Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería Campus Tlaxcala

Proyecto Final

Software para gestión de empleados

Fundamentos de Programación

Dr. Esaú Escobar Juárez

Presenta Equipo 4:

Sergio de Jesús Castillo Molano

Miguel Alejandro Flores Sotelo

Juan Carlos Flores Mora

Oscar Ayala Elizalde

Índice

Índice de figuras

1. Introducción

1.1. Planteamiento del problema

En la actualidad optimizar las tareas cotidianas, nos ayuda ahorrar tiempo ya que es uno de los recursos mas importantes que existen en el campo empresarial, es por ello que se recreara un software para la gestión de empleados eficiente, para que el administrador pueda manipular datos de los empleados. El problema se aborda utilizando un conjunto de datos que contiene información sobre los empleados, como el nombre, el área de trabajo, la identificación del empleado, su sueldo, horas semanales a trabajar y su número de celular.
El manejo adecuado de esta información resulta crucial para optimizar los procesos internos, garantizar la correcta asignación de tareas, llevar un registro actualizado de la información de los empleados y facilitar la toma de decisiones basadas en datos.

1.2. Objetivos

Consiste en crear un software destinado para el administrador, que controle la gestión de empleados. En este software se puede realizar la carga de empleados, en el cual se visualizarán por medio de una lista, así como la búsqueda y comprobación de que algún empleado esté registrado dentro del sistema, como también se presenta la opción de dar de alta a empleados o dar de baja, dependiendo de lo que el administrador solicite, consultar los datos de cada empleado, o también poder modificarlos. Por otra parte, se encuentra la opción que el mismo empleado, con su ID y contraseña, tiene la posibilidad de observar sus datos correspondientes.
Además se busca aplicar lo aprendido durante el curso de fundamentos de programación, tratando de utilizar la mayoría de los conocimientos adquiridos a lo largo del semestre.

2. Análisis Realizado

2.1. Resolución del problema en el ámbito computacional

Trabajar con software dentro de las empresas ha sido una herramienta favorable, desde que la informática evolucionó para adaptarse a las necesidades empresariales. Ha ayudado a optimizar tareas rutinarias dentro de los diversos departamentos que existen en las instituciones, así como utilizar y aprovechar los datos para extraer información y formar estrategias para que las empresas aunmenten su rendimiento en todos los aspectos.

Estas razones fueron de gran impulso para el desarrollo de la tecnología en la computación, dando así el concepto de las bases de datos. Una base de datos es un conjunto de datos ordenados que están relacionados entre sí y almacenados sistemáticamente para su posterior uso. Las bases de datos permiten el almacenamiento de grandes cantidades de datos de forma organizada.

Los primeros avances que dieron inicio al desarrollo del concepto de base de datos fue la creación del tabulador de tarjetas perforadas, desarrollado por Herman Hollerith en 1884, con el objetivo de ayudar en el resumen de información y, posteriormente, a la contabilidad.

Figura 1: Ilustración de base de datos.
Nota. Adaptado de Breve historia del origen de las bases de datos, por Araujo J., 2017, Platzi (https://larepublica.es/wp-content/uploads/2017/02/bases-de-datos-1472639644731.jpg)

En la década de los 60 los ordenadores bajaron de precio con el fin de que las compañías las pudieran adquirir, esto dio paso a que se popularizara el uso de los discos duros (hecho que fue un buen adelanto para la época, ya que estos tienen información persistente, en otras palabras, que perdura en el tiempo). En esta misma época comenzó la primera

generación de bases de datos de red y las bases de datos jerárquicas, ya que abría la posibilidad de almacenar estructuras de datos en listas y árboles.

En la década de los 70, un científico de la informática, llamado Edgar Frank Codd aclaró el modelo relacional, en el cual sus elementos se organizan como un conjunto de tablas con columnas y filas[1]; y a la vez que publicó una serie de reglas para los sistemas de datos relacionales. Este hecho dio el nacimiento de la segunda generación de los Sistemas Gestores de Bases de Datos (SGBD).

En la década de los 90 surgió las bases de datos orientadas a objetos que tuvieron, incluso en la actualidad, bastante éxito en el momento de ejecutar datos complejos en los lugares donde las bases de datos relacionales no han podido desenvolverse con una manera eficaz. De esta forma se desarrollaron herramientas como Excel y Access (Figura 2). Así se creó la tercera generación de Sistemas Gestores de Bases de Datos.

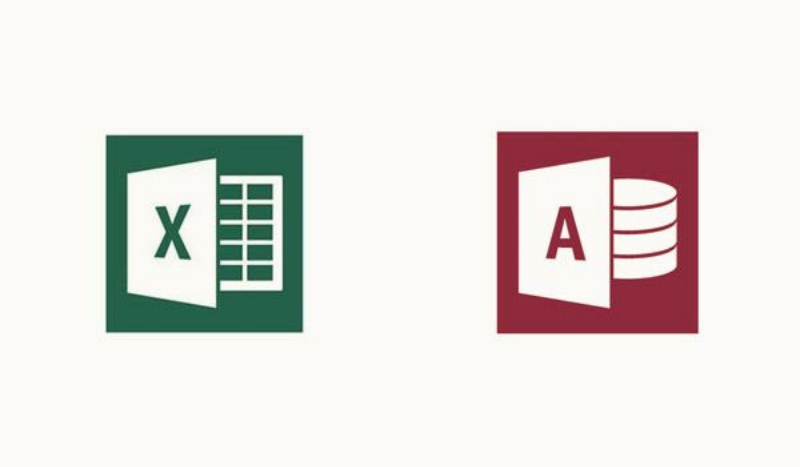

Figura 2: Logo de Excel y de Access de Microsoft.
Nota. Adaptado de Breve historia del origen de las bases de datos, por Araujo J., 2017, Platzi (https://static.comunicae.com/photos/notas/1188185/1498219039_Curso_Excel_Y_Access.jpg)

En la actualidad, las tres compañías que dominan el mercado de las Database (DB) son: IBM, Microsoft y Oracle. Sin embargo, existen otros fuertes competidores que se encargan de proveer excelentes sistemas, como MongoDb y PostgreSQL.[2]

[1]Oracle.(s.f). *¿Qué es una base de datos?* Oracle. Recuperado el 7 de junio de 2023 de https://www.oracle.com/mx/database/what-is-database/

