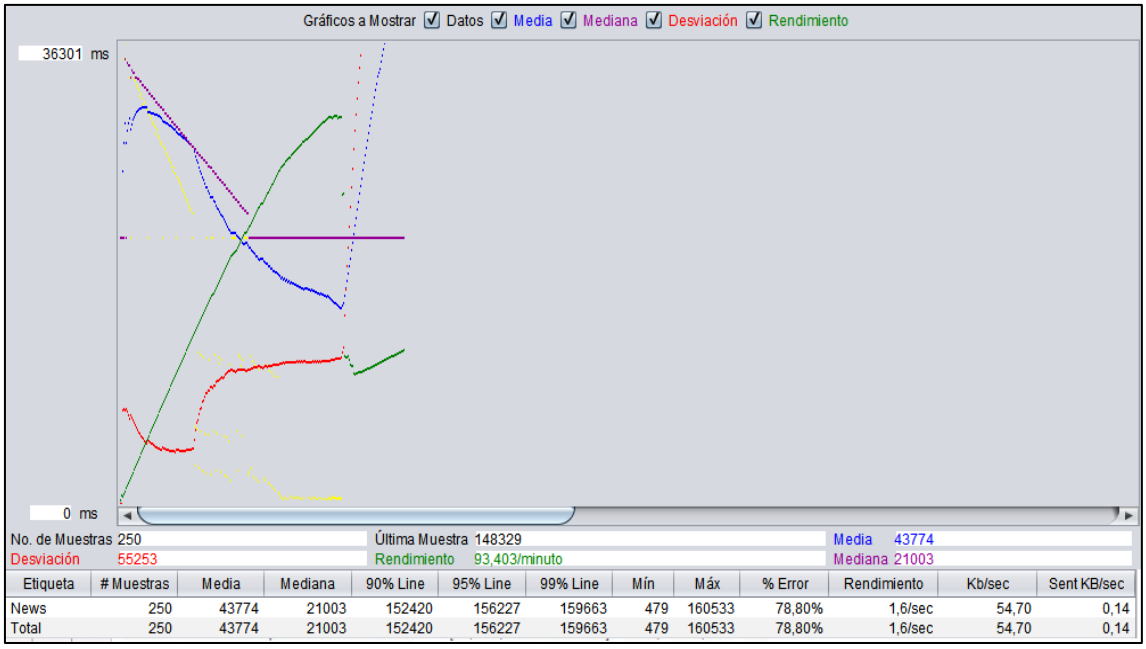



***Figura 86.*** Escenario con 250 usuarios

En la Figura 86 para un número de 250 usuarios se puede apreciar un error del 78.8% y en media de tiempo para peticiones se eleva el tiempo a un 43774 ms. Motivo por el cual en la Figura 87 se analiza uno de los mensajes de error (identificado con el código 500) y al no responder a la petición el servidor refleja un tiempo de latencia alto en este ejemplo de esta muestra tarda 60137ms.

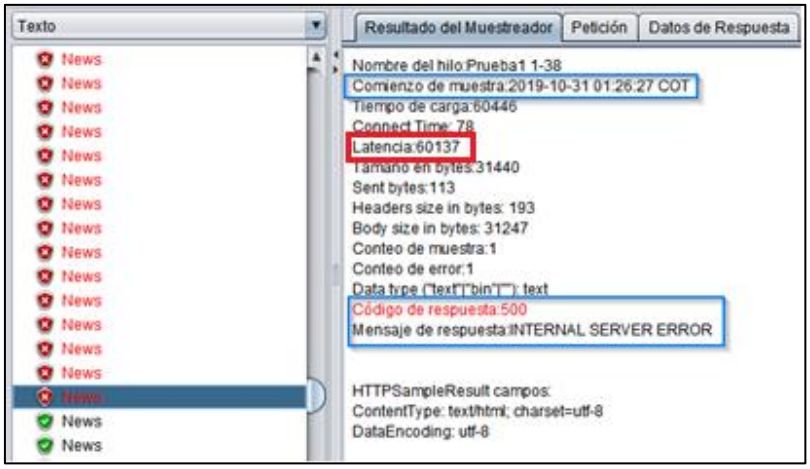



***Figura 87.*** Resultado del muestreador (250 usuarios)

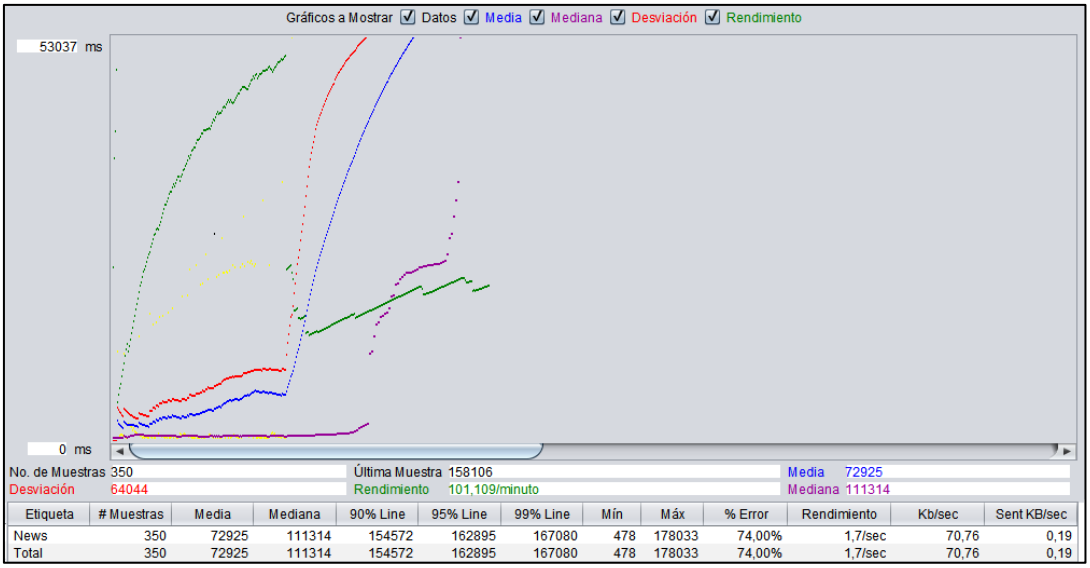



***Figura 88.*** Escenario con 350 usuarios

En la Figura 88 se observa que para un número de 350 usuarios existe una media de 72925 ms y un error de 74,0%. Se continúa incrementando el número de usuarios y así poniendo a prueba el tiempo de respuesta del servidor a estas peticiones; por lo que en la Figura 89 con 500 usuarios se muestra una media de 62147 ms y un error de 87,4%.

