Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Plataforma para la simulación en tiempo real de operaciones de tránsito

Proyecto de título presentado por Diego Hernán Parraguez Mardones

para obtener el grado de Ingeniero Civil Informático

Dirigida por Tatiana Gutiérrez y Patricio Galdames

2018

Agradecimientos

En este amplio y arduo camino que ha estado lleno de dificultades y alegrías tanto en el desarrollo de éste proyecto de título como en mi carrera universitaria deseo realizar una pausa en mi vida y agradecer a las personas que me han apoyado en este proceso de mi vida. No hubiese sido ésto posible sin la participación de personas e instituciones que han facilitado las cosas para que este trabajo llegue a un feliz término.

A mi hija, Helen, por ser la luz en mi camino, gracias a ti he recorrido este camino hasta el final. Gracias hija mía por ser la fuente de mi esfuerzo y todas las energías requeridas, por ser mi motor de vida. Espero que entiendas y comprendas a futuro que la recompensa que nos espera de este sacrificio que realizamos juntos sea provechoso para ambos y que el tiempo demandado en este trabajo compensará el tiempo.

Debo agradecer de manera especial y sincera a mi Profesora Guía la Doctora Tatiana Gutiérrez Bunster por aceptarme para realizar éste proyecto de título bajo su dirección. Su apoyo y confianza en mi trabajo y su capacidad para guiar mis ideas ha sido fundamental, no solo durante el desarrollo de éste trabajo si no que también en mi formación como estudiante y futuro profesional.

De la misma manera, quiero expresar mi mas sincero y especial agradecimiento al Director del Departamento de Ingeniería Civil y Ambiental al Doctor Patricio Álvarez Mendoza por su importante aporte, ayuda y participación activa en el desarrollo de mi proyecto. Por entregarme las herramientas necesarias, por depositar en mi su confianza para que esto fuera posible, muchas gracias profesor.

A mis profesores, por entregarme las herramientas, los conocimientos y la sabiduría necesaria

durante el transcurso de mi carrera universitaria.

A mis padres por ser el pilar fundamental en todo lo que ahora soy. Por su educación tanto académica como personal, por su amor incondicional y apoyo mantenido intacta mente a través del tiempo. Por confiar en mi y ver que ésto es posible, muchas gracias a ambos. Faltan palabras para expresar mi gratitud hacia ustedes. Sin ustedes este trabajo no podría haberlo hecho solo.

A mi novia, Katheryne, por tu apoyo incondicional y motivación para la culminación de mis estudios profesionales. Gracias por todo, por tus palabras de aliento, por tu confianza, por tus consejos, por todo, te estoy eternamente agradecido.

A mi madre, Sara Mardones, por darme la vida, por tu amor absoluto, por creer en mi y porque siempre me has apoyado en todo. Madre mía, gracias, todo esto te lo debo a ti. Te amo.

A mis amigos, Iván, Luis, Gustavo, Nicolás, Matias, Javier, Ricardo y a todos los demas, que si bien no los nombro saben quienes fueron lo que estuvieron conmigo con su apoyo, confianza, consejos, por compartir los buenos momentos, gracias a todos y a cada uno.

A mi hermano, Alvaro, que si bien no esta en estos momentos conmigo siempre a estado incondicionalmente, apoyándome, aconsejándome, y siempre queriendo lo mejor para mi, gracias hermano, por estar conmigo y ser un apoyo fundamental en mi vida.

Resumen

Palabras Clave — TSS, AIMSUN, Python, ITS, microsimulador, MTA, API, UOCT, scripts

Este proyecto de titulo proviene de la necesidad que se presenta con el software de microsimulación de licencia pagada llamado AIMSUN, el cual posee una amplia gama de herramientas dispuestas a ser utilizadas mediante el uso de la programación, siendo compatible con los lenguajes Python, C# y C++. De modo que no posee una manera de entregar la información en tiempo de simulación. Es por esto que se trabajará tanto en investigación como en desarrollo al no haber trabajos previos al respecto, lo que conlleva a realizar un estudio detallado a la documentación del programa, para luego desarrollar la solución requerida por el cliente. Generando un software mas completo y funcional entregando la información detallada y clara al operador del programa en tiempo de utilización, permitiendo generar un software capaz de entrenar a nuevos operadores como de la misma manera perfeccionando la toma de decisiones al momento de ocurrir algún infortunio en el desarrollo de la simulación.

Así para presentar un caso de estudio relacionándose a lo descrito anteriormente.

La configuración que se debe realizar en software Aimsun, se encuentra dentro de “ANEXOS”

Índice general

Lista de Figuras

Índice de Tablas

Capítulo 1

Introducción

Debido al rápido crecimiento de la ingeniería de tránsito y sus sistemas en términos de demanda y complejidad, el estudio de este tipo de sistemas ha adquirido cada vez más importancia tanto en entornos universitarios, como por parte de las administraciones públicas y privadas. En este sentido, la mayoría de los esfuerzos invertidos en el área han estado encaminados hacia el desarrollo de Sistemas Inteligentes de Transporte (ITS, en inglés), más seguros y eficientes a través de la utilización óptima de la infraestructura disponible (intersecciones, semáforos, sistemas de peaje automático, etc.). Teniendo en cuenta que el control del tránsito en redes urbanas se hace principalmente a través del manejo de los semáforos en intersecciones y que existe diversidad en los sistemas de autopistas desarrollados, resulta relevante analizar las alternativas disponibles en la actualidad para llevar a cabo una planificación y gestión apropiada mediante la utilización de software especializado en la ingeniería de tránsito. En problemas con características de complejidad y magnitud como lo son el tránsito de mallas urbanas o redes interurbanas, no es viable validar las estrategias de control directamente sobre las vías de tránsito vehicular, la utilización de softwares de modelización y simulación de tránsito se encuentran fundamentalmente ligada a las estrategias de planificación y gestión de tránsito. Los principales softwares de modelización y/o simulación pueden clasificarse según el modelo de tráfico que utilicen. Estos modelos pueden ser macroscópicos, mesoscópicas o microscópicos.

Modelos macroscópicos: Estos son válidos para aplicaciones de gran escala donde las

principales variables de interés se encuentran relacionadas con las características del flujo. Entre los programas de modelización macroscópica más relevantes se encuentran: TransCAD, EMME, VISUM, FREFLO, TRANSYT, NETVACI, KRONOS, entre otros.

Modelos mesoscópicas: Estos presentan una aproximación intermedia entre los microscópicos y los macroscópicos en la medida en que mezclan conceptos y herramientas de ambos modelos al analizar el comportamiento de grupos de vehículos. Los softwares de simulación que utilizan modelos mesoscópicas son, entre otros, DYNASMART, DYNAMIT, METROPOLIS e INTEGRATION.

Modelos microscópicos: Estos modelos, presentan la escala más pequeña para el acercamiento al análisis de los sistemas de ingeniera de tránsito. Sus variables de interés se relacionan con el comportamiento de vehículos individuales respecto a la infraestructura y a los demás vehículos en ella. Hay que destacar que este tipo de modelos tienen el objetivo de representar comportamientos humanos por lo que su manejo es relativamente complejo y costoso. Las principales herramientas de microsimulación son en la actualidad TransModeler, VISSIM, AIMSUN, etc.

La utilización específica de uno (o varios) de estos programas se realiza en base a las fortalezas, debilidades y alcance identificados para cada paquete de software, contrastados con las características de la situación particular a analizar. Es por ello que el modelo a trabajar será el microscópico, desempeñándose particularmente con la herramienta de microsimulación de tránsito AIMSUN (MTA)[1]. Se escogerá este software en particular por poseer licencia en la Universidad del Bío-Bío. Es por ello que se desarrollará una Interfaz de Programación de Aplicaciones (API), el cual, se comunicará con el microsimulador AIMSUN, generando un software funcional y permitiendo a su vez el control de diversos parámetros de entrenamiento del MTA. De ésta forma, el software generará entrenamientos mediante simulaciones para luego ser evaluados y así optimizar la toma de decisiones de los operadores de la Unidad Operativa de Control de Tránsito (UOCT)[2] de la ciudad de Concepción, Chile.

A continuación, se cita, de manera resumida, el propósito del software AIMSUN:

[1] Microsimulador de Tránsito AIMSUN - MTA
[2] Unidad Operativa de Control de Tránsito - UOCT

"Aimsun es un software de simulación de tráfico que permite simular todo tipo de elementos, desde un carril bus hasta la totalidad de Manhattan. Con miles de usuarios con licencia en organizaciones gubernamentales, consultorías y universidades en todo el mundo, Aimsun destaca por la velocidad excepcional de sus simulaciones y por la combinación de la modelización de la demanda y la asignación estática y dinámica de tráfico con simulaciones mesoscópicas, microscópicas e híbridas microscópica-mesoscópica en una misma aplicación de software.

Aimsun permite evaluar operaciones de tráfico de cualquier escala y complejidad. La cantidad de aplicaciones es interminable, pero algunas de las más habituales son las siguientes:

- Evaluación y optimización de los planes de Prioridad de Señales de Tráfico (TSP) y de Autobuses de Tránsito Rápido (BRT)
- Estudios de viabilidad sobre Vehículos de Alta Ocupación (VAO) y carriles de alta ocupación en peajes
- Análisis del impacto del diseño de infraestructuras como la mejora o la construcción de una nueva carretera
- Análisis del impacto medioambiental
- Tarificación de peajes y carreteras
- Evaluación de las estrategias de gestión de la demanda de tráfico
- Optimización de Plan de control de señales (interfaz Aimsun-TRANSYT) y evaluación del control adaptativo
- Análisis de seguridad
- Evaluación de las políticas de velocidad variable y de otros Sistemas Inteligentes de Transporte (ITS)
- Análisis del Manual de Capacidad de Carreteras (HCM)
- Gestión de zonas con obras" [1]

Capítulo 2

Descripción de la Empresa

2.1. Introducción

El número de proyectos de desarrollo informático ha ido en constante alza independiente del área de aplicación. Es cada vez más necesario el poder adentrarse en el mundo de la tecnología para tener una mejor llegada a los clientes, que sea más transparente y rápida en sus procesos. En el trabajo a desarrollar se modelará un proyecto informático para el departamento de Ingeniería Civil y Ambiental de la Facultad de Ingeniería de la Universidad del Bío-Bío. En el informe se detalla el proceso de modelamiento y análisis del sistema mediante el tipo de metodología de desarrollo de software (cascada para este caso en particular). Los beneficios que se obtendrá con el desarrollo del software contemplará diferentes aspectos, por ejemplo:

o Interfaz gráfica capaz de modificar los parámetros durante la simulación del MTA (en tiempo real).

o Mejora de eficiencia a sistemas complementarios.

o Entrenamiento a operadores de la UOCT.

o Entrenamiento a carabineros para adquirir mejores planes de contingencia.

o Entrenamiento a directores de tránsito.

o Uso académico.

2.2. Empresa

Descripción de la Empresa o Nombre o razón social: UNIVERSIDAD DEL BIO BIO

o RUT: 60.911.006-6

o Representante legal. HÉCTOR GAETE FERES

o Dirección postal: Collao Nº1202 Casilla 5-C. C.P: 4081112. Concepción

o Sitio Web: www.ubiobio.cl

o Teléfono: +56-413111200

2.3. Área de Trabajo

El software será desarrollado para la Facultad de Ingeniería, particularmente en el Departamento de Ingeniería Civil y Ambiental de la Universidad del Bío-Bío.

El Departamento de Ingeniería Civil y Ambiental es un organismo integrado a la Facultad de Ingeniería de la Universidad del Bío-Bío, que desarrolla actividades de docencia, investigación, asistencia técnica y extensión en el campo de la Ingeniería Civil. Está conformado por profesionales con valores compartidos cuya misión es generar, aplicar y difundir a través de sus actividades el conocimiento de la ciencia y tecnología aplicada a Ingeniería Civil; en permanente interacción con la sociedad. Además, contribuirá a la atención eficaz de las demandas de la sociedad en ámbitos relacionados con la disciplina, mediante el desarrollo de líneas de investigación y fomentando el uso tecnologías emergentes. Lo anterior en estrecha y permanente vinculación con el medio, contando con recurso humano de excelencia, e infraestructura suficiente y adecuada.[3]

2.4. Descripción de la problemática

En la actualidad, la gestión del tránsito en vías públicas se realiza a través de centros de control del tránsito, a cargo de Unidades Operativas de Control de Tránsito (UOCT). Éstas unidades son organismos técnicos a nivel regional, dependientes de la Coordinación de Planificación y Desarrollo del Ministerio de Transportes y Telecomunicaciones de Chile, que se encargan de administrar y operar los sistemas de control de tránsito y otros sistemas complementarios de

apoyo, como circuitos cerrados de televisión, letreros de mensaje variable, estaciones automáticas de conteo vehicular, entre otras tecnologías ITS[1]. Estos centros de control de tránsito se encuentran en ciertas ciudades del país, como lo son; Antofagasta, Valparaíso, Santiago y Concepción. Es en esta última ciudad en la que nos centraremos, el cual, no posee una herramienta para la simulación en tiempo real de un evento[2] ocurrido en las calles de la ciudad.

El MTA es un microsimulador de tránsito, con interfaz gráfica, capaz de simular una red completa de tránsito, permitiendo conocer el comportamiento de la simulación, sin poseer la propiedad de realizar algún cambio en sus parámetros[3] durante la simulación. Es aquí donde nace la necesidad de desarrollar un software que sea capaz de interactuar con el MTA, por no poseer en una interfaz gráfica capaz de manipular los parámetros de la simulación en tiempo real, lo cual, al realizar la API en un lenguaje compatible con el MTA (C++ y Scripts de Python), permitirá administrar entrenamientos mediante los componentes de la simulación para optar y formar la(s) solución(es) optima(s) perteneciente al evento generado por la simulación.