[2]Araujo,J.(2017). *Breve historia del origen de las bases de datos.* Platzi. Recuperado el 7 de junio de 2023 de https://platzi.com/blog/historia-origen-bases-de-datos/

2.2. Ejemplos seleccionados para elaboración del proyecto final

Para la elaboración de este proyecto se consideró el modelo relacional de las bases de datos, concepto mencionado anteriormente. Esto se debe a que las bases de datos modernas requieren servidores y demás elementos para considerarse como tal, por lo que se ve un tanto limitado. Sin embargo, estas limitaciones no impiden el hecho de realizar la base de datos de manera local con el lenguaje de programación C, además de que se tomaron como ejemplo otros programas de gestión empresarial para desarrollar una idea de cómo realizarlo, los cuales son los siguientes:

1. **MongoDB.** MongoDB (del inglés humongous, "enorme") es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito en C++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico.[3]

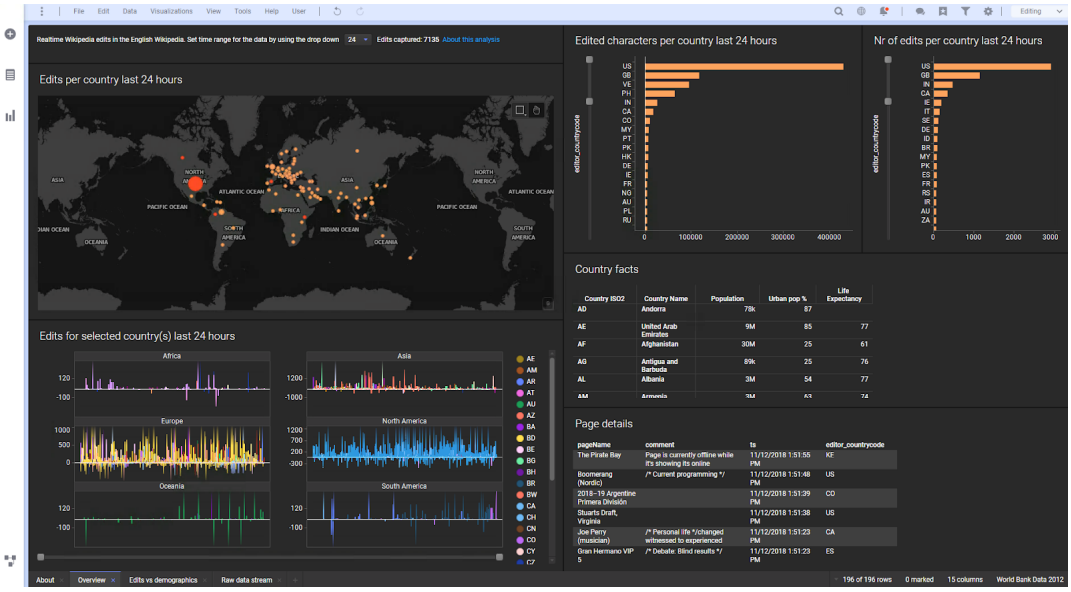

Figura 3: Interfaz de las gráficas en MongoDB
Nota. Adaptado de TIBCO Spotfire, por Capterra, (s.f), Capterra (https://lh5.googleusercontent.com/IxFqDGkrgTS__ugmBfhm08LdihRAhtfva3gYagrI2Q-lxQvn39nFCaL-A-IyNxUykt1eWXAFevCGsC1NMwrD-8GCdioyQJkP__0rRj08x-xZm__Bli2zM4ySBqHetGVofl86OMSMto1)

A partir de este sistema, sólo se tomó una característica, la cual, consiste en la graficación de datos respecto a un cierto aspecto o característica de los empleados

[3]Robledano,A.(2019, octubre 18). *Qué es MongoDB y características*. OpenWebinars. Recuperado el 8 de junio de 2023 de https://openwebinars.net/blog/que-es-mongodb/

(Figura 3), en este caso se eligió la cantidad de trabajadores que hay en todas las áreas de la empresa.

2. **MySQL.** MySQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad al estar basada en código abierto. MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle.[4]

   MySQL posee un software llamado "MySQL Workbench", el cual, se tomó como ejemplo para hacer el registro de los datos de los trabajadores (Figura 4). En este software se puede agregar datos de distintos tipos, ya sea enteros, cadenas o inclusive fechas, los cuales se ven reflejados en una tabla conforme se registren los datos.

