Análisis y modelado de un sistema para la gestión de formularios

López, Carlos – 20162021295

López González, Brayan Arnulfo – 20162021313

6 de marzo de 2019

Universidad Distrital Francisco José de Caldas

Ingeniería de Sistemas

Ciencias de la computación I

Bogotá, D. C.

Tabla de contenido

[Introducción 3](#_Toc2617439)

[Objetivos 4](#_Toc2617440)

[General 4](#_Toc2617441)

[Especifico 4](#_Toc2617442)

[Descripción de la situación 5](#_Toc2617443)

[Requerimientos funcionales 6](#_Toc2617444)

[Modelado y análisis 8](#_Toc2617445)

[Diagrama de casos de uso 8](#_Toc2617446)

[Diagrama de clases 10](#_Toc2617447)

[Organización del proyecto 10](#_Toc2617448)

[Diagrama de paquetes 11](#_Toc2617449)

[Manejo información 11](#_Toc2617450)

[Conclusiones 12](#_Toc2617451)

# Introducción

El presente documento es un complemento o soporte para un sistema de software encargado de ayudar en la gestión de formularios en una empresa con varias sucursales a través del mundo. El sistema usa estructuras de datos como multilistas para guardar la información en memoria principal, optimizar su búsqueda y modificación. En memoria secundaria se trabaja a través de archivos planos.

El documento está dividido en dos partes, la primera explica detalladamente la situación o problema sobre el que se trabaja. La segunda parte corresponde al modelo y análisis que se realizó para resolver el problema.

# Objetivos

## General

Realizar un sistema para facilitar la gestión de formularios en una empresa.

## Especifico

Entender la importancia de usar estructuras de datos como listas, multilistas, arboles, entre otras y las diferentes maneras en las cuales esto puede facilitar el desarrollo de software.

Aplicar los conceptos y habilidades adquiridas durante el semestre.

# Descripción de la situación

Una empresa desea recoger información acerca de sus empleados con el fin de hallar perfiles que les permitan enfocar el desarrollo de las labores de manera más eficiente. Para ello, los gerentes ordenan la elaboración de formularios que son entregados en cada una de las sucursales ubicadas a lo largo y ancho del país (puede haber más de una sucursal en la misma ciudad). La información que se debe consignar allí es la siguiente:

* Nombre
* Apellido
* Tipo de identificación: puede ser uno de los siguientes valores: Cédula de ciudadanía, cédula de extranjería o tarjeta de identidad.
* Número de identificación
* Sexo: F o M
* Teléfono celular
* Teléfono fijo
* e-mail
* Fecha de Nacimiento
* Ciudad de Nacimiento
* País de Nacimiento
* Ciudad de Residencia
* Dirección
* Barrio
* Actividad laboral: Artes, ciencias sociales, ingenierías, áreas de la salud, otros
* Tiene hijos?: como respuesta se acepta S ó N
* Número de hijos
* Listado con el nombre y fecha de nacimiento de cada uno de los hijos: 0-5, 6-10, 11-15, mayor de 16
* Nombre de la sucursal en la que trabaja

Datos de la sucursal.

* Nombre de la sucursal
* Dirección de la sucursal
* Barrio
* Nombre del gerente de la sucursal.

Para evitar que la solicitud de información se haga demasiado grande, el programa debe estar en capacidad de calcular la edad de cada una de las personas encuestadas.

## Requerimientos funcionales

Con el fin de optimizar las búsquedas deberán utilizarse listas y la multilista definida. Las cuales se construyen al iniciar el programa y a partir de la información almacenada en los archivos.

Debe garantizar la eficiencia en las búsquedas aprovechando las estructuras de datos que considere pertinentes.

El programa presenta un menú y submenús que facilitan las consultas.

Adicionalmente, las listas deben crearse de manera genérica, de tal forma que puedan agregarse fácilmente otras consultas adicionales por parte de la empresa.

El programa debe estar en capacidad de calcular la edad de cada una de las personas registradas.

Para la entrega y sustentación final deben incluir los archivos planos indispensables con no menos de 30 registros.

Debe aplicar programación orientada a objetos y el principio de alta cohesión y bajo acoplamiento.

Debe aplicar los conceptos de optimización estudiados en el curso. No limitarse a garantizar el funcionamiento del programa.

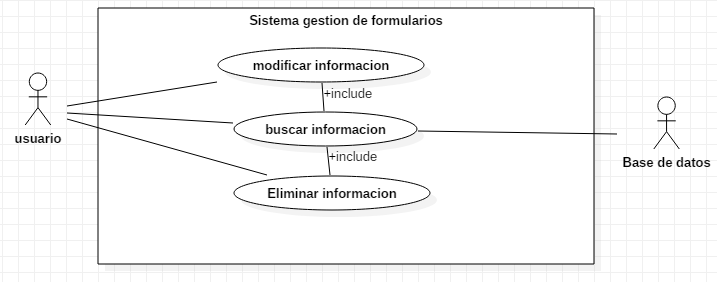
Al finalizar el programa se actualizan los archivos incluyendo los cambios realizados durante la ejecución. Se agregaron o retiraron feligreses.