La referencia a componentes, se habla de la manipulación de parámetros pertenecientes al entrenamiento, simulación y evaluación para los operadores del UOCT, como, por ejemplo; el tiempo de luz verde de un semáforo en una intersección, información del trazado que está realizando el vehículo, aumentar o disminuir los carriles vehiculares de un sector en particular de la ciudad, desviar el camino para evitar congestión vehicular, como también realizar casos de estudios, por ejemplo, generar un evento en particular dentro de la ciudad. Cuando se habla a un evento, se refiere puntualmente a situaciones relacionadas con accidentes vehiculares, disturbios en las vías de tránsito, vehículo detenido en un carril de una avenida, entre otras situaciones concernientes al flujo vehicular en las vías de tránsito de la ciudad de Concepción.

[1] ITS:Intelligent Tutoring System o Sistema de Tutoría Inteligente.

[2] evento: eventualidad, hecho imprevisto, o que puede acaecer(suceder).

[3] parámetros: son los datos (como variables) que utilizará la API con el MTA, es decir, por ejemplo, tiempo de luz roja en una esquina (variable 1), cambios de dirección de las vías (variable 2), detención inoportuna de un automóvil (variable 3), etc.

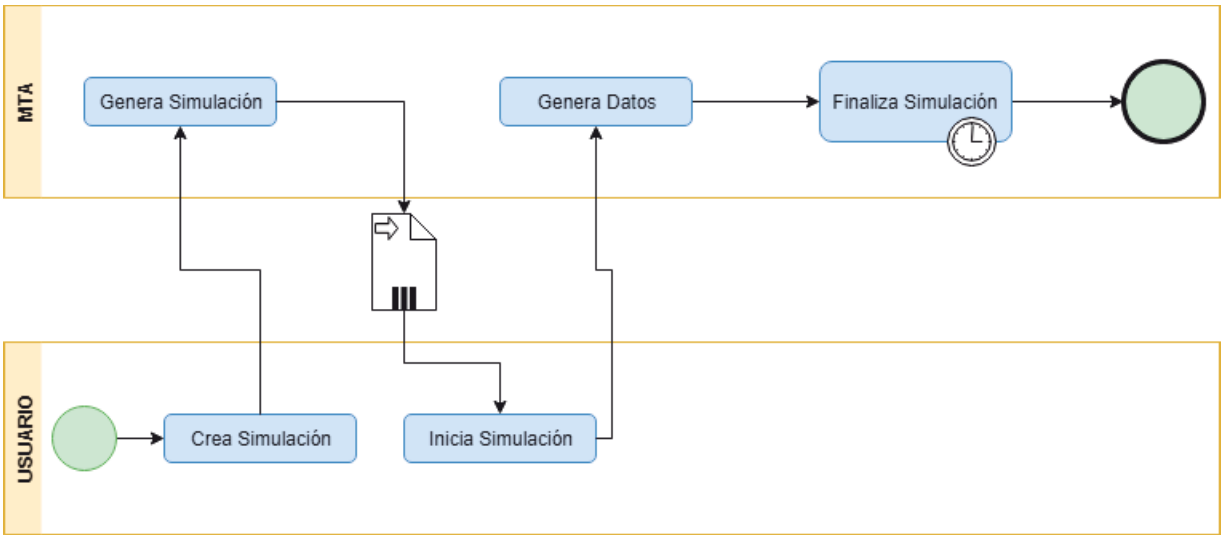


Figura 2.1: BPMN MTA - Cliente/Usuario

En la figura 2.1 se muestra el BPMN en función al Usuario y el MTA. El inicio del proceso comienza en el cliente o usuario que trabajará con el MTA. El cliente debe de crear una simulación para luego el MTA genere dicha simulación. Seguidamente, el MTA genera un archivo el cual contiene todas las funciones e información referente a la simulación, por la cual, el cliente inicia dicho archivo para iniciar la simulación, generando datos gráficamente (a través de los flujos de las calles, mostrando el comportamiento de los vehículos mediante una red de tránsito) dentro del MTA. Para luego, pasado el tiempo de la simulación, el MTA finaliza la simulación terminando el proceso entre él y el usuario.

Capítulo 3

Definición Proyecto

3.1. Objetivos del proyecto

Desarrollo de una plataforma para la simulación en tiempo real de operaciones de tránsito entre la plataforma y el software de microsimulación de tránsito Aimsun.

Objetivos Específicos

OE1. Estudiar la documentación del microsimulador de tránsito Aimsun. Identificando la funcionalidad del software para el desarrollo de la plataforma.

OE2. Realizar una toma de requerimientos (al cliente) para el diseño de la interfaz gráfica que poseerá la plataforma.

OE3. Desarrollar una plataforma con interfaz gráfica para obtener la información requerida por el cliente.

OE4. Validar y exponer la información obtenida entre la API y el MTA.

OE5. Implementación de un caso de estudio para la validación de la plataforma.

Figura 3.1: Diagrama Funcionamiento actual UOCT

En la Figura 3.1 muestra el actual funcionamiento de la UOCT en todas sus oficinas, la cual el sistema de control de tránsito es utilizado por la UOCT y éste sistema se comunica con los sistemas complementarios desplazando la información bidireccionalmente, desarrollados por la Coordinación de Planificación y Desarrollo del Ministerio de Transportes y Telecomunicaciones.

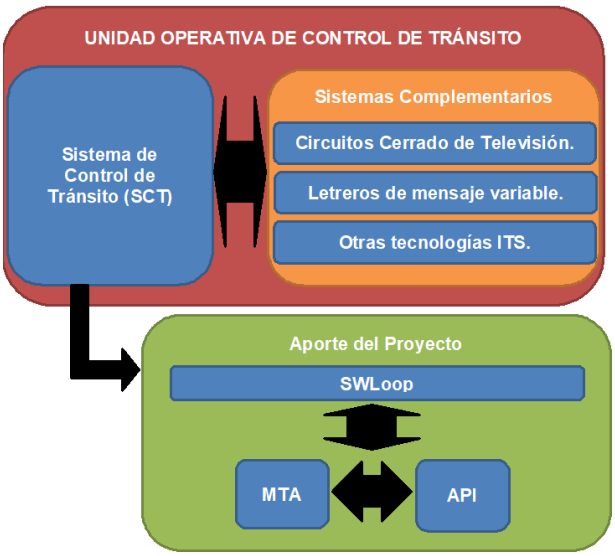


Figura 3.2: Diagrama futuro funcionamiento UOCT

En la Figura 3.2 se muestra el diagrama del futuro funcionamiento de la UOCT. El cual, se describe de la misma forma que la Figura 1, pero incorporando la futura plataforma denominada SWLoop, con un flujo de datos unidireccional, traspasando la información desde el SCT[1] hacia el

[1]SCT: Sistema de Control de Transito.

SWLoop. El flujo de datos es de ésta manera debido a que el MTA funciona con datos de sensores para la generación de simulaciones, por lo cual, obtendrá dicha información perteneciente al SCT para así crear la simulación y generar los entrenamientos.

El aporte de este proyecto recibe el nombre de SWLoop por provenir de "Software in the loop" [4], la cual, ésta es una técnica utilizada para el desarrollo, comprobación y simulación de sistemas embebidos en tiempo real. Éste tipo de software es utilizado por sus cortos ciclos de desarrollo, su interacción más amigable con el usuario qué con el sistema de control, además, con la posibilidad de automatizar su funcionamiento. Es por esto que se desarrollará la plataforma utilizando este tipo de software, por los beneficios que se obtendrán del proyecto, por ejemplo; mejoras de tiempo de respuesta a eventos ocurridos en la vía de tránsito, integración de nueva maquinaria (en caso necesario), mejora de eficiencia a sistemas complementarios, entrenamiento a operadores de la UOCT, entrenamiento a carabineros para adquirir planes de contingencia, entrenamiento a directores de tránsito o simplemente, uso académico.

Así, el propósito de éste proyecto busca la implementación de un software para obtener una herramienta de entrenamiento a través de la simulación, pensado en los diversos sectores del área de tránsito y así obtener una evaluación para optimizar la toma de decisiones mediante los entrenamientos realizados por el usuario, la API y el software AIMSUN.

3.2. Ambiente de Ingeniería de Software

Es de suma importancia definir metodologías de desarrollo, especificación de técnicas y notaciones así como estandares de documentacion. Implicando una forma coherente de formulación y desarrollo para el presente proyecto, dando no solo una coherencia de la información entregada sino formalidad en el proyecto.

El desarrollo de la API que éste proyecto propone son de las siguientes caracteristicas:

- **Metodología de desarrollo** : Modelo Lineal Cascada, dado que se estipula un listado de requerimientos los cuales se congelan para el desarrollo, y se generarán documentos entregables y existe un tiempo de desarrollo definido.
- Para la especificación de las Funcionalidades del sistema, se utilizará modelo de casos de

uso, diseñados en la herramienta online Draw.io (https://www.draw.io)

- El desarrollo de Diagramas BPMN se utiliza la herramienta online Draw.io (https://www.draw.io)
- La Interfaz Gráfica de la API se realizará con la librería Tkinter, donde esta misma utiliza librerías gráficas Tcl/Tk. Se utilizará estas librerías en particular por venir previamente instalados por el software AIMSUN, por lo cual, facilitará la integración de la API con el sistema MTA.

3.3. Definiciones, Siglas y Abreviaciones

MTA: Microsimulador de Tránsito AIMSUN. Es un potente y versátil paquete de simulación, aplicable a una amplia gama de tareas de planeamiento y modelamiento de tráfico. AIMSUN puede simular toda clase de redes de viales, desde autopistas hasta calles de los centros de las ciudades, y puede analizar redes multimodales de áreas extensas con gran detalle y fidelidad. Usted puede animar el comportamiento de sistemas de tráfico complejos para ilustrar la circulación de tráfico, la operación semafórica, y el funcionamiento conjunto de la red.

BPMN: Business Process Model and Notation (notación y modelado de procesos de negocios), notación gráfica estandarizada que permite el modelado de procesos de negocio

UOCT: La Unidad Operativa de Control de Tránsito (UOCT) es un organismo dependiente del Ministerio de Transporte y Telecomunicaciones de Chile que se encarga de administrar y operar los sistemas de control de tránsito en las principales ciudades de Chile.

Los sistemas de control monitorean en tiempo real de la operación de la mayoría de los semáforos existentes en las ciudades por medio de una red de comunicaciones que conecta cada semáforo con el sistema de control.

API: La interfaz de programación de aplicaciones, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

SCT: Sistema de control de tránsito. La Unidad Operativa de Control de Tránsito (UOCT) de Santiago dispone de un Sistema Centralizado de Control de Tránsito que permite administrar y

supervisar las condiciones de operación, en tiempo real, de las 2.850 intersecciones semaforizadas integradas al sistema.

En la práctica, casi la totalidad de los cruces semaforizados de Santiago están interconectados a este sistema de control. Una de las principales características de este Sistema de Control es que permite monitorear y modificar en línea las programaciones de cada uno de los semáforos, o bien implementar planes especiales de tiempos para mitigar problemas de congestión, producto de incidentes o alto flujo vehicular.[2]

Librerías gráficas: es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, es este caso Python, que ofrece una interfaz bien definida para la funcionalidad que se invoca, específicamente en la GUI de Python.

GUI Python: son las librerías que soporta el lenguaje para crear a través de líneas de código una interfaz gráfica de usuario (GUI en ingles). Los tipos de librerías que soporta Python son los siguientes:

- Tkinter: basadas en las librerías gráficas Tcl/Tk[2].
- WxPython: se basa en WxWidgets[3].
- PyQT: basado en la librería Qt.
- PyGTK: basado en la librería GTK.

Qt: es un framework multiplataforma orientado a objetos ampliamente usado para desarrollar programas (software) que utilicen interfaz gráfica de usuario, así como también diferentes tipos de herramientas para la línea de comandos y consolas para servidores que no necesitan una interfaz gráfica de usuario.[5]

GTK: "GIMP Tool Kit" es una biblioteca del equipo GTK+, la cual contiene los objetos y funciones para crear la interfaz gráfica de usuario. Maneja widgets como ventanas, botones, menús, etiquetas, deslizadores, pestañas, etc.[6]

GTK+: o The GIMP Toolkit es un conjunto de bibliotecas multiplataforma para desarrollar interfaces gráficas de usuario (GUI), principalmente para los entornos gráficos GNOME, XFCE y ROX (usados en los sistemas operativos GNU/Linux) aunque también se puede usar en el

[2] librerías gráficas que se utilizan para la creación de interfaces gráficas.
[3] se caracteriza por ser multiplataforma, por lo que permite el desarrollo rápido de aplicaciones gráficas.

escritorio de Windows, Mac OS y otros.

Inicialmente fueron creadas para desarrollar el programa de edición de imagen GIMP, sin embargo actualmente se usan bastante por muchos otros programas en los sistemas GNU/Linux. Junto a Qt es una de las bibliotecas más populares para Wayland y X Window System.[7]

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86 (solo para equipos Apple) o en un PowerPC.[8]

Capítulo 4

Análisis - Especificación de requerimientos de Software

4.1. Alcances

Formular un sistema de información que apoye el proceso de simulación del MTA. Este sistema tiene como propósito mejorar el proceso actual que tiene el sistema, el cual, el futuro sistema debe generar una interfaz gráfica desarrollada a través de los scripts de Python el que debe de ser capaz de controlar los distintos parámetros utilizados en la simulación del MTA. El MTA está compuesto por diferentes variables que controlan la simulación, el que posee una interfaz gráfica propia para la programación de la simulación, pero sin la propiedad de modificar en el momento dichas variables, es decir, modificar el tiempo de luz verde en un semáforo, por ejemplo, o generar un evento fortuito en las vías de tránsito. Por lo cual, el alcance de éste proyecto tiene como finalidad el desarrollo de un script en lenguaje Python capaz de manipular los parámetros utilizados en la simulación y así modificar en tiempo real la simulación generada por el MTA. Se estima un tiempo de desarrollo del sistema como máximo 6 meses.