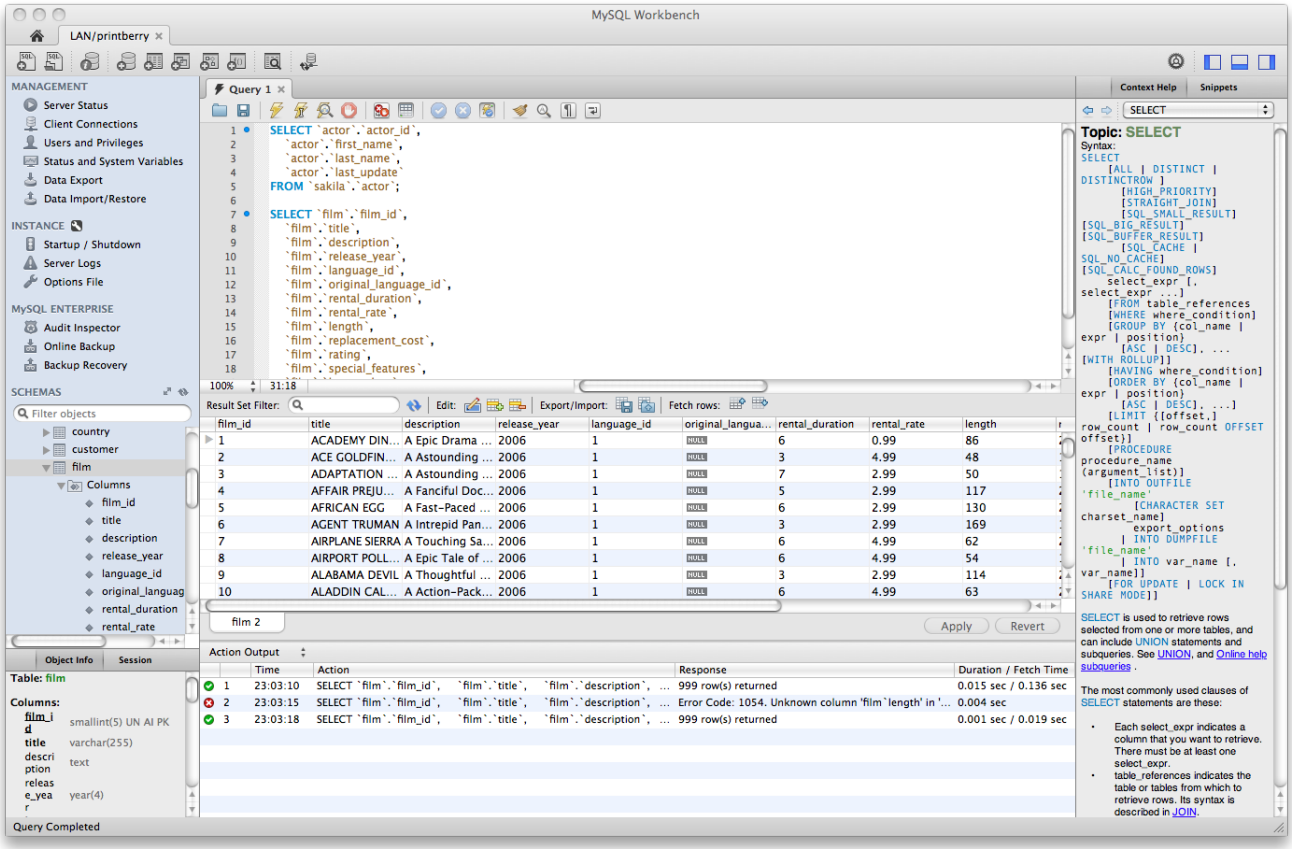

Figura 4: Interfaz gráfica de MySQL Workbench
Nota. [Imagen de MySQL Workbench], por Quora, (s.f), Quora (https://www.mysql.com/common/images/products/MySQL_Workbench_Editor_General_Mac.png)

[4]Robledano,A.(2019, septiembre 24). *Qué es MySQL: Características y ventajas*. OpenWebinars. Recuperado el 8 de junio de 2023 de https://openwebinars.net/blog/que-es-mysql/

2.3. Análisis de la abstracción y de la descomposición en este proyecto

Para realizar el proyecto se fragmentó en varias partes, esto con el fin de resolver cada uno de los aspectos del proyecto a una menor escala y sin aumentar la dificultad más de la que ya tiene, de modo que se divide de la siguiente manera:

1. **Interfaz empleado-administrador.** En este aspecto se determinó que tanto el administrador como el empleado pueden consultar esta base de datos, con la diferencia de que el empleado sólo puede consultar sus datos, mientras que el administrador podrá consultar los datos de todos los trabajadores, así como modificarlos. En ambos casos, para iniciar sesión se le solicita su contraseña, que para el caso del administrador es una contraseña que puede ser establecida o puede generar una nueva, y para el empleado es su número de identificación o ID.

2. **Datos de los trabajadores.** Dentro de este aspecto existen muchos datos que una base de datos puede recopilar y almacenar acerca de un empleado, por lo que se limitó toda esa variedad de datos a los más elementales, de modo que quedan los siguientes: nombre completo, área en la que está asignada, número de identificación o ID, salario que percibe mensualmente, horas que labora el trabajador a la semana y su número telefónico.
   Evidentemente, son pocos datos para ser una base de datos y que se puede agregar más información de distintos aspectos, por ejemplo, datos acerca de la salud de los empleados, el desempeño que tuvo cierto trabajador en cierto tiempo y demás aspectos, sin embargo, esto implica un manejo de información más grande que el que se estableció, así como el aumento de la complejidad para manejar toda esa información.

3. **Menú del administrador.** Una vez que el administrador ingrese su contraseña, se le dirige a un menú en el que le aparecerán distintas opciones por hacer, esto con el objetivo de manipular la información de la base de datos. En este menú se encuentran 6 opciones, los cuales se hablarán en los siguientes puntos.

   a) **Alta de empleados.** El administradror ingresa los datos correpondientes del empleado y se añade hasta el final de la lista.

   b) **Baja de empleados.** En esta opción, el administrador selecciona cuál trabajador desea dar de baja por medio de su ID.

c) **Modificación de datos.** En este apartado, el administrador busca al trabajador por medio de su ID y selecciona el dato que desea modificar.

d) **Mostrar todos los datos.** En esta opción, se muestra el registro de todos los trabajadores que están en la base de datos, desplegando los datos de cada uno de ellos. Además, muestra al final del listado una gráfica que señala cuántos trabajadores se encuentran en cada una de las áreas de la empresa (el cual, se hablará más adelante).

e) **Búsqueda de empleado.** El administrador puede buscar a un empleado en específico a través de su ID, simplemente despliega la información acerca de ese trabajador. Es una función muy similiar al del inciso anterior.

f) **Salir.** Cuando el administrador haya acabado de utilizar la base, esta opción cierra el programa.

4. **Gráfica.** La gráfica muestra el número de trabajadores que se encuentran en cada área. Esta función tiene un grado de complejidad un poco mayor, ya que los datos se ven afectados al momento de añadir, eliminiar o modificar la información de uno o varios trabajadores.

3. Requisitos del sistema

3.1. Hardware

Para ejecutar el programa debe contar con ciertas características:

1. El procesador debe ser al menos de 2Hz o superior para poder ejecutar el software (programa).
2. Memoria RAM: Se recomienda tener al menos 4GB de memoria RAM o superior, para garantizar un rendimiento óptimo.
3. Espacio de almacenamiento 120 SSD o Mecanico: Debe contar con suficiente espacio para el sistema operativo, el software de desarollo, la base de datos.

3.2. Software

1. Un sistema operativo compatible en este caso puede ser: Windows,Linux o macOS.
2. Un compilador en C, en este caso se usó DEV C.
3. Un editor de texto o un entorno de desarollo integrado, esto para escribir el código fuente del programa.

3.3. Otros recursos necesarios

1. Conocimientos previos.

4. Diseño

4.1. Diseño del algoritmo en pseudocódigo o diagrama de flujo

Antes de comenzar en el proceso de escribir cualquier código, es importante tener definido a dónde se desea llegar y saber cual es el mejor medio para cumplir el objetivo, por ello hemos creado un diagrama de flujo(figura 5.) en el que basamos nuestro proceso, el cual procedemos a explicar a continuación.

