# Introducción

Dado el problema de crear un sistema o programa que cree conexiones entre localidades de las distintas provincias de Argentina, se pide implementar una representación basada en Grafos en el cual los vértices serían las distintas localidades y las aristas las conexiones telefónicas entre ellas.

Además, se pide implementar Árbol Generador Mínimo (AGM) el cual creara un Grafo con la mínima cantidad de aristas, que a su vez tiene el menor peso posible, dando así el costo mínimo por conectar todos los vértices. Por otro lado, se nos otorga la librería JMapViewer como una posibilidad para mostrar en pantalla toda la información mencionada anteriormente.

# Implementación

Para comenzar se crearon las clases de negocio con sus respectivos test unitarios, para tratar de minimizar la cantidad de bugs. Teniendo en cuenta que se busca modelar un conjunto de conexiones con localidades, se crearon las clases que se incluyen en dicho apartado.

## Clase Localidad

La clase “Localidad” representa una localidad de una provincia de Argentina, para ello necesitará tener el nombre de la localidad, el nombre de la provincia, la latitud y la longitud geográfica. Además, tendrá las validaciones para que no se pueda ingresar localidades que no entren dentro del mapa.

## Clase CableDeRed

Luego tenemos la clase “CableDeRed” la cual representará una conexión/arista entre una localidad y otra, para lograrlo recibirá las dos localidades que estarán unidas, y además el costo que tiene hacer esa conexión. La misma, almacenara los dos vértices, el costo, y la distancia en km entre ambos puntos.

## Clase GrafoLocalidad

La clase GrafoLocalidad (GL) usará como vértice la Clase Localidad y como arista la Clase CableDeRed. En GL habrá métodos generales de un Grafo, además de que para crearlo se le deberá pasar los costos fijos y porcentuales que se desea usar: costo por KM, porcentaje añadido se una conexión tiene más de 300KM y costo añadido si dos localidades son de distintas provincias.

Dicha clase hace usó de la Clase auxiliar llamada Utilidades la cual contiene funciones para calcular el costo y la distancia (algoritmo de Haversine) de una localidad a otra.

## Clase GrafoLocalidadCompleto

Esta clase se encarga de crear un “GrafoLocalidad” pero con la particularidad de que siempre se encuentra completo a partir de los vértices que se agregan o con los que se inicializó la variable. Asimismo, como el grafo siempre está completo, nunca se van a poder ingresar o quitar aristas, ya que si no seguiría siendo un grafo completo.

## Clase ArbolGeneradorMinimo

Para crear el algoritmo de Árbol Generador Mínimo (AGM) nos pareció interesante establecer la clase “ArbolGeneradorMinimo”, en el que para generar dicho árbol utilizamos el algoritmo de Prim dado que nos devuelve un Grafo que tiene las mínimas aristas posibles y además con el peso mínimo. El algoritmo incrementa continuamente el tamaño de un árbol, comenzando por un vértice inicial al que se le van agregando sucesivamente vértices cuya distancia a los anteriores es mínima

Para poder crear el AGM con Prim primeros se requería que el Grafo pasado como parámetro sea conexo y para verificar esto debíamos implementar una función que indique que es conexo por eso creamos la Clase BFS, la cual contiene un método que devuelve si un Grafo es conexo o no.

Finalmente, establecimos que al llamar al método “Prim” de la clase y pasándole un GrafoLocalidad como argumento, esta nos va a devolver el AGM de dicho grafo.

## Clase Sistema

La Clase Sistema, se ocupa de manejar todo lo relacionado con agregar vértices, crear agms, carga y guardado de datos. Además de brindar todos los datos necesarios para que puedan ser mostrados como mejor convenga. Todos sus métodos son static debido a que nos pareció una buena decisión de diseño.

## Clase BaseDeDatos

Siguiendo con la clase BaseDeDatos, esta es la encargada de cargar y guardar archivos dentro del Sistema, almacenando todos los AGM que han sido guardados y las localidades posibles para cargar. Para ello nos apoyamos del formato. JSON y la librería Gson para manejar dichos formatos.

## Clase MapaPerzonalizado

Esta clase extiende de JMapViewer y crea un mapa personalizado que contiene métodos para agregar y sacar tanto polígonos como marcadores, además de que limita al zoom que se le puede hacer al mismo.

