



UNIVERSIDAD NACIONAL  
AUTÓNOMA DE MÉXICO  
FACULTAD DE INGENIERÍA

PROGRAMACIÓN ORIENTADA A OBJETOS

# PROYECTO

Profesor: Jorge Rommel Santiago Arce

Grupo: 2

## INTEGRANTES:

- Flores Cruz Karina Alejandra
- Tapia Sandoval Eduardo Herminio
- Vázquez Méndez Enrique
- Velázquez Rodríguez Diego

Fecha de entrega: 19/11/2019

## ÍNDICE

❖ Objetivo.....	3
❖ Desarrollo.....	3
• Descripción del programa.....	3
• Instrucciones de uso.....	4
• Firmas de las funciones usadas en el programa.....	6
○ ProyectoPOO.java.....	6
○ archivos.java.....	6
○ trabajador.java.....	7
○ Tarea.java.....	8
○ TareasAdmin.java.....	8
• Diagramas UML.....	8
• Análisis de diseño.....	10
• Análisis de la solución.....	11
• Implementación.....	12
❖ Comentarios.....	13
• Flores Cruz.....	13
• Tapia Sandoval.....	14
• Diego Velázquez.....	14
• Vázquez Méndez.....	15

## OBJETIVO

Aplicar los conceptos de la programación orientada objetos para implementar en un programa.

## DESARROLLO

### 1. Descripción del programa

Dentro del programa tenemos una aplicación que es capaz de administrar tareas, que pueden asignarse a un desarrollador. Dentro del programa tenemos un usuario administrador, al cual se le llama líder de proyecto, puede editar las actividades, es decir, crearlas, eliminarlas y asignarlas a los distintos desarrolladores. Cada desarrollador puede editar sus actividades asignadas de acuerdo con los siguientes estados:

a) Pendiente: Estado en el que se encuentra la actividad cuando el desarrollador del proyecto se la asigna.

b) En progreso: Una vez que el desarrollador empieza a trabajar en la tarea, debe ponerla en este estado.

c) Terminada: Una vez que el desarrollador termina su actividad y ha comprobado que la terminó, debe ponerla en este estado.

d) Entregada: Una vez que el líder de proyecto ha comprobado el funcionamiento de la actividad, el desarrollador puede ponerla en este estado.

Esas son las funciones básicas que nuestro programa puede realizar, este puede tener muchas aplicaciones, ya que es algo con que podemos trabajar todo el tiempo. Nuestro programa trabaja con archivos, de esa manera puede acceder a poner todos los estados en los que puede estar nuestra tarea.