De la misma manera en esta formulación del sistema de información no debería por ningún caso que no funcione al momento de implementarse, esto quiere decir que tiene que ser compatible con el MTA y obtener la información requerida por el cliente.

4.2. Objetivo del Software

El objetivo de la API es ayudar y complementar el proceso actual que posee el MTA en la generación de simulaciones.

¿De qué manera será posible mejorar el sistema?

Mediante la implementación de un sistema API a través de los scripts de Python. Para ésto se creará el scripts capaz de manipular las funciones del MTA y los datos descritos anteriormente, modificando la información acorde las necesidades del cliente, además el scripts debe generar una interfaz gráfica para mostrar y modificar los datos, por lo cual, el cliente tendrá un sistema gráfico amigable y acorde a las necesidades para intervenir la simulación.

Será posible para el cliente el ingreso y modificación de datos mediante la interfaz gráfica, administrando de manera optima la simulación acorde a sus necesidades.

4.3. Descripción Global del Producto

4.3.1. Interfaz de usuario

La interfaz de usuario que se implementa en el sistema de información de acuerdo a las necesidades del cliente, serán:

- Uso de ventanas externas y menú en relación al MTA
- Uso de teclado y cursor
- Iconos e Imágenes relacionados con MTA
- Uso de colores legibles
- Uso de Layout para la distribución de elementos en el diseño

Figura 4.1: Mockup Interfaz Gráfica Principal

En la Figura 4.1 se muestra el Mockup[1] de la ventana principal que posee el sistema. En élla, se observan 3 botones principales, el cual, al pulsar el botón "Vehículos" entregará la información relacionada a todos los automóviles que la simulación crea para su funcionamiento, por lo cual, en donde se posiciona el "Multi-line textarea" aparecerá la información requerida por el usuario que esté operando el sistema, en este caso, entregará el "ID" del vehículo y la "Distancia Total" que recorre en el punto que se solicita la información.

De la misma manera, al presionar el botón "Calles", el sistema realiza la consulta al MTA para obtener la información de las calles creadas en la simulación. Donde se encuentra el "Multi-line textarea" se posiciona los datos de "ID" de la calle y "Capacidad" que posee.

Finalmente, al pulsar el botón "SALIR", se cierra la ventana en la que se presiona.

[1] es un prototipo que proporciona una parte de la funcionalidad de un sistema

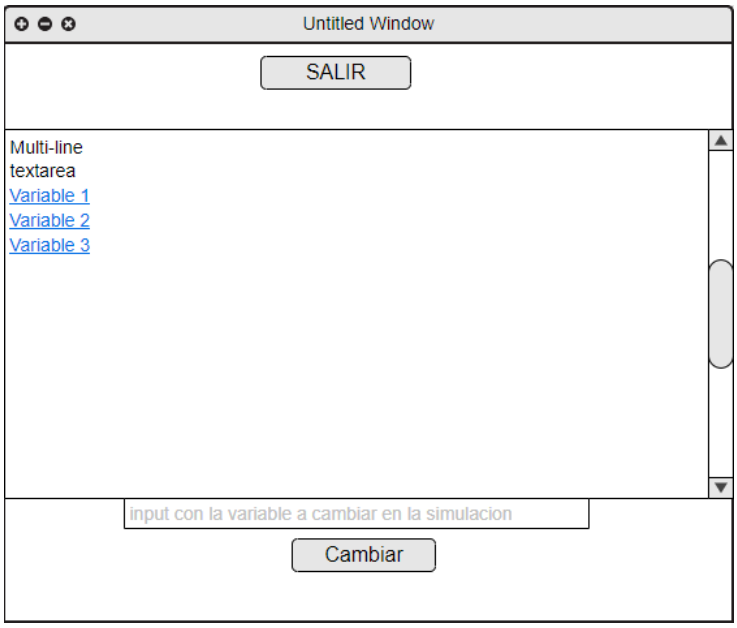


Figura 4.2: Mockup Interfaz Gráfica para el cambio de variables (vehículos y semáforos)

En la Figura 4.2 se muestra el Mockup de la ventana secundaria que posee el sistema. Ésta ventana se inicia al presionar en alguna variable mostrada en el Mockup de la figura 4.1 dentro del "Multi-line textarea". Al pulsar alguna de ésas variables tanto de los semáforos como de los vehículos, muestra los datos completos de dicha variable que se relaciona con el "ID" del dato, entregando una información mas clara y detallada de los datos obtenidos del MTA, por ejemplo, si es acorde a los vehículos, muestra el "ID", distancia recorrida, carril por el cual se encuentra, entre otros.

En la parte inferior de la ventana, se aprecia un "Input", seguido del botón "Cambiar". Ése "Input" es usado al momento de querer modificar alguna variable que se muestra en la ventana secundaria, el cual, al pulsar el botón "Cambiar", el dato se cambia y se guarda en el dato escogido en el "Multi-line textarea". Así el sistema modifica las variables obtenidas en la consulta hacia el MTA.

4.3.2. Interfaz de Hardware

Se utiliza periféricos de comunicación de entrada/salida, específicamente cursor, teclado y pantalla.

Uso de un equipo de trabajo (desktop o laptop) con tarjeta de video externa para el uso del Software AIMSUN.

Especificaciones mínimas del equipo de trabajo:

- Procesador Intel o AMD capaz de soportar 64 bits.
- Windows Vista o superior.
- 1 GB de espacio disponible en el disco duro.
- Monitor con resolución mínima 1024x768.
- 1 puerto USB disponible (para el uso de la llave del software AIMSUN.)
- Tarjeta gráfica Nvidia, ATI o Intel con aceleración de hardware OpenGL 2.0 y 256 MB de RAM de video.

4.3.3. Interfaz de Software

El sistema funcionará de manera paralela al MTA, el cual, sin la ejecución de la simulación del MTA no se podrá lanzar la API para la administración de las variables que utiliza el MTA.

Software AIMSUN previamente instalado en el equipo de trabajo con su respectiva llave USB (Dongle[2])

4.3.4. Interfaces de Comunicación

La interfaz de comunicación proveniente del MTA junto a la API se llevará a cabo a partir de las funciones[3] y métodos[4] provenientes del MTA para comunicarse con el lenguaje Python, es decir, el MTA contiene una amplia cantidad de funciones que pueden ser utilizado (y/o llamados) tanto con el lenguaje C++ como en Python, por lo cual, la comunicación entre el MTA y la API

[2] es un pequeño dispositivo, que se conecta a otro dispositivo para aportar una función adicional.[9]

[3] función: es un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor.

[4] método: conjunto de instrucciones a las que se les asocia un nombre de modo que si se desea ejecutarlas, sólo basta con referenciarlas a través de dicho nombre en vez de tener que escribilas.

será a través de las funciones que contiene el MTA mediante Python, para así manipular los datos necesarios que requiera la API. En particular, se utilizarán los datos de los vehículos generados en la simulación como así también la información de los semáforos.

4.3.5. Consideraciones ambientales

La realización de este sistema en la empresa busca la implementación de un software para obtener una herramienta de entrenamiento a través de la simulación, pensado en los diversos sectores del área de tránsito y así obtener una evaluación para optimizar la toma de decisiones mediante los entrenamientos realizados por el usuario, la API y el software AIMSUN, es decir, la ejecución del proyecto busca mejorar tiempos de respuesta a eventos ocurridos en la vía de tránsito, integrar nueva maquinaria o tecnología, mejorar la eficiencia a sistemas complementarios, entrenamiento a operadores de la UOCT, entrenamiento a carabineros de Chile para adquirir planes de contingencia, entrenamiento a directores de tránsito o simplemente para uso académico.

4.4. Requerimientos Específicos

4.4.1. Requerimientos Funcionales del sistema

ID	Nombre	Descripción	Desarrollador Responsable
01	RFE1	Generar una interfaz gráfica capaz de obtener los datos provenientes de una intersección (datos de vehículos y calles) cualquiera dentro de la simulación.	Diego Parraguez
02	RFE2	Mostrar los datos obtenidos en una ventana relacionados con las calles creados en la simulación.	Diego Parraguez
03	RFE3	Mostrar los datos obtenidos en una ventana relacionados con los vehículos creados en la simulación.	Diego Parraguez
04	RFE4	Generar una ventana nueva para modificar los datos de los vehículos originados en la simulación.	Diego Parraguez
05	RFE5	Generar una ventana nueva para modificar los datos de las calles originadas en la simulación.	Diego Parraguez

Tabla 4.1: Requerimientos Funcionales del sistema

4.4.2. Interfaces externas de entrada

ID	Nombre del ítem	Detalle de datos contenidos en ítem
DE_01	Datos Vehículos (Ventana Principal)	ID Vehículo, Distancia recorrida.
DE_02	Datos Calles (Ventana Principal)	ID Calle, capacidad (vehículo por hora).
DE_03	Datos Vehículo (Ventana Secundaria)	ID Vehículo.
DE_04	Datos Callees (Ventana Secundaria)	ID Calle.

Tabla 4.2: Interfaces externas de entrada

4.4.3. Interfaces externas de salida

ID	Nombre del ítem	Detalle de los datos contenidos en ítem	Medio Salida
IS_01	Datos Vehículos (Ventana Principal)	ID Vehículo, Distancia recorrida.	Pantalla - Ventana Externa.
IS_02	Datos Calles (Ventana Principal)	ID Calle, Limite máximo de velocidad, capacidad (vehículo por hora).	Pantalla - Ventana Externa.
IS_03	Datos Vehículos (Ventana Secundaria)	ID Vehículo, Tipo, ID Sección, Número del carril por el cual va, Desde que sección va, Hasta que sección recorrerá, velocidad actual, Tiempo detenido, Distancia Total recorrida	Pantalla - Ventana Externa.
IS_04	Datos Calles (Ventana Secundaria)	ID Calle, Velocidad límite, capacidad, largo de la calle, Variación del tiempo de reacción de la calle	Pantalla - Ventana Externa.

Tabla 4.3: Interfaces externas de salida

4.4.4. Atributos del producto

- **USABILIDAD - FÁCIL DE COMPRENSIÓN.** El sistema debe constar de una alta entendibilidad. Es clave para la generación de las distintas ventanas que requiere para la entrega de información. Por lo que el correcto uso de sus diferentes unidades métricas será uno de los puntos críticos en relación a los datos requeridos al momento de solicitar la información al MTA.
- **EFICIENCIA - COMPORTAMIENTO EN EL TIEMPO.** El sistema garantiza un tiempo de actualización de estados de datos inferior a los 2 segundos considerando 1 usuario conectado al sistema (al ser utilizado por un solo usuario al requerir una llave USB para el funcionamiento del MTA). Evaluado con un promedio de 720 transacciones por simulación con usuario en condiciones normales[5] y un peak de 3600 por simulación.
- **FUNCIONALIDAD - EXACTITUD.** El sistema debe ser exacto. Las consultas al MTA debe ser con extremo cuidado, es por esto que se hace hincapié en las unidades de medida de los datos y en la forma en que la API realiza las consultas al sistema (por los datos requeridos al realizar la consulta). Toda entrega de información por el sistema está basado en que los datos han sido consultadas de manera correcta mediante las funciones adecuadas.
- **FIABILIDAD - CAPACIDAD DE RECUPERACIÓN.** En caso de fallas, el sistema debe ser capaz de reenviar la última información en la cual se estaba trabajando. No debe ocurrir pérdidas de datos.
- **MANTENIBILIDAD - FACILIDAD DE CAMBIO.** El sistema estará constantemente bajo cambios requeridos por el usuario. Por lo tanto, el sistema debe ser capaz de asumir dichos cambios sin crear mayores problemas para el usuario y el MTA.
- **PORTABILIDAD - FACILIDAD DE AJUSTE.** El sistema debe ser sencillo de trasladarse a los distintos tipos de SO de windows. Es decir que este debe ser posible para el sistema moverse de Windows 7 a 8 sin perder funcionalidades.

[5] refiriéndose a una simulación con un máximo de 1 hora de funcionamiento y realizando consultas al MTA cada 5 segundos y sin anomalías (eventos relacionados con el tráfico) en el transcurso de la simulación.

4.5. Estudio de Factibilidad

4.5.1. Factibilidad Técnica

En este estudio se evalúa si es que se cuenta con tecnologías y recursos tecnológicos, tanto de software como hardware para poder desarrollar el sistema. Además, se evalúa si es que las personas participantes poseen los conocimientos técnicos suficientes para llevar a cabo el correcto desarrollo de este.

4.5.1.1. Recursos de Software

El Software que se utilizará para llevar a cabo la realización del proyecto son los siguientes:

- **Software Aimsun Next 8.2:** "permite llevar a cabo evaluaciones de operaciones de tráfico de cualquier escala y complejidad". Se utiliza para pronosticar dinámicamente las condiciones futuras del tráfico en función del estado actual de la red y para evaluar la respuesta a incidentes o las estrategias de gestión del tráfico.
- **Python:** "Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma". A través de éste lenguaje de programación se desarrollará el proyecto junto a las herramientas de programación que posee (interpretador de código y editor de texto).
- **TeXstudio:** "es un editor de LaTeX de código abierto y Multiplataforma que proporciona un soporte moderno de escritura". Será utilizado para la realización de los informes durante el desarrollo del proyecto.

4.5.1.2. Recursos de Hardware

Para que el proyecto pueda realizarse de manera correcta, se debe contar con un computador que soporte los requerimientos mínimos del software AIMSUN, los cuales son los siguientes:

Versión de Windows®

- Windows® Vista / 7/8 / 8.1 / 10, Windows Server 2008 R2 / 2012/2016, procesador de 64 bits.
- 8 GB de RAM
- 400 MB de capacidad de disco duro
- Monitor a color con una resolución de al menos 1024 × 768
- 1 ranura USB
- Dispositivo señalador compatible con Microsoft Windows®. Se recomienda el mouse de dos botones con rueda de desplazamiento
- Tarjeta gráfica Nvidia, ATI o Intel con aceleración de hardware OpenGL 2.0 y 256 MB de RAM de video.

