# Centro de vuelos Internacionales

*Manual Técnico*


**Integrantes**:

Luis Estuardo Azurdia Cárcamo 201408606
Celeste Marilú Duarte Amaya 201318666
Kevin Alberto Morán Orellana 201403762

# Objetivos

- Este manual técnico va enfocado a que los usuarios de la aplicación Centro de vuelos Internacionales puedan conocer su estructura de código y así poder tener control total y poder realizar cambios y mejoras.

- Dar a conocer las herramientas utilizadas y definir la lógica utilizada para que los usuarios que requieran comprender la parte técnica o realizar cambios puedan hacerlo sin ningún inconveniente.

# Librerías de Simio utilizadas

- SimioAPI

# Descripción de Clases

Form: Contiene los métodos para la conexión con simio y la interacción con el usuario

- Métodos y Procedimientos utilizados

  Form_Load(): Carga un modelo predefinido a la aplicación para realizar sobre él las cargas correspondientes.

  guardarModeloSalida(): Guarda el proyecto en la ruta especificada.

  seekForName(string nombre): Devuelve la entidad que tenga el nombre indicado.

  button1_Click(): Realiza la carga csv de aeropuertos a la aplicación.

  button2_Click(): Realiza la carga csv de vuelos a la aplicación.

  button3_Click(): Realiza una llamada para crear el archivo .spfx utilizando los datos cargados a la aplicación.

CSView: Muestra una vista a los archivos .csv cargados a la aplicación

ReadCsv: Contiene los métodos para la lectura de los archivos .csv

- Métodos y Procedimientos utilizados

  ReadCsv(string path): Llama al método para abrir el archivo en la ruta indicada.

  open(string path): Lee el archivo csv de la ruta indicada.

  getline(): Devuelve un arreglo de strings con los datos de cada celda de la siguiente línea a leer de el archivo csv cargado.

Main: Clase que controla la lógica para crear el archivo .spfx según los datos leídos previamente.

- Métodos y Procedimientos utilizados

  start(string path_air, string path_rut): Ejecuta todas las llamadas a los métodos de creación de los source, objeto sink, rutas y escenarios y guardar el modelo creado.

  createRuta(): Llama al creador de rutas enviándole el inicio y el fin de la ruta por cada ruta indicada.

  createAirport(): Llama al creador de combiners, asignándole atributos parent y member

SimioFile: Clase que crea todos los objetos del modelo de simio.

- Métodos y Procedimientos utilizados

  SimioFile(): Carga el modelo predefinido a la aplicación.

  saveFile(): Guarda el modelo en la carpeta bin/Debug de la aplicación.

  createObject(string tipo, string nombre, int x, int y, int z): crea el objeto del tipo indicado con el nombre y en las posiciones indicadas.

  createServer(string nombre, int x, int y, int z): crea un servidor con el nombre y en las posiciones indicadas.

  createCombiner(string nombre, int x, int y, int z): crea un combiner con el nombre y en las posiciones indicadas y con los valores de los procesos creados en simio para escribir la bitacora de clientes y el total de aviones.

  addConnector(INodeObject airplane, INodeObject parent, object p): crea un link de tipo Conector entre los INodeObject indicados.

  addPath(INodeObject inicio, INodeObject fin): crea un link de tipo path entre el objeto de inicio y el objeto de fin.

  createSource(string name, int x, int y, int z): crea un objeto tipo source con el nombre indicado y en dichas posiciones.

  createSink(string name, int x, int y, int z): crea un objeto tipo sink con el nombre indicado y en dichas posiciones.

  createTransferNode(string name, int x, int y, int z): crea un objeto tipo TransferNode con el nombre indicado y en dichas posiciones.

getNodeOutput(IIntelligentObject objeto): Devuelve el nodo de salida del objeto indicado.

getInput(IIntelligentObject objeto): Devuelve el nodo de entrada del objeto indicado.

getParentInput(IIntelligentObject objeto): Devuelve el nodo padre de un objeto combiner que se le indique.

getMemberInput(IIntelligentObject objeto): Devuelve el nodo miembro de un objeto combiner que se le indique.

seekForName(string name): Devuelve el objeto que tenga el nombre indicado.

getObjectList(): Devuelve todos los objetos existentes en el modelo.

addFailure(IIntelligenteObject objeto, string falla): Asigna la falla indicada al objeto indicado.

createProperty(): Crea el Referenced Property que utilizará el experimento.

addProperty(IIntelligenteObject objecto): Asigna el Referenced Property al objeto indicado.

createExperiment(string name): Crea el experimento y setea sus propiedades a las deseadas para simular un día de trabajo.

addScenarios(IExperiment experimento): Crea los escenarios requeridos en el experimento.

addTiempos(IExperiment experimento): Asigna los tiempos solicitados a los controles de los escenarios del experimento.

addResponses(IExperiment experimento, string expresionResponse): Crea los responses en el experimento que se le indique con la expresión indicada.