Etapa 1: Identificación del problema.

Contexto del problema:

- Los profesores no tienen un software que les permita gestionar eficientemente las operaciones CRUD sobre una base de datos de personas.
- No hay un programa que genere la creación de registros de personas para el 2020 simulando el crecimiento de la población
- Para generar los nombres completos, se deben tomar los nombres de un dataset específico.
- Los apellidos se deben tomar del archivo completo de un dataset específico. La combinación de todos los nombres con todos los apellidos debe producir la cantidad (o similar) de personas que se desea deseamos.
- La fecha de nacimiento debe ser generada aleatoriamente, suponiendo una distribución de edad para toda América basada en una distribución específica. La distribución de la población en sexo puede asumirse una cantidad igual de hombres y mujeres.
- La estatura debe ser generada aleatoriamente en un intervalo que tenga sentido.
- La nacionalidad debe ser asignada a cada persona, generada de tal forma que se mantengan los porcentajes relativos de población de cada país respecto del continente de acuerdo a unos datos de población específico (se puede también asignar exactamente la misma población de cada país, y si hay diferencia en el total, dicha diferencia -positiva o negativa- puede quedar en el país con mayor población).

Definición del problema:

Se debe desarrollar un prototipo de software que gestione eficientemente las operaciones CRUD sobre una base de datos de personas de nuestro continente

Especificación de requerimientos:

- El programa debe tener una opción para empezar a generar los datos siguiendo las especificaciones.
- Debe tener una barra de progreso si el proceso tarda más de 1 segundo en terminar, y debe indicar cuánto tiempo se demoró la operación.
- La opción de generar debe tener un campo de texto en el cual se pueda digitar cuántos registros se desea generar.
- Una vez los datos se generan, debe haber una opción para guardarlos en la base de datos del programa, y así poderlos consultar posteriormente.

- Todos los datos del programa deben ser persistentes (es decir, si se cierra el programa, deben seguir allí una vez se inicie nuevamente).
- El programa desarrollado por su equipo debe tener la posibilidad de llevar a cabo cualquiera de las funciones CRED.
- La interfaz con el usuario deberá contar con un formulario para agregar a una nueva persona, otro para buscar a una persona por los criterios mencionados más adelante y uno más para actualizar los campos de una persona existente. Este último formulario debe también tener una opción para eliminar a una persona.
- El formulario para agregar debe tener todos los campos requeridos para la información de una persona y la opción de Guardar.
- El formulario para actualizar una persona, debe tener todos los campos editables de información de una persona, cargados con su información actual, la opción de Actualizar (para guardar los cambios, si hubo) y la opción Eliminar (si se desea eliminar a esta persona).
- El formulario para buscar debe tener la posibilidad de realizar la búsqueda por cualquiera de los siguientes criterios: nombre, apellido, nombre completo y código, debe estar de forma excluyente.

Etapa 2: Recopilación de información

CRUD: la base de la gestión de datos
El concepto CRUD está estrechamente vinculado a la gestión de datos digitales. CRUD hace referencia a un acrónimo en el que se reúnen las primeras letras de las cuatro operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos:

- Create (Crear registros)
- Read bzw. Retrieve (Leer registros)
- Update (Actualizar registros)
- Delete bzw. Destroy (Borrar registros)

En pocas palabras, CRUD resume las funciones requeridas por un usuario para crear y gestionar datos. Varios procesos de gestión de datos están basados en CRUD, en los que dichas operaciones están específicamente adaptadas a los requisitos del sistema y de usuario, ya sea para la gestión de bases de datos o para el uso de aplicaciones. Para los expertos, las operaciones son las herramientas de acceso típicas e indispensables para comprobar, por ejemplo, los problemas de la base de datos, mientras que para los usuarios, CRUD significa crear una cuenta (créate) y utilizarla (read), actualizarla (update) o borrar (delete) en cualquier momento. Dependiendo de la configuración regional, las operaciones CRUD pueden implementarse de diferentes maneras, como lo muestra la siguiente tabla:

CRUD-Operation	SQL	RESTful HTTP	XQuery
Create	INSERT	POST, PUT	insert
Read	SELECT	GET, HEAD	copy/modify/return
Update	UPDATE	PUT, PATCH	replace, rename
Delete	DELETE	DELETE	delete

fuente: ¿Qué es CRUD? | Operaciones CRUD en programación - IONOS

First Names - dataset by alexandra | data.world

Frequently Occurring Surnames from the Census 2010 - dataset by uscensusbureau | data.world

Population by Country - 2020 | Kaggle

Create, read, update and delete - Wikipedia

Etapa 3: Búsqueda de soluciones creativas.

- Nuestro equipo de trabajo tuvo la idea de utilizar árboles binarios para hacer de una tarea algo más sencilla la ardua acción de realizar la extensa búsqueda de la larga lista de nombre que hay, ademas de que la búsqueda se pueda realizar no únicamente con el nombre sino con cualquier dato de la persona autogenerada es decir, apellido, código, nombre completo, y la solución más efectiva para realizar esta acción es utilizando árboles binarios.