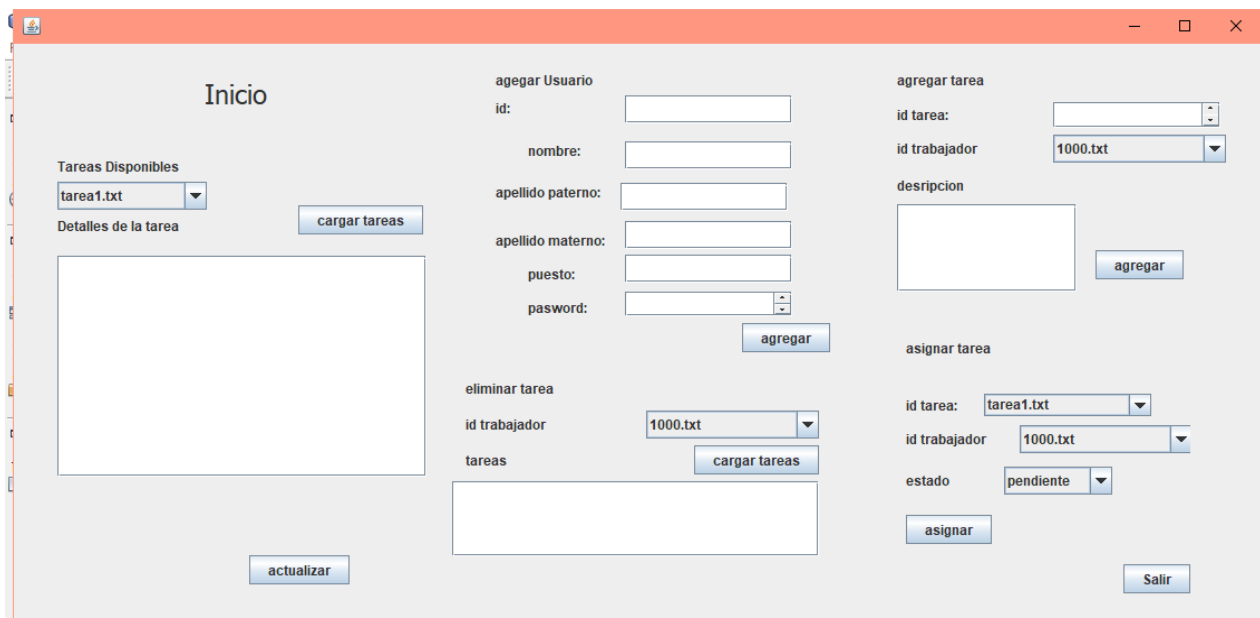
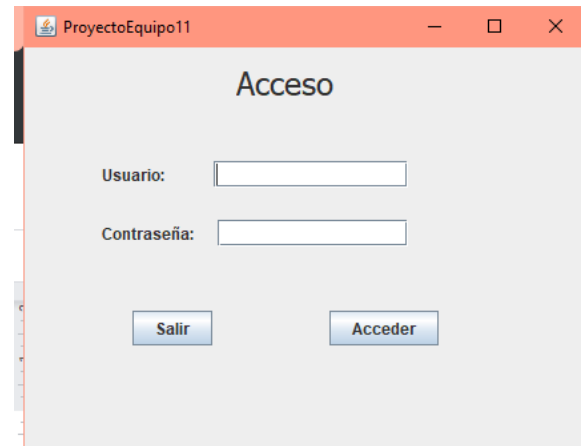
### 2. Instrucciones de uso

Dentro del programa primero tenemos una interfaz gráfica, la cual pedirá iniciar sesión para poder entrar a la aplicación. Para poder tener acceso a la plataforma, tenemos que los usuarios y contraseñas están

en la carpeta de usuarios a la cual podemos acceder. Por otro lado, como ejemplo tenemos un administrador con usuario 1000 y contraseña 111, al igual podemos tener a un trabajador con usuario 1010 y contraseña 6789. Dentro de la imagen podremos ver la primera interfaz, que solicita el usuario y la contraseña para poder acceder.

Al haber iniciado sesión dentro de nuestra aplicación, saldrá una siguiente interfaz, dependiendo de si quien ingreso es un administrador o un trabajador.

Suponiendo que ingresamos con un administrador, veríamos la siguiente interfaz:



Dentro de esta sección en la parte superior izquierda, tenemos las tareas disponibles y abajo tenemos un recuadro para poder ingresar detalles de la tarea, además tenemos los botones para actualizar, en dado caso de que hayamos hecho un cambio, también tenemos otro botón para cargar las tareas.

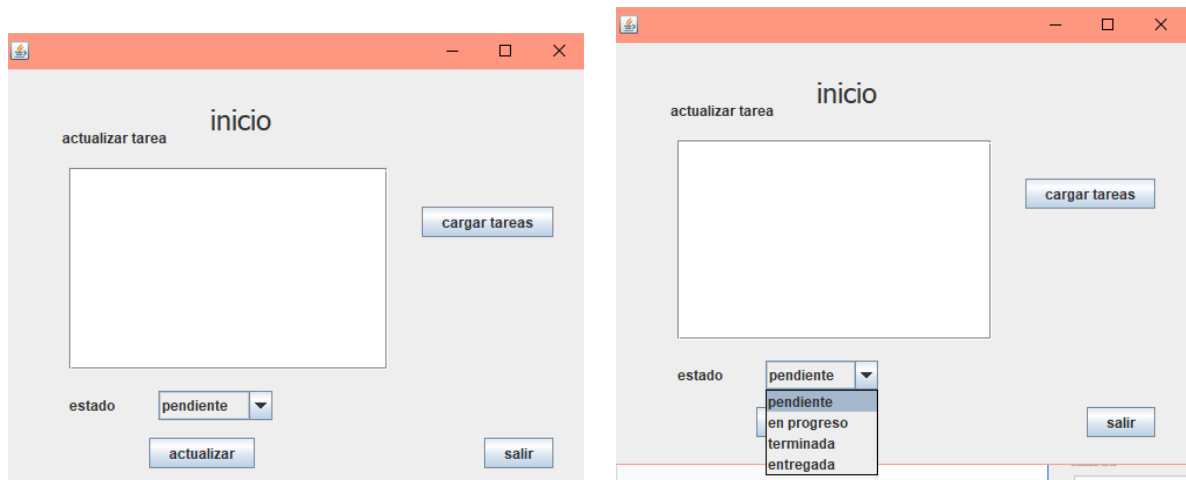
En la parte central de nuestra interfaz, tenemos una sección para agregar un usuario, donde antes de agregarlo, nos solicita unos datos

