Proyecto:

Algoritmos de Clustering

ITESO

Universidad Jesuita
de Guadalajara

Programación orientada a objetos
Profesor: Juan Pablo González

Daniel Hernández Navarro
Eloir Corona Hermosillo

Contenido

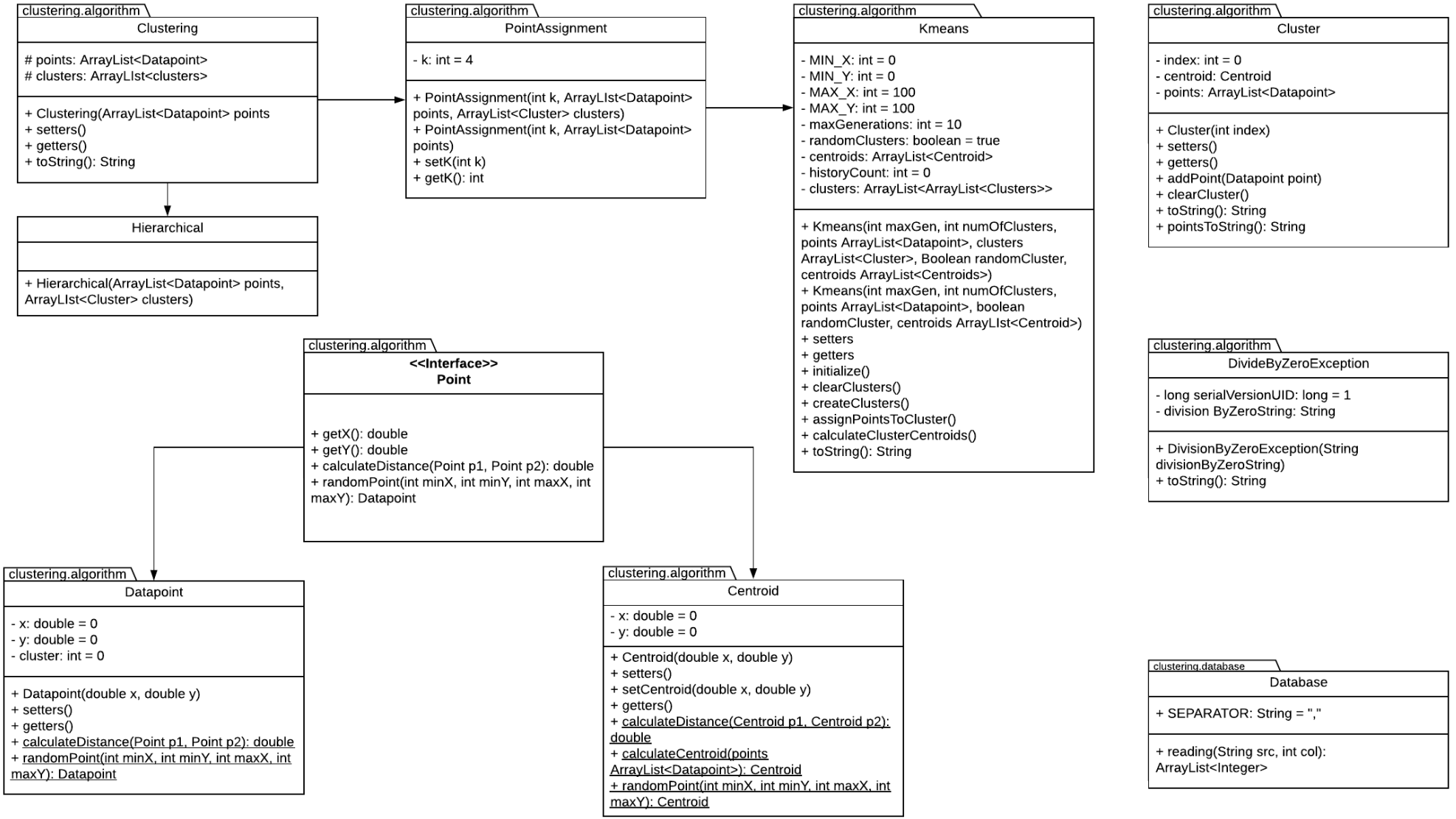
Introducción

Este proyecto busca que como alumnos

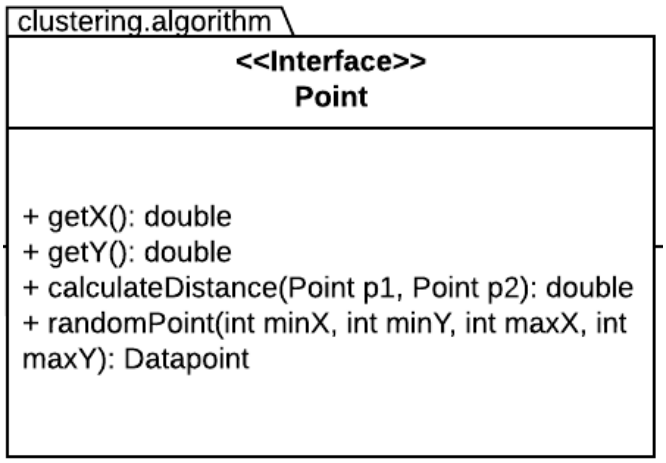
El clustering es una técnica que nos permite la agrupación de grandes volúmenes de datos para una interpretación más sencilla, esta técnica puede ser utilizado en el llamado machine learning pero en nuestro caso solo queremos que nos agrupe información, en otras palabras queremos hacer minería de datos. Dentro de esta práctica se pueden encontrar varios algoritmos para ejecutarla, en nuestro caso elegimos el algoritmo de nombre Kmeans debido a ser uno de los más sencillos de implementar.

En nuestro caso particular se eligió obtener la información de una base de datos del inegi, esta base de datos contiene toda la información sobre la población del estado de Jalisco en base a las localidades