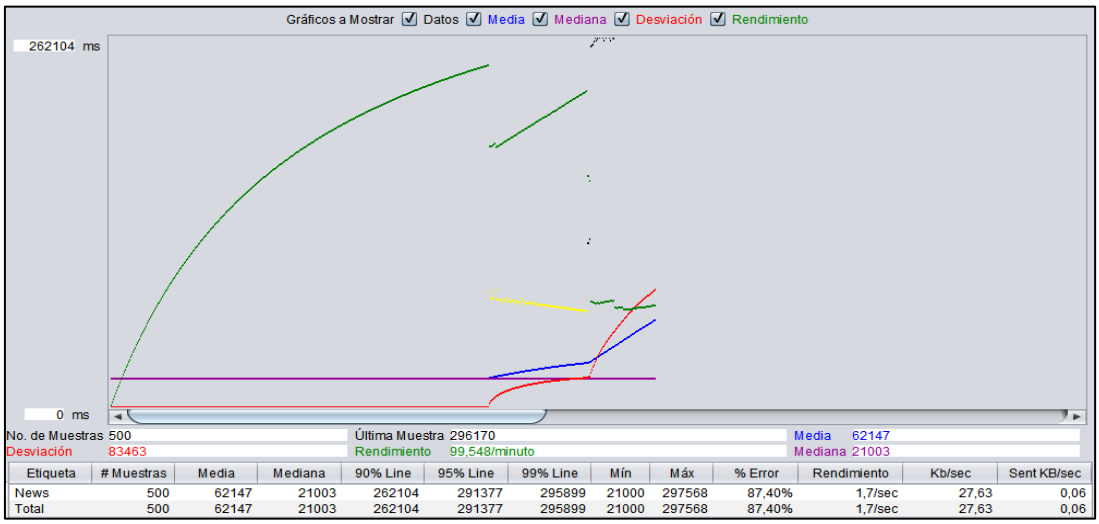



***Figura 89.*** Escenario con 500 usuarios

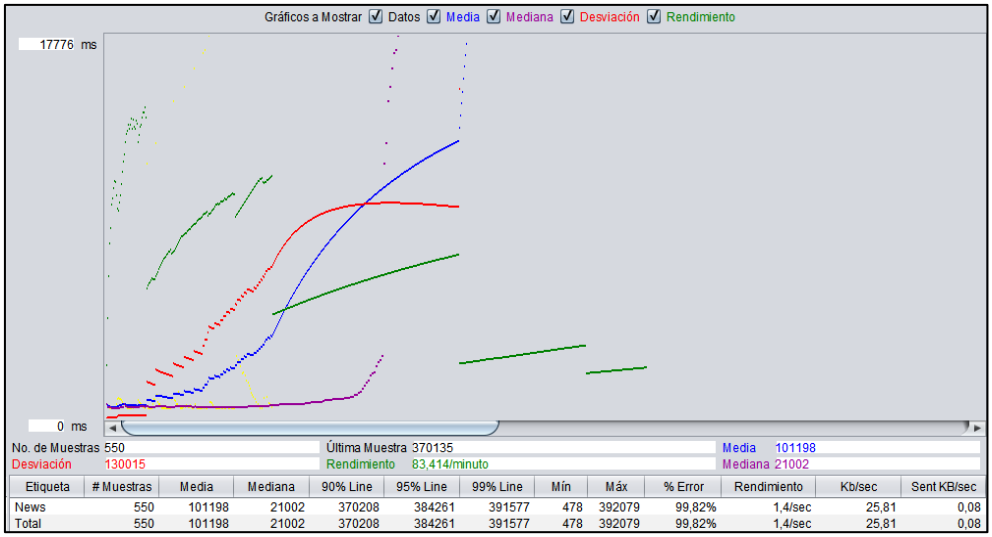



***Figura 90.*** Escenario con 550 usuarios

Por último, en la Figura 90 se observa el resultado de las pruebas con 550 usuarios y nos da un valor en media de 101198 ms (equivale 1,68 min) y un error de 99,82%. Al finalizar esta prueba con 550 usuarios por el error que nos da casi un cien por ciento, se intenta el ingreso al servidor con la IP en un navegador el cual falla evidentemente como se observa en Figura 91.

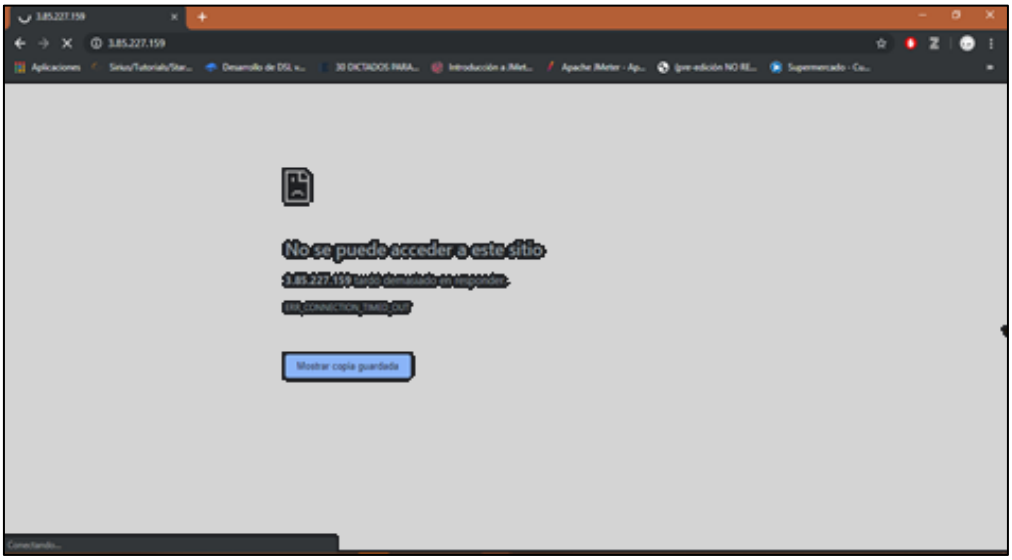



***Figura 91.*** Error excede tiempo de conexión

Tabla 33.

*Resumen resultados pruebas con JMeter.*

| # Muestras | Media (ms) | % Error | Rendimiento |
|---|---|---|---|
| **50** | 10651 | 0,00 | 41,0/min |
| **100** | 85491 | 7,00 | 33,4/min |
| **250** | 43774 | 78,8 | 1,6/sec |
| **350** | 72925 | 74,00 | 1,7/sec |
| **500** | 62147 | 87,40 | 1,7/sec |
| **550** | 101198 | 99,82 | 1,4/sec |

En la Tabla 33 se tiene un resumen de resultados de las pruebas realizadas con el incremento de usuarios y aquí se puede observar que conforme se incrementan los mismos, el error aumenta al igual que el tiempo medio de respuesta a las peticiones del servidor. El máximo error al que se llega es de 99,8% en el cual se satura al servidor evitando que por unos segundos no se pueda acceder.

**Comparación predicción Cliente Web y Google Maps**

En la siguiente Figura 92 se puede apreciar la predicción que realiza Google Maps, en este caso para un día Jueves a las 06H00. En este aplicativo también cuenta con la funcionalidad para predecir el tráfico de un día típico con las opciones a elegir de lunes a domingo en el rango de hora desde las 06H00 hasta las 22H00. Al igual que en el cliente Web, se puede comprender el estado del tráfico mediante la escala de colores como se detalla en la Tabla 34; siendo el verde un tráfico fluido (rápido), continuando con el naranja que es un indicador de que el tráfico cuenta con un leve estancamiento y conforme aumenta la densidad de tráfico, el color rojo se intensifica significando estancamientos fuertes en las avenidas. De esta manera se hace fácil la comprensión para el usuario e identifica rápidamente vías congestionadas.

Tabla 34.

*Código de colores indicador de circulación de la carretera*

| Símbolo | Color | Descripción |
|---|---|---|
| | Verde | No hay retenciones de tráfico. |
| | Naranja | Hay una densidad de tráfico media. |
| | Rojo | Hay retenciones de tráfico. Cuanto más oscuro sea el rojo, más lenta será la velocidad de circulación. |

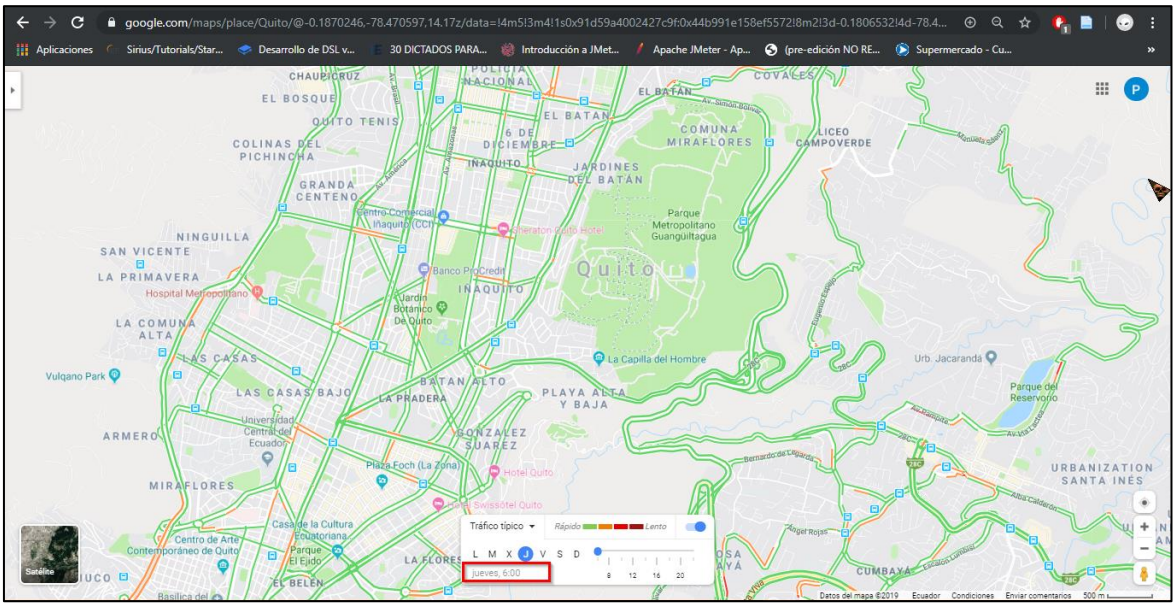



***Figura 92.*** Predicción de tráfico Google Maps. (2019-10-10 06H00)

En la siguiente Figura 93 se puede observar los resultados de predicción de la arquitectura diseñada en el presente proyecto. Se realizó las pruebas el día martes 01/10/2019 solicitando se prediga el estado de tráfico para la siguiente fecha y hora: jueves 10/10/2019 a las 06H00 y de manera cualitativa se puede observar que los resultados son similares a la que nos brinda Google Maps. En ambos casos la mayoría de las calles se pintan de color verde y naranja.