Versión Mac OS X

- OS X 10.9 - 10.13, procesador de 64 bits.
- 8 GB de RAM
- 400 MB de capacidad de disco duro
- Monitor a color con una resolución de al menos 1024 × 768
- 1 ranura USB
- Dispositivo señalador compatible con Mac OS X. Se recomienda el mouse de dos botones con rueda de desplazamiento
- Tarjeta gráfica Nvidia, ATI o Intel con aceleración de hardware OpenGL 2.0 y 256 MB de RAM de video.

Versión de Linux®

- Distribución compatible con Ubuntu 14/16/17, procesador de 64 bits.
- 8 GB de RAM
- 400 MB de capacidad de disco duro
- Monitor a color con una resolución de al menos 1024 × 768
- 1 ranura USB
- Dispositivo señalador compatible con Linux. Se recomienda el mouse de dos botones con rueda de desplazamiento

- Tarjeta gráfica Nvidia, ATI o Intel con aceleración de hardware OpenGL 2.0 y 256 MB de RAM de video.

Requisitos adicionales para aplicaciones de gama alta

Visualización 3D intensiva

- 16 GB de RAM
- Tarjeta gráfica de alta gama
- 1-20 GB de capacidad adicional del disco duro
- Monitor a color con un tamaño de pantalla de al menos 19 pulgadas y una resolución de al menos 1440 x 900
- Tarjeta gráfica de la estación de trabajo Nvidia o ATI con soporte OpenGL 2.0 y al menos 1 GB de RAM
- Una aplicación de modelado y visualización 3D como Autodesk 3ds Max o Maya, SketchUp Pro, etc.
- Una aplicación de edición de video como Camtasia Studio, Adobe Premiere, etc.

Redes muy grandes

- CPU multi-core más rápido que 3 GHz
- 16 GB de RAM (se recomiendan 32 GB)
- Soporte para el intercambio de datos de 1 Gbit

Servidor computacional

- CPU de 6 núcleos más rápido que 3 GHz
- 64 GB de RAM
- Soporte para el intercambio de datos de 1 Gbit

Para la realización y desarrollo del proyecto, utilizaremos la versión de Windows, por lo que se utilizará el siguiente hardware:

- Procesador Intel Core i5 7300HQ
- 8GB de RAM
- Pantalla LED 15.6"
- Almacenamiento 1 TB

- Tarjeta de Video NVIDIA GeForce GTX 1050 (4GB)

Notebook: ASUS FX503VD-DM210T

4.5.1.3. Recursos Humanos

Para realizar este proyecto se necesitan los siguientes roles:

- Jefe de proyecto
- Analista
- Diseñador
- Programador
- Tester

Se poseen los conocimientos necesarios para afrontar cada uno de los roles que se necesitan para el desarrollo del proyecto, esto por los conocimientos adquiridos en la carrera Ingeniería Civil en Informática, al encontrarme en el último año de carrera. Por lo que realizaré cada uno de los roles necesarios para la realización del proyecto.

Por lo tanto, se puede concluir, que, desde un punto de vista técnico, es factible la realización del software ya que se cuentan con todas las tecnologías necesarias para desarrollar el proyecto de manera correcta, además, se cuenta con el personal adecuado para dirigir, analizar, diseñar, programar y realizar las pruebas de testeo necesarias.

4.5.2. Factibilidad Operativa

La factibilidad operativa comprende el análisis para que el nuevo sistema a desarrollar sea utilizado de manera correcta, utilizando al máximo las funcionalidades que éste provea. Además, se analiza el impacto que el sistema ocasiona a la empresa y a los usuarios del mismo.

Los usuarios de éste sistema son:

- Alumnos(as) de la Universidad del Bío-Bío que utilicen el software AIMSUN.
- Usuarios con los conocimientos para utilizar el software AIMSUN.
- Operadores de la Unidad Operativa de Control de Tránsito de Concepción (UOCT)

Los niveles de conocimiento necesarios por los usuarios para la correcta utilización del siste-

ma son medios-avanzados, ya que comprenden del uso del software AIMSUN (nivel de usuario intermedio). Se asume que estos poseen dichos conocimientos ya que será por medio de AIMSUN el uso del software a crear.

El impacto que generará la implementación del nuevo sistema hacia los usuarios, será completamente positivo, ya que mejorará el uso del programa Aimsun, creando una aplicación mas dinámica por poseer funcionalidad que se utiliza en tiempo de ejecución al modificar datos acorde al transcurso de la simulación. De ésta manera, se creará un software de entrenamiento para los usuarios descritos anteriormente.

Con esto se puede concluir, que, desde el punto de vista operativo, el sistema es factible.

4.5.3. Factibilidad Económica

En el estudio de factibilidad económica, se han considerado los costos del proyecto en términos del prototipo actual. A continuación, se detallan los montos necesarios que se deberán asumir para la realización de este proyecto en pesos chilenos (CLP):

	Descripción	Valor
Recursos Humanos	Remuneración del integrante del grupo de trabajo	$0
Llave AIMSUN	USB utilizado como llave para poder iniciar el software AIMSUN	$ 1.500.000.- (en el caso del proyecto $ 0[1])
Python	Lenguaje de programación utilizado para el desarrollo de la aplicación.	$ 0
TeXstudio	es un editor de LaTeX de código abierto y Multiplataforma que proporciona un soporte moderno de escritura	$ 0
Draw.io	Aplicación utilizada para la creación de distintos diagramas y modelos.	$ 0
TOTAL		$ 1.500.000.- ($0 en el proyecto)

Tabla 4.4: Factibilidad Económica

El hardware necesario para la realización del proyecto (terminales) no se incluye, ya que, estos corresponden al equipo personal (notebooks) del alumno.

4.5.4. Beneficios

Como el desarrollo de la aplicación posee múltiples alcances para los usuarios, estos se detallarán de la siguiente manera:

- Alumnos(as) de la Universidad del Bío-Bío (UBB).
- Usuarios de la UOCT.

Para los Alumnos(as) de la UBB adquieren el conocimiento del uso del software AIMSUN

[1]Se queda como costo $ 0 porque el proyecto al ser de la UBB, ésta posee una llave para el software AIMSUN.

junto a la API, por lo que monetaria mente no posee costo.

Contrariamente, a los alumnos(as) de la UBB, los Operadores de la UOCT si se deben de realizar cálculos acorde a los beneficios que conlleva realizar el proyecto, los cuales son los siguientes:

Costos sin el proyecto.

- Duración contrato a plazo de un nuevo operador: 1 meses (180 Horas).
- Sueldo mensual del operador: $400.000.-
- Valor de hora hombre (HH): $2.222.-

180 horas x $ 2.222 HH = $399.960.-

El valor que acabamos de calcular corresponde al tiempo llevado a dinero, que ocupa el usuario al usar el sistema actual durante 1 mes (4 semanas), si esto lo llevamos a nivel de 3 meses, el cual dura un contrato a plazo, el resultado sería el siguiente:

$ 399.960 x 3 meses = $1.199.880.-

Ahora bien, con la implementación del proyecto, el operador debe aumentar su curva de aprendizaje bajo el nuevo sistema, por lo que se estima el siguiente cálculo:

Costos con el proyecto.

- Duración contrato a plazo de un nuevo operador: 2 semanas (90 Horas).
- Sueldo mensual del operador: $400.000.-
- Valor de hora hombre (HH): $2.222.-

90 horas x $ 2.222 HH = $ 199.980.-

El valor que se acaba de calcular corresponde al tiempo llevado a dinero en relación al entrenamiento para un operador con nuevo sistema. Ésto, llevado a un plazo de 1 mes, sería de la siguiente manera:

180 horas x $ 2.222 HH = $ 399.960.-

Con el nuevo sistema, un operador de la UOCT se tendría que demorar aproximadamente 1 mes en adquirir el conocimiento para el manejo de manera mas optima para la toma de decisiones relacionado a una red de tránsito, por lo que la UOCT puede ahorrarse 2 meses de sueldo de un operador para el entrenamiento de éste.

De esta manera se puede estipular lo siguiente:

Ahorro de la UOCT $ 799.920 por operador.

Capítulo 5

Análisis - Casos de uso

5.1. Proceso de negocio - Futuro

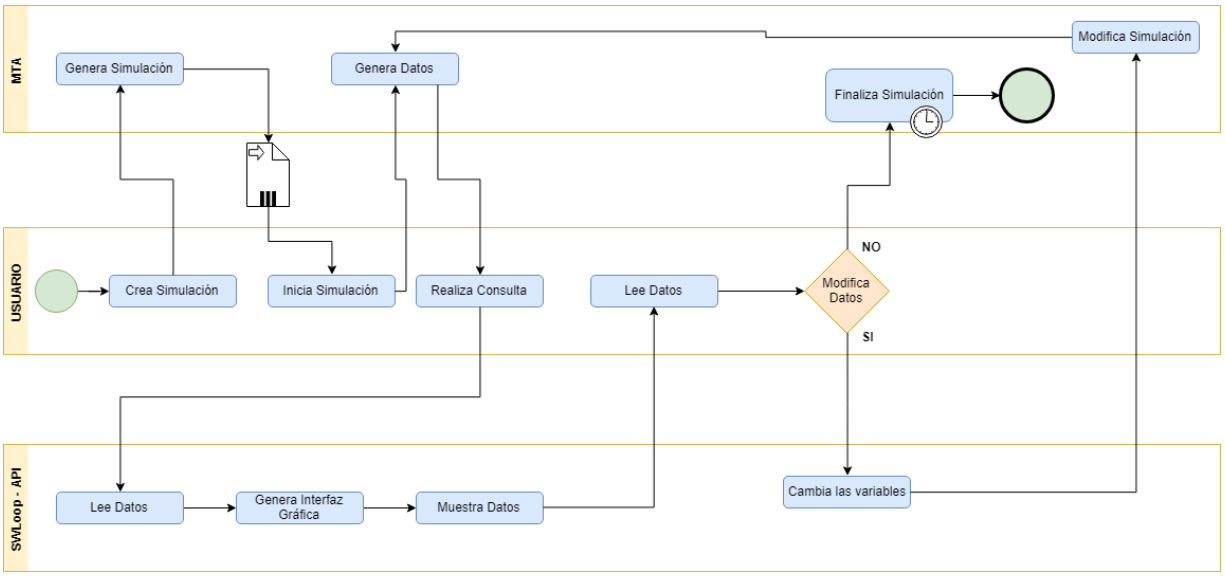


Figura 5.1: BPMN MTA - Cliente/Usuario - API

En la figura 5.1 se puede observar el nuevo modelo de negocio, representado por la notación BPMN para el futuro sistema. Se identifica como iniciador general de las tareas al usuario/cliente, relacionándose directamente entre el MTA y el SWLoop - API.

5.2. Modelo de Casos de Uso

5.2.1. Diagrama de Casos de Uso

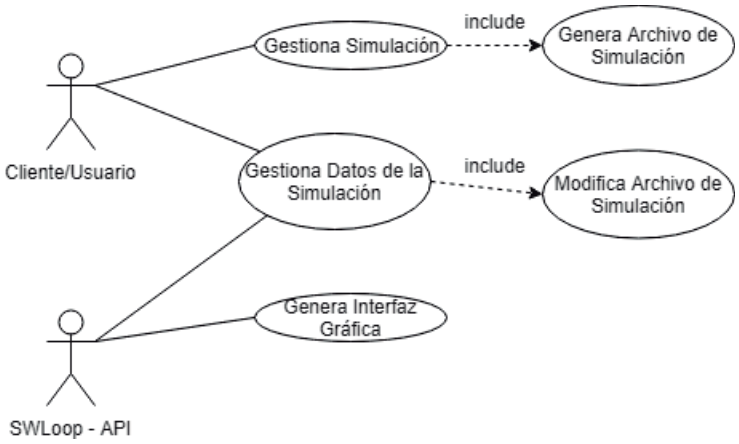


Figura 5.2: Modelo de Casos de Uso para Sistema de simulación en tiempo real

5.2.2. Actores

Actor	Cargo	Descripción cargo/funciones	Nivel de conocimientos técnicos	Nivel de privilegios del sistema
Usuario Cliente	Administrador	Encargado de utilizar el sistema, gestiona las simulaciones del MTA y los datos de ésta.	Medio - avanzado en el uso del MTA.	Nivel alto, control total del sistema.
SWLoop API	Gestión de los datos de la simulación	Encargado de gestionar los datos del MTA.	Avanzados	Nivel restringido para el uso del MTA.

Tabla 5.1: Actores

5.2.3. Especificación de los Casos de Uso

Caso de Uso	Desarrollador
Gestionar Simulación	Diego Parraguez
Gestionar datos de simulación	Diego Parraguez
Generar interfaz gráfica	Diego Parraguez

Tabla 5.2: Tabla resumen con los Casos de Uso y Desarrollador encargado

5.2.3.1. Caso de Uso: Gestionar simulación

- **Descripción:** en este caso de uso el usuario/cliente podrá crear, modificar y eliminar la simulación en el sistema.
- **Actores:** Usuario/Cliente.
- **Pre-condiciones:** El usuario/cliente debe iniciar el Software AIMSUN y poseer la llave USB pertinente. Tener al menos un archivo generado por el MTA para la modificación y eliminación de la simulación.
- **Flujo de eventos básico:**

Actor	Sistema
1.- Éste caso de uso inicia cuando el usuario requiere gestionar una simulación	2.-
3(a).- El usuario crea la simulación en el MTA.	4.- Genera los datos solicitados de manera gráfica dentro de la ventana principal del software, generando un archivo de salida con los datos de la simulación (red de tránsito).