de esta, además esta información está contenida en un archivo CSV del cual se extrae la información, se extrae la localidad como un número y el número de población perteneciente a esa localidad

Diagrama de relaciones



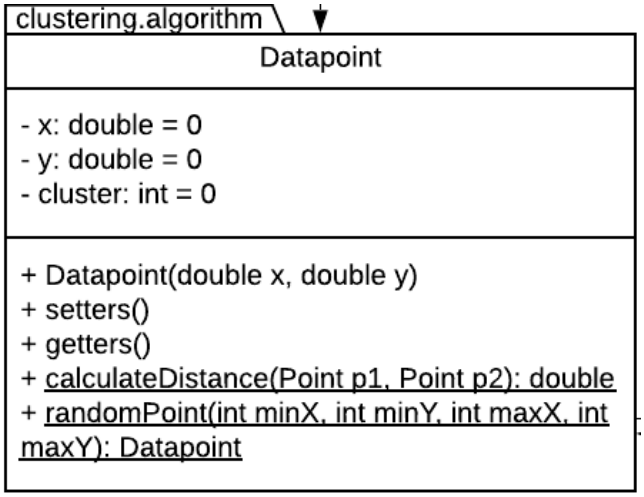
Diagramas UML



La interface se encarga de describir características que necesitamos que los puntos (vectores de dos dimensiones) tengan para el programa,

calculateDistance() se encarga obtener la distancia que hay entre dos puntos

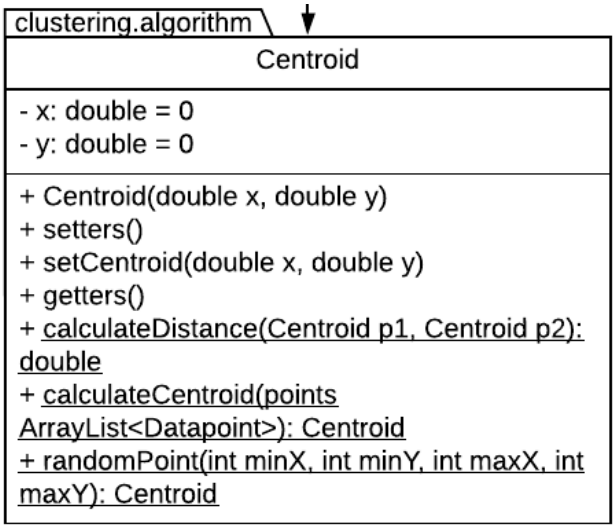
randomPoint() se encarga de generar un punto aleatorio dentro del rango permitido, esto nos sirve al momento de crear centroides para el algoritmo



Datapoint implementa la interfaz **Point**, se trata de los puntos otorgados por el medio de información, en este caso la base de datos, no crea métodos más allá de los dictados por la interface.

El contructor se usa para asignar los valores pasados como argumentos, las coordenadas x,y, y ponerlos en los atributos con los mismo nombres.