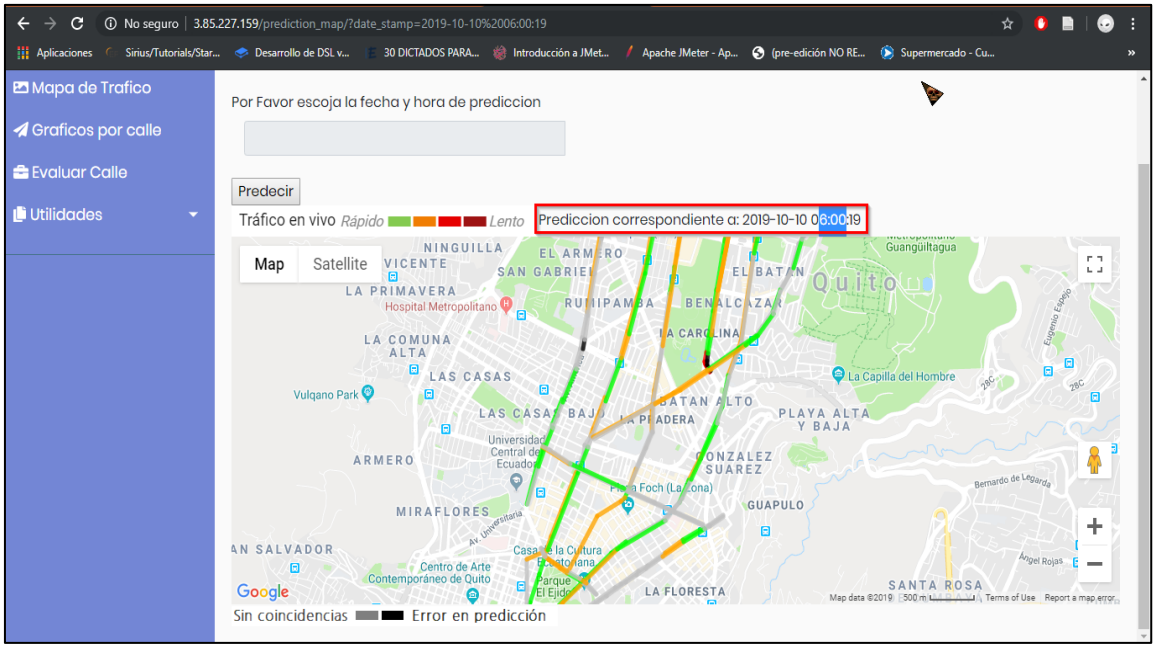



***Figura 93.*** Predicción tráfico Cliente Web. (2019-10-10 06H00)

En la Figura 94 la obtención de resultados sobre predicción de tráfico por medio de Google Maps al igual que el cliente Web como se muestra en la Figura 95. Los resultados se configuraron para el 10-10-2019 a las 17H40 a simple vista se puede encontrar similitudes de la información y concuerda con lo que habitualmente se conoce que esta hora pico donde la circulación vehicular es complicada en las calles más transitadas de Quito; tales como la Shyris, Patria, Colón, 12 de Octubre, 6 de Diciembre, etc. Cabe recalcar que igual al caso anterior se realizan las pruebas el 01/10/2019, con una anticipación de 9 días de anticipación.

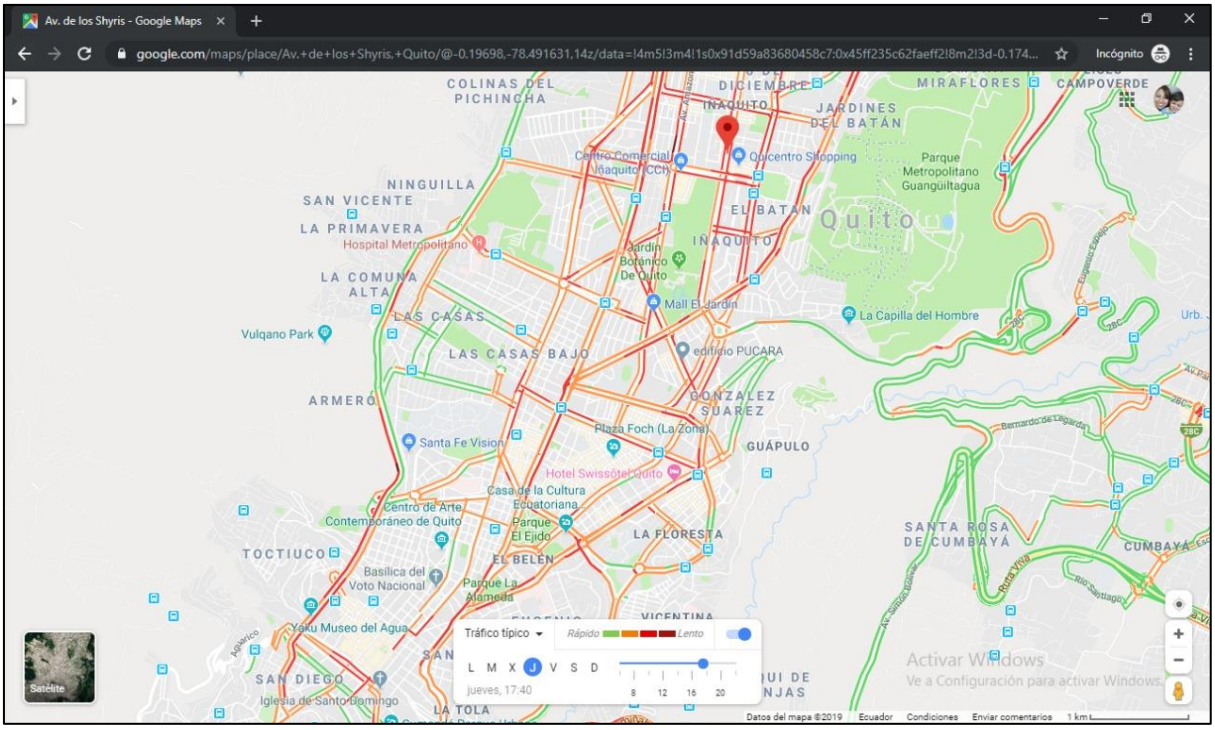



***Figura 94.*** Predicción de tráfico Google Maps. (2019-10-10 17H40)

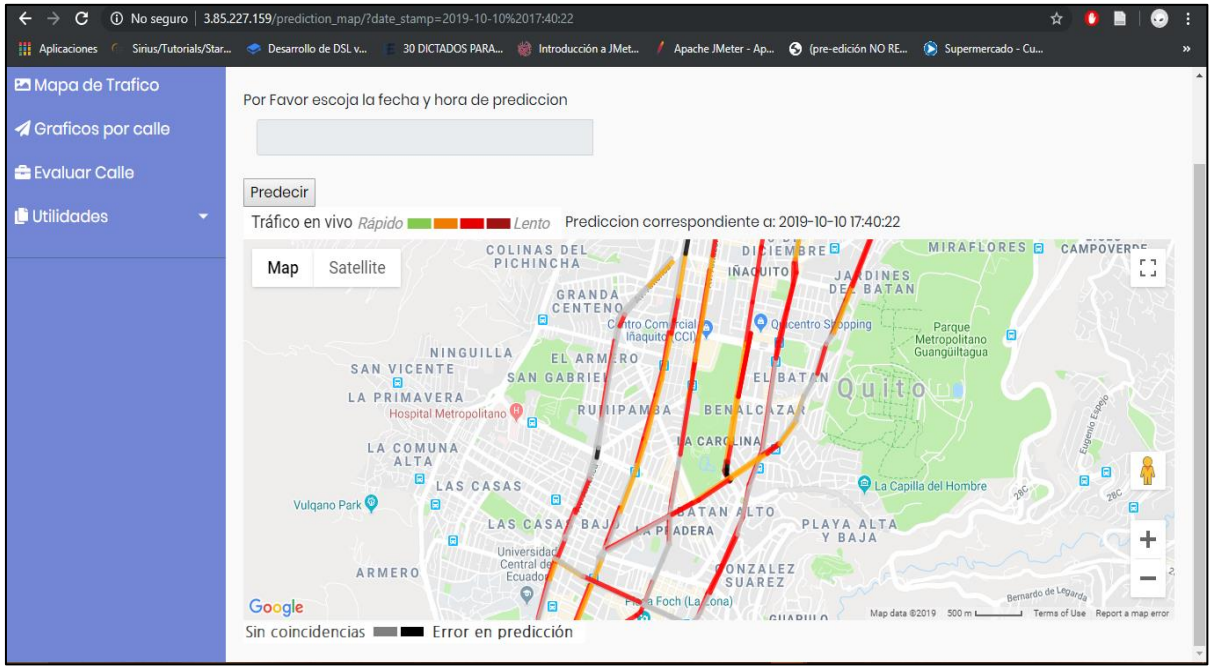



***Figura 95.*** Predicción tráfico Aplicativo Web. (2019-10-10 17H40)

**Comparación de predicción por BigML y Google Maps**

Las pruebas que a continuación se muestran fueron realizadas el 01/10/2019. En la siguiente Tabla 36 se presenta un análisis como ejemplo del resultado de la predicción de BigML, se extrae el estado de tráfico de cada intersección que constituye a la calle La Patria.