Tabla 5.3: Caso de Uso Gestionar simulación - Crear Simulación

- **Flujo de eventos alternativos:**

Actor	Sistema
3(b).- El usuario modifica la simulación en el MTA.	4.- Realiza los cambios dentro de la simulación.
5.- Guarda las modificaciones	6.- Sobrescribe el archivo original con los cambios realizados y muestra gráficamente la red de tránsito modificada.

Tabla 5.4: Flujo Alternativo - Modificación de la Simulación

Actor	Sistema
3(c).- El usuario selecciona el archivo generado y lo elimina.	4.- Muestra mensaje de confirmación de eliminación.
5.- El usuario selecciona la opción de confirmar.	6.- Elimina todo registro de la simulación.

Tabla 5.5: Flujo Alternativo - Eliminar Simulación

- **Post-condiciones:** En el caso de modificación de la simulación, los cambios quedan realizados en el archivo generado por el sistema. Para el caso de flujo de eventos alternativo 3(c), elimina el archivo generado por el sistema.

5.2.3.2. Caso de Uso: Gestionar datos de simulación

- **Descripción:** en este caso de uso el usuario/cliente podrá leer y modificar los datos de la simulación en el MTA.
- **Actores:** Usuario/Cliente.
- **Pre-condiciones:** El usuario/cliente debe iniciar el Software AIMSUN y poseer la llave USB pertinente. Tener al menos un archivo generado por el sistema para la lectura y

modificación de datos de la simulación.

- **Flujo de eventos básico:**

Actor	Sistema
1.- Éste caso de uso inicia cuando el usuario requiere gestionar los datos de la simulación	2.-
3.- El usuario inicia el script de la API.	4.- Despliega la ventana principal de la API.
5(a).- El usuario selecciona el botón de vehículos.	6.- Entrega la información requerida por medio de la API, mostrando los datos a través de la ventana principal de la API.
7.- El usuario selecciona el vehículo que requiere información.	8.- Se despliega una ventana nueva con toda la información pertinente al vehículo seleccionado.

Tabla 5.6: Caso de Uso Gestionar datos de simulación - Leer datos vehículos

- **Flujo de eventos alternativos:**

Actor	Sistema
5(b).- El usuario selecciona el botón calles.	6.- Entrega la información requerida por medio de la API, mostrando los datos a través de la ventana principal de la API.
7.- El usuario selecciona la calle que requiere información.	8.- Se despliega una ventana nueva con toda la información pertinente a la calle seleccionada.

Tabla 5.7: Flujo Alternativo - Leer datos de Calles

Actor	Sistema
5(c).- El usuario selecciona el botón de vehículos.	6.- Entrega la información requerida por medio de la API, mostrando los datos a través de la ventana principal.
7.- El usuario selecciona el vehículo a modificar.	8.- Se despliega una ventana nueva con los campos disponibles para editar (en este caso solo la velocidad máxima del vehículo).
9.- Selecciona el campo "velocidad máxima", modifica su valor y se envía (por medio de la API) el dato modificado a través del botón correspondiente.	10.- Recibe el dato modificado y se actualiza la simulación.

Tabla 5.8: Flujo Alternativo - Modificar datos de vehículos

Actor	Sistema
5(d).- El usuario selecciona el botón de calles.	6.- Entrega la información requerida por medio de la API, mostrando los datos a través de la ventana principal de la API.
7.- El usuario selecciona la calle a modificar	8.- Se despliega una ventana nueva con los campos disponibles para editar (en este caso solo la velocidad límite).
9.- Selecciona el campo "Velocidad límite", modifica su valor y se envía (por medio de la API) el dato modificado a través del botón correspondiente.	10.- Recibe el dato modificado y se actualiza la simulación.

Tabla 5.9: Flujo Alternativo - Modificar datos de Calles

- **Post-condiciones:** En el caso de modificación de datos de la simulación, los cambios quedan realizados en el archivo generado por el sistema.

5.2.3.3. Caso de Uso: Gestionar datos de simulación

- **Descripción:** en este caso de uso el SWLoop - API podrá leer y modificar los datos de la simulación en el MTA.
- **Actores:** SWLoop - API.
- **Pre-condiciones:** El usuario/cliente debe iniciar el Software AIMSUN, poseer la llave USB pertinente e iniciar el script de la API. Tener al menos un archivo generado por el sistema para la lectura y modificación de datos de la simulación.
- **Flujo de eventos básico:**

Actor	Sistema
1.- Éste caso de uso inicia cuando la API requiere gestionar los datos de la simulación	2.-
3(a).- La API solicita los datos de los vehículos.	4.- Entrega la información requerida y muestra los datos a través de la ventana principal de la API.
5.- La API requiere información detallada de un vehículo en particular.	6.- Se despliega una ventana nueva con toda la información pertinente al vehículo requerido.

Tabla 5.10: Caso de Uso Gestionar datos de simulación - Leer datos vehículos (API)

- **Flujo de eventos alternativos:**

Actor	Sistema
3(b).- La API solicita los datos de las calles.	4.- Entrega la información requerida y muestra los datos a través de la ventana principal de la API.
5.- La API requiere información detallada de una calle en particular.	6.- Se despliega una ventana nueva con toda la información pertinente a la calle requerida.

Tabla 5.11: Flujo Alternativo - Leer datos de calles (API)

Actor	Sistema
3(c).- La API solicita los datos de los vehículos.	4.- Entrega la información requerida y muestra los datos a través de la ventana principal de la API.
5.- La API requiere información detallada de un vehículo en particular.	6.- Se despliega una ventana nueva con toda la información pertinente al vehículo requerido.
7.- La API solicita el campo “velocidad máxima”, se modifica su valor y se envía el dato modificado a través del botón correspondiente.	8.- Recibe el dato modificado y se actualiza la simulación.

Tabla 5.12: Flujo Alternativo - Modificar datos de vehículos (API)

Actor	Sistema
3(c).- La API solicita los datos de las calles.	4.- Entrega la información requerida y muestra los datos a través de la ventana principal de la API.
5.- La API requiere información detallada de una calle en particular.	6.- Se despliega una ventana nueva con toda la información pertinente a la calle requerida.
7.- La API solicita el campo "Velocidad límite", se modifica su valor y se envía el dato modificado a través del botón correspondiente.	8.- Recibe el dato modificado y se actualiza la simulación.

Tabla 5.13: Flujo Alternativo - Modificar datos de calles (API)

- **Post-condiciones:** En el caso de modificación de datos de la simulación por medio de la API, éstos quedan realizados en el archivo generado por el sistema.

5.2.3.4. Caso de Uso: Generar interfaz gráfica

- **Descripción:** en este caso de uso el SWLoop - API gestionará una interfaz gráfica dentro de la simulación en el MTA.
- **Actores:** SWLoop - API.
- **Pre-condiciones:** El usuario/cliente debe iniciar el Software AIMSUN, poseer la llave USB pertinente e iniciar el script de la API. Tener al menos un archivo generado por el sistema para el funcionamiento de la simulación.
- **Flujo de eventos básico:**

Actor	Sistema
1.- Éste caso de uso inicia cuando la API genera una interfaz gráfica	2.-
3.- La API solicita las librerías para crear la interfaz gráfica.	4.- Entrega la información de las librerías solicitadas.
5.-	6.- Crea la ventana principal por medio de las librerías del sistema (Tcl/Tk).

Tabla 5.14: Caso de Uso Generar interfaz gráfica

Capítulo 6

Caso de Estudio

En este capitulo se abarcará un caso de estudio para el uso del MTA junto a la API. Éste estudio contemplará la inserción de un accidente en las vías de tránsito, por lo cual, modificará variables en la ejecución de la simulación.

Al estudiar y comprender el contenido del manual del MTA[10–14] se contempla una función en particular que tiene que ver con los incidentes dentro de la simulación[11], la cual, lleva parámetros específicos para su implementación. Su estructura esta conformada por los siguientes valores;

int **AKIGenerateIncident**(int **asection**, int **alane**, double **position**, double **length**, double **initime**, double **duration**, double **visibilityDistance**, bool **updateIdGroup**, bool **applySpeedReduction**, double **upstreamDistanceSR**, double **downstreamDistanceSR**, double **maxSpeedSR**)

en donde:

- asection: Identificador de la sección (o calle) donde se generará el incidente.
- alane: Carril donde se generará el incidente (1..N).
- position: Posición del incidente.
- length: Longitud del incidente.
- initime: Tiempo absoluto de la simulación cuando el incidente comience (segundos).
- duration: Duración del incidente(segundos).

- visibilityDistance: Distancia de visibilidad en metros del incidente. El valor predeterminado es de 200 metros al crear el incidente utilizando la interfaz gráfica del software.
- updateIdGroup: Verdadero cuando el incidente es un nuevo grupo de incidentes y falso si los incidentes se deben tratar como parte del último incidente creado (al crear incidentes en carriles adyacentes que deben tratarse como un todo).
- applySpeedReduction: Verdadero para aplicar una reducción de velocidad alrededor del incidente (reducir la velocidad de los vehículos a medida que pasan) y en caso contrario, falso.
- upstreamDistanceSR: si se aplica la reducción, la distancia antes del incidente. El valor predeterminado es 200 m.
- downstreamDistanceSR: si se va a aplicar la reducción, la distancia después del incidente. El valor predeterminado es 200 m.
- maxSpeedSR: si se aplica la reducción, el objetivo reduce la velocidad. El valor predeterminado es 50 km / h.

Así, utilizando el MTA, creando un nuevo script de Python, se coloca en marcha la función descrita anteriormente para así generar un evento o accidente dentro de la simulación y de este modo colocar en práctica el caso de estudio.

El poner en práctica el caso de estudio tiene como funcionalidad descubrir la toma de decisiones que poseen los operadores del MTA relacionado con los accidentes en la red de tránsito, para así descubrir, si las hay, las falencias que hay en ellos y de ésta manera afrontar dichas carencias, entrenándolos de mejor forma en relación a la toma de decisiones de eventos fortuitos dentro de una red de tránsito. Dicho de otra forma, el principal objetivo del caso de estudio es el entrenamiento de operadores del MTA relacionado a la toma de decisiones en cuanto a accidentes en las vías del tránsito.

Capítulo 7

Conclusión

7.1. Conclusiones

El desarrollo de este proyecto de sistema de información implicó un acabado conocimiento del ambiente donde se debe integrar el sistema. Al utilizar metodología lineal Cascada para el desarrollo, es necesario aplicar ingeniería de requerimientos en la primera etapa de desarrollo y por tanto, es necesario entrevistarse con el cliente para generar en forma conjunta los requerimientos funcionales y no funcionales del sistema.

Considerando el tipo de proyecto que se desarrolla, se puede determinar que la metodología escogida (Cascada) es la más adecuada para el desarrollo, ya que se pueden congelar los requerimientos luego de definirlos, y generar un contrato de desarrollo. También la empresa a la que se le desarrolla (el cliente) expresa que el sistema trabaja con información delicada y por tanto se necesita un muy bajo nivel de fallas, y requiere documentación necesaria para realizar mantención futura del sistema y que permite ciertos niveles de escalabilidad.

Por tanto se puede decir que en comparación con otras metodologías, las ventajas que presenta cascada son contundentes y permite tener una completa documentación de cada etapa de desarrollo permitiendo al desarrollador y al cliente mantener un orden en cuanto a costos y tiempo de desarrollo e implementación del sistema.

Durante este proyecto, se ha tenido la posibilidad de valorar la documentación del software

AIMSUN, por ser fundamental para el desarrollo del sistema, debido a que el software, si bien es una herramienta completa para la simulación de redes de tránsito, no posee una herramienta para la entrega y modificación de información en tiempo real, por lo que, al poseer un editor y compilador del lenguaje Python permite realizar dicha funcionalidad a través de dicho lenguaje de programación.

La documentación entregada por el MTA ayuda a comprender el funcionamiento del software, entregando las herramientas para el desarrollo del proyecto descrito anteriormente. Sin un estudio minucioso de la documentación del MTA [1, 10–14] no es posible realizar el proyecto desarrollado previamente, por la gran cantidad de funciones que contiene AIMSUN en sus librerías. Sin embargo al utilizar y estudiar detalladamente su documentación, se encontró una falla en el sistema, esto ocurrió al querer utilizar las librerías gráficas Tcl/Tk, teniendo que contactar al soporte de AIMSUN, dando una pronta respuesta y enviándome los archivos necesarios para solucionar el problema, agradeciendo así mismo por notificar dicha falla y encargándose de solucionar lo para la futura versión del microsimulador.

Adicionalmente, para utilizar las funciones que AIMSUN entrega, se debe realizar una configuración previa para que, durante la simulación, entregue los datos que se solicitan. Dicha configuración queda registrada en el capítulo 8.

Por otra parte, es necesario capacitar a los operadores del software (MTA), por incorporar una nueva ventana en el funcionamiento de este, complementando de manera gráfica al sistema final, entregando datos específicos relacionados con los vehículos y las calles que la simulación posee, por lo que ahora, en tiempo de simulación, se pueden modificar ciertos parámetros. En el caso de los vehículos, la modificación que se puede realizar corresponde a la velocidad máxima por la que éste puede circular dentro de la red de tránsito de la simulación, en cambio, en las calles se puede modificar la velocidad límite que pueden circular por dicha calle, adicionalmente, muestra datos estadísticos pertenecientes a la calle seleccionada, por lo que se puede interpretar para el uso de la administración de la simulación.

En relación a la creación de una simulación dentro del software, éste posee un tutorial para captar a nuevos usuarios, pero cabe destacar que una persona sin conocimientos previos, tardará

en realizar una simulación completa de una red de tránsito, debido a la amplia cantidad de herramientas para la creación y estructuración que necesita una microsimulación, por lo que se recomienda, para hacer funcionar de manera eficiente y optima el software, una simulación y el sistema completo, es poseer experiencia previa.

