Entrega Parte A

Programación avanzada ICI3241-1

Nombres: Valentina San Martín, Marlene Lagos y Matías Alevropulos

Fecha de entrega: 02 de Mayo de 2021

Repositorio GitHub de Versiones de Proyecto:

https://github.com/Arakito14/Fiscalia/

Fiscalías Regionales de Chile (19, 4 en RM):

http://www.fiscaliadechile.cl/Fiscalia/quienes/fiscaliaReg.jsp

Distritos Electorales de Chile (28):

https://es.wikipedia.org/wiki/Divisi%C3%B3n_electoral_de_Chile

EP1.1. Realizar un análisis de los datos a utilizar y principales funcionalidades a implementar que dan sentido a la realización del proyecto.

El proyecto a realizar tratará acerca de los casos que manejan los fiscales dentro de la Fiscalía de Chile. El objetivo del programa será brindar al Ministerio Público una forma de mantener organizadas las causas y los fiscales que tienen a su disposición a lo largo de todo el país.

Los datos a utilizar son:

-Causas que posee el Ministerio Público, incluyendo el código que las identifica, los datos del fiscal a cargo del caso, la información sobre los procedimientos realizados en el caso,el estado del caso y el lugar en donde se desarrolla.

-Los fiscales que trabajan para el Ministerio Público, incluyendo sus nombres, rut, su especialidad en el ámbito jurídico, su ubicación actual y los casos que tiene a su cargo.

Las funcionalidades a implementar son:

1.- Mostrar fiscales: Muestra por pantalla la información de todos los fiscales dentro de la fiscalía.

2.-Mostrar Causas: Muestra por pantalla la información de todas las causas dentro de la fiscalía, incluyendo una lista de sus procedimientos.

3.-Buscar Fiscal: El usuario ingresa el rut del fiscal y el programa le muestra la información de ese fiscal, incluyendo la información sobre sus causas y los procedimientos de las mismas.

4.-Buscar causa: El usuario ingresa el código de la causa y el programa le muestra la información de esa causa, incluyendo la información del Fiscal a cargo y sus procedimientos.

5.-Nuevo fiscal: Se le pide al usuario ingresar la información del fiscal y a partir de eso ingresar el nuevo fiscal al sistema.

6.-Nueva causa: Se le pide al usuario ingresar codigo de causa nueva, su estado (abierto,cerrado), tipo, su distrito, creará una causa a nuestro mapa.

7.-Agregar procedimiento a una causa: Se pide al usuario o fiscal ingresar código de una causa existente, se ingresa nombre de peritaje, nombre del participante, su rol, si desea ingresar mas participantes y el resultado de su peritaje.

8.-Asignar una causa a un fiscal: Primero se pregunta qué causa (sin Fiscal asignado) quieres asignar, luego entregamos una sugerencia de Fiscales recomendados para la causa en específico (Basado en distrito y especialidad)

9.-Modificar el distrito de un fiscal: A partir del RUT de nuestro fiscal, se asignará un nuevo distrito basado en la elección del usuario.

10.-Cambiar el fiscal encargado de una causa: Se le pide al usuario ingresar código de causa y el RUT del nuevo fiscal encargado de tal causa.

11.-Cambiar el resultado de un procedimiento: Se le pide al usuario ingresar código de causa, se muestra causa por pantalla, se muestran procedimientos, se le pide al usuario el número de procedimiento a cambiar resultado.

12.-Eliminar Fiscal: Se elimina fiscal a partir de su RUT.

13.-Eliminar Causa: Se elimina la causa a partir de su Código.

14.-Eliminar Procedimiento: Se elimina el procedimiento a partir de su Código

Errores menores: Errores de excepción en caso de resultado no esperado, algunos errores menores de formato.

EP1.2. Diseño conceptual de clases del Dominio y su código en Java.

Las clases a implementar en el programa son:

1. Procedimiento: Esta clase guardará la información de los diferentes procedimientos que se vayan realizando en sus respectivos casos. Sus atributos son:

a) String nombreProc: Guarda el nombre del procedimiento, o sea si se trata de una orden de cateo, una toma de testimonio, una autopsia, etc.
b) String participantes[]: Guarda los nombres de todos los implicados en el respectivo procedimiento. Es un Array para poder almacenar todos los participantes.
c) String roles[]: Guarda los roles que cumplen los participantes del procedimiento actual, o sea si son policías, testigos, el abogado,etc. Es un Array para poder almacenar todos los roles.
d) String Resultado: Guarda una breve descripción de cómo resultó el respectivo procedimiento.

2. Causa: Esta clase guardará la información de las diferentes causas que se ingresen en el programa. Sus atributos son:

a) String codigo: Guarda el código numérico que se suele usar para identificar cada caso dentro de la fiscalía. Es tipo string porque a pesar de ser un número no necesitaremos hacer ninguna operación con él
b) Fiscal encargado: Guarda los datos del fiscal que está a cargo de la causa. Es un objeto de la clase Fiscal porque será útil para las funcionalidades a implementar.
c) Procedimientos peritajes[]: Guarda en orden los procedimientos que se han realizado en su respectiva causa. Es un Array para poder almacenar todos los peritajes.
d) String estado: Guarda el estado actual de la causa.
e) String tipoCaso: Guarda la temática del delito investigado en la causa. Se incluye este dato debido a que normalmente se elige al fiscal considerando la temática que sea su especialidad (crímenes sexuales, económicos, crimen organizado, etc).
f) int distrito: Guarda el distrito en donde se desarrolla la causa. Se incluye este dato debido a que también se elige a los fiscales por la cercanía al lugar del suceso.
g) boolean asignada: Booleano que indica si la causa ya ha sido asignada a algún fiscal.

