

INFORME

TRABAJO PRÁCTICO 2 - PROGRAMACIÓN III

CLUSTERING HUMANO



Universidad
Nacional de
General
Sarmiento



Integrantes del grupo:

Abel Aquino
Lautaro Moreno
Karin Pellegrini

Comisión : 01

Profesores:

Patricia Bagnes
Ignacio Sotelo
Gabriel Carrillo

Introducción

Dentro del presente informe, se detalla el programa realizado por los alumnos expuestos anteriormente, para poder implementar una aplicación para identificar grupos de personas en base a datos, definidos como el interés por un determinado tema, representados por números del 1 al 5 (siendo 5 el máximo interés por ese tema, y 1 como el más bajo). Los 2 grupos resultantes de la división se podrán visualizar en forma de puntos en un mapa, conectados por líneas hasta conformar el grupo. Estos grupos estarán compuestos por personas con un índice de similitud cercano, es decir que son personas con intereses parecidos.

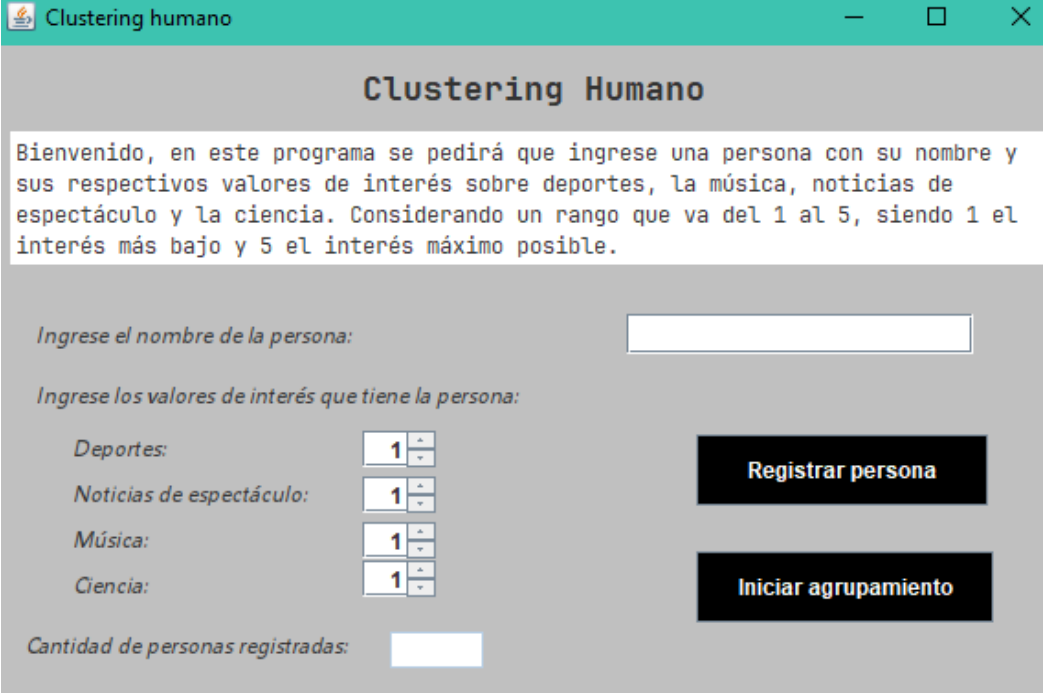
Los intereses están divididos en 4 temas: Interés por la musica - interés por las noticias de espectáculo - interés por los deportes - interés por la ciencia

La presentación visual está dividida siguiendo la arquitectura MVP vista en clase. Usamos la librería de WindowBuilder para poder diseñar las interfaces llamadas "pantallaCargarGrupos" y "pantallaIngresarPersonas" en el plazo de 4 semanas desde que se anunció.