Se propone un caso de estudio para el estudio y entrenamiento a operadores del software AIMSUN, por lo que hará el sistema mas completo al momento de utilizar el software, permitiendo ademas el uso académico a usuarios con conocimientos del software.

7.2. Trabajos futuros

El MTA al poseer una amplia gama de librerías relacionado con la gestión de información y administración de redes de tránsito, es posible ampliar el sistema realizado, pudiendo adicionar funcionalidad al software, por lo que, al tener mas de 230 funciones éste proyecto fue acotado a relacionarse, hasta el momento, con dos parámetros importante dentro de una simulación, como lo son los vehículos y las calles de la simulación, por lo que ir adicionando funcionalidad al software es posible junto al lenguaje Python.

Un ejemplo de un futuro trabajo para éste proyecto es la incorporación de la manipulación del transporte publico en una red de tránsito, ampliando el espectro de trabajo y estudio al utilizar el software. Otro trabajo que puede llevarse a cabo a posterior, es el cambio de tiempo en los semáforos que contiene la simulación, para así, de manera mas rápida administrar y manejar la red vehicular simulada bajo el software de AIMSUN.

Para realizar los futuros trabajos descritos anteriormente se debe tener presente la lectura del capitulo 8 para la previa configuración que necesita el software AIMSUN.

Capítulo 8

Anexos

8.1. Manual de Usuario

8.1.1. Introducción

En este manual se encuentra los problemas encontrados durante la realización del proyecto, junto a la soluciones y configuración previa que debe de contener el software AIMSUN para el correcto funcionamiento de la API.

Se realiza dicho manual para entregar todos los conocimientos adquiridos durante el desarrollo de la aplicación y así aportar con los trabajos posteriores que se pueden realizar al software AIMSUN.

8.1.2. Problemas - Solución

La forma en que el software AIMSUN genera una interfaz gráfica a través el lenguaje Python es mediante la librería Tcl/Tk, la cual, viene incorporada en AIMSUN.

Ahora bien, al comenzar a utilizar los scripts de Python para generar la interfaz gráfica se utiliza la librería Tcl/Tk, la que se llama utilizando la siguiente linea de código:

```
import Tkinter as tk
```

Al momento de utilizar dicha línea de código ya se pueden utilizar las clases que la librería Tcl/Tk contiene, las que particularmente se utilizan, por ejemplo; la clase tk.Tk()[1], Frame[2], Scrollbar[3], Listbox[4], entre otros.

```
import Tkinter as tk

win = tk.Tk()
win.geometry('800x600')
win.title("Pantalla Principal - SWLoop")

win.mainloop()
```

Luego de comenzar a programar la ventana del sistema se generaron errores en el log de AIMSUN, los que iban directamente relacionados a la programación que se estaba realizando con la ventana principal

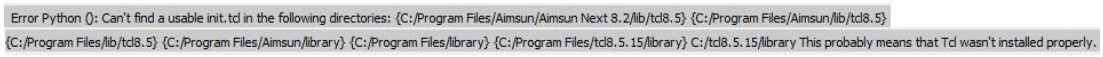


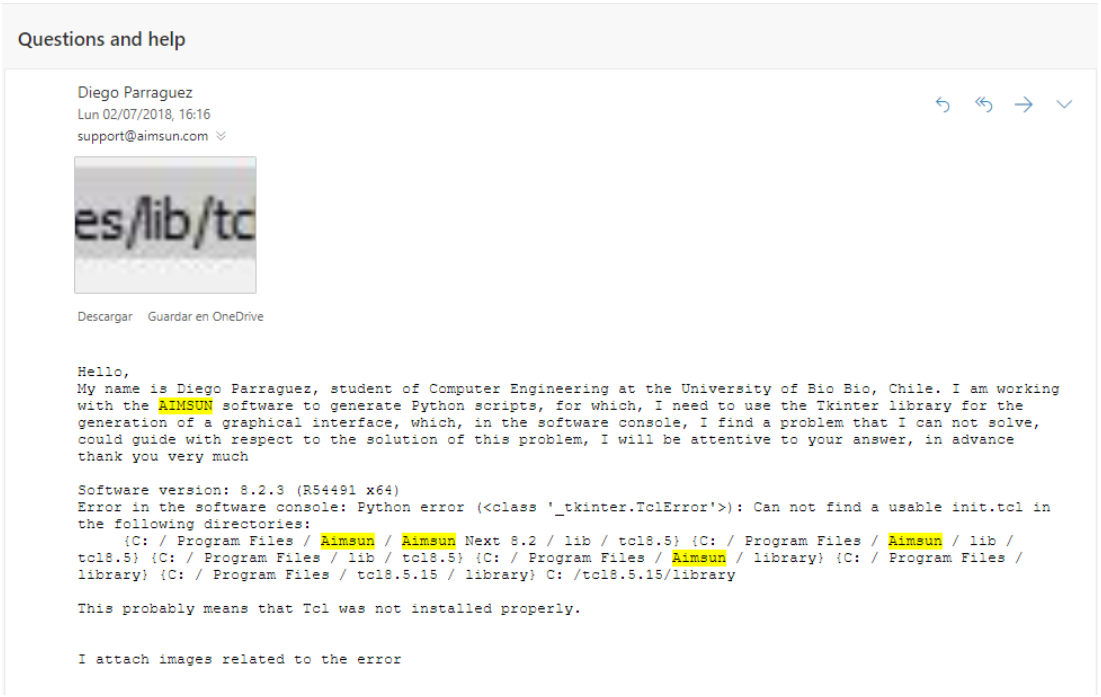
Figura 8.1: Error log AIMSUN

Al encontrarme con el error de la figura 8.1, comencé a buscar en internet las posibles soluciones a dicho error, encontrando resultados culpando a que no se encontraba instalada la librería Tcl/Tk. Por lo que seguidamente comencé la búsqueda para posteriormente instalar dicha librería de manera manual. Ya finalizado el proceso, se comenzaron nuevamente las pruebas dentro del software AIMSUN encontrándome nuevamente con el error. Así indagando e investigando no encontré solución alguna, primero, porque todo se relacionaba con el lenguaje Python y no bajo algún software como lo utiliza AIMSUN y segundo, el software AIMSUN es pagado, por lo cual, todo problema encontrado con el programa se utiliza la vía de ayuda que provee el software, en

[1] Clase utilizada para generar una ventana gráfica.
[2] Clase para crear un marco, utilizado para maquetar una ventana gráfica.
[3] Clase para crear una barra de desplazamiento, va orientado al Frame creado.
[4] Clase que genera una lista de atributos, va orientado junto al Frame.

primera instancia se estudió la documentación del software, la cual, al no encontrar solución en ésta, se recurrió a la siguiente vía de ayuda la que corresponde vía e-mail.

De esta forma se redacta y envía el siguiente correo:


Questions and help

Diego Parraguez
Lun 02/07/2018, 16:16
support@aimsun.com

Descargar Guardar en OneDrive

Hello,
My name is Diego Parraguez, student of Computer Engineering at the University of Bio Bio, Chile. I am working with the AIMSUN software to generate Python scripts, for which, I need to use the Tkinter library for the generation of a graphical interface, which, in the software console, I find a problem that I can not solve, could guide with respect to the solution of this problem, I will be attentive to your answer, in advance thank you very much

Software version: 8.2.3 (R54491 x64)
Error in the software console: Python error (<class '_tkinter.TclError'>): Can not find a usable init.tcl in the following directories:
{C: / Program Files / Aimsun / Aimsun Next 8.2 / lib / tcl8.5} {C: / Program Files / Aimsun / lib / tcl8.5} {C: / Program Files / lib / tcl8.5} {C: / Program Files / Aimsun / library} {C: / Program Files / library} {C: / Program Files / tcl8.5.15 / library} C: /tcl8.5.15/library

This probably means that Tcl was not installed properly.

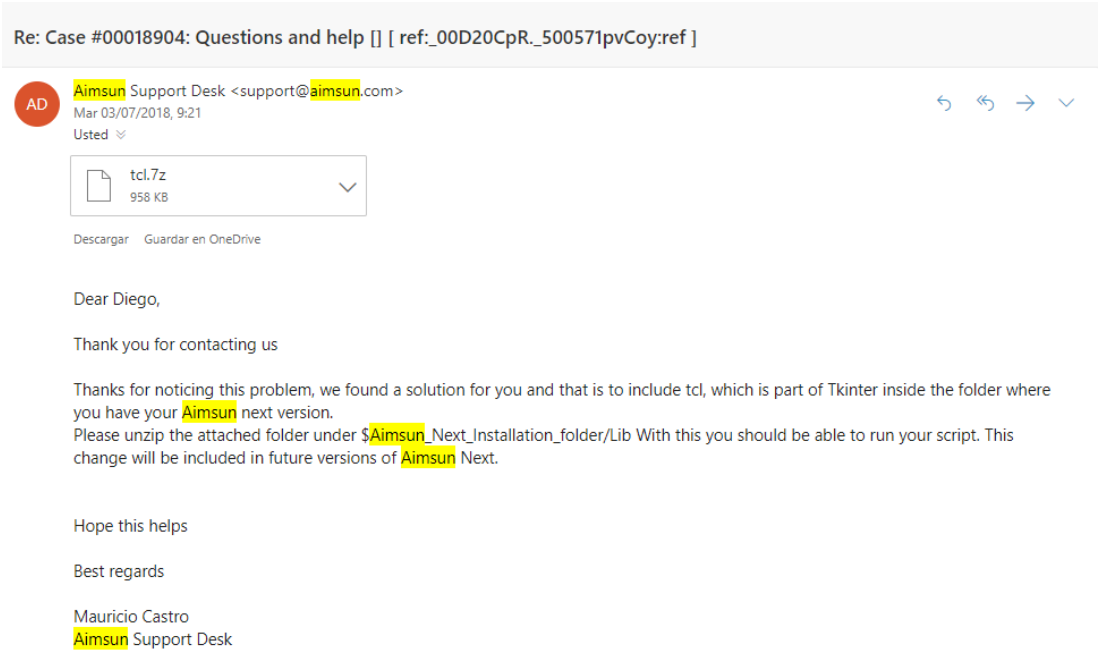
I attach images related to the error

Figura 8.2: Correo problema librería Tcl/Tk

Traduciendo lo principal del correo de la figura 8.2 podemos destacar los siguiente:

"... Necesito usar la biblioteca Tkinter para la generación de una interfaz gráfica, que, en la consola del software, encuentro un problema que no puedo resolver, podría guiar con respecto a la solución de este problema..."

Seguidamente respondieron lo siguiente:


Re: Case #00018904: Questions and help [] [ref:_00D20CpR._500571pvCoy:ref]

AD Aimsun Support Desk <support@aimsun.com>
Mar 03/07/2018, 9:21
Usted

tcl.7z
958 KB

Descargar Guardar en OneDrive

Dear Diego,

Thank you for contacting us

Thanks for noticing this problem, we found a solution for you and that is to include tcl, which is part of Tkinter inside the folder where you have your Aimsun next version.
Please unzip the attached folder under $Aimsun_Next_Installation_folder/Lib With this you should be able to run your script. This change will be included in future versions of Aimsun Next.

Hope this helps

Best regards

Mauricio Castro
Aimsun Support Desk

Figura 8.3: Correo solución librería Tcl/Tk

En el correo de la figura 8.3 menciona lo siguiente;

"Querido Diego:

Gracias por contactarnos

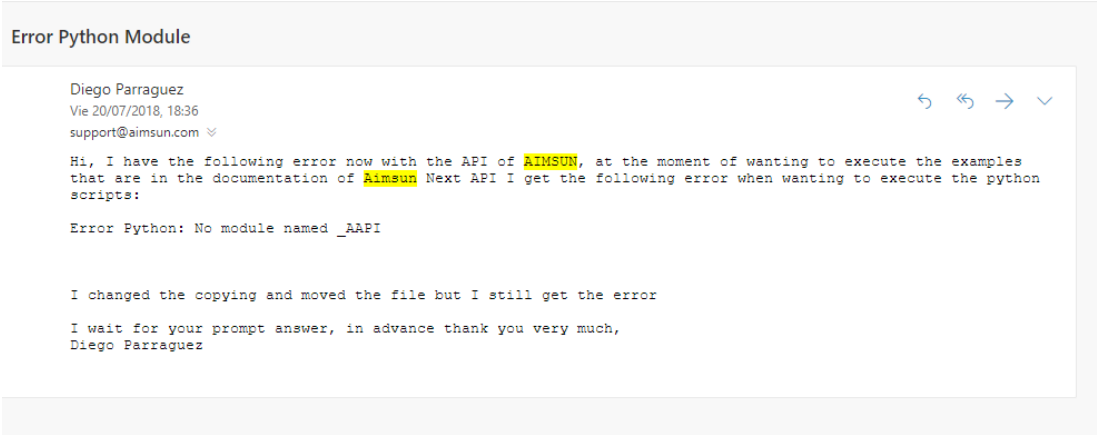
Gracias por notar este problema, encontramos una solución para ti y es incluir tcl, que es parte de Tkinter dentro de la carpeta donde tienes tu versión de Aimsun next. Descomprima la carpeta adjunta en $ Aimsun_Next_Installation_folder / Lib. Con esto, debería poder ejecutar su script. Este cambio se incluirá en futuras versiones de Aimsun Next.

Espero que esto ayude"

Dentro del correo de la figura 8.3 se adjuntaron los archivos necesarios para la utilización de la librería Tlc/TK, así mismo, se descomprimieron los archivos enviados dentro de la carpeta especificada en el correo, así solucionando el problema que se originaba al intentar generar una ventana dentro de los scripts de Python de Aimsun.