1. Inicio: Con una figura ovalada se muestra cual es el principio de un diagrama de flujo.

2. Mensaje de inicio: Impresión de mensaje de inicio del programa con el que el usuario tendrá interacción, esto se muestra con un paralelogramo, usado para impresión de datos.

3. ¿Empleado o gestor?: Se hace uso de una representación en la que se debe elegir el tipo de usuario del programa, en el que se define por empleado o administrador.

4. Inicio de sesión: Administrador debe registrar sus nuevos datos de acceso si es primer uso de sesión, lo cual se representa como procesamiento de información, o el administrador debe ingresar sus credenciales en caso de haber usado el programa antes.

5. Mostrar menú de administrador: Por medio de paralelogramos se muestran diferentes opciones a elegir: alta de empleado para registrar nuevos datos de empleados, baja de empleado para eliminar información de un empleado, lista de empleado para mostrar los empleados registrados, consultar datos para buscar y mostrar datos de un empleado en específico, modificar datos registrados de empleados y salir del programa.

6. Menú empleado: El apartado del diagrama para empleados limita las funciones del programa a solo solicitar ID de empleado y mostrar su información general.

7. Volver a mostrar menú: Después de hacer las operaciones solicitadas por el administrador o empleado, el programa vuelve mostrar el menú, esto se vuelve un ciclo hasta que el usuario seleccione la opción de salida.

8. Fin: Si el usuario elige la opción de salida, el programa se cerrará, esto también se representa con una figura ovalada.

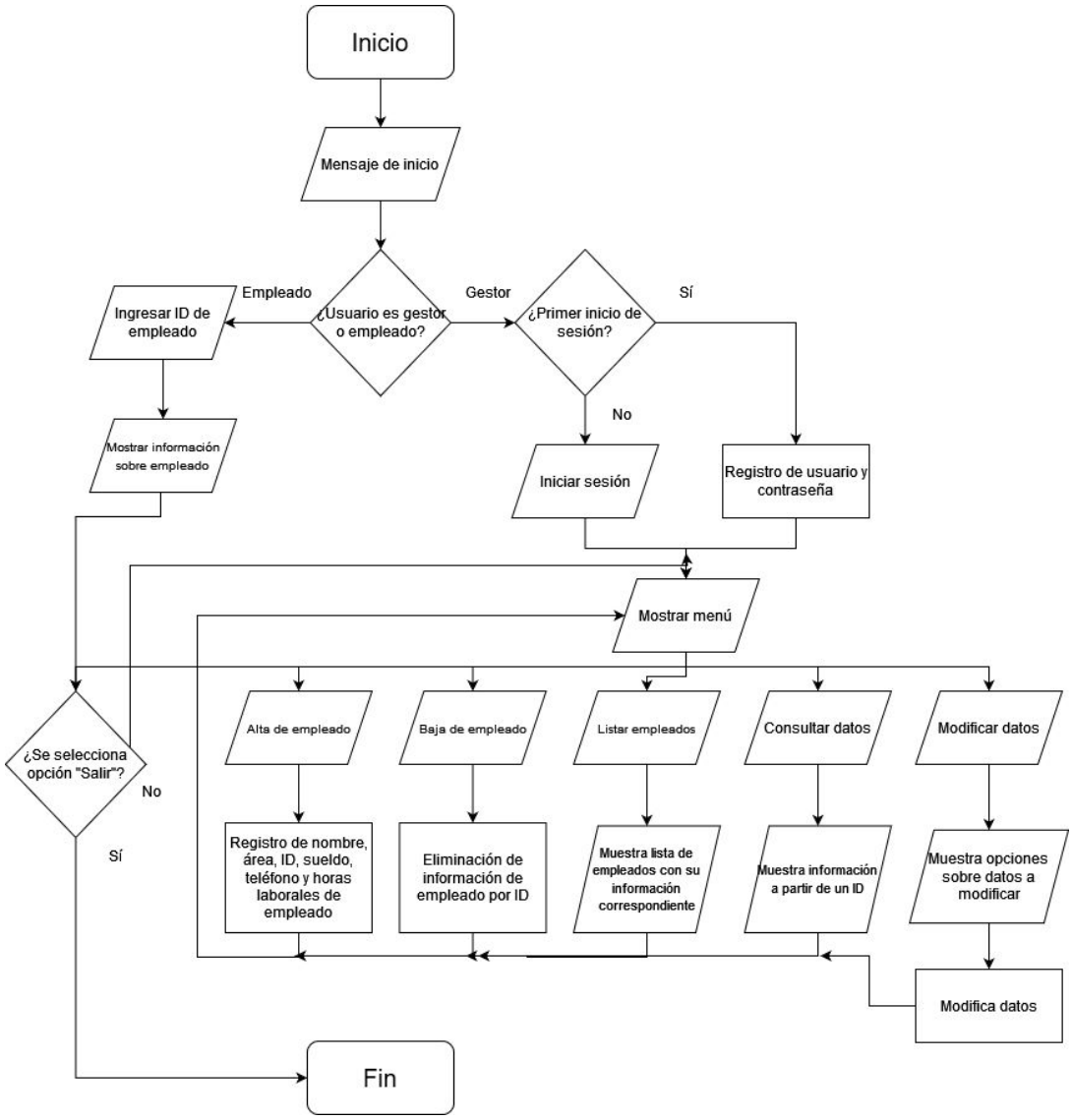

Figura 5: Diagrama de Flujo - Proyecto Final Softwork

4.2. Descripción de la interfaz con el usuario

Menú de opciones disponibles:

1. Cargar cargo de empleado: solicitar el ingreso del nombre del cargo y agregarlo a la lista de cargos existentes.

2. Mostrar listado de empleados: mostrar en pantalla un listado con los nombres, apellidos, cargos, puestos y sueldos de todos los empleados existentes.

3. Dar de alta empleado: solicitar el ingreso de los datos personales del empleado (nombre, apellido, fecha de nacimiento, etc.), su cargo, puesto y sueldo, y agregarlo a la lista de empleados existentes.

4. Dar de baja empleado: solicitar el ID del empleado a dar de baja, buscarlo en la lista de empleados y eliminarlo de la misma.

5. Ver datos de empleado por ID: solicitar el ID del empleado a buscar, buscarlo en la lista de empleados y mostrar sus datos personales, cargo, puesto y sueldo.