básicos para poder agregarlo, es necesario que todos estos campos estén llenos, en caso contrario, no se podrá agregar al usuario, además de que los únicos puestos disponibles son "worker" o "Admin"..

Igualmente, en la parte central inferior, tenemos una sección que es para eliminar las tareas, donde se solicita el id del trabajador para poder eliminar la tarea.

Finalmente, en la de la izquierda tenemos la sección en la cual el administrador puede agregar una tarea, se solicita el id de tarea y del trabajador, además, solicita una descripción, posteriormente tenemos la sección para asignarle la tarea al usuario, dentro de esta sección se le puede dar el estado a la tarea.

Por otro lado, si suponemos que se ingresó un trabajador, nos aparecería la siguiente interfaz:



Dentro de la interfaz de usuario tenemos un botón para cargar tareas, donde se mostrarían las tareas que tenemos, por otro lado tenemos una sección para actualizar las tareas, también se puede modificar el estado en el cual se encuentra la tarea. En dado caso que al apretar el botón de actualizar nos marque un error, simplemente es cuestión de volver a oprimir el botón, y nuestra tarea ya quedaría actualizada.

### 3. Firmas de las funciones usadas en el programa

Dentro de nuestro programa tenemos varias clases que son:

- ProyectoPOO.java

Dentro de esta clase, tenemos nuestra función main, dentro de esta función tenemos el acceso principal, el cual nos sirve para brindar la primera interfaz gráfica, que da paso a la segunda interfaz gráfica.

- archivos.java

- *public static ArrayList archivos(boolean dir)*

Dentro de esta función teníamos a todos los archivos y donde iban a estar, lo que necesitaba como parámetro es un booleano.

- *public static ArrayList<String>obtenerCreadenciales(String nombre, boolean caso)*

Esta función como su nombre lo dice es para obtener las Credenciales, o las identificaciones del usuario o administrador, como esta en la función necesitábamos el nombre de tipo string y el booleano que correspondía al caso.

- *public static ArrayList<String>obtenerRelacion(String nombre)*

Esto es para poder obtener la relación, como bien su nombre lo dice, para esta necesitamos un único parámetro que es el nombre.

- *public static void crearArchTarea(String id)*

Esta función está creada para crear el archivo de la tarea, para esto igual marca su ubicación, como parámetros necesitamos el id.

- *public static void crearArchRelacion((String id,String idTarea,String estado)*

En esta función se crea la relación con el archivo, necesitamos como parámetros el id, el id de la tarea y el estado de esta.

- *public static void setArchRelacion((String id)*

Función que trabaja sobre la relación con el archivo, para esta función, el único parámetro que necesitamos es el id.

- *public static void eliminarTarea(String id)*

Esta función, como bien su nombre lo dice es para eliminar la tarea que ya no necesitamos, lo único que necesitamos como parámetro es el id.

- *public static void asignarDescripcion(String id,String msj)*

Función creada para asignar la descripción a la tarea que designe el administrador.

- *public void crearWorker(String id)*

Función que crea al trabajador, para ello solo necesitamos su id.

- *public void llenarWorker(trabajador worker, String id)*

Esta función nos sirve para llenar los datos del trabajador, por lo que necesitamos el trabajador y su id.

- *public static void crearArchTarea(){*

este es un método con sobrecarga, lo que realiza es crear un archivo estadísticas.text en el directorio estadísticas

- *public static void setEstadisticas(int p, int pro, int ter, int ent)*

Lo que realiza este método es escribir el conteo de las tareas en un archivo en el directorio estadísticas.