En la Tabla 35 se asigna símbolos de representación para los resultados de la comparación entre la predicción de los algoritmos de aprendizajes utilizados y los de Google Maps; siendo ‘A’ para los aciertos, ‘B’ para desaciertos y ‘0’ para cuando devuelven los algoritmos el color Gris que representa ‘Sin Coincidencias’ ya que no hay información para que se adapte a esta intersección.

Tabla 35.
*Representación resultados para la comparación.*

| REPRESENTACIÓN | SIGNIFICADO |
|---|---|
| **0** | Sin Coincidencias |
| **A** | Acierto |
| **B** | Desacierto |

Se puede observar que para los resultados presentado en la Tabla 36 y obtenidos del análisis BigML y Google Maps existe consistencias de un acierto y dos desaciertos.

Haciendo a un lado los resultados identificados por ‘0’ siendo los menos relevantes en este caso de la comparación realizada.

Tabla 36.

*Estado de tráfico calle La Patria – predicción BigML vs. Google Mpas. (2019-10-10 06H00)*

| Intersección (av. Shyris) | BigML | Google maps | Resumen |
|---|---|---|---|
| **Patria-12 de Octubre NS** | GRIS | VERDE | 0 |
| **Patria-6 de Diciembre NS** | VERDE | VERDE | A |
| **Patria-Amazonas NS** | NARANJA | VERDE | B |
| **Patria-9 de Octubre NS** | NARANJA | VERDE | B |
| **Patria-10 de Agosto NS** | GRIS | VERDE | 0 |
| **Patria-Portoviejo NS** | GRIS | VERDE | 0 |
| **Patria-18 de Septiembre NS** | GRIS | VERDE | 0 |

En la Tabla 37 se puede apreciar dos aciertos ‘2 A’ y un acierto ‘B’ y además 4 ceros de las intersecciones en las cuales no se pudo encontrar coincidencias y aplicar la predicción de tráfico. Los resultados son diferentes a la Tabla 36 pese a que sea el mismo algoritmo y calle analizada porque se están presentando en diferentes horarios en la misma fecha.

Tabla 37.

*Estado de tráfico Av. Patria – predicción BigML vs. Google Mpas. (2019-10-10 17H40)*

| INTERSECCIÓN (AV. SHYRIS) | BIGML | GOOGLE MAPS | RESUMEN |
|---|---|---|---|
| **Patria-12 de Octubre NS** | GRIS | ROJO | 0 |
| **Patria-6 de Diciembre NS** | NARANJA | NARANJA | A |
| **Patria-Amazonas NS** | NARANJA | NARANJA | A |
| **Patria-9 de Octubre NS** | NARANJA | ROJO | B |

**CONTINÚA →**

| Patria-10 de Agosto NS | GRIS | NARANJA | 0 |
|---|---|---|---|
| **Patria-Portoviejo NS** | GRIS | NARANJA | 0 |
| **Patria-18 de Septiembre NS** | GRIS | NARANJA | 0 |

## Comparación predicción de SARIMAX y Google Maps

Al igual que en la sección anterior se obtiene la Tabla 38 en la cual se puede observar el resultado de la comparación de predicción de SARIMAX y de Google Maps y que para este caso se tiene 5 desaciertos y 5 aciertos. Para esto se extrae el estado de tráfico de cada intersección que constituye a la Av. Shyris y se pone a prueba bajo la misma fecha y hora que se hizo con BigML.

Tabla 38.

*Estado de tráfico calle Shyris – predicción BigML vs. Google Mpas. (2019-10-10 06H00)*

| INTERSECCIÓN (AV. SHYRIS) | SARIMAX | GOOGLE MAPS | RESUMEN |
|---|---|---|---|
| **Shyris-Rio Coca NS** | NARANJA | VERDE | B |
| **Shyris-Tomas de Berlanga NS** | NARANJA | VERDE | B |
| **Shyris-Isla Floreana NS** | VERDE | VERDE | A |
| **Shyris - Gaspar de Villarroel NS** | NARANJA | VERDE | B |
| **Shyris - El Telegrafo NS** | VERDE | VERDE | A |
| **Shyris - Naciones Unidas NS** | NARANJA | VERDE | B |
| **Shyris-Portugal NS** | VERDE | VERDE | A |
| **Shyris- Rep. Del Salvador NS** | VERDE | VERDE | A |
| **Shirys- Rusia NS** | VERDE | VERDE | A |
| **Shirys- Eloy Alfaro NS** | ROJO | VERDE | B |

En la Tabla 39 se presenta un escenario de la misma Av. Shyris en hora pico y aquí se puede observar que los resultados de la comparación son 3 desaciertos y 5 aciertos.

Tabla 39.

*Estado de tráfico – predicción BigML vs. Google Mpas. (2019-10-10 17H40)*

| INTERSECCIÓN (AV. SHYRIS) | SARIMAX | GOOGLE MAPS | RESUMEN |
|---|---|---|---|
| **Shyris-Rio Coca NS** | ROJO | NARANJA | B |
| **Shyris-Tomas de Berlanga NS** | ROJO | NARANJA | B |
| **Shyris-Isla Floreana NS** | ROJO | NARANJA | B |
| **Shyris - Gaspar de Villarroel NS** | ROJO | ROJO | A |
| **Shyris - El Telegrafo NS** | ROJO | ROJO | A |
| **Shyris - Naciones Unidas NS** | ROJO | ROJO | A |
| **Shyris-Portugal NS** | NARANJA | NARANJA | A |
| **Shyris- Rep. Del Salvador NS** | NARANJA | NARANJA | A |
| **Shirys- Rusia NS** | NARANJA | ROJO | B |
| **Shirys- Eloy Alfaro NS** | ROJO | ROJO | A |

**Comparación entre valores reales y valores predictivos de velocidad**

En la base de datos se cuenta con valores de predicción que se realizaron en el módulo de machine learnnig, por lo tanto, para realizar estas pruebas se procedió a exportar los datos en el rango de fechas: desde 17/10/2019 hasta el 24/10/2019. Además, paralelamente en las mismas fechas se tomó datos reales de las intersecciones Shyris - Naciones Unidas y Amazonas – Gaspar

de Villarroel con la API GDMA. Se escoge la Shyris - Naciones Unidas para este caso de forma aleatoria y la intersección Amazonas – Gaspar de Villarroel de manera empírica basada en la caracterización de las calles que se explica en el capítulo 3 y 4; las dos intersecciones antes nombradas tienen parámetros semejantes ideales para realizar pruebas de comparación respecto a la velocidad con predicción SARIMAX y BIGML.

Con la información de velocidad se realizó el cálculo de cuan dispersos están los datos con respecto a la media, es decir, la desviación estándar.

En la Figura 96, el eje 'X' representa la fecha y hora en la cual los datos fueron adquiridos y el eje 'Y' la velocidad vehicular de la intersección Shyris y Naciones Unidas, la velocidad predictiva representada por color rojo y de color azul la velocidad real.

Luego de un análisis de la información, se percibió que existían valores fuera del comportamiento habitual de los datos vecinos por lo que se les tomó como condiciones de no importa y con los datos restantes se calcula la desviación estándar para este caso un valor de 14.7% y visualmente la gráfica de predicción sigue el patrón de comportamiento de la gráfica con valores reales.

Para la Figura 97 que corresponde a la intersección Amazonas y Gaspar de Villarroel se mantienen los mismos parámetros asignados para el eje x (fecha y hora) y el eje y (velocidad), en este caso la desviación estándar es igual a 19.6 % y visualmente la gráfica de predicción sigue el comportamiento de la gráfica de los valores reales, aunque contiene picos fuera de la media. Esto se debe a varios factores uno de ellos es que en el momento de predecir y acoplar las características a la intersección que se asemeja.