Pasos a seguir:

1. Esperar que el administrador seleccione una opción.

2. Según la opción seleccionada, realizar la operación correspondiente:

   - Cargar cargo de empleado: solicitar el ingreso del nombre del cargo y agregarlo a la lista de cargos existentes.
   - Mostrar listado de empleados: mostrar en pantalla un listado con los nombres, apellidos, cargos, puestos y sueldos de todos los empleados existentes.
   - Dar de alta empleado: solicitar el ingreso de los datos personales del empleado (nombre, apellido, fecha de nacimiento, etc.), su cargo, puesto y sueldo, y agregarlo a la lista de empleados existentes.
   - Dar de baja empleado: solicitar el ID del empleado a dar de baja, buscarlo en la lista de empleados y eliminarlo de la misma.
   - Ver datos de empleado por ID: solicitar el ID del empleado a buscar, buscarlo en la lista de empleados y mostrar sus datos personales, cargo, puesto y sueldo.

3. Confirmar el éxito o fracaso de la acción realizada y volver al menú principal.

El programa deberá contar con una estructura de datos que permita almacenar información de los empleados y los cargos existentes. Además, deberá manejar adecuadamente los casos en los que se ingresen datos incorrectos o acciones no permitidas.

Salidas:

- Confirmación del éxito o fracaso de la acción realizada.
- Listado de empleados con sus respectivos datos, según corresponda.
- Datos de un empleado en particular si se ingresa su ID para buscarlo.

5. Detalles del funcionamiento del programa

El programa de gestión de empleados sigue los siguientes pasos:

1. El programa se inicia y muestra un menú con las siguientes opciones disponibles:
   - Cargar cargo de empleado: Permite ingresar el nombre de un nuevo cargo y agregarlo a la lista de cargos existentes.
   - Mostrar listado de empleados: Muestra en pantalla los nombres, apellidos, cargos, puestos y sueldos de todos los empleados existentes.
   - Dar de alta empleado: Permite ingresar los datos personales (nombre, apellido, fecha de nacimiento, etc.), cargo, puesto y sueldo de un nuevo empleado y lo agrega a la lista de empleados existentes.
   - Dar de baja empleado: Permite ingresar el ID de un empleado y eliminarlo de la lista de empleados existentes.
   - Ver datos de empleado por ID: Permite ingresar el ID de un empleado y mostrar sus datos personales, cargo, puesto y sueldo.
2. El administrador selecciona una opción del menú ingresando el número correspondiente.
3. Según la opción seleccionada, se realiza la operación correspondiente:
   - Cargar cargo de empleado:
     - El programa solicita al administrador que ingrese el nombre del cargo a cargar.
     - Verifica que el nombre del cargo no esté vacío y que no exista previamente en la lista de cargos existentes.

- Si cumple con las condiciones, agrega el nombre del cargo a la lista de cargos existentes y muestra un mensaje de confirmación.
- Si no cumple con las condiciones, muestra un mensaje de error indicando el motivo (nombre vacío o ya existente).

- Mostrar listado de empleados:
  - El programa verifica si la lista de empleados está vacía.
  - Si la lista no está vacía, muestra en pantalla un listado con los nombres, apellidos, cargos, puestos y sueldos de todos los empleados existentes.
  - Si la lista está vacía, muestra un mensaje indicando que no hay empleados registrados.
- Dar de alta empleado:
  - El programa solicita al administrador que ingrese los datos personales del empleado, como nombre, apellido, fecha de nacimiento, etc.
  - Luego, se solicita el cargo, puesto y sueldo del empleado.
  - Verifica que los campos obligatorios no estén vacíos y que el cargo ingresado exista en la lista de cargos existentes.
  - Si cumple con las condiciones, crea un nuevo empleado con los datos ingresados y lo agrega a la lista de empleados existentes.
  - Muestra un mensaje de confirmación indicando que el empleado se dio de alta correctamente.
  - Si no cumple con las condiciones, muestra un mensaje de error indicando el motivo (campos vacíos o cargo inexistente).
- Dar de baja empleado:
  - El programa solicita al administrador que ingrese el ID del empleado que se desea dar de baja.
  - Verifica si el ID ingresado es válido y si existe en la lista de empleados.
  - Si el ID es válido y se encuentra en la lista, elimina al empleado de la lista de empleados existentes.
  - Muestra un mensaje de confirmación indicando que el empleado se dio de baja correctamente.
  - Si el ID no es válido o no se encuentra en la lista, muestra un mensaje de error indicando el motivo (ID inválido o no encontrado).
- Ver datos de empleado por ID:

- El programa solicita al administrador que ingrese el ID del empleado que se desea buscar.
- Verifica si el ID ingresado es válido y si existe en la lista de empleados.
- Si el ID es válido y se encuentra en la lista, muestra en pantalla los datos personales, cargo, puesto y sueldo del empleado.
- Si el ID no es válido o no se encuentra en la lista, muestra un mensaje indicando que no se encontraron datos para el ID especificado.

4. Después de realizar la operación seleccionada y mostrar el mensaje correspondiente, el programa vuelve al menú principal para que el administrador pueda seleccionar otra opción o salir del programa.