- trabajador.java

- *public trabajador(String id,String Nombre, String ApeMat, String ApePat, String puesto, String password)*

Dentro de esta función se asigna y por lo tanto se necesitan como parámetros su id, nombre, puesto y contraseña.

- *public trabajador(String id, ArrayList <String> usersArch)*

Dentro de esta función se asignan los datos al trabajador, para esta función necesitamos el id, y la lista con los usuarios.

- *public void setTarea(String tarea)*

Función creada para asignar las tareas, necesitamos un parámetro de tipo string que nos dice la tarea.

- *public String getTarea()*

Función que retorna las tareas asignadas, para esta no necesitamos ningún parámetro.

- Tarea.java

- *public static String obtenerDescripcion(String id)*

Esta función está hecha para poder obtener la descripción de la tarea, lo único que necesitamos como parámetro es el id.

- *public static void estadisticas()*

Este método nos permite crear una lista con los nombres de los archivos relación, posteriormente se accede a cada uno de estos archivos para conocer el estado de cada tarea y generar un conteo de las tareas pendientes, en progreso, terminadas y entregadas. además se invoca al método crearArchivosTarea y sets Estadísticas.

- Admin.java

- *public Admin(String id, ArrayList<String> usersArch)*

Tenemos esta función la cual se encarga de darle los mismos parámetros ya que esta clase hereda de trabajador.

- *public Admin(String id, ArrayList<String> usersArch, trabajador worker)*

En esta función, que es una sobrecarga del método, en este caso tenemos que se pide un id, la lista, usersArchs y al trabajador, está hecha para darle los parámetros que le corresponden, ya que hereda de trabajador.

- *public void asignarTarea(trabajador worker, String tarea)*

Función hecha para que el usuario pueda asignar las tareas, necesitamos como parámetros un trabajador y un string que hace referencia a la tarea.