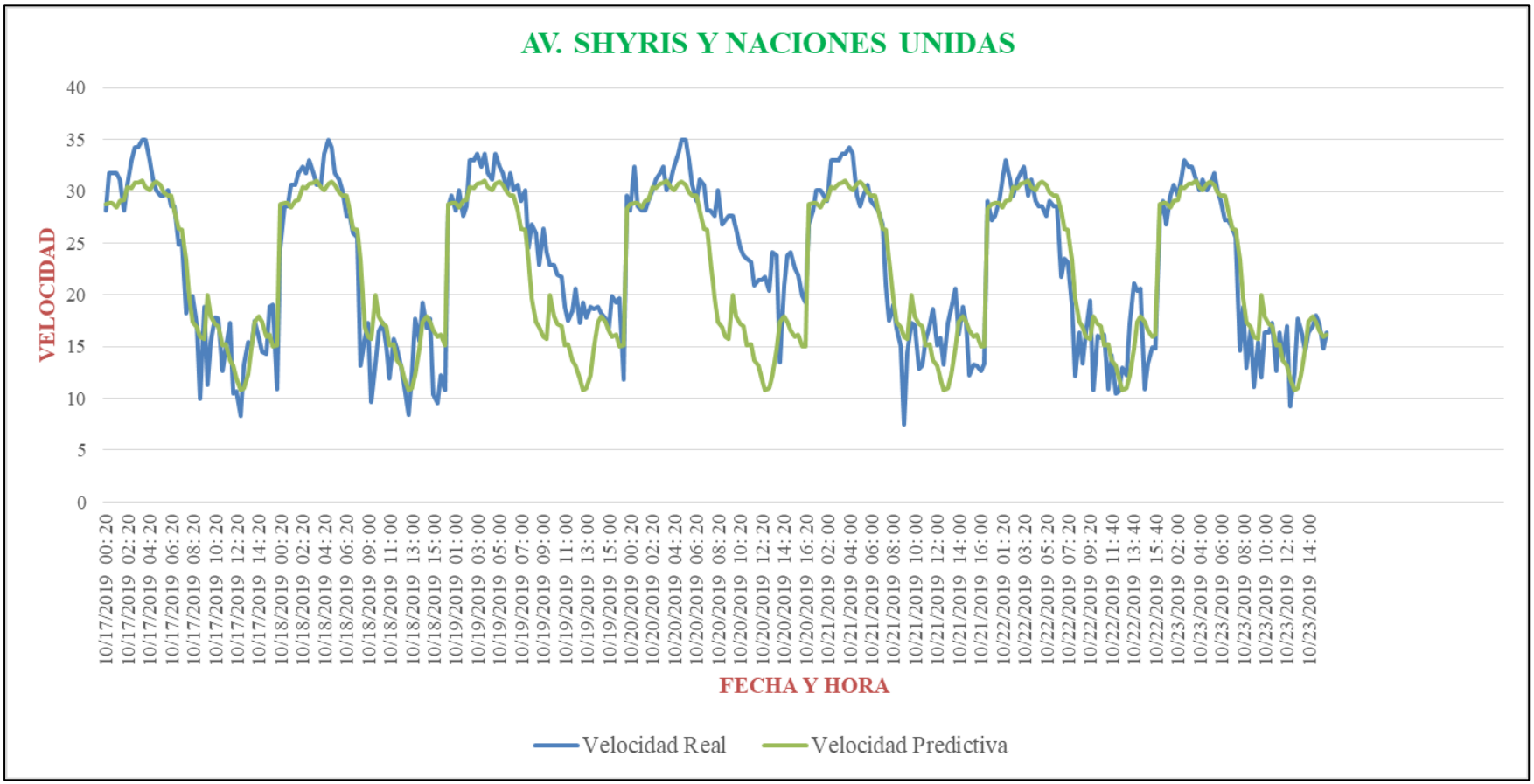



***Figura 96.*** Shyris y Naciones Unidas

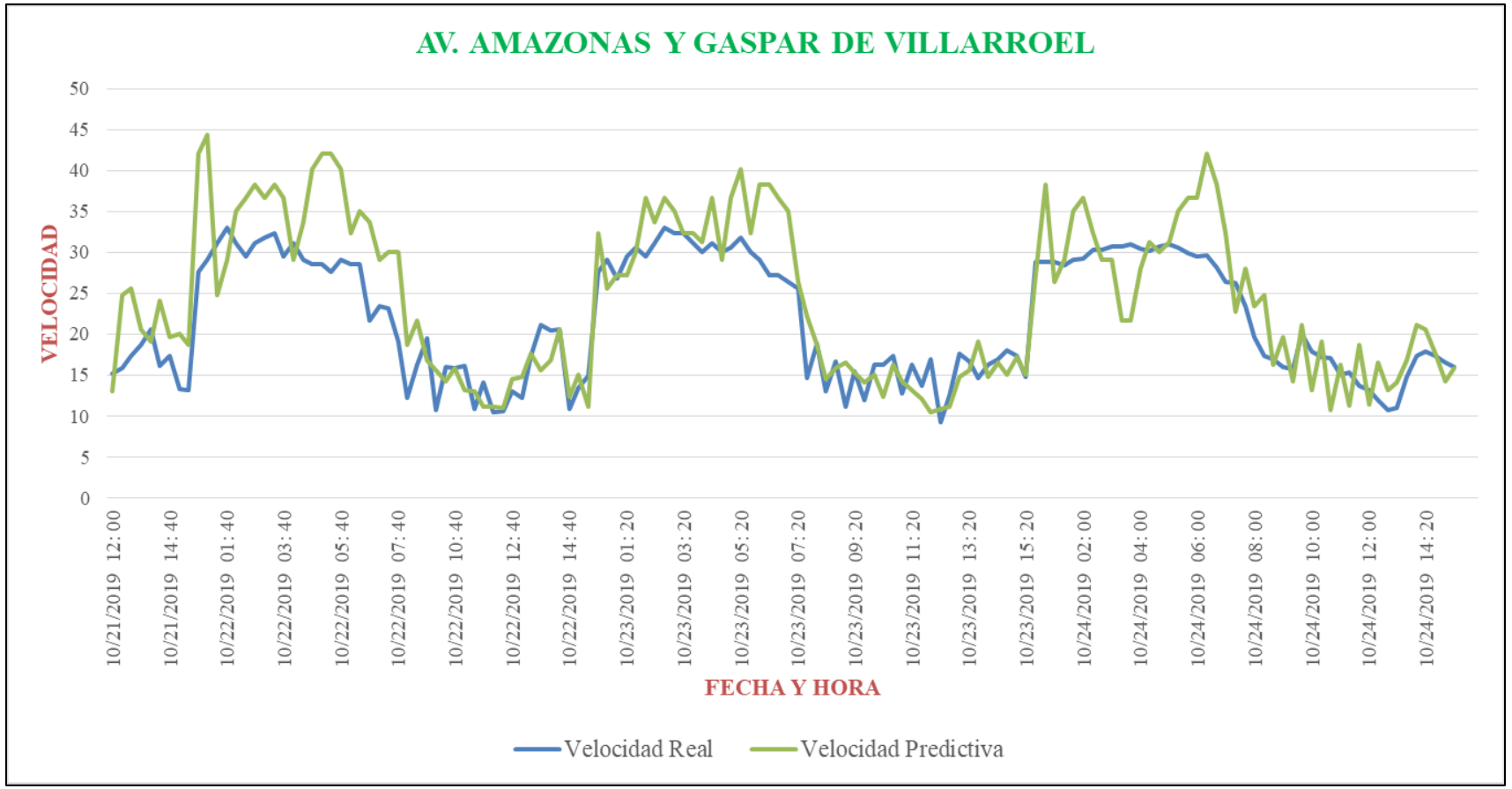



***Figura 97.*** Amazonas y Gaspar de Villaroel

**Pruebas de Usabilidad**

En esta sección se va a realizar pruebas de usabilidad del cliente móvil que sirve para la recolección de datos, interfaz que se muestra en la Figura 98.


***Figura 98.*** Interfaz inicial de la aplicación para recolección de datos

Para este caso se hicieron pruebas con 15 diferentes usuarios de edades entre 18 a 22 años acerca de manejo, conocimiento y uso del cliente Web. Se entrega una encuesta en la cual se tiene 10 preguntas las cuales pueden ser valorados en escala del 1 al 5, siendo 1 total desacuerdo, 3 no seguro de contestar y 5 total acuerdo, cómo se muestra en la siguiente Tabla 40.