Etapa 4: Transición de las ideas a los diseños preliminares.

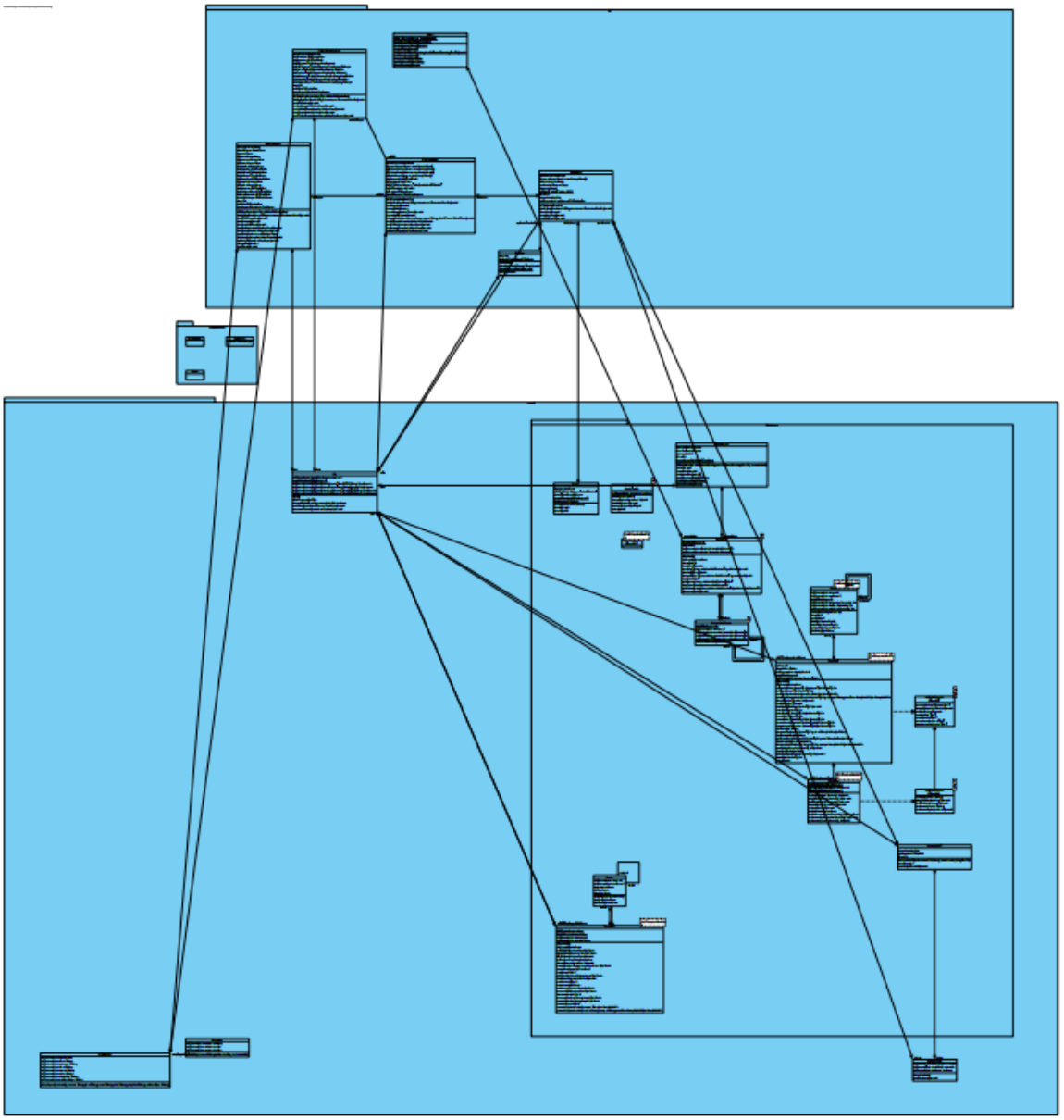
Debido a que todos los aportes del grupo fueron consistentes no se realizó el descarte de ninguna idea además de que fueron correctamente planteadas desde un principio. Además también se optó por agregar linkedlist para ayudar en la solución del problema

Etapa 5: Evaluación y selección de la mejor solución.

Brinda una solución satisfactoria	Realiza una solución óptima	Realiza todo con los requerimientos específicos	No deja ningún punto sin resolver
x	x	x	x

La idea que se seleccionó sigue siendo la planteada desde el inicio ya que de acuerdo a la rúbrica nos brinda la mejor solución posible para poder desarrollar de una manera adecuada las específicas peticiones que se realizaron.