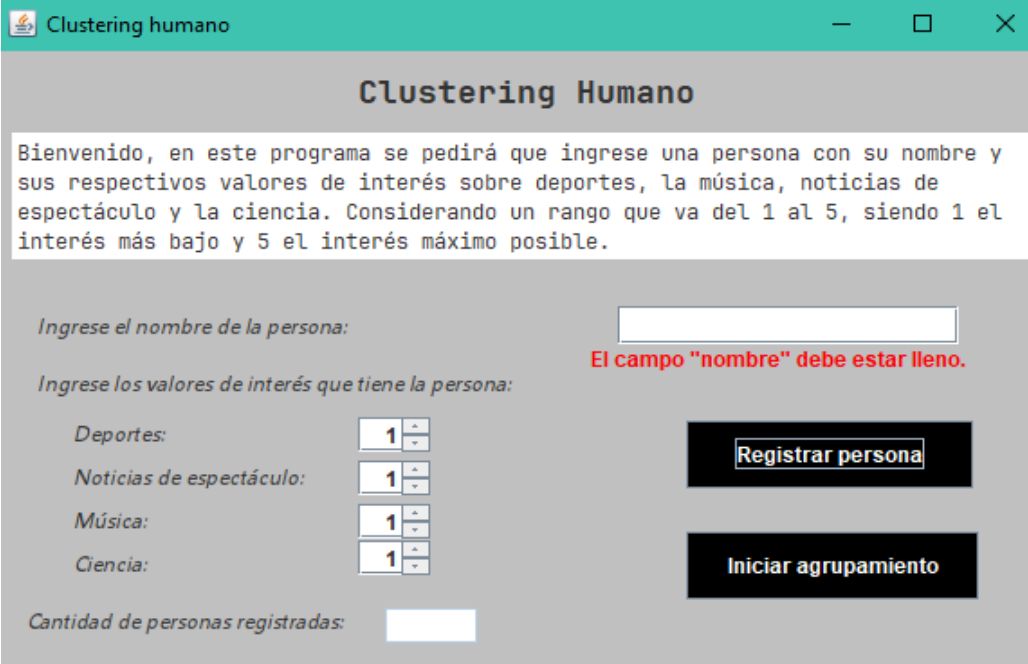
A continuación, vamos a mostrar las clases de las interfaces, sus casos de uso divididos en secciones, y luego mostraremos las clases implementadas para el correcto funcionamiento interno de la aplicación.

Pantalla ingresar personas

Dentro de esta interfaz se le pide al usuario registrar un mínimo de dos personas para poder hacer el Clustering Humano. Se le pide que ingrese el nombre de una persona y sus cuatro campos donde se seleccionan valores del 1 al 5 según el interés respectivo.



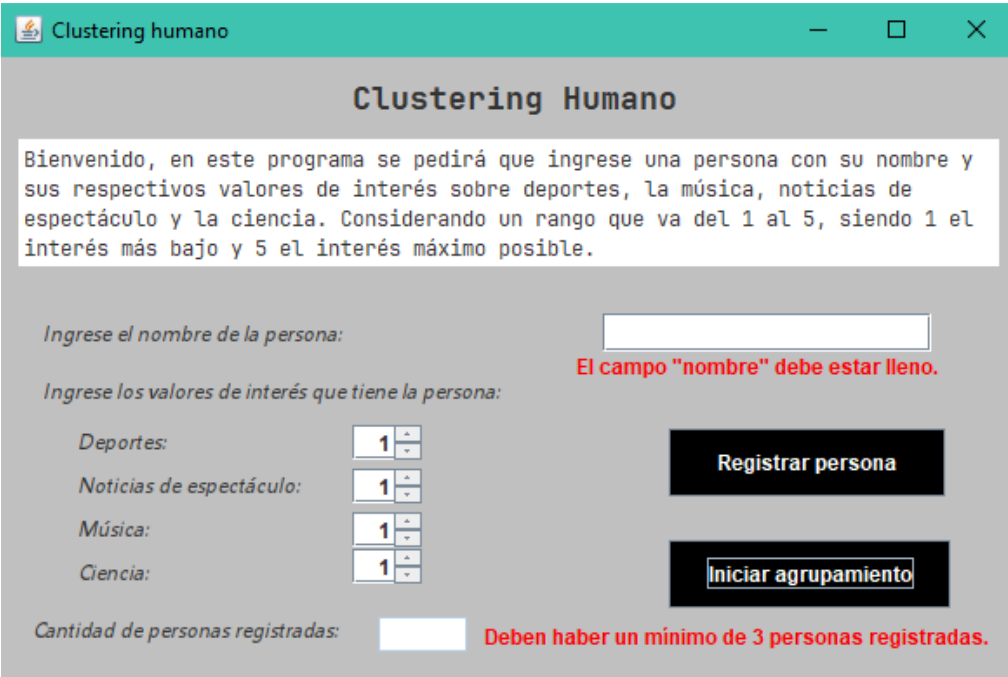
El nombre de la persona debe ser ingresado de forma obligatoria ya que se necesita identificar a la persona de alguna forma. Caso contrario, saldrá un mensaje que le indique al usuario ingresar dicho dato.