#### 4. Diagramas UML

*Diagrama de clases:*

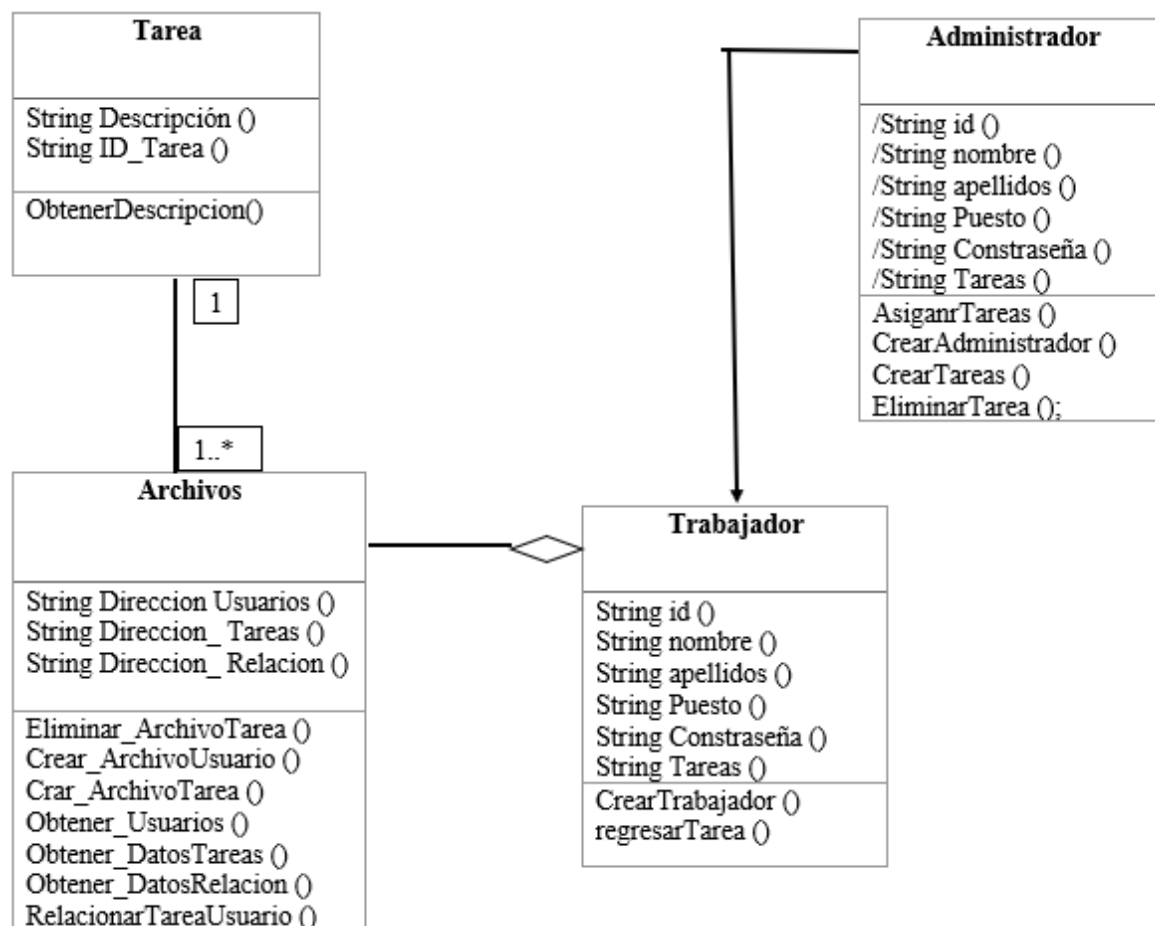


Diagrama de objetos;

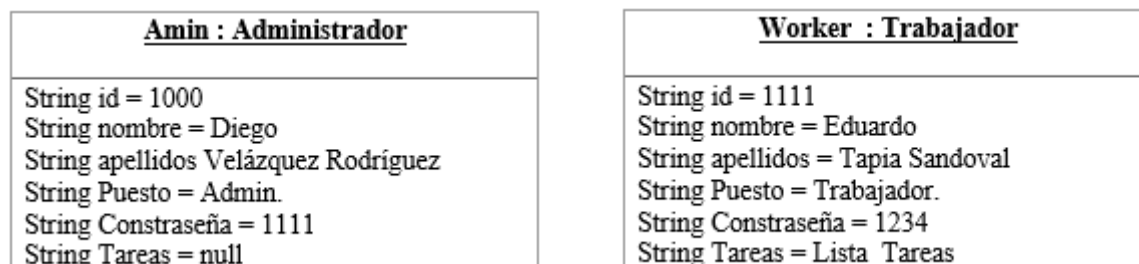


Diagrama de casos de uso;

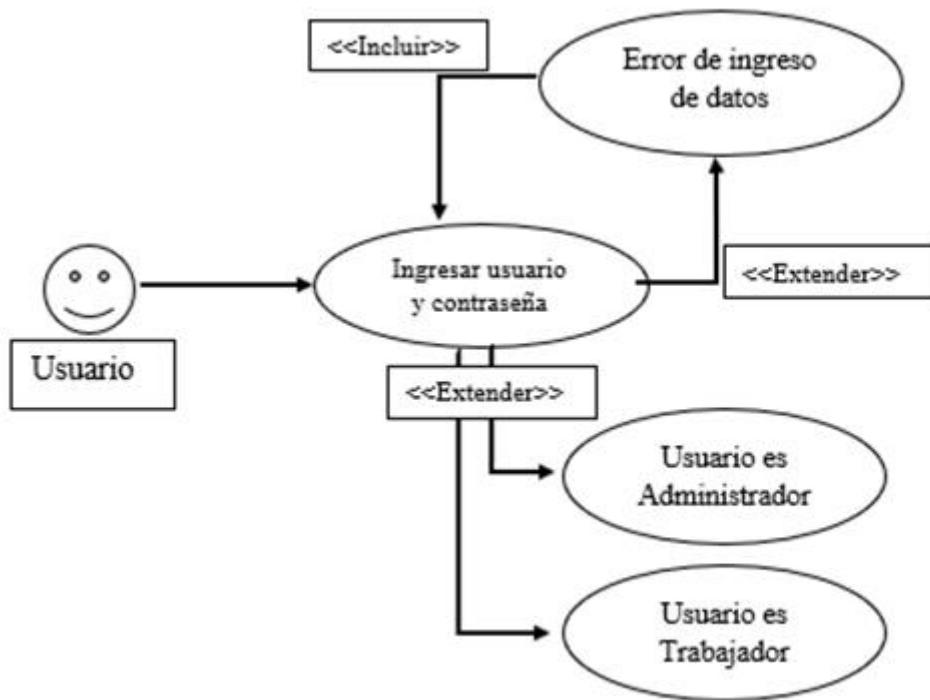


Diagrama de Estados;