Tabla 40.

*Escala para puntuación encuesta.*

| ESCALA | SIGNIFICADO |
|---|---|
| **1** | Total, desacuerdo |
| **5** | Total, acuerdo |
| **3** | No está seguro de contestar |

En base a los resultados entregados por los usuarios en las encuestas (Anexo E) se realiza el cálculo del puntaje sobre 100 puntos para la aplicación como se muestra en la siguiente Figura 99. Se puede observar que las puntuaciones no bajan de los 75 y la máxima son 100 puntos.

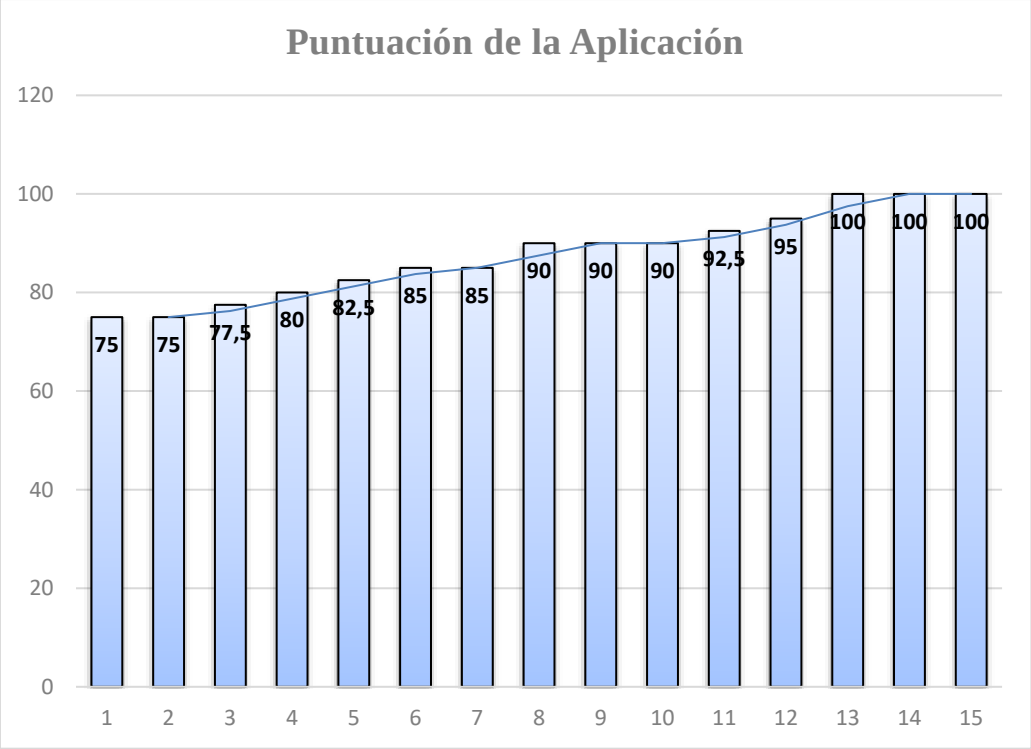



***Figura 99.*** Promedio de la puntuación en la encuesta por usuario

Para tres de los usuarios encuestados su apreciación para la aplicación es excelente de un promedio de 100 puntos y nueve de los encuestados los promedios oscilan entre 80 a 95 puntos indicando una apreciación para la aplicación que es muy buena. Por último, los tres usuarios restantes otorgan una puntuación entre los 75 a 76 puntos y, además, se debe tomar en cuenta que en este caso se da la oportunidad de otorgar una puntuación de 3, la cual significa que no está seguro de contestar. Sin embargo, las puntuaciones dan indicadores altos y en promedio el porcentaje es de 87.83% que significa que la interfaz es fácil de manejar por los usuarios e incluso estarían dispuestos a seguir utilizándola.

**Prueba de funcionalidad**

Con respecto a la parte de televisión digital se realiza una prueba de funcionalidad, la cual consiste en realizar una consulta al servidor y que se nos devuelva el resultado, el mismo que se presente en pantalla de TVD.

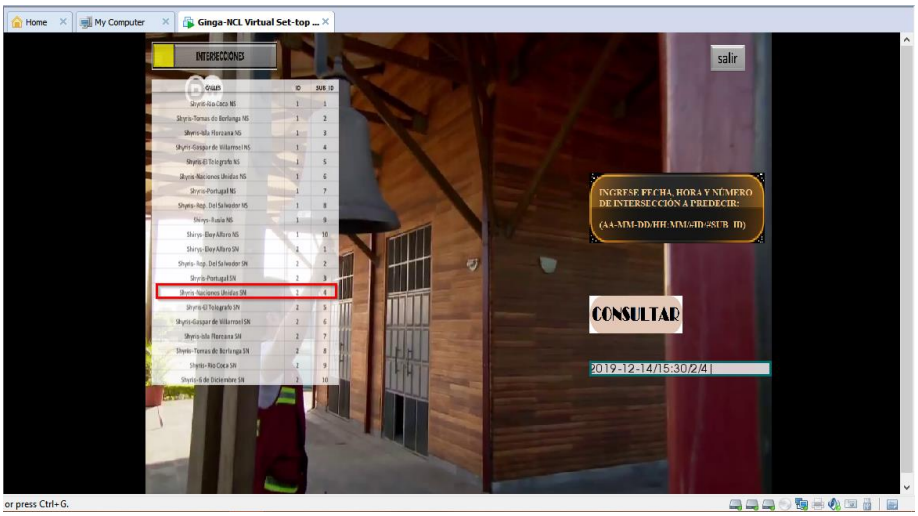



***Figura 100.*** Ingreso consulta para predicción

En la Figura 100 se puede observar como el usuario debe llenar el campo de texto con la fecha, hora y calle que requiere predecir. Una vez que se ha llenado correctamente se presiona el botón de consultar para que envíe la solicitud al servidor y conteste. La respuesta a la

solicitud se puede observar en la Figura 101 de lo que se ha enviado para la consulta; aquí nos entrega la calle consultada que coincide con la que se ha seleccionado, la fecha, la velocidad con la se podrá transitar en este tiempo. Así se prueba la funcionalidad en televisión digital que es correcta a lo que se diseñó inicialmente.

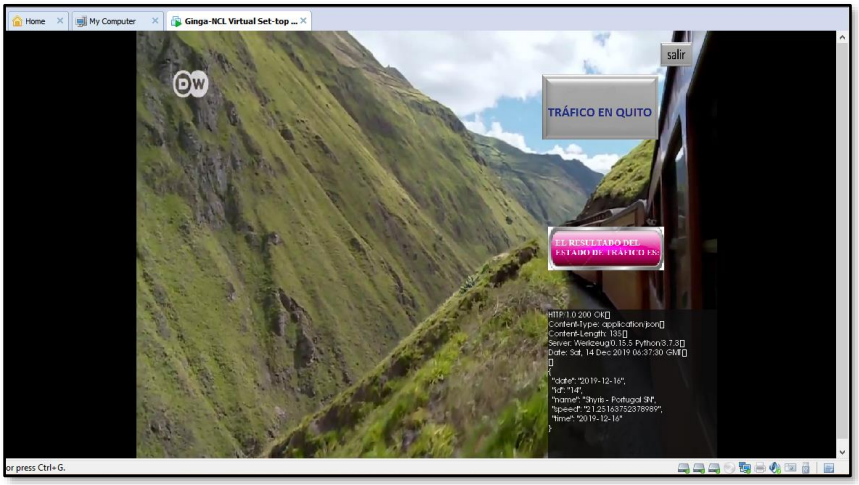



***Figura 101.*** Predicción de lo consultado

# CAPÍTULO 6

# CONCLUSIONES Y RECOMENDACIONES

## Conclusiones

- Para el proyecto se analizaron dos tipos de plataformas en la nube, la primera es la plataforma AWS que accede a productos y servicios con una capa gratuita con ciertas limitaciones, mientras que la segunda plataforma Google Cloud ofrece un bono de 300 dólares abierto a todos los servicios. En nuestro caso el orquestador y base de datos se encuentra en AWS dentro de la capa gratuita, y Google Cloud fue usado en la instancia de machine learning porque los recursos necesarios para este módulo son más robustos y en AWS hubiese implicado gastos.
- En el presente proyecto se diseñó una arquitectura *cross-platform* para el análisis de tráfico vehicular; compuesta de tres plataformas: a) la primera el cliente móvil para captar la información, b) la segunda el cliente Web y c) la tercera el cliente TVD, las dos últimas usadas para mostrar la información. Todas las plataformas antes mencionadas usan recursos de computación en la nube para adquirir, almacenar, predecir y presentar la información; permitiendo al usuario que pueda acceder desde multiplataformas en cualquier momento o lugar, utilizando las características y capacidades específicas de cada cliente.
- El proyecto está formado por un conjunto de microservicios y a su vez segmentado en módulos independientes para una fácil implementación. El módulo orquestador establece la comunicación entre todos los servicios, recibe solicitudes y envía respuestas entre los diferentes dispositivos de software y hardware. Esto permite una

arquitectura distribuida y hace al sistema más estable, por ende en caso de ataques no colapsará todo el sistema sino solo el módulo en particular.