# Modelado y análisis

Se intentó trabajar usando un modelo de proceso iterativo incremental, en el que cada iteración aumenta una parte del sistema.

A continuación se encuentra el análisis y modelado del sistema, se trabajó teniendo en cuenta todo lo explicado en la descripción del problema o situación. Se comenzó realizando un diagrama de casos de uso, donde se detallan los comportamientos del sistema y su interacción de los actores del mismo; partiendo del anterior diagrama se realizó el de clases y paquetes.

## Diagrama de casos de uso

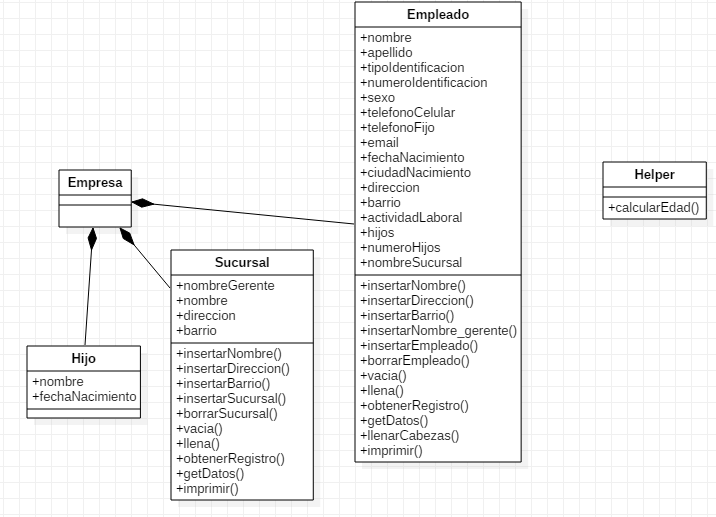


*Fig. 1. Diagrama de casos de uso. Fuente: elaboración propia en StarUML.*

En la figura 1 se observa el diagrama de casos de uso, que posee tres casos de uso y dos actores. A continuación se encuentran los casos de uso detallados.

|  |  |
| --- | --- |
| **NOMBRE: Buscar información** | **ID: 001** |
| **Descripción:** El usuario solicita información al sistema, este lo busca en la base de datos y lo retorna. | |
| **Actor:** Usuario | |
| **Precondición:** | |
| **Evento: Actor** | **Evento: Sistema** |
| 1. El usuario ingresa la información que requiere. | 2. El sistema lee la información, corrobora que sea válida y envía la búsqueda a la base de datos.  3. Se comprueba que la información se encuentre en la base de datos y se retorna al usuario |
| **Poscondición:** Se genera un informe con la información que solicito el usuario, en caso de encontrarse se le informa al usuario que la información no hace parte de la base de datos. | |
| **Restricción:** El sistema hace la búsqueda de forma textual, en caso de que el usuario ingrese mal una letra o le falte algún símbolo, el sistema no encontrara la información buscada. | |

## Diagrama de clases



*Fig. 2. Diagrama de clases. Fuente: elaboración propia en StarUML.*

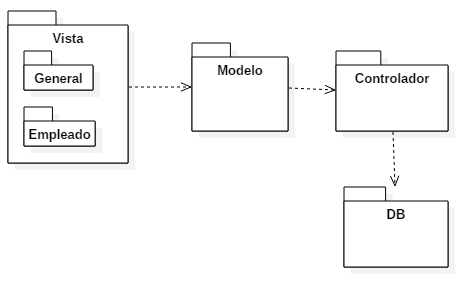
Como se puede ver en la figura 2, el proyecto consta de cinco clases. La clase empresa que contiene las clases empleado, sucursal e hijo. La clase helper ayuda en funciones básicas como calcular la edad de los empleados.

## Organización del proyecto

El proyecto usa la arquitectura de modelo-vista-controlador. En el modelo están definidas las estructuras de datos que servirán para la gestión de formulario. La vista es presentada por consola, esta permite al usuario poder acceder a las diferentes funcionalidades del programa. El controlador verifica los datos ingresados por el usuario y conecta la vista con el modelo.

A continuación en la figura 3 se encuentra el diagrama de paquetes en el cual se muestra la distribución del sistema en cuatro paquetes, Vista, Modelo, Controlador y DB.

### *Diagrama de paquetes*



*Fig. 3. Diagrama de paquetes. Fuente: elaboración propia en StarUML.*

## Manejo información

Memoria principal y secundaria.

# Conclusiones

Usar estructuras de datos como multilistas permite estructurar un proyecto de forma adecuada para realizar consultas y hacer las modificaciones correspondientes de forma rápida.