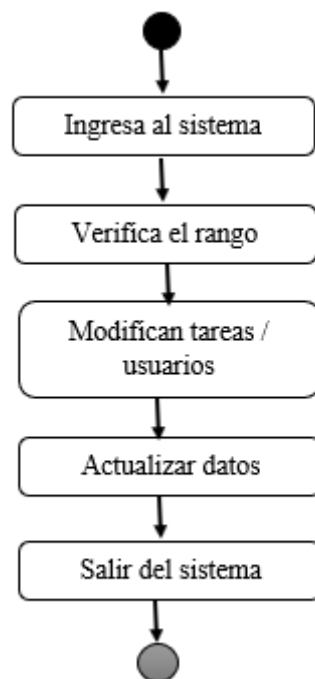
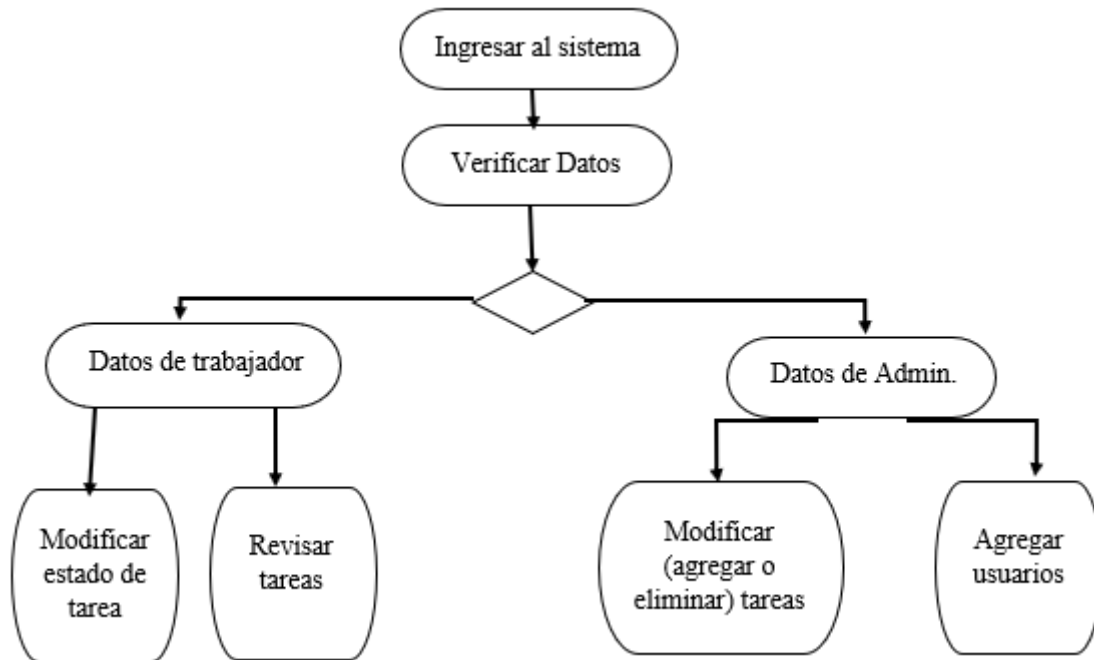


Diagrama de actividades;



## 5. Análisis de diseño

Para la solución de este problema se hace uso de archivos, dentro del directorio principal del proyecto se tienen tres diferentes carpetas para almacenar la información del programa.

La primer carpeta es "usuarios", en ella se sigue una metodología especial para localizar los archivos. El nombre del archivo, corresponde al id del trabajador, este elemento es una clave única para cada usuario. Se podría llevar un mejor control de estas claves con el uso de una base de datos, pero por la complejidad de implementación se omitió esta consideración.

Dentro de cada archivo de usuario, se encuentra la información relacionada, como nombre, apellido paterno, apellido materno, puesto y password. De esta manera solo, con el método "archivos" podemos obtener una lista con los nombres de los archivos.