3. Fiscal: Esta clase almacena los datos de cada fiscal junto con sus causas en proceso y su especialidad, la cual será utilizada para la asignación de casos posteriormente. Sus atributos son:

a) String nombre: Almacena el nombre de cada fiscal.
b) String rut: Almacena el rut de cada fiscal para facilitar la búsqueda de estos.
c) Causa causasActuales[]: Almacena en un arreglo todas las causas que están siendo procesadas por cada fiscal.
d) String especialidad: Almacena el área de especialidad de cada fiscal, la cual será utilizada como discriminador para la asignación de los casos.
e) String región: Almacena la región donde ejerce su trabajo cada fiscal, la cual servirá posteriormente como filtro para la asignación de casos.

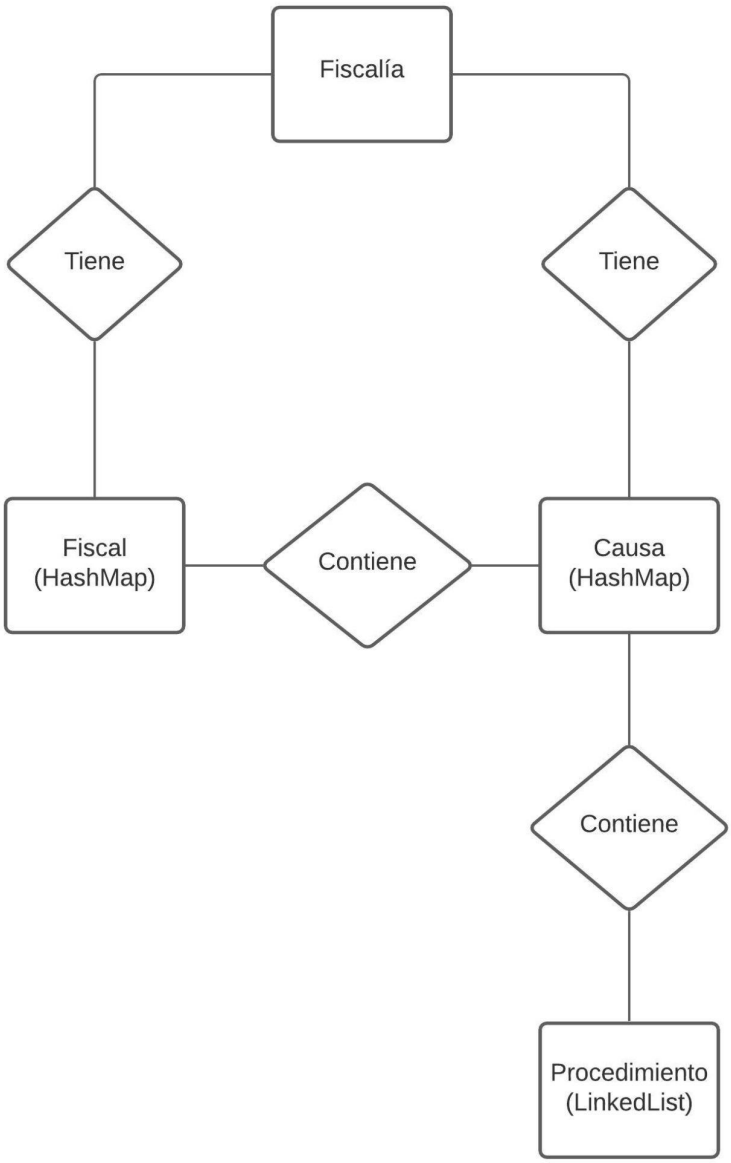
4.Fiscalía: Esta clase almacena las colecciones principales de fiscales y causas para mantener un registro dentro del programa. También almacena una contraseña que más adelante se usará para iniciar sesión como administrador dentro del programa. Sus atributos son:

a) HashMap<String,Fiscal> Fiscales: Almacena la información de todos los fiscales.
b) HashMap<String, Causa> Causas: Almacena la información de todas las causas.
c) String contraseña: Guarda la contraseña de administrador.

EP2. Diseño conceptual de la anidación de colecciones

Las colecciones que se manejan en este programa son un mapa de fiscales, mapas de causas y mapas de procedimientos. En primera instancia el mapa de fiscales y el mapa de causas

existen de forma separada para llevar el registro de los elementos existentes, pero cada elemento de la clase Fiscal lleva dentro otro mapa de causas en donde contiene las causas que le corresponden en ese momento. Por otra parte, todas las causas contienen en su interior una lista enlazada de procedimientos que muestra el progreso de la causa en cuestión.



EP3. Diseño de diagrama de clases UML de lo considerado hasta la EPA

El diagrama de clases UML se encuentra adjunto en la entrega.

Revisar, además repositorio Github, readme con instrucciones en página inicial, gracias!