6. Pruebas o ejemplos de funcionamiento

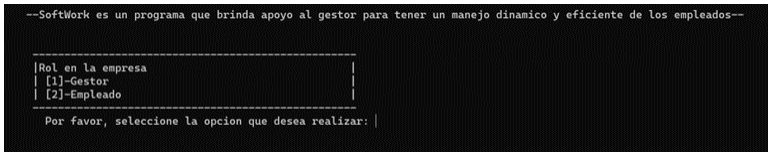

Figura 6: Inicio del programa

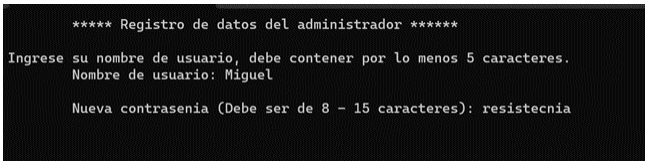

Figura 7: Registro de credenciales

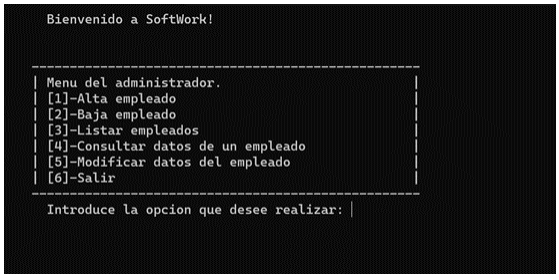

Figura 8: Menú de opciones del gestor

Figura 9: Listando emppleados

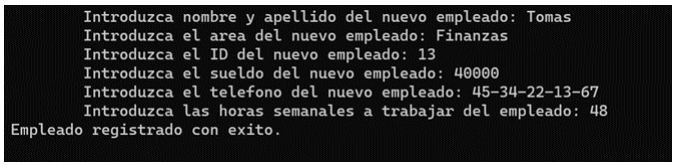

Figura 10: Registro de empleado

7. Conclusiones

En conclusión, el programa de gestión de empleados implementa una serie de funcionalidades para administrar y manipular información relacionada con los empleados de una organización. A través de un menú interactivo, el administrador puede realizar diversas operaciones como cargar nuevos cargos, dar de alta y baja empleados, mostrar listados de empleados y ver los datos completos de un empleado específico.

El programa utiliza una estructura de datos para almacenar la información de los empleados y los cargos existentes, lo que permite una gestión eficiente de los datos. Además, se realizan validaciones de los datos ingresados para asegurar su integridad y se brindan mensajes claros de confirmación o error al administrador.

Gracias a este programa, se simplifica y automatiza el proceso de administración de los empleados, facilitando tareas como la creación, modificación y eliminación de registros de sus datos, así como la visualización de información relevante. Esto puede contribuir a una gestión más eficiente de los recursos humanos de la organización y a un mejor seguimiento de la información de los empleados.

Referencias

[1] Araujo,J.(2017). *Breve historia del origen de las bases de datos*. Platzi. Recuperado el 7 de junio de 2023 de https://platzi.com/blog/historia-origen-bases-de-datos/

[2] Araujo,J.(2017). *Breve historia del origen de las bases de datos*. Platzi. Recuperado el 7 de junio de 2023 de https://larepublica.es/wp-content/uploads/2017/02/bases-de-datos-1472639644731.jpg

[3] Araujo,J.(2017). *Breve historia del origen de las bases de datos*. Platzi. Recuperado el 7 de junio de 2023 de https://static.comunicae.com/photos/notas/1188185/1498219039__Curso__Excel__Y__Access.jpg

[4] Capterra. (s.f). *TIBCO Spotfire*. Capterra. Recuperado el 8 de junio de 2023 de https://lh5.googleusercontent.com/IxFqDGkrgTS__ugmBfhm08LdihRAhtfva3gYagrI2Q-lxQvn39nFCaL-A-IyNxUykt1eWXAFevCGsC1NMwrD-8GCdioyQJkP__0rRj08x-xZm__Bli2zM4ySBqHetGVofl86OMSMto1

[5] Oracle.(s.f). *¿Qué es una base de datos?* Oracle. Recuperado el 7 de junio de 2023 de https://www.oracle.com/mx/database/what-is-database/

[6] Quora. (s.f). [Imagen de MySQL Workbench]. Quora. Recuperado el 8 de junio de 2023 de https://www.mysql.com/common/images/products/MySQL__Workbench__Editor__General__Mac

[7] Robledano,A.(2019, octubre 18). *Qué es MongoDB y características*. OpenWebinars. Recuperado el 8 de junio de 2023 de https://openwebinars.net/blog/que-es-mongodb/

[8] Robledano,A.(2019, septiembre 24). *Qué es MySQL: Características y ventajas*. OpenWebinars. Recuperado el 8 de junio de 2023 de https://openwebinars.net/blog/que-es-mysql/

8. Apéndice del código del programa

A continuación se muestra el código completo descrito en este documento:

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <string.h>

struct empleado{
    char nombre[200];
    char area[200];
    int id;
    float sueldo;
    char num_cel[200];
    int hrs_sem_trbj;
};
void mostrarmenu();
int cargarempleados(struct empleado empleados[100],int producc,int finan,int rechum,
void guardar_empleados(struct empleado empleados[100],int num_empleados);
void listar_empleados(struct empleado empleados [100],int num_empleados);
int alta_de_empleado(struct empleado empleados[100],int num_empleados,int producc,in
int baja_de_empleado(struct empleado empleados[100],int num_empleados,int producc,in
void consultar_empleado_gestor(struct empleado empleados[100],int num_empleados);
void consultar_empleado_empleado(struct empleado empleados[100],int num_empleados);
void modificar_area(struct empleado empleados[100],int num_empleados,int producc,int
void modificar_sueldo(struct empleado empleados [100],int num_empleados);
void modificar_num_cel(struct empleado empleados[100],int num_empleados);
void modificar_hrs_sem_trbj(struct empleado empleados [100],int num_empleados);
int comprobacion_existencia_empleado(struct empleado empleados[100], int num_emplead
int lecturaCredenciales(char *iniusuAdm, char *iniclaveAdm);
void inicioSesion(char *iniusuAdm, char *iniclaveAdm);
void graficadora(int grafica[50][49],int produccion,int finanzas,int rechumanos,int
int claveUsuario(void);

int main(){
    int elecc=0;
    int grafica[50][49]={0};
    int produccion=0,finanzas=0,rechumanos=0,ingenieria=0,gerencia=0,limpieza=0,tran
    int opc=0;
    int num_empleados=0;
    struct empleado empleados[100];
    num_empleados=cargarempleados(empleados,produccion,finanzas,rechumanos,ingenieri
```

```
int id_empleado;
int opc_modificacion_empleado=0;
char cntclave, iniusuAdm[20], iniclaveAdm[20];
FILE *sesion;

printf("␣␣␣␣␣--SoftWork␣es␣un␣programa␣que␣brinda␣apoyo␣al␣gestor␣para␣tener␣un␣
printf("\n␣␣␣␣␣␣-----------------------------------------------------");
printf("\n␣␣␣␣␣␣|Rol␣en␣la␣empresa\t\t\t\t|\n␣␣␣␣␣␣|\t[1]-Gestor\t\t\t\t\t|\n␣␣␣
printf("\n␣␣␣␣␣␣-----------------------------------------------------");
printf("\n\tPor␣favor,␣seleccione␣la␣opcion␣que␣desea␣realizar:␣");
scanf("%d",&elecc);
system("cls");

switch(elecc){
    case 1:
        sesion = fopen("credenciales.txt", "r");
        if (sesion == NULL){
            claveUsuario();
                    }
                    else {
                            fclose(sesion);
                            lecturaCredenciales(iniusuAdm, iniclaveAdm);
                            inicioSesion(iniusuAdm, iniclaveAdm);
                    }
        system("cls");
        printf("\tBienvenido␣a␣SoftWork!\n\n");
        mostrarmenu();
        scanf("%d",&opc);

        while(opc!=6){
            if(opc==1){
                system("cls");
                int creado=alta_de_empleado(empleados,num_empleados,produccion,f
                if(creado==1){
                    num_empleados++;
                }
            }

            else if(opc==2){
                system ("cls");
                int eliminado=baja_de_empleado(empleados,num_empleados,produccio
                if(eliminado==1){
                    num_empleados--;
                }
```