La segunda carpeta se llama "tareas" en esta carpeta se sigue la misma metodología que en la carpeta anterior, el nombre del archivo corresponde al nombre de la tarea y dentro de este archivo, la descripción de la tarea.

Por último, la carpeta relacion, funciona igual que las anteriores, con la diferencia de que el nombre del archivo corresponde al nombre del

usuario con el cual se ha vinculado la tarea, esto es posible puesto que los archivos con nombres iguales se encuentran en diferentes directorios. Dentro de cada uno de los archivos se encuentra la relación de cada uno de los trabajadores con las tareas y el estado de estas.

Con este sistema optimizamos los procesos, puesto que solo se carga el archivo necesario para mostrar la información además de que las diferentes operaciones se simplifican.

Cuando un usuario administrador desee agregar o reasignar una tarea, solo se selecciona al usuario, internamente lo que se hace es únicamente modificar el archivo relación. Al eliminar la tarea, se elimina la relación y el archivo del directorio tareas.

## **6. Análisis de la solución**

Para este proyecto consideramos los casos en los cuales el programa era cargado desde diferentes computadores, por ello se debían de guardar los datos de las diferentes sesiones de trabajo, puesto que si dicha relación se realizaba mediante objetos, solo se guardaban en memoria principal y al terminar de ejecutar el programa los datos no se guardaban. Como solución a esto, se decidió implementar una carpeta donde se guardan todas las relaciones entre la tarea, el estado y el trabajador que la realiza.

En este caso, únicamente un trabajador puede tener una tarea, pero la tarea puede asignarse a varios trabajadores. Con esto en mente podemos decir que la complejidad algorítmica de nuestro programa es de  $O(n)$ , puesto que, cuando se cargan las relaciones de usuarios, relaciones de usuarios etc, sólo se utiliza un ciclo for para cargar los datos a los diferentes comboBox. Lo anterior se sostiene pensando en que el método `contains()` realiza la búsqueda con binary sort, pero este proceso solo se realiza un vez en el programa, cuando se carga la relación de usuarios registrados.

## 7. Implementación

Lo que realiza el programa es dado un directorio con archivos de usuario, es cargar la lista con los nombre de los archivos, que corresponde al id del trabajador, posteriormente, si encuentra al usuario, quiere decir que está registrado, por lo que se carga el archivo con sus datos, estos son almacenados en un arreglo con el método split, y posteriormente se crea el objeto con los tributos.

Después se comprueba que la contraseña ingresada coincida con la contraseña del trabajador. si es así se permite el acceso al sistema.

En el caso de un administrador, se despliega la interfaz donde podrá, agregar usuarios, crear tareas relacionadas a usuarios, asignar tareas y eliminarlas.

Los nombres de tareas y usuarios se cargan a diferentes combobox. en el caso de crear un usuario se deben de asignar todos los campos, para la creación de la tarea, se debe seleccionar el trabajador, el nombre de la tarea y la descripción de esta. en reasignación se selecciona el trabajador, la tarea y el estado con el que será reasignada. en eliminación se muestra los usuarios con tareas y el nombre y estado de la tarea asignada,

En el caso del trabajador sólo se puede visualizar la tarea y cambiar el estado de la misma.

La forma en cómo se leen los datos es muy controlada, se decidió implementar combo box para llevar un mejor control de los trabajadores y así evitar errores al asignar las tareas. la lectura de datos se realiza con cuadros de texto, las cuales retornan un string con la información ingresada.

Cada vez que se seleccionan botones se activan procesos creación de los archivos con la información recabada o volver a cargar la información a los combo box.

## COMENTARIOS

- Flores Cruz Karina Alejandra

Dentro de este programa siento que se lograron aplicar todos los temas vistos en el curso, dentro del curso, ya que desde las cosas más

básicas que hemos visto hasta las más complejas fueron aplicadas. Al igual dentro de este proyecto tuvimos que implementar búsquedas que hicimos para poder desarrollar en el trabajo con los archivos, de igual manera, fue algo que llevo mucho trabajo. El desarrollo del programa fue algo muy complejo, donde se tenía que estar al pendiente de muchas cosas, debido a que cualquier error en el código podría provocar que tuviéramos muchísimas cosas mal, eso fue algo desesperante, y que no me gusto mucho, porque nos podíamos confundir fácilmente. Dentro de la interfaz gráfica fue lo más fácil de hacer e ir creando el código poco a poco para llegar a algo más complejo fue algo muy impresionante.