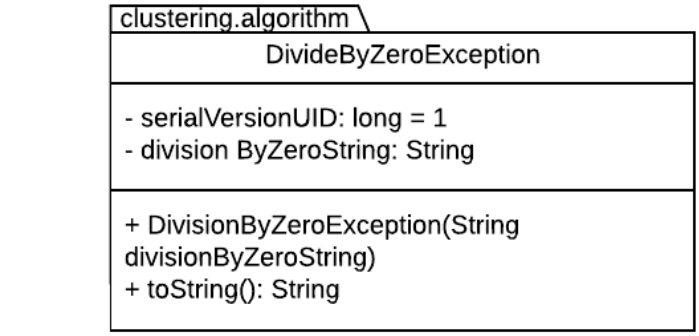
El atributo *cluster* guarda a que cluster o grupo pertenece le punto.



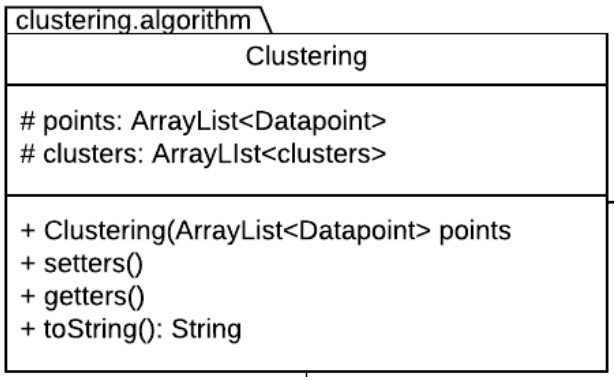
Centroid es una clase que también implementa la interface Point, su única diferencia con ***Datapoint*** es el método **calculateCentroid()** el cual simplemente genera centroides aleatorios para delimitar los clusters.

setCentroid() asigna en un solo método lo valores de xy al punto

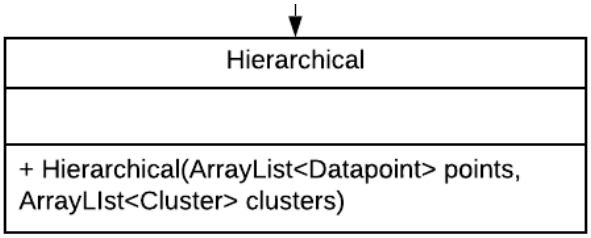
calculateCentroid() en base a los puntos pertenecientes al centroide, obtiene el promedio de las coordenadas de todos los puntos



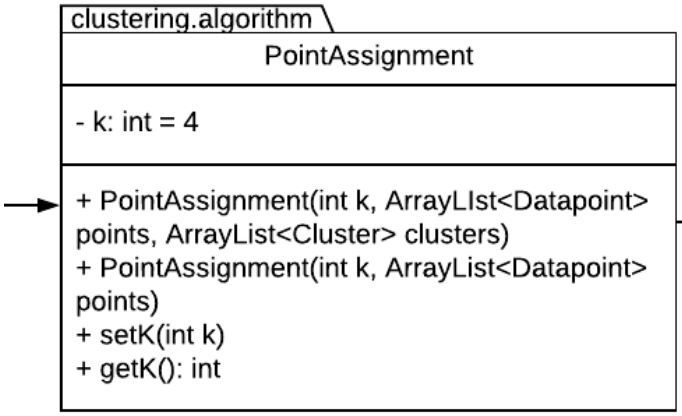
Esta clase simplemente nos presenta control de excepciones, como su nombre indica, si se intenta hacer una división sobre cero,



Es la clase padre de todo lo relacionado al agrupamieto, nos presenta una forma de guardar puntos y clusters en un objeto como atributos.