The screenshot shows a window titled "Clustering humano" with a teal header. The main content area has a title "Clustering Humano" and a welcome message: "Bienvenido, en este programa se pedirá que ingrese una persona con su nombre y sus respectivos valores de interés sobre deportes, la música, noticias de espectáculo y la ciencia. Considerando un rango que va del 1 al 5, siendo 1 el interés más bajo y 5 el interés máximo posible." Below this, there are input fields for "Ingrese el nombre de la persona:" (empty) and "Ingrese los valores de interés que tiene la persona:". The interest values are set to 1 for "Deportes:", "Noticias de espectáculo:", "Música:", and "Ciencia:". At the bottom left, there is a field for "Cantidad de personas registradas:". On the right, there are two buttons: "Registrar persona" and "Iniciar agrupamiento". A red error message "El campo 'nombre' debe estar lleno." is displayed next to the name input field.

Las acciones que el usuario puede hacer dentro de esta interfaz son las de registrar una persona (con los valores previamente descritos) e iniciar el agrupamiento (refiriéndonos a separar el conjunto de personas recibido en 2 grupos con intereses similares).

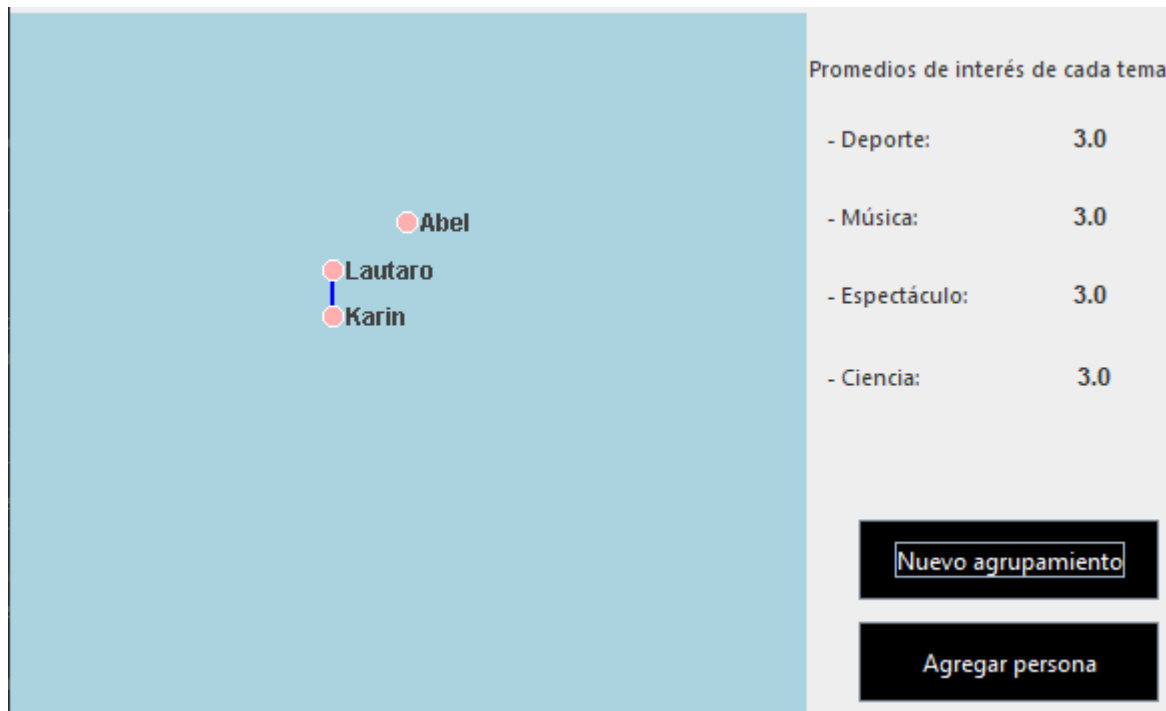
Para realizar el agrupamiento debe haber un mínimo de 3 personas registradas, ya que sino no tendría sentido dividir en dos grupos a una cantidad menor. En caso de que se haya registrado solo una persona o ninguna saldrá el siguiente mensaje indicando que debe registrarse 3 personas como mínimo.