Considero que ahora, con este código aprecio muchísimo más los programas que son más complejos o las aplicaciones, ya que lleva un gran trabajo poder desarrollarlos, por otro lado, considero que aplicamos todo lo visto dentro del curso en este proyecto para poder implementar el programa, fue algo muy laborioso, pero con orden se podía lograr, considero entonces que los objetivos del proyecto se cumplieron.

- Tapia Sandoval Eduardo Herminio

Este proyecto fue el cierre de lo aprendido a lo largo del curso, o por lo menos de la gran mayoría de conceptos, fue complicada la planeación y mucho más aún la realización, conforme avanzábamos en el desarrollo nos percatamos de fallas o casos que no consideramos en un inicio, la manera de solucionar esto no se presentaba como una decisión unánime, al contrario, cada uno de los miembros del equipo proponían diferentes maneras de atacar el problema, a mi parecer, este fue el mayor inconveniente que se nos presentó en el proyecto. Sin embargo, cuando la solución era planteada, llevarla a cabo para resolver los conflictos era un trabajo relativamente fácil. El problema principal del proyecto era "sencilla" pero en cuanto nos pusimos a desglosar todo lo que conlleva, nos dimos cuenta de que algo que aparenta facilidad, puede llegar a complicarse en extremo si no se planea bien la manera de trabajar, ahí es donde los diagramas toman suma importancia para el proyecto.

En sí, el proyecto representa lo que aprendimos a lo largo del curso, nos basamos en todo lo conocido para poder terminar el programa, aunque no aparecen conceptos como el paralelismo, considero que los pilares fundamentales de la POO si están presentes,

el lenguaje usado en el desarrollo fue java, el cual desarrollamos a lo largo de todo el curso, en proyectos como este, es donde se aprecia realmente la variedad de cosas las cuales puedes llevar a cabo con este lenguaje, la manera de trabajar del mismo e incluso, las limitaciones que posee.

Normalmente, no soy partidario de los proyectos desarrollados en equipo, pero este, me permitió ver una nueva perspectiva del potencial que posee el trabajo grupal, por la lluvia de ideas y conceptos que generamos y porque pude apreciar que cada programador tiene su manera de programar y que es complicado seguir al cien por ciento lo que otro programador intenta hacer, aun si ya conoces la idea. Por esta razón, consideró que el objetivo del proyecto se cumplió en general.

- Diego Velázquez Rodríguez

con la realización de este proyecto, pudimos poner en práctica los conceptos vistos en clase, como lo son la realización de instancias de objetos, diferentes modificadores de acceso, herencia, entre otras.

además de que pudimos trabajar son uno de los aspectos más interesantes de java, la implementación de interfaces gráficas, los cuales nos permiten dar una apariencia más profesional al trabajo.

Se presentaron diferentes problemas en la realización de este proyecto, puesto que se tienen que tener muchas consideraciones para la realización de este mismo. De cierta forma los diagramas UML sirvieron como una guía para el desarrollo pero aun así era difícil tener un panorama completo.

- Vázquez Méndez Enrique

Para este proyecto, a pesar de que en ocasiones resultó algo difícil su programación, pudimos aplicar nuestros conocimientos sobre Java y sus conceptos de Programación Orientada a Objetos, que adquirimos durante el curso. Al igual que también tuvimos que aplicar los cuatro pilares de la Programación Orientada a Objetos que, aunque parezca insignificante, resulta de suma importancia a la hora de implementar una correcta solución a un problema, sobre todo en nuestra vida profesional como Ingenieros.

Considero que este proyecto fue bastante completo, porque adicionalmente a los conceptos básicos acerca del lenguaje, también

tuvimos que aplicar lo aprendido sobre el manejo de archivos y la interfaz gráfica, que para nuestra suerte el IDE de Netbeans tiene una herramienta con la que resulta más fácil el programar esta parte, la cual con sólo arrastrar los elementos necesarios al área de trabajo, genera automáticamente el código asociado a ese elemento.