Al momento de estudiar la documentación del programa se encontró información relacionada al desarrollo del proyecto (muestra y modificación de datos en tiempo de simulación), por lo que

se comenzó a programar la muestra de datos en la ventana principal dentro de los scripts de Python, encontrando un nuevo error proviniendo de la utilización de las librerías del software, por lo que nuevamente se envió un correo solicitando la solución pertinente. Hay que destacar para que AIMSUN muestre los datos de la simulación se utiliza un archivo nombrado AAPI con extensión .py[5] o .cxx[6] para entregar los datos solicitados, en este caso en particular se utiliza el archivo .py por ser archivos del lenguaje Python (lenguaje que se desarrolla el proyecto) la que se mostrará la información mediante una interfaz gráfica.


Error Python Module

Diego Parraguez
Vie 20/07/2018, 18:36
support@aimsun.com

Hi, I have the following error now with the API of AIMSUN, at the moment of wanting to execute the examples that are in the documentation of Aimsun Next API I get the following error when wanting to execute the python scripts:

Error Python: No module named _AAPI

I changed the copying and moved the file but I still get the error

I wait for your prompt answer, in advance thank you very much,
Diego Parraguez

Figura 8.4: Correo problema archivo AAPI.py

```
"Tengo el siguiente error ahora con la API de AIMSUN, al momento de ↩
    querer ejecutar los ejemplos que estan en la documentacion de ↩
    Aimsun Next API, aparece el siguiente error cuando quiero ejecutar↩
     los scripts de Python:

Error Python: ningun modulo llamado _AAPI"
```

En la figura 8.4 muestra el problema descrito previamente, por lo que el soporte de AIMSUN contesta con lo siguiente:

[5] Archivos del lenguaje Python
[6] Archivos del lenguaje C++

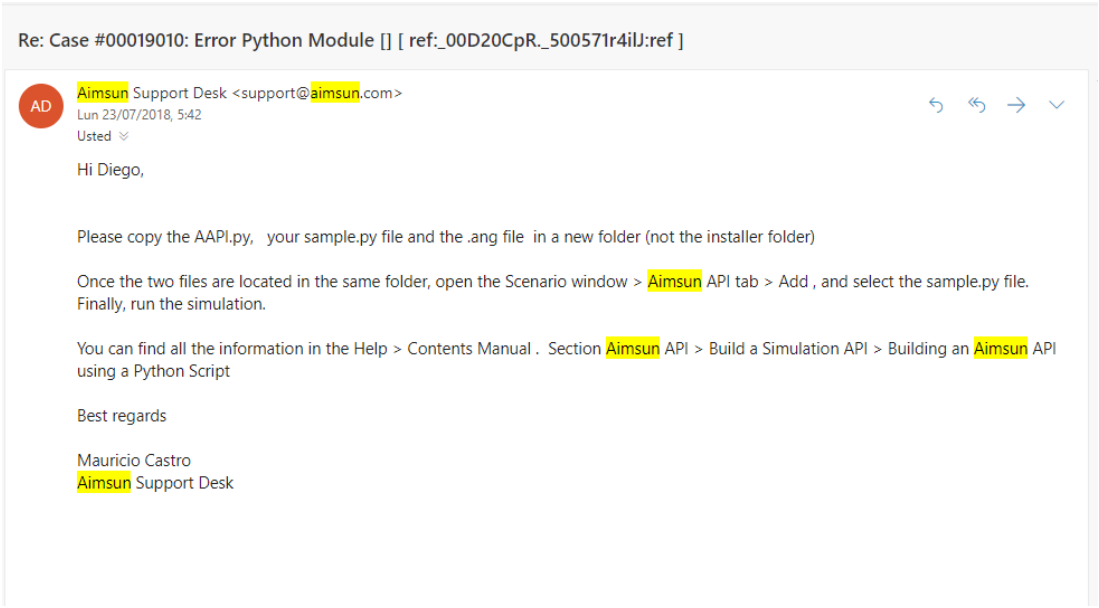


Figura 8.5: Correo solución archivo AAPI.py

```
"Hola Diego,

Copie AAPI.py, su archivo sample.py y el archivo .ang en una nueva carpeta (no en la carpeta del instalador)
Una vez que los dos archivos se encuentren en la misma carpeta, abra la ventana Escenario> pestaña API Aimsun> Agregar y seleccione el archivo sample.py.
Finalmente, ejecuta la simulación.
Puede encontrar toda la información en el Manual de Ayuda> Contenido. Sección API de Aimsun> Crear una API de simulación> Crear una API de Aimsun utilizando una secuencia de comandos de Python
Atentamente"
```

8.1.3. Configuración Previa

Ya solucionado los problemas encontrados en 8.1.2, se realizan las configuraciones correspondientes mostrados y detallados a continuación:

Para utilizar el software del proyecto hay que realizar una configuración perteneciente al archivo AAPI.py. Éste archivo debe ser incorporado a la simulación para que de esa manera pueda ser utilizado en ella.

Figura 8.6: Archivo que contiene la simulación

- Se abre el archivo de la simulación "Concepción_3.ang" que se muestra en la figura 8.6, que posee la simulación en la que se trabajará (contiene la red de tránsito).
- Seguidamente, dentro de la interfaz de AIMSUN, nos vamos especificamente a la barra lateral derecha, como lo muestra la figura 8.7

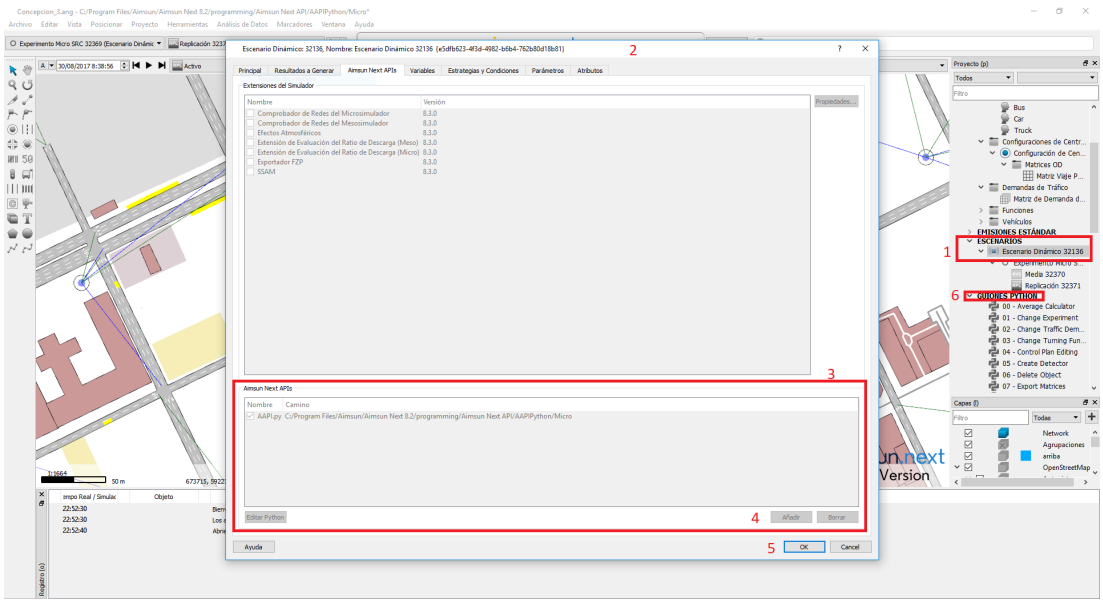


Figura 8.7: Configuración archivo AAPI.py

- Al presionar con el puntero del mouse sobre (1 - Escenario Dinámico) nos vamos a propiedades.
- Posteriormente nos abre la ventana (2) y nos vamos a la pestaña "AIMSUN NEXT APIs".
- Luego a ésto, en la parte inferior a la ventana (2) nos vamos a la sección (3) y presionamos el botón (4), como lo muestra la figura 8.7.

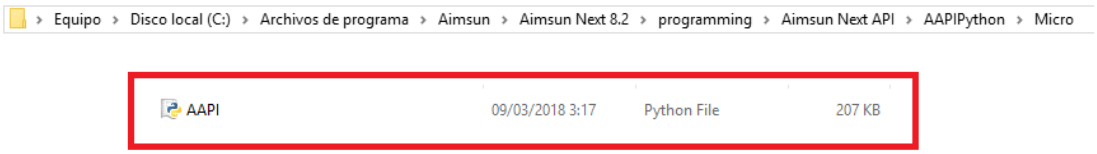


Figura 8.8: Ruta archivo AAPI.py

- Seguidamente se abrirá una ventana en la que debemos seleccionar el archivo AAPI.py, el cual, para ubicar dicho archivo debemos ingresar a la dirección como lo muestra la figura 8.8 en su parte superior.
- A continuación, aparecerá el archivo AAPI.py dentro de la sección (3) como se muestra en la figura 8.7, para luego presionar el botón (5) y así proveer de la información que entrega

AIMSUN gracias al archivo configurado.

- Ahora, para utilizar el archivo AAPI.py debemos crear un "Scripts o guión de Python" por lo que nos vamos nuevamente a la barra de herramientas que se encuentra a la derecha de la figura 8.7 y nos vamos al apartado (6), en donde, con el puntero sobre "GUIONES PYTHON" presionamos el botón secundario del mouse y seleccionamos la opción "Crear Guión Python".

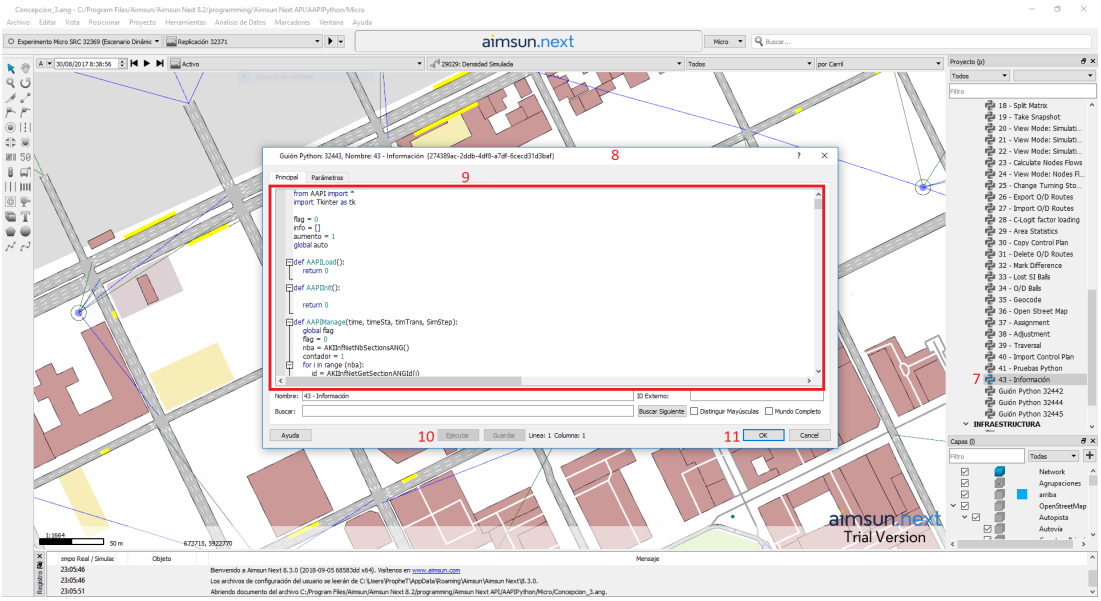


Figura 8.9: Ejecución de un script de Python

- Utilizamos la barra de desplazamiento de la barra de herramientas para descender por los guiones de python para encontrarnos con el guión creado, como se percibe en (7) en la figura 8.9.
- Se abrirá una nueva ventana (8) la que posee un editor de texto, que sirve para escribir el código del programa como se observa en (9)
- Una vez escrito el código en el apartado (9) se ejecuta el scripts presionando el botón (10)
- Para guardar el scripts dentro de la simulación, debe presionar el botón (11) como se muestra en la figura 8.9.

De ésta manera se configura el software previamente para su correcto uso, de no realizar

dichas configuraciones el software no entregará los datos descrito en los capítulos anteriores.