Etapa 6: Preparación de Informes y Especificaciones



Etapa 7: Implementación del diseño

```
12  public class Main {
13
14      private static final String SRC_FILE = "data/Data.txt";
15      private static linkList<Country> countries;
16
17      public static void main(String[] args) throws Exception {
18          mainDriver();
19      }
20
21      public static void mainDriver() throws Exception {
22          BufferedReader br = new BufferedReader(new FileReader("data/Names_2010Census.csv"));
23          BufferedReader rb = new BufferedReader(new FileReader("data/babynames-clean.csv"));
24          Cd fb = new Cd(); //Crud
25          init(br, rb, fb);
26          ObjectOutputStream os = new ObjectOutputStream(new FileOutputStream(SRC_FILE));
27          os.writeObject(fb);
28          os.close();
29      }
30
31      public static void init(BufferedReader br, BufferedReader rb, Cd fb) throws IOException {
32          try {
33              readCountry();
34          } catch (IOException e) {
35              e.printStackTrace();
36          }
37
38          int num = 1;
39          int num1 = 1;
40          DecimalFormat df = new DecimalFormat("#.00");
41          String cord = br.readLine();
42
43
44          while (cord != null && num < 20) {
45              try {
46                  cord = br.readLine();
47                  if (cord != null) {
48                      String name = cord.split(",")[0];
49
50                      String cord1 = rb.readLine();

50                      String cord1 = rb.readLine();
51
52                      while (cord1 != null  && num1 < 100){
53                          cord1 = rb.readLine();
54
55                          String ln = cord1.split(",")[0];
56                          String sex = (Math.random() < 0.5)? "Female": "Male";
57                          String bd = randomDate();
58                          String height =  df.format((Math.random()*(1.80-1.50)+1.50));
59                          String nationality = getNationality();
60
61                          Employee p = new Employee(fb.getTotalEmployees() + 1, name, ln, sex, bd, height, nationality); //Person
62
63                          fb.addEmployee(p);
64                          num1++;
65                      }
66                  }
67              } catch (Exception e) {
68                  e.printStackTrace();
69              }
70              num++;
71              num1 = 1;
72          }
73          try {
74              br.close();
75              rb.close();
76          } catch (IOException e) {
77              e.printStackTrace();
78          }
79      }
80
81      public static int[] distribution(){
82          Calendar calendar = Calendar.getInstance();
83          int year = calendar.get(Calendar.YEAR);
84          double random = Math.random();
85          int sYear = 0; //Start of year
86          int eYear = 0; //End of year
87
88          if (random >= 0 && random < 0.19){
89              sYear = year - 14;
```

```
            sYear = year - 14;
            eYear = year;
        }else if (random >= 0.19 && random < 0.32){
            sYear = year - 24;
            eYear = year - 15;
        }else if(random >= 0.32 && random < 0.72){
            sYear = year - 54;
            eYear = year - 25;
        }else if(random >= 0.72 && random < 0.85){
            sYear = year - 64;
            eYear = year - 55;
        }else{
            sYear = 1900;
            eYear = year - 65;
        }
        return new int[]{sYear, eYear};
    }



    public static String randomDate() {
        int[] limit = distribution();
        Calendar cal = Calendar.getInstance();
        cal.set(cal.get(Calendar.YEAR), ((int) (Math.random()*(13-1)+1)), ((int) (Math.random()*(31-1)+1)));

        return new SimpleDateFormat("dd").format(cal.getTime()) + "/" + new SimpleDateFormat("MM").format(cal.getTime()) + "/" + ((int) (Math.random()*(limit[1]-limit[0])+limi
    }

    public static String getNationality(){
        double random = Math.random();
        boolean flag = true;
        int cont = 0;

        while (flag){
            if (random >= countries.get(cont).getInit() && random < countries.get(cont).getEnd()){
                flag = false;
                return countries.get(cont).getName();
            }
            cont++;
        }
        return null;
    }

    public static void readCountry() throws IOException {
        BufferedReader brr = new BufferedReader(new FileReader("data/population_by_country_2020.csv"));
```

```
        double random = Math.random();
        boolean flag = true;
        int cont = 0;

        while (flag){
            if (random >= countries.get(cont).getInit() && random < countries.get(cont).getEnd()){
                flag = false;
                return countries.get(cont).getName();
            }
            cont++;
        }
        return null;
    }

    public static void readCountry() throws IOException {
        BufferedReader brr = new BufferedReader(new FileReader("data/population_by_country_2020.csv"));
        countries = new linkList<>();
        String cord = brr.readLine();
        double init = 0;

        while (cord != null) {
            cord = brr.readLine();

            if (cord != null) {

                String[] data = cord.split(",");
                String name = data[0];
                double end;

                end = ((Double.parseDouble(data[1])/(790*1000000))/10);

                if (countries.isEmpty()) {
                    countries.add(new Country(name, init, end));

                }else{
                    init = (countries.get(countries.size()-1).getEnd());
                    end = end + init + 0.0000496;
                    countries.add(new Country(name, init, end));
                }
            }else {
                break;
            }

        }
        brr.close();
    }
}
```