This screenshot is identical to the previous one, showing the "Clustering humano" window with the same form and error message. However, a second red error message, "Deben haber un mínimo de 3 personas registradas.", is now visible at the bottom right of the window, next to the "Cantidad de personas registradas:" field.

Pantalla cargar grupos

Una vez que se cargan los datos, si el usuario toca el botón de “Iniciar agrupación” se abrirá esta interfaz, donde se visualizarán dos grupos de personas registradas como puntos conectados por líneas dentro de un mapa. Estos dos grupos son grafos, que a su vez son componentes conexos, con un único camino entre una persona y otra.

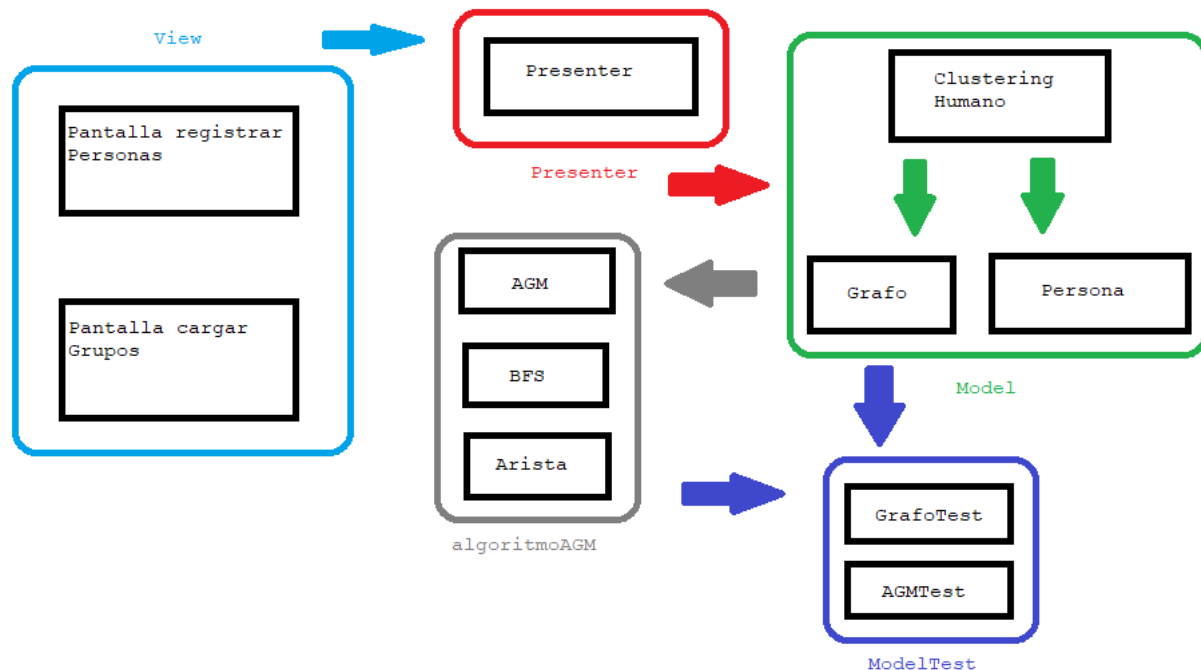


En este apartado podemos ver que tenemos dos secciones, la primera donde podemos ver gráficamente los dos grafos, los cuales representan los grupos que tienen un índice de similitud. En la segunda, en el apartado derecho de la pantalla, se observa los promedios, indicando el índice promedio por cada interés. En la parte inferior se observan dos botones, uno que tiene la función de realizar un nuevo agrupamiento y otro que agrega una o varias personas. La diferencia radica en que en el primero se inicializa el grafo nuevamente, por lo que el grafo anterior es reemplazado por el nuevo ingresado por el usuario.

Arquitectura técnica

El proyecto está construido sobre la arquitectura MVP, en la que se encuentra dividido en 3 módulos, el “view”, el “presenter”, y por último, el “model”. Además, cuenta con otro 2 módulos extra, el “algoritmoAGM”, donde se encuentran alojados los componentes responsables para convertir un grafo a un árbol generador mínimo, y el “modeloTest” donde se encuentran las clases, a nuestro parecer, requieren tener test de pruebas unitarias ya que necesitan estar lo mejor implementadas que se pueda. Cada módulo cuenta con sus clases y cada cual tiene su responsabilidad, logrando un código más amigable para futuras funcionalidades.

Nuestra arquitectura implementada queda dividida de la siguiente forma:



Módulo model

Dentro de este módulo se encuentran tres clases principales, dos que sirven de componentes y una que se encarga de resolver las cosas.

Estas clases son: Grafo - Persona - ClusteringHumano.