- El sistema de predicción de tráfico está basada en metodologías MDA para el despliegue y la construcción de la plataforma, ya que puede ser adaptado a otros lenguajes y escalable en el tiempo para cubrir necesidades al abstraer características generales de la propuesta, con lo cual se puede ampliar la arquitectura para proyectos futuros. La arquitectura en base a metamodelos separa los elementos para que sea mucho más fácil implementar módulo por módulo, reduciendo costos y amenorando la complejidad al escoger herramientas óptimas que se usarían para la resolución de los problemas planteados inicialmente.
- En función de las investigaciones realizadas el tráfico vehicular es proporcional al crecimiento de la población la cual va en aumento. Por lo tanto, es necesario tener un registro permanente de parámetros necesarios en una base de datos para luego determinar el estado de tráfico vehicular de las calles. Motivo por el cual, se usó plataformas de computación en la nube que cuentan con almacenamiento expandible para la implementación del proyecto. Se ha utilizado ingeniería de modelos por ejemplo para separar cada uno de los módulos y establecer sus características individuales con lo que es posible establecer las interconexiones con otros módulos para facilitar la implementación a futuro y la ampliación del sistema.
- La herramienta de machine learning tiene varios métodos de aprendizaje, por lo que previo a su uso es importante analizar el comportamiento de los datos para tener una mejor asertividad en los resultados de la predicción. El proyecto tienen dos tipos de comportamientos en los datos: a) en primera instancia los datos de velocidad son adquiridos por GDMA para el entrenamiento, de los cuales se obtienen tendencias

temporales y estacionarias, por lo que se determinó el uso del modelo SARIMAX, y b) en segundo lugar los datos son de caracterización de las calles por lo cual se usa la predicción por clasificación y se implementó el algoritmo de árbol de decisiones en la herramienta de análisis predictivo BigML.

- Al ser una arquitectura *cross-platform* el diseño necesita ser realizado de manera muy ordenada, por lo tanto los diagramas BPMN permiten ver el flujo de la secuencia del sistema y como se va a interrelacionar con el resto de componentes.
- Al realizar pruebas de dos intersecciones de calles de la ciudad de Quito entre valores reales y valores predichos, se obtuvo la desviación estándar de la resta de las dos gráficas en la predicción SARIMAX la desviación fue del 14.7% para la intersección Shyris y Naciones Unidas, lo cual es un valor aceptable y tiene un comportamiento similar a los datos reales. Además, la desviación en el método de árbol de decisiones es 19.6% para la intersección Amazonas y Gaspar de Villarroel. El entrenamiento y aprendizaje de árbol de decisiones fue realizado por caracterización cualitativa y la desviación aumenta debido a que acumula la suma de RSME de la predicción SARIMAX y además de la predicción de caracterización, por lo tanto la confiabilidad del punto entrenado hereda a la intersección con método de clasificación la suma de RSME.
- En la comparación cualitativa que se realizó se pudo apreciar que los colores en la predicción de nuestro cliente Web con la de Google Maps son similares, pese a que se desconoce y no hay información del rango de velocidades con las que maneja Google para imponer el estado de tráfico (código de colores).
- El entrenamiento del sistema depende de la cantidad y la calidad de información verídica que se adquiere para la predicción e identificar el tipo de algoritmo que se

debe usar para procesar la información de manera acertada. En este proyecto las fuentes utilizadas son API Google Distance Matrix y el cliente móvil, debido a que permite que los usuarios puedan acceder y enviar la información en cualquier momento y lugar.

- El uso de un integrador automático (orquestador), permite el manejo, la unificación y coordinación de los microservicios para la implementación del patrón arquitectónico Modelo Vista Controlador y de esta manera evitar una aplicación monolítica.
- En base a las pruebas realizadas con la herramienta JMeter se puso a prueba el tiempo de respuesta del servidor al incrementar la carga, llegando a un límite de saturación con 500 usuarios e impidiendo el acceso al URL (página Web), esto por las especificaciones con las que cuenta el servidor son básicas debido a costos.
- Para validar la usabilidad de la aplicación del cliente móvil se empleó la herramienta de SUS, obteniendo una puntuación excelente y sobresaliente en general del 87.83%. Cabe mencionar que la encuesta fue realizada con pocos usuarios dando un valor aceptable del índice de usabilidad y colocándolo en el rango de lo mejor posible en la escala de SUS.
- Para realizar la predicción de datos es necesario contar con datos históricos de entrenamiento. El porcentaje de error es inversamente proporcional a los datos, esto quiere decir que, si los datos son mayores, el error es menor. Para el trabajo presente los datos tomados de la calle Shyris de la ciudad de Quito fueron durante 4 meses (febrero a mayo del 2019) cada 20 minutos diariamente, por lo tanto, el nivel de confianza disminuye a lo largo del tiempo.

- El cliente móvil es muy importante para la toma de datos haciendo uso del GPS con el que cuentan los dispositivos móviles y que es una herramienta muy utilizada en la actualidad y que al momento la mayoría de las personas cuentan. Esto permite tener una herramienta que solventa las necesidades de ciudades que no tienen desplegados sensores todavía en sus calles. El cliente móvil es el medio para la toma de datos y es indispensable que se tenga una gran cantidad de usuarios para una mejor alimentación de la base de datos llamado *crowsourcing.*
- Además se ha utilizado un cliente TVD en vista de que en Ecuador se optó por el estándar ISDBT por lo cual se emplea este medio para poder especificar la información y llegar al público la información de predicción de tráfico en cualquier momento, siendo de utilidad para el usuario ya que la movilidad es un aspecto cotidiano e importante; además que en la actualidad la televisión es una herramienta muy utilizada la misma que ha evolucionado permitiendo desplegar aplicaciones.
- La caracterización de las calles se lo realizó cualitativamente por medio del servidor de Google Maps con vista 360° en ciertas calles de Quito y en base a la experiencia de los administradores y se eligió los siguientes parámetros: restaurantes, centros educativos, semáforos, centros comerciales, parada de buses, dirección cardinal, parques, área ejecutiva y transporte metropolitano entre otros que puedan causar congestión vehicular. Los resultados tuvieron una desviación estándar del 19.6% de la información real y con un comportamiento que tiende a asemejarse a los datos reales.

## 6.1 Recomendaciones

- La toma de datos es vital para la predicción, por lo tanto, se recomienda tomar los datos de manera continua sin dejarlos de adquirir por el tiempo total del proyecto de investigación.
- La API de Google Distance Matrix se habilita en google cloud. Luego de terminar su uso se debe deshabilitar para que el costo no siga aumentando luego de ya no necesitar sus servicios.
- Es recomendable que el alcance del tiempo de la predicción sea el doble del tiempo de adquisición de datos es decir si la duración de la toma de datos fue de 4 meses la predicción sería para los siguientes 4 meses para tener niveles de confianza aceptables en el pronóstico.

## 6.2 TRABAJOS FUTUROS

- Se propone realizar una clasificación de datos con parámetros basados en registros oficiales de la ciudad, tales como: la Dirección Metropolitana de Catastro, uso de suelos e identificación de las intersecciones de forma personal para tener una mejor precisión del panorama de las características de las calles en la ciudad de Quito. De esta forma se tendrían datos más reales y vigentes.
- En un entorno real los datos serían muy grandes por lo que sería demasiado lento tratar a los datos como se lo hace tradicionalmente, se propone trabajar con un tipo de base de datos diferente que trate información de grandes volúmenes datos por ejemplo el almacenamiento de información a grafos.

- En las pruebas realizadas con un solo servidor Web y en función a los resultados obtenidos, se pudo observar que se llegó a un límite de usuarios simultáneos realizando solicitudes o peticiones al servidor, por lo tanto, en próximos trabajos se plantea revisar resultados con balanceadores de carga (Ribbon) y de esta forma probar si sería conveniente y eficaz la respuesta con un número superior a 500 usuarios. De igual manera realizar pruebas con (Hystrix) para la tolerancia a fallas.
- El cliente móvil fue desarrollado para un sistema operativo Android 8, en próximos trabajos se propone depurar la aplicación para los sistemas operativos actuales y cubrir las exigencias de estas versiones nuevas. Además, se sugiere buscar algunas estrategias para expandir la aplicación a nuevos usuarios, una alternativa podría ser subir al playstore y ponerlo al alcance de cualquier usuario.