## Clase VentanaPrincipal

Esta clase es la ventana principal de la aplicación y se encarga de mostrar el mapa en donde se va a poder observar el árbol generador mínimo con todas sus localidades y cables de telefónicos.

A partir de esta ventana se pueden ingresar los valores en el mapa de las localidades ya predefinidas en el sistema o localidades personalizadas en las que se deben ingresar los datos de la latitud, altitud, provincia y localidad. El sistema no va a dejar que se ingrese localidades personalizadas sin alguno de esos campos o con algunos de esos campos con tipo de datos erróneos. Con estas localidades cargadas y apretando el botón generar AGM podemos generar el árbol generador mínimo con las localidades que se encuentra en el mapa de la pantalla.

Asimismo, la interfaz al presionar el botón de “Crear AGM” nos deja cambiar las aristas de un origen destino a otro destino que genera un árbol generador, pero no mínimo.

Por otro lado, en la ventana principal tenemos el botón borrar localidades que borra todas las localidades del mapa para así volver a empezar a agregar vértices y volver a poder generar el árbol generador mínimo.

Para terminar, tenemos el botón guardar que nos da guardar nuestro conjunto de localidades y tenemos el botón cargar para cargar nuestros conjuntos de localidades. Dichas localidades se guardan en un directorio predeterminado y también se cargan desde este mismo directorio.

## Clase VentanaOpciones

Esta ventana va a tener las opciones de los valores disponibles para asignarle a los diferentes costos, los cuales son costo por kilómetro que nos indica cuanto sale el cable de red por kilómetro, el porcentaje si supera los 300 km (el porcentaje tiene que estar escrito de 0 a 1) y por último el costo fijo de sí las localidades son de diferentes provincias.

Si desea guardar los valores para que el próximo AGM los tenga en cuenta se presione guardar en caso contrario si no desea esto último apriete volver y los valores con los que se crea el AGM van a ser los últimos guardado.

## Clase VentanaEstado

La Clase VentanaEstado crea una ventana paralela la cual funciona para poder intercambiar una arista por otra, siempre teniendo en cuenta que siga siendo AGM.

## Clase Controlador

Esta clase es la encargada de controlar que ventanas van abrirse y cerrarse además de que les brinda a las ventanas la información que están solicitan pero que se encuentran en el sistema.

# PROBLEMAS

En este apartado vamos a enumerar las problemáticas y soluciones que tuvimos a lo largo del desarrollo del proyecto.

* Para el uso de archivos .JSON fue requerido parsear de manera “manual” el grafo para guardarlo/cargado, además de la cantidad de información sobre .JSON lo cual dificulto un poco esa parte.
* Tuvimos que re factorizar varias veces el algoritmo de Prim para que su complejidad esté en O (m.n) (m = cantidad de aristas, n = cantidad de vértices), y O(n^2) si el grafo es muy denso.
* Para delimitar el uso de localidades solo de Argentina hubo que acotar los valores de longitud y latitud a un rango aproximado, con lo cual puede ser que no haya mucha precisión al colocar algunas localidades.
* Los valores de longitud y latitud de algunas localidades no dan de forma exacta en la zona donde está esa localidad, sino que de forma aproximada.
* En la BaseDeDatos hay muchos objetos cargados en memoria dinámica.
* La BaseDeDatos no está desacoplada del código de negocio
* Para la gestión de ventanas, para evitar agregar un método get por ventana agregada se usó un static EnumMap<Ventana, JFrame> ventanas, con lo cual solo hay que añadir la Constante en el enum Ventana y añadir el frame a ventanas, esto evita la creación de muchos getters en el futuro.
* No pudimos validar la entrada de una localidad personalizada (en la cual el usuario ingresa latitud y longitud) respecto de si esa localidad se encuentra en los límites de la provincia ingresada (si se sale o si está dentro).

# Conclusión

En síntesis, pudimos aplicar muchos conceptos vistos en clases tales como Grafo, BFS, AGM, uso de Tests, uso de librerías como Gson y JMapViewer dado que el proyecto resuelve la geolocalización para gestión de redes en distintos puntos de un país y esto nos hizo ver diferentes conceptos aplicados a un entorno real aproximado, entendiendo la importancia del uso de Tests y herramientas como GitLab con el uso de ramas.