A continuación, se dará un breve vistazo a las clases involucradas y algunos métodos dentro de estas clases que se consideran importantes para el correcto funcionamiento del programa:

Clase Persona: Permite registrar una persona con su nombre y los intereses adecuados. Además, calcula el índice de similitud que tienen dos personas, restando sus intereses por un determinado tema.

```
Persona(nombre, iDeportes, iMusica, iEspectaculo,
iCiencia) // Registra la persona con su nombre y atributos de
interés.
```

```
calcularSimilitud(persona) // Calcula el índice de
similitud entre dos personas, restando sus valores de interés
y aplicando la función módulo para que siempre sea positivo el
valor
```

Clase Grafo: Esta estructura es representada de dos formas, como una matriz donde cada fila y columna representan un vértice, y una lista que contiene los vecinos de cada vértice asumiendo que cuando se le pide los vecinos de una posición específica se piden los vecinos del vértice ingresado. Dentro de la matriz,

cada celda representa el peso de una arista, el cual es mayor o igual a 0 si existe la arista. En el caso donde la arista no existe (como por ejemplo, cuando el valor de la fila y la columna son iguales) se le asigna un -1 en esa posición. Sus métodos más usados son:

```
Grafo(cantVertices) // Crea la matriz con la cantidad de
filas y columnas del mismo tamaño que el valor ingresado de
vértices, además de crear la lista donde se contendrán los
vecinos de los vértices. Por último, hace que no hayan aristas
en la matriz, para luego añadirle aristas en caso de pedir
eso.
```

```
agregarArista(verticeOrigen,verticeDestino,peso) // Añade
una arista en la matriz entre el vértice de origen y destino.
Además, los agrega a la lista de vecinos ya que si existe
arista entre dos vértices es porque ambos son vecinos.
```

```
eliminarArista(verticeOrigen,verticeDestino) //
Simplemente elimina la arista de la matriz de pesos y
actualiza la lista de vecinos de los vértices.
```

```
vecinosDelVertice(vertice) // Me devuelve una lista con
los vértices donde se forman aristas con el vértice pedido.
```

```
sacarAristaMasGrande() // Necesaria para cumplir con el
funcionamiento de la división de grupos luego de obtener un
AGM. Elimina la arista con mayor peso dentro de la matriz y
actualiza los vecinos de los vértices relacionados.
```