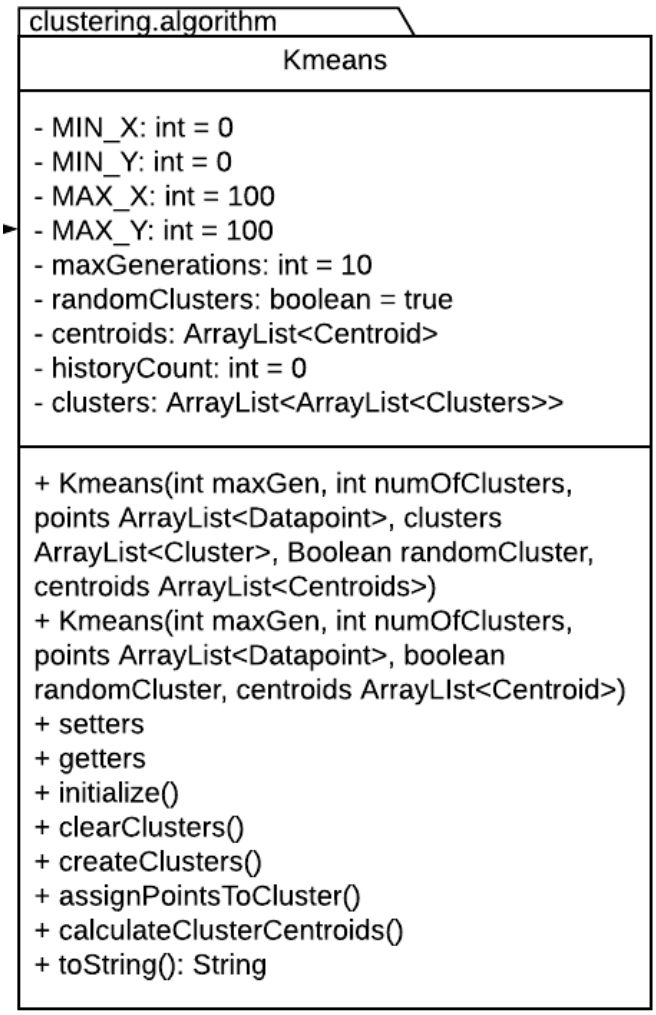


Es una especialización de **clustering**, se puso simplemente como forma de visualizar los grupos de algoritmos de clustering



Especializa un poco la clase **clustering**, se mueve en torno a los algoritmos no jerárquicos o **PointAssignment**.

Esta especialización nos provee otro atributo, *k*, que nos indica el número de clusters que se tendrá al momento de agrupar.



Es el algoritmo de clustering que se eligió para el proyecto,

Los MIN y MAX de X y Y, nos presentan los límites de valores para los puntos propuestos por la información para el funcionamiento del algoritmo,

maxGenerations nos indica la cantidad de veces que se recalcularán lo centroides en base a los puntos que se asignen.

randomClusters simplemente es para saber si se generarán centroides aleatorios o se tendrán unos predeterminados.

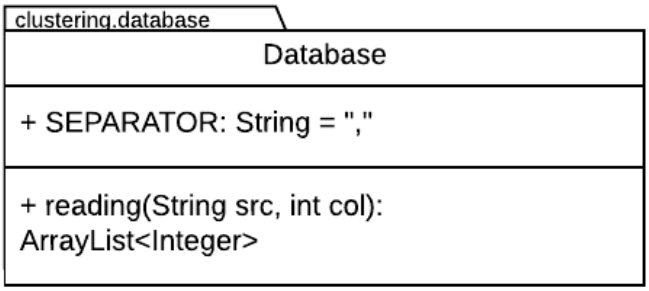
Los constructores simplemente pasan los valores de los argumentos a los atributos del objeto con el mismo nombre.

Initializaze() realiza cada una de las iteraciones del algoritmo de clustering, limpia los puntos asignados y se calculan los nuevos puntos y centroides hasta llegar a un máximo de generaciones.

createCluster() crea el número de clusters indicados con k

assignPointsToCluster() calcula la longitud máxima que puede haber entre dos puntos, con los valores de MAX y MIN de X y Y, y calcula la distancia que hay del punto a cada uno de los clusters creado, si la distancia está dentro de lo permitido asigna ese punto al cluster con el que menos distancia encuentre.

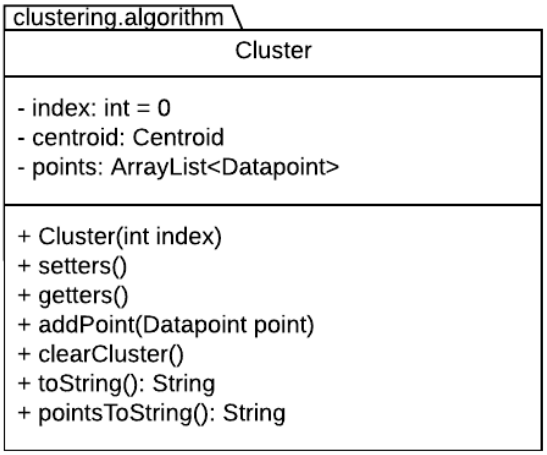
calculateClusterCentroid() asigna un centroide a cada cluster, ya sea aleatorio o predeterminado.



Nos ofrece lo necesario para extraer la información del archivo CSV.

Los archivos CSV son archivos separados por comas, por lo que cada coma se separan los elementos, de ahí SEPARATOR,

reading() se encarga de extraer la información de la columna deseada(col), en base al archivo con la ruta que especifiquemos(src)



La clase **Cluster** nos permite crear cluster o grupos, estos tienen un centroide y los puntos que lo rodean.