8.2. Código Fuente

```
from AAPI import *
import Tkinter as tk

flag = 0
info = []
aumento = 1
global auto

def AAPILoad():
return 0

def AAPIInit():

return 0

def AAPIManage(time, timeSta, timTrans, SimStep):
global flag
flag = 0
nba = AKIInfNetNbSectionsANG()
contador = 1
for i in range (nba):
id = AKIInfNetGetSectionANGId(i)
nb = AKIVehStateGetNbVehiclesSection(id,True)
for j in range(nb):
infVeh = AKIVehStateGetVehicleInfSection(id,j)
astring = "ID: " + str(infVeh.idVeh)
```

```
editArea.insert(contador, "%s\n" %(astring))
contador+=1
nbj = AKIInfNetNbJunctions()
for i in range(nbj):
id = AKIInfNetGetJunctionId(i)
nb = AKIVehStateGetNbVehiclesJunction(id)
for j in range(nb):
infVeh = AKIVehStateGetVehicleInfJunction(id,j)
astring = "ID: " + str(infVeh.idVeh)
editArea.insert(contador, "%s\n" %(astring))
contador+=1
return 0


def AAPIManage2(time, timeSta, timTrans, SimStep):
global flag
flag = 1
contador = 1
sectionType = model.getType( "GKSection" )
for types in model.getCatalog().getUsedSubTypesFromType( sectionType ):
for s in types.itervalues():
if (s.getId() != "null" and s.getName() != ""):
_valor = "Calle Id: %d - Nombre: %s" %(s.getId(), s.getName())
editArea.insert(contador, "%s\n" %(_valor))
contador+=1
return 0

def salir(): win.destroy()


def imprimir_en_label():
label1.after(100, imprimir_en_label) # Llamada recursiva con .after
```

```
global auto

ind = editArea.curselection()
global aumento
if editArea.curselection() != ():
sel = editArea.get(ind)
auto = sel
if  sel != ():
if aumento == 1:
aumento+=1
editArea.bind('<FocusOut>', lambda e: editArea.selection_clear(0, tk.END))
llamar_ventana(aumento, auto)
#mitexto.set(sel)




def llamar_ventana(aumento, auto):

if aumento == 2:
aumento = 3
ven = tk.Tk()
ven.geometry('800x600')
ven.title("INFORMACION - SWLoop")
def destruir():
global aumento
aumento = 1
ven.destroy()

def AAPIMana():
global auto
global info
```

```
nbaaux = AKIInfNetNbSectionsANG()
conta = 1
contador2 = 0
nbjaux = AKIInfNetNbJunctions()
for i in range(nbjaux):
idaux = AKIInfNetGetJunctionId(i)
nbaux = AKIVehStateGetNbVehiclesJunction(idaux)
for j in range(nbaux):
infVehaux = AKIVehStateGetVehicleInfJunction(idaux,j)
astringaux = "ID: " + str(infVehaux.idVeh)
if auto.split() == astringaux.split():
astringaux = "ID: " + str(infVehaux.idVeh)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idVeh)
contador2+=1
astringaux = "Tipo: " + str(infVehaux.type)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.type)
contador2+=1
astringaux = "ID Calle: " + str(infVehaux.idJunction)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idJunction)
contador2+=1
astringaux = "Numero Carril: " + str(infVehaux.numberLane)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.numberLane)
contador2+=1
astringaux = "Desde donde viene: " + str(infVehaux.idSectionTo)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idSectionTo)
contador2+=1
astringaux = "Hasta donde va: " + str(infVehaux.idSectionFrom)
```

```
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idSectionFrom)
contador2+=1
astringaux = "Velocidad: " + str(infVehaux.CurrentSpeed)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.CurrentSpeed)
e.insert(0, str(infVehaux.CurrentSpeed))
contador2+=1
astringaux = "Tiempo Detenido: " + str(infVehaux.CurrentStopTime) + " ↩
    segundos"
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.CurrentStopTime)
contador2+=1
astringaux = "Distancia Total: " + str(infVehaux.TotalDistance)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.TotalDistance)
contador2+=1
for i in range (nbaaux):
idaux = AKIInfNetGetSectionANGId(i)
nbaux = AKIVehStateGetNbVehiclesSection(idaux,True)
for j in range(nbaux):
infVehaux = AKIVehStateGetVehicleInfSection(idaux,j)
astringaux = "ID: " + str(infVehaux.idVeh)
if auto.split() == astringaux.split():
astringaux = "ID: " + str(infVehaux.idVeh)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idVeh)
contador2+=1
astringaux = "Tipo: " + str(infVehaux.type)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.type)
contador2+=1
```

```
astringaux = "ID Calle: " + str(infVehaux.idSection)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idJunction)
contador2+=1
astringaux = "Numero Carril: " + str(infVehaux.numberLane)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.numberLane)
contador2+=1
astringaux = "Desde donde viene: " + str(infVehaux.idSectionTo)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idSectionTo)
contador2+=1
astringaux = "Hasta donde va: " + str(infVehaux.idSectionFrom)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.idSectionFrom)
contador2+=1
astringaux = "Velocidad: " + str(infVehaux.PreviousSpeed)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.CurrentSpeed)
e.insert(0, str(infVehaux.CurrentSpeed))
contador2+=1
astringaux = "Tiempo Detenido: " + str(infVehaux.CurrentStopTime) + " ↩
    segundos"
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.CurrentStopTime)
contador2+=1
astringaux = "Distancia Total: " + str(infVehaux.TotalDistance)
area.insert(contador2,"%s" %(astringaux))
info.insert(contador2, infVehaux.TotalDistance)
contador2+=1
else:
sectionType = model.getType( "GKSection" )
```

```
for types in model.getCatalog().getUsedSubTypesFromType( sectionType ):
for s in types.itervalues():
if (s.getId() != "null" and s.getName() != ""):
_valor = "Calle Id: %d  -  Nombre: %s" %(s.getId(), s.getName())
if (auto.split() == _valor.split()):
if contador2 <12:
A2KSectionInf = AKIInfNetGetSectionANGInf(s.getId())
StructAkiEstadSection = AKIEstGetGlobalStatisticsSection( s.getId(), 0)
astringaux = "ID: " + str(A2KSectionInf.id)
area.insert(contador2, "%s\n" %(astringaux))
info.insert(contador2, A2KSectionInf.id)
contador2+=1
astringaux = "Velocidad Limite: " + str(A2KSectionInf.speedLimit)
area.insert(contador2, "%s\n" %(astringaux))
info.insert(contador2, A2KSectionInf.speedLimit)
e.insert(0, str(A2KSectionInf.speedLimit))
contador2+=1
astringaux = "Capacidad: " + str(A2KSectionInf.capacity)
area.insert(contador2, "%s\n" %(astringaux))
info.insert(contador2, A2KSectionInf.capacity)
contador2+=1
astringaux = "Tiempo Reaccion: " + str(A2KSectionInf.↩
   reactionTimeAtTrafficLightVariation)
area.insert(contador2, "%s\n" %(astringaux))
info.insert(contador2, A2KSectionInf.reactionTimeAtTrafficLightVariation)
contador2+=1
if StructAkiEstadSection.report == 0:
area.insert(contador2, "\n")
contador2+=1
area.insert(contador2, "———————— DATOS ESTADISTICOS ————————————\n")
contador2+=1
astringaux = "ID CALLE: " + str(StructAkiEstadSection.Id)
```

```
area.insert(contador2, "%s\n" %(astringaux))
contador2+=1
astringaux = "Promedio Tiempo de Viaje: " + str(StructAkiEstadSection.TTa)
area.insert(contador2, "%s\n" %(astringaux))
contador2+=1
astringaux = "Flujo (veh/hr): " + str(StructAkiEstadSection.Flow)
area.insert(contador2, "%s\n" %(astringaux))
contador2+=1
astringaux = "Promedio Velocidad (km/hr): " + str(StructAkiEstadSection.Sa↩
    )
area.insert(contador2, "%s\n" %(astringaux))
contador2+=1
astringaux = "Densidad (veh/km): " + str(StructAkiEstadSection.Density)
area.insert(contador2, "%s\n" %(astringaux))
contador2+=1
astringaux = "Promedio Detenciones (num/veh): " + str(↩
    StructAkiEstadSection.NumStops)
area.insert(contador2, "%s\n" %(astringaux))
contador2+=1
astringaux = "Tiempo Total de Viaje (en seg): " + str(↩
    StructAkiEstadSection.TotalTravelTime)
area.insert(contador2, "%s\n" %(astringaux))

def cambio_variable():
global flag
AKIPrintString(str(flag))
if flag <= 0:
veloc = e.get()
ver = AKIVehSetAsTracked(int(info[0]))
if ver == 0:
data = AKIVehTrackedModifySpeed(int(info[0]), float(veloc))
if data == 0:
```

```
e.delete(0, tk.END)
e.insert(0, "CAMBIO REALIZADO")
InfVeh = AKIVehTrackedGetInf(int(info[0]))
aux = 0
area.delete(0, tk.END)
stringaux = "ID: " + str(InfVeh.idVeh)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Tipo: " + str(InfVeh.type)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "ID Calle: " + str(InfVeh.idSection)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Numero Carril: " + str(InfVeh.numberLane)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Desde donde viene: " + str(InfVeh.idSectionTo)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Hasta donde va: " + str(InfVeh.idSectionFrom)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Velocidad: " + str(InfVeh.CurrentSpeed)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Tiempo Detenido: " + str(InfVeh.CurrentStopTime) + " segundos↩
    "
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Distancia Total: " + str(InfVeh.TotalDistance)
area.insert(aux, "%s\n" %(stringaux))
```

```
AKIPrintString('OK AUTO')
else:
e.delete(0, tk.END)
e.insert(0, "ERROR VEHICULO: " + str(data))
else:
    e.delete(0, tk.END)
    e.insert(0, "ERROR: " + str(ver))
else:
if flag == 1:
aux = 0
veloc = e.get()
if AKIActionAddSpeedAction(int(info[0]), float(veloc), 0, 0) !='NULL':
report = intp()
behParams = AKIInfNetGetSectionBehaviouralParam(int(info[0]), report)
if report.value() == 0:
behParams.speedLimit = float(e.get())
AKIInfNetSetSectionBehaviouralParam(int(info[0]), behParams, True)
AKIPrintString("OK CALLE")
e.delete(0, tk.END)
e.insert(0, "CAMBIO REALIZADO")
area.delete(0, tk.END)
A2KSectionInf = AKIInfNetGetSectionANGInf(int(info[0]))
stringaux = "ID: " + str(A2KSectionInf.id)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Velocidad Limite: " + str(A2KSectionInf.speedLimit)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
stringaux = "Capacidad: " + str(A2KSectionInf.capacity)
area.insert(aux, "%s\n" %(stringaux))
aux+=1
```

```
stringaux = "Tiempo Reaccion: " + str(A2KSectionInf.↩
    reactionTimeAtTrafficLightVariation)
area.insert(aux, "%s\n" %(stringaux))
else :
AKIPrintString("Not OK")
else:
e.delete(0, tk.END)
e.insert(0, "ERROR CALLES")
else:
area.insert(0, "ERROR")
def llamada():
AAPIMana()
marco1 = tk.Frame(ven, borderwidth=1, relief="sunken")
barra = tk.Scrollbar(marco1)
area = tk.Listbox(marco1, width=125, height=30,
yscrollcommand=barra.set,
borderwidth=0, highlightthickness=0)
barra.config(command=area.yview)
barra.pack(side=tk.RIGHT, fill="y")
area.pack(side=tk.LEFT, fill="both", expand=True)

marco0=tk.Frame(ven)
marco0.pack(side="top")

marco3=tk.Frame(marco0)
boto = tk.Button(marco3, text="      SALIR      ", command=destruir)
boto.pack()
marco3.pack(side="right")

marco1.pack(side="top")
marco2=tk.Frame(ven)
marco2.pack()
```

```
e = tk.Entry(marco2)
e.pack(side=tk.LEFT)
e.focus_set()
cambiar=tk.Button(marco2, text="CAMBIAR", command=cambio_variable)
cambiar.pack(side=tk.RIGHT)
llamada()
ven.mainloop()
else:
return 0



def hacer_click():
editArea.delete(0, tk.END)
AAPIManage(360.0, 25260.0, 300.0, 1.0)

def semaforos():
editArea.delete(0, tk.END)
AAPIManage2(360.0, 25260.0, 300.0, 1.0)
win = tk.Tk()
win.geometry('800x600')
win.title("Pantalla Principal - SWLoop")

frame1 = tk.Frame(win, borderwidth=1, relief="sunken")
scrollbar = tk.Scrollbar(frame1)
editArea = tk.Listbox(frame1, width=125, height=30,
yscrollcommand=scrollbar.set,
borderwidth=0, highlightthickness=0)
scrollbar.config(command=editArea.yview)
scrollbar.pack(side=tk.RIGHT, fill="y")
editArea.pack(side=tk.LEFT, fill="both", expand=True)

frame0=tk.Frame(win)
```

```
frame0.pack(side="top")
frame2=tk.Frame(frame0)
boton = tk.Button(frame2, text="VEHICULOS", command=hacer_click)
boton.pack()
frame2.pack(side="left")

frame3=tk.Frame(frame0)
boton2 = tk.Button(frame3, text="      SALIR      ", command=salir)
boton2.pack()
frame3.pack(side="right")

frame4=tk.Frame(frame0)
boton3 = tk.Button(frame4, text="CALLES", command=semaforos)
boton3.pack()
frame4.pack(side="right")



frame1.pack(side="top")

mitexto= tk.StringVar()
label1= tk.Label(win, textvar=mitexto)
label1.pack()

imprimir_en_label()
win.mainloop()
```

Referencias

[1] aimsun-next (2018). About Aimsun Next. Recuperado de https://www.aimsun.com/aimsun-next/.

[2] Unidad Operativa de Control de Tránsito (2018). Sistemas complementarios y de apoyo a la gestión de tránsito. Recuperado de http://www.uoct.cl/sistemas-complementarios-y-de-apoyo-a-la-gestion-de-transito/.

[3] Departamento Ingeniería Civil y Ambiental (2018). Misión y Visión del Departamento. Recuperado de http://www.dic.ubiobio.cl/#Departamento.

[4] Software-In-the-Loop (SIL), Desarrollo y comprobación de sistemas embebidos en tiempo real, en la industria automotriz y espacial, Marzo 2015, Recuerado de http://www.eluniversalqueretaro.mx/content/desarrollo-y-comprobacion-de-sistemas-embebidos-en-tiempo-real-en-la-industria-automotri

[5] Qt(Biblioteca) (2018), Propósitos y características. Recuperado de https://es.wikipedia.org/wiki/Qt_(biblioteca)

[6] GTK (2018), Definición. Recuperado de https://es.wikipedia.org/wiki/GTK

[7] GTK+ (2018), Definición. Recuperado de https://es.wikipedia.org/wiki/GTK%2B

[8] Multiplataforma (2018). Definición Multiplataforma (informática). Recuperado de http://www.alegsa.com.ar/Dic/multiplataforma.php

[9] Dongle (Mayo 2018). Recuperado de https://es.wikipedia.org/wiki/Dongle.

[10] Aimsun 8 API Manual, Mayo 2013. 2006 - 2013 TSS-Transport Simulation Systems.

[11] Aimsun Next 8.2 Help, Users Manual, Aimsun Next API Incidents (2018).

[12] Aimsun Next 8.2 Help, Users Manual, Aimsun Next API Control Meterings (2018).

[13] Aimsun Next 8.2 Help, Users Manual, Aimsun Next API Vehicles Information (2018).

[14] Aimsun Next 8.2 Help, Users Manual, Aimsun Next API Management Actions (2018).