Clase ClusteringHumano: Esta clase es la encargada de usar las estructuras previamente descritas, permite realizar el agrupamiento de las personas y calcular su índice de similitud, crea un Árbol Generador Mínimo y le saca la arista de mayor peso para poder dividir los grupos. Los métodos más relevantes son:

```
ClusteringHumano() // Crea una lista con las personas que
se irán registrando por medio de la interfaz, además de crear
un grafo para poder manipularlo de tal forma que me queden dos
grupos.
```

```
registrarPersona(nombre,iDeportes,iMusica,iEspectaculo,IC
iencia) // Permite registrar una persona en la lista de
personas con sus datos.
```

```
construirGrafoCompleto() // Crea un grafo donde todos los
vértices tienen aristas entre ellos, y le asigna un peso a las
aristas, salvo cuando el vértice se relaciona consigo mismo
(lo cual no debe pasar, representado con un -1).
```

```
construirAGMConPrim() // Usando el algoritmo de Prim
(visto en clase), crea un árbol generador mínimo donde no hay
bucles entre los vértices.
```

```
sacarAristaDeMayorPeso() // Luego de obtener un AGM, este
```

método me permite ver cual es la arista de mayor peso que hay y luego la saca del grafo. De esta forma, el grafo queda conformado por dos componentes conexos.

Cabe resaltar que también hay métodos usados para calcular la similaridad promedio que tienen todas las personas en común según su tipo de interés.

Módulo presenter

En este módulo se encuentra sólo una clase, la cual se llama presenter, encargada de mandar información que solicita la interfaz 'PantallaRegistrarPersonas' para poder registrar personas e iniciar el agrupamiento con las personas registradas. Internamente, esta clase se compone del objeto capaz de manejar los componentes dentro del módulo model, es decir, el 'ClusteringHumano'.

Las funciones más importantes son:

```
registrarPersona(nombre,iDeportes,iMusica,iEspectaculo,iCiencia) // Se encarga de mandar la información al clustering para que registre la persona. En caso de que el tamaño de las personas registradas sea mayor a 2, se realizará el agrupamiento (grafo completo, tener un AGM y sacarle la arista más pesada) cada vez que se agregue una persona. De esta forma se irán actualizando los grupos cada vez que se inicie el agrupamiento.
```

```
iniciarAgrupamiento() // Una vez se realiza la llamada a esta función, el clustering construye un grafo completo con la cantidad de personas registradas hasta el momento (mayor a 2), construye el AGM con el algoritmo de Prim y saca la arista de mayor peso, obteniéndose de esta forma los dos grupos.
```

Módulo View

Dentro de este módulo están las interfaces que el usuario vé, una se encarga de registrar a las personas y la otra se encarga de cargar los datos ingresados anteriormente para visualizarlos en forma de 2 grupos. Cabe resaltar, que ambas interfaces se autogestionan, es decir que cada una guarda un objeto del mismo tipo que la clase de la otra interfaz. Esto está hecho así para poder facilitar la salida de una interfaz a otra.

El módulo se compone por las interfaces 'PantallaIngresarPersona' y 'PantallaCargarGrupos'. Ambas vistas anteriormente de forma visual y sus casos de uso.

Los métodos importantes de 'PantallaIngresarPersona' son los eventos que tienen los botones de "Registrar persona" y "Iniciar agrupamiento".

Cuando se toca el botón "Registrar persona", se registran los datos ingresados y se contabiliza la cantidad de personas que llevan registradas. En caso de que el nombre no esté ingresado saldrá un mensaje de advertencia debajo del campo nombre.

Cuando se presiona el botón “Iniciar agrupamiento”, se verifica que la cantidad de personas registradas sea mayor a 2. En caso afirmativo, usa al objeto ‘presenter’ para iniciar el agrupamiento y sale de esta interfaz, haciéndola invisible.

Los métodos importantes de ‘PantallaCargarGrupos’ son los eventos que tienen los botones de “Nuevo agrupamiento” y “Agregar persona”.