```
}

else if(opc==3){
    system("cls");
    listar_empleados(empleados,num_empleados);
    graficadora(grafica,produccion,finanzas,rechumanos,ingenieria,ge
}

else if(opc==4){
    system("cls");
   consultar_empleado_gestor(empleados,num_empleados);
}

else if(opc==5){
    system("cls");
    printf("\n␣␣␣␣␣␣-------------------------------------------------
    printf("␣␣␣␣␣␣|\t[1]-Area\t\t\t\t\t|\n␣␣␣␣␣␣|\t[2]-Sueldo\t\t\t\
    printf("␣␣␣␣␣␣-------------------------------------------------
    printf("\tIntroduce␣la␣opcion␣que␣desee␣realizar:␣");
    scanf("%d",&opc_modificacion_empleado);
    switch (opc_modificacion_empleado){
        case 1:
            modificar_area(empleados,num_empleados,produccion,finanz
            break;
        case 2:
            modificar_sueldo(empleados,num_empleados);
            break;
        case 3:
            modificar_num_cel(empleados,num_empleados);
            break;
        case 4:
            modificar_hrs_sem_trbj(empleados,num_empleados);
            break;
        default:
            printf("Opcion␣no␣valida.\n");
    }
}

else{
    printf("Opcion␣no␣valida.\n");
}
guardar_empleados(empleados,num_empleados);
printf("\n");
mostrarmenu();
```

```
                scanf("%d",&opc);
            }
            system("cls");
            printf("\tSoftWork␣le␣desea␣un␣excelente␣dia.␣Hasta␣pronto!\n");
            break;

        case 2:
            printf("Bienvenido␣a␣SoftWork!\n\n");
            consultar_empleado_empleado(empleados,num_empleados);
            break;

        default:
            printf("No␣existe␣la␣opcion␣ingresada\n");
            break;
      }
    return 0;
}

void mostrarmenu(){
    printf("\n␣␣␣␣␣␣----------------------------------------------------");
    printf("\n␣␣␣␣␣␣|␣Menu␣del␣administrador.\t\t\t\t|\n");
    printf("␣␣␣␣␣␣|\t[1]-Alta␣empleado\t\t\t\t|\n␣␣␣␣␣␣|\t[2]-Baja␣empleado\t\t\t\t|
    printf("␣␣␣␣␣␣----------------------------------------------------\n");
    printf("\tIntroduce␣la␣opcion␣que␣desee␣realizar:␣");
}
int cargarempleados(struct empleado empleados[100],int producc,int finan,int rechum,
    int num_empleados=0;
    FILE *f;
    f=fopen("empleados.txt","r");
    char cadena[250];
    char delimitador[]=";";

    while(feof(f)==0){
        struct empleado e;
        fgets(cadena,250,f);
        char *token= strtok(cadena,delimitador);

        if(token!=NULL){
            int campo=1;
            while (token!=NULL){
                if (campo==1){
                    strcpy(e.nombre,token);
                }
                else if(campo==2){
```

```
                strcpy(e.area,token);
                if(strcmp(e.area,"Produccion")==0){
                        producc++;
                    }
                    if(strcmp(e.area,"Finanzas")==0){
                        finan++;
                    }
                    if(strcmp(e.area,"Ingenieria")==0){
                        ingen++;
                    }
                    if(strcmp(e.area,"R. Humanos")==0){
                        rechum++;
                    }
                    if(strcmp(e.area,"Limpieza")==0){
                        limp++;
                    }
                    if(strcmp(e.area,"Gerencia")==0){
                        geren++;
                    }
                    if(strcmp(e.area,"Transporte")==0){
                        transp++;
                    }

            }
            else if(campo==3){
                e.id=atoi(token);
            }
            else if(campo==4){
                e.sueldo=atof(token);
            }
            else if(campo==5){
                strcpy(e.num_cel,token);
            }
            else if(campo==6){
                e.hrs_sem_trbj=atoi(token);
            }
            campo++;
            token=strtok(NULL,delimitador);
        }
        empleados[num_empleados]=e;
        num_empleados++;
    }
}
fclose(f);
```

```
    *produccion=producc;
   *finanzas=finan;
   *ingenieria=ingen;
   *rechumanos=rechum;
   *limpieza=limp;
   *gerencia=geren;
   *transporte=transp;
    return num_empleados;
}

void guardar_empleados(struct empleado empleados[100],int num_empleados){
    int i;
        FILE *f;
    f=fopen("empleados.txt","w");

    for(i=0;i<num_empleados-1;i++){
        fprintf(f,"%s;%s;%d;%.2f;%s;%d\n",empleados[i].nombre,empleados[i].area,empl
    }
    fprintf(f,"%s;%s;%d;%.2f;%s;%d",empleados[num_empleados-1].nombre,empleados[num_
    fclose(f);
}


void listar_empleados(struct empleado empleados[100], int num_empleados){
        int i;
    printf("LISTANDO␣LOS␣DATOS␣DEL␣EMPLEADO\n\n");
    for(i = 0; i < num_empleados; i++){
        printf("-------------------------------------------------------------\n");
        printf("|Nombre␣del␣empleado\t␣|␣%s\n",empleados[i].nombre);
        printf("-------------------------------------------------------------\n");
        printf("|Area␣de␣trabajo\t␣|␣%s\n",empleados[i].area);
        printf("-------------------------------------------------------------\n");
        printf("|ID␣del␣empleado\t␣|␣%d\n",empleados[i].id);
        printf("-------------------------------------------------------------\n");
        printf("|Sueldo␣percibido\t␣|␣%.2f\n",empleados[i].sueldo);
        printf("-------------------------------------------------------------\n");
        printf("|Numero␣de␣telefono␣\t␣|␣%s\n",empleados[i].num_cel);
        printf("-------------------------------------------------------------\n");
        printf("|Horas␣por␣semana\t␣|␣%d\n",empleados[i].hrs_sem_trbj);
        printf("-------------------------------------------------------------\n");
        printf("\n");
    }
}
```