El atributo **index** nos indica el número de cluster que estamos trabajando.

toString() nos regresa las coordenadas del cluster con su número y su centroide correspondiente.

pointsToString() nos regresa las coordenadas de los puntos correspondientes a ese cluster

Caso de aplicación: agrupación de localidades

Como se mencionó en la introducción la aplicación pensada para la API es la agrupación de localidades en Jalisco, información otorgada por el inegi, y de esta manera agrupar los datos conforme estos dos aspectos.

Con esto en mente los resultados obtenidos variaron en base a cada implementación, ya que se usaron datos aleatorios para determinar las condiciones de cada cluster, pero aquí abajo se puede observar una de estas implementaciones:

```
History to String
-----------------------
Gen: 0
-----
Cluster 0
Centroide (0.0,0.0)
Puntos: 0
-----
Cluster 1
Centroide (496802.0,2375963.0)
Puntos: 3046
-----
Cluster 2
Centroide (1.0,97750.0)
Puntos: 1
-----------------------
Gen: 1
-----
Cluster 0
Centroide (496801.0,2323788.0)
Puntos: 3045
-----
Cluster 1
Centroide (0.0,0.0)
Puntos: 0
-----
Cluster 2
Centroide (2.0,149925.0)
Puntos: 2
-----------------------
```

```
Gen: 2
-----
Cluster 0
Centroide (0.0,0.0)
Puntos: 0
-----
Cluster 1
Centroide (496802.0,2375963.0)
Puntos: 3046
-----
Cluster 2
Centroide (1.0,97750.0)
Puntos: 1
----------------------
Gen: 3
-----
Cluster 0
Centroide (496801.0,2323788.0)
Puntos: 3045
-----
Cluster 1
Centroide (0.0,0.0)
Puntos: 0
-----
Cluster 2
Centroide (2.0,149925.0)
Puntos: 2
----------------------

Gen: 4
-----
Cluster 0
Centroide (0.0,0.0)
Puntos: 0
-----
Cluster 1
Centroide (496802.0,2375963.0)
Puntos: 3046
-----
Cluster 2
Centroide (1.0,97750.0)
Puntos: 1
```

Para esta implementación se puso que fueran simplemente 5 generaciones, es decir que la calculación de los centroides se hiciera 5 veces, después de eso dejara los puntos y los grupos así, y 3 clusters, para esta implementación se utilizó el código de la clase test:

```
public static void main(String[] args) throws
DivisionByZeroException {
        String src =
"C:\\Users\\OEM\\Desktop\\Entrega\\iter_14_cpv2010-1.csv";
        ArrayList < Integer > data_x = Database.reading(src, 4);
        ArrayList < Integer > data_y = Database.reading(src, 9);

        ArrayList < Datapoint > points = new ArrayList<>();
        ArrayList < Centroid > centroids = new ArrayList<>();

        int maxGenerations = 5;
        int numberOfClusters = 3;
```

```
        boolean randomCluster = true;

        for(int i = 0; i < data_x.size(); i++) {
            Datapoint p1 = new Datapoint(data_x.get(i),
data_y.get(i));
            points.add(p1);
        }

        Kmeans k = new Kmeans(maxGenerations, numberOfClusters,
points, randomCluster, centroids);

        k.initialize();
        System.out.println(k.toString());

    }
```

Como podemos observar el procedimiento para la prueba es bastante sencillo primero tenemos dos ArrayList de Integer, uno para guardar el número de la localidad (data_x) y el segundo para guardar el número de población de la localidad correspondiente (data_y), así pues tenemos por debajo dos ArrayList, uno para puntos y el otro para los centroides, el de puntos se llena con la información obtenida del archivo CSV, declaramos el número de generaciones, clusters y si queremos que los centroides sean aleatorios o no, se llama al algoritmo (Kmeans) y se le entregan como argumentos los datos antes inicializados este algoritmo requiere de todas las clases propuestas en el proyecto, se usa el método initialize que lo que hace es ejecutar el algoritmo de asignación y recalculación las veces indicadas y por último se imprime el toString de K con todos los datos de los clusters y sus respectivos puntos.

Conclusión

Con este proyecto pudimos poner en práctica los conocimientos de la programación orientada a objetos mientras resolvíamos el reto que fue el algoritmo de clustering, pudimos reforzar los aprendido en clase, lo que mas usamos creo yo fueron los contenedores genéricos, y aprender algunas cosas nuevas, como el manejo de archivos, al final fue un proyecto retador pero lo compenso siendo bastante interesante.