Cuando el usuario presiona el botón “Nuevo agrupamiento”, la pantalla donde se cargan los grupos se dejará de ver y se abrirá una nueva pantalla para registrar nuevos grupos de personas.

Cuando se presione “Agregar persona”, la interfaz donde se registraron los datos de las personas (que sigue estando sólo que no es visible para el usuario) será visible y podrá registrar a más personas, para luego volver a iniciar el agrupamiento con un grupo mucho más amplio de personas.

Módulo modelTest

Dentro de este paquete se encuentran los test de las clases ‘Grafo’, ‘AGM’ y ‘ClusteringHumano’. Es importante ya que nos permite saber si cada clase implementada funciona correctamente y poder estudiar su comportamiento con diferentes parámetros. No se hablará mucho de lo que contiene cada clase ya que son los mismos métodos de las clases antes mencionadas pero con distintos parámetros.

Módulo algoritmoAGM

En este último módulo se encuentran las clases que sirven para solucionar determinados problemas. Se divide en 3 clases: AGM - BFS - Arista

La clase Arista nos permite guardar la intersección entre dos vértices, guardando los dos vértices involucrados y un tercer parámetro el cual es el peso que tiene la arista. En caso de no existir una arista entre esos dos vertices el peso será -1.

En la clase AGM, se utiliza el método de Prim para poder armar un grafo conexo y sin bucles, lo que por definición es un Árbol Generador Mínimo. Recorre el grafo y evalúa si cada vértice está contenido dentro del conjunto de vértices del árbol, en caso de que no lo esté lo añade. Y termina devolviendo un nuevo grafo conformado por los vértices del grafo original pero con las aristas con peso mínimo. Usa el algoritmo BFS para saber si es conexo como última validación.

Por último, en la clase BFS hay métodos para saber si un grafo es conexo (es decir, si hay un camino para llegar a todos los vértices desde el origen).

Conclusión

Para concluir con este informe, podemos decir que el proyecto se pudo desarrollar cumpliendo con los objetivos pedidos por la consigna, a lo largo de las cuatro semanas el grupo fue dando ideas, algunas implementadas y otras descartadas. Entendemos que se logró resolver los desafíos como:

- La correcta implementación del algoritmo de Prim utilizado para crear el AGM.
- Crear dos grafos a partir de la eliminación de la arista de mayor peso y su correcta visualización en la interfaz utilizando la librería JMapView, la cual fue propuesta por los profesores.
- Respetar la arquitectura MVP.
- Mostrar el promedio de interés de cada tema.
- Permitir que se agreguen nuevas personas al grafo y recalcular los grupos.
- Realizar test unitarios para evaluar el comportamiento de las unidades de la aplicación.
- En el trabajo pasado tuvimos inconvenientes para gestionar las pantallas que aparecían y se cerraban ya que sólo las ocultábamos del usuario pero seguían cargadas en memoria. No obstante, en este trabajo solucionamos eso para evitar que cada pantalla de cargar grupos se mantenga “invisible” pero sin cerrarse, usando el método `.dispose()`.
- Implementamos lo visto en las lecturas y charlado en clases, más concretamente las reglas de cleanCode, no importar librerías sin usar ni tampoco dejar impresiones en consola, y usar nombres declarativos para las variables y funciones.

Así como tuvimos problemas que logramos solucionar, también surgieron contratiempos que nos impidieron implementar más funcionalidades. Dentro de estos pendientes encontramos, por mencionar algunos:

- Mostrar estadísticas de los dos grupos generados por separado.
- Permitir que se generen más de dos grupos, preguntándole al usuario cuántos grupos quiere generar. En este último caso, nos damos una idea de cómo debe ser implementado el algoritmo. Siendo que, según la cantidad de grupos que desee el usuario, se debe sacar la arista de mayor peso por cada grupo deseado. Es decir, en cada iteración se elimina la arista de mayor peso hasta lograr dividir el grafo en la cantidad de grupos que el usuario desea.
- Poder mostrar los valores de interés de cada persona cuando se hace un click sobre el vértice del grafo mostrado.