## LISTA DE REFERENCIAS

Abhishek Sharma, Svetlozar Nestorov & Boris Jukić. (2015). *Augmenting Data Warehouses with Big Data*. https://www.tandfonline.com/doi/abs/10.1080/10580530.2015.1044338

Adrián Rodríguez. (2018). *Internet de las Cosas contra el tráfico vehicular—Telcel Empresas*. https://telcelempresas.com/internet-de-las-cosas-contra-el-trafico-vehicular/

Ana María Carvajal. (2019, febrero 21). *Investigación mundial sobre movilidad ubica a Quito en el puesto 26 entre 200 ciudades con más problemas de tráfico*. El Comercio. https://www.elcomercio.com/actualidad/congestion-vehicular-ranking-movilidad-amt.html

Andrade, A. (2018). *1 Introduccion Series de Tiempo.pdf*.

Apache JMeter. (2019). *Apache JMeter*. http://jmeter.apache.org/

Apiumhub. (2017). Técnicas de testeo de software y herramientas. En *Técnicas de testeo de software y herramientas*. https://apiumhub.com/es/tech-blog-barcelona/tecnicas-de-testeo-de-software/

Bahit Eugenia. (2012). *MVC avanzado desde la óptica del Arquitecto de Software y de la del Programador*. http://46.101.4.154/Libros/(pre-edici%C3%B3n%20NO%20REVISADA)%20Arquitecturas%20Web%20modulares%20con%20MVC%20en%20Python%20y%20PHP.pdf

Başţanlar, Y., & Özuysal, M. (2014). Introduction to Machine Learning. En M. Yousef & J. Allmer (Eds.), *miRNomics: MicroRNA Biology and Computational Analysis* (Vol. 1107, pp. 105-128). Humana Press. https://doi.org/10.1007/978-1-62703-748-8_7

Brownlee, J. (2018a, agosto 16). A Gentle Introduction to SARIMA for Time Series Forecasting in Python. *Machine Learning Mastery*. https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/

Brownlee, J. (2018b, octubre 23). How to Grid Search SARIMA Hyperparameters for Time Series Forecasting. *Machine Learning Mastery*. https://machinelearningmastery.com/how-to-grid-search-sarima-model-hyperparameters-for-time-series-forecasting-in-python/

Chakray. (2019, mayo 15). Microservicios. *Chakray*. https://www.chakray.com/es/que-son-los-microservicios-definicion-caracteristicas-y-ventajas-y-desventajas/

Coyago, A. P. R., Ortega, S. F. C., & Pinargote, A. J. P. (2017). Análisis de la aplicación del pico y placa en la ciudad de Quito. *INNOVA Research Journal*, *2*(6), 136-142.

Dans E. (2012). *Bigml-modelizacion-e-inteligencia-artificial*. https://www.enriquedans.com/2012/07/bigml-modelizacion-e-%20inteligencia-artificial.html

de la Fuente Fernández, S. (2017). *Modelo-arima.pdf*. http://www.estadistica.net/ECONOMETRIA/SERIES-TEMPORALES/modelo-arima.pdf

Departamento Tecnologías de Tráfico Madrid. (2013). *PuntosMedidaTraficoMadrid .pdf*.

García, M. (2017). *MVC (Modelo-Vista-Controlador): ¿qué es y para qué sirve?* https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve

García Molina, J., & García Rubio, F. O. (2013). *Desarrollo de software dirigido por modelos conceptos, métodos y herramientas*. Ra-Ma.

Google Cloud. (2019). *Servicios de computación en la nube*. Google Cloud. https://cloud.google.com/?hl=es-419

Hafedh, U. L., Taewoo, U. at A., & Hans, U. of W. (2012). *Understanding Smart Cities: An Integrative Framework*. http://www.ctg.albany.edu/media/pubs/pdfs/hicss_2012_smartcities.pdf

Herrera, J. F. G. (2018). *Diseño e implementación de aplicaciones interactivas basadas en ginga-ncl para televisión digital enfocadas en la temática del medio ambiente*. 127.

IMF business school. (2018, mayo 18). Nosql vs sql: Diferencias y principales tecnologías. *Blog de Tecnología - IMF BS*. https://blogs.imf-formacion.com/blog/tecnologia/nosql-vs-sql-diferencias-principales-tecnologias-201805/

INEC. (2018, diciembre). *Anuario de Estadísticas de Transporte 2017*. http://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Economicas/Estadistica%20de%20Transporte/2017/2017_TRANSPORTE_PRESENTACION.pdf

INRIX. (2018a). *INRIX Global Traffic Scorecard*. INRIX - INRIX. http://inrix.com/scorecard/

INRIX. (2018b). *Quito's Scorecard Report*. INRIX - INRIX. http://inrix.com/scorecard-city/

Jose Cavanillas. (2015). *New horizons for a data-driven economy: A roadmap for usage and exploitation of big data in Europe*. Springer Berlin Heidelberg.

Márquez, J. E. M. (2018). *Una arquitectura orientada a servicios y dirigida por eventos para el control inteligente de UAVs multipropósito* [Http://purl.org/dc/dcmitype/Text, Universidad de Extremadura]. https://dialnet.unirioja.es/servlet/tesis?codigo=212831

McCaffrey, J. (2012). *Test Run—Classification and Prediction Using Neural Networks*. https://msdn.microsoft.com/en-us/magazine/jj190808.aspx

MeasuringU. (2019). *SUS*. https://measuringu.com/

Mell, P., & Grance, T. (2011). *The NIST Definition of Cloud Computing*. 7.

MONT. (2017, septiembre 27). *De la Escala de Usabilidad de Sistemas*. Medium. https://medium.com/@m00nt/de-la-escala-de-usabilidad-de-sistemas-97c935e68fa9

Morejón, J. F., Cueva, R. A. L., & Carrera, V. (2015). *Análisis del desempeño de algoritmos de detección de eventos vulcanológicos basados en machine learning*. 88.

Municipio del DMQ. (2009). *Plan Maestro Movilidad DMQ 2009 2025*. calameo.com. https://www.calameo.com/books/000006297102a6f33ba37

Ordóñez, J. (2018, mayo). *¿Qué es una API REST?* Idento. https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/

Peter Harrington. (2012). *Machine Learning in Action*.

Pons, C. F., Giandini, R. S., & Pérez, G. A. (2010). *Desarrollo de software dirigido por modelos*.

Sánchez, R. P. (2011). *Soporte a la trazabilidad en el desarrollo de líneas de producto software*. 123.

Shaw Mary. (1996). *Software Architecture: Perspectives on an Emerging Discipline*.

Silva, T. H., de Melo, P. O. S. V., Viana, A. C., Almeida, J. M., Salles, J., & Loureiro, A. A. F. (2013). Traffic Condition Is More Than Colored Lines on a Map: Characterization of Waze Alerts. En A. Jatowt, E.-P. Lim, Y. Ding, A. Miura, T. Tezuka, G. Dias, K. Tanaka, A. Flanagin, & B. T. Dai (Eds.), *Social Informatics* (Vol. 8238, pp. 309-318). Springer International Publishing. https://doi.org/10.1007/978-3-319-03260-3_27

Stuart Kent. (2002). *Model Driven Engineering*. https://www.cs.kent.ac.uk/projects/kmf/Documents/ifm02paper.pdf

Sun Bruce. (2011, agosto 3). *Arquitectura multinivel para la construcción de servicios web RESTful*. http://www.ibm.com/developerworks/ssa/library/wa-aj-multitier/index.html

Torres, J. (2011, junio 28). Comunidad Ginga Ecuador: Middleware Ginga. *Comunidad Ginga Ecuador*. http://comunidadgingaec.blogspot.com/2011/06/middleware-ginga.html

UTFSM. (2013). *GINGA -TVD*. http://www2.elo.utfsm.cl/~elo323/ncl.html

Vergara, N. M., Rivera, J. E., Romero, J. R., & Moreno, A. V. (2007). *Desarrollo de software dirigido por modelos*.

Villavicencio, J. (2010). *Introducción a Series de Tiempo*. 33.

WAZE Mobile. (2006). *Aplicación gratuita para indicaciones de viaje, información del tráfico y navegación GPS de Waze*. https://www.waze.com/es/contact

Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, *1*(1), 22-32. https://doi.org/10.1109/JIOT.2014.2306328

# ANEXOS