```
int alta_de_empleado(struct empleado empleados[100],int num_empleados,int producc,in
    int creado=0;
    struct empleado nuevoemp;
    fflush(stdin);
    printf("\tIntroduzca␣nombre␣y␣apellido␣del␣nuevo␣empleado:␣");
    gets(nuevoemp.nombre);
    fflush(stdin);
    printf("\tIntroduzca␣el␣area␣del␣nuevo␣empleado:␣");
    gets(nuevoemp.area);
    if(strcmp(nuevoemp.area,"Produccion")==0)
        producc++;
    else if (strcmp(nuevoemp.area,"Finanzas")==0)
        finan++;
    else if (strcmp(nuevoemp.area,"Ingenieria")==0)
        ingen++;
    else if (strcmp(nuevoemp.area,"R.␣Humanos")==0)
        rechum++;
    else if (strcmp(nuevoemp.area,"Limpieza")==0)
        limp++;
    else if (strcmp(nuevoemp.area,"Gerencia")==0)
        geren++;
    else if (strcmp(nuevoemp.area,"Transporte")==0)
        transp++;
    fflush(stdin);
    printf("\tIntroduzca␣el␣ID␣del␣nuevo␣empleado:␣");
    scanf("%d",&nuevoemp.id);
    fflush(stdin);
    printf("\tIntroduzca␣el␣sueldo␣del␣nuevo␣empleado:␣");
    scanf("%f",&nuevoemp.sueldo);
    fflush(stdin);
    printf("\tIntroduzca␣el␣telefono␣del␣nuevo␣empleado:␣");
    gets(nuevoemp.num_cel);
    fflush(stdin);
    printf("\tIntroduzca␣las␣horas␣semanales␣a␣trabajar␣del␣empleado:␣");
    scanf("%d",&nuevoemp.hrs_sem_trbj);

    int existe=comprobacion_existencia_empleado(empleados,num_empleados,nuevoemp.id)
    if(num_empleados<100){
        if(existe==0){
            empleados[num_empleados]=nuevoemp;
            creado=1;
            printf("Empleado␣registrado␣con␣exito.\n");
        }
        else{
```

```
            printf("No se puede crear el empleado. El ID ingresado ya existe.\n");
        }
    }
    else{
        printf("No se puede dar de alta el empleado. Limite de registros alcanzado.\
    }
    *produccion=producc;
    *finanzas=finan;
    *ingenieria=ingen;
    *rechumanos=rechum;
    *limpieza=limp;
    *gerencia=geren;
    *transporte=transp;
    return creado;
}

int baja_de_empleado(struct empleado empleados[100],int num_empleados,int producc,in
    int eliminado=0;
    int id, i, j;
    int indice;
    printf("\tIntroduza el ID del empleado a dar de baja: ");
    scanf("%d",&id);
    int existe=comprobacion_existencia_empleado(empleados,num_empleados,id);
    if(existe==1){
        for(i=0;i<num_empleados;i++){
            if(empleados[i].id==id){
                indice=i;
            }
        }
        if(strcmp(empleados[indice].area,"Produccion")==0){
            producc--;
        }
        if(strcmp(empleados[indice].area,"Finanzas")==0){
            finan--;
        }
        if(strcmp(empleados[indice].area,"Ingenieria")==0){
            ingen--;
        }
        if(strcmp(empleados[indice].area,"R. Humanos")==0){
            rechum--;
        }
        if(strcmp(empleados[indice].area,"Limpieza")==0){
            limp--;
        }
```

```
        if(strcmp(empleados[indice].area,"Gerencia")==0){
            geren--;
        }
        if(strcmp(empleados[indice].area,"Transporte")==0){
            transp--;
        }
        *produccion=producc;
        *finanzas=finan;
        *ingenieria=ingen;
        *rechumanos=rechum;
        *limpieza=limp;
        *gerencia=geren;
        *transporte=transp;
        for(j=indice;j<num_empleados-1;j++){
            empleados[j]=empleados[j+1];
            struct empleado aux;
            empleados[j+1]=aux;
        }
        eliminado=1;
        printf("Empleado␣eliminado␣con␣exito.\n");

    }
    else{
        printf("No␣se␣puede␣dar␣de␣baja␣el␣␣empleado.␣No␣existe␣el␣ID␣ingresado.\n")
    }
    return eliminado;
}

void consultar_empleado_gestor(struct empleado empleados[100],int num_empleados){
        int id;
        int i;
        printf("Introduzca␣el␣ID␣del␣empleado␣a␣consultar␣porfavor:␣");
        scanf("%d",&id);
        int existe=comprobacion_existencia_empleado(empleados,num_empleados,id);
        if(existe==1){
                for(i=0;i<num_empleados;i++){
                                if(empleados[i].id==id){
                                printf("\nCargando␣los␣datos␣del␣empleado␣con␣ID␣%d\
                                printf("-----------------------------------------
                printf("|Nombre␣del␣empleado\t␣|␣%s\n",empleados[i].nombre);
                printf("-------------------------------------------------------\
                printf("|Area␣de␣trabajo\t␣|␣%s\n",empleados[i].area);
                printf("-------------------------------------------------------\
                printf("|ID␣del␣empleado\t␣|␣%d\n",empleados[i].id);
```

```
                printf("-------------------------------------------------------------\
                printf("|Sueldo percibido\t |  %.2f\n",empleados[i].sueldo);
                printf("-------------------------------------------------------------\
                printf("|Numero de telefono \t |  %s\n",empleados[i].num_cel);
                printf("-------------------------------------------------------------\
                printf("|Horas por semana\t |  %d\n",empleados[i].hrs_sem_trbj);
                printf("-------------------------------------------------------------\
                printf("\n");
        }
        }
                    }

        else{
            printf("ID no recnocido.EMPLEADO INEXISTENTE\n\n");
        }
}

void consultar_empleado_empleado(struct empleado empleados[100],int num_empleados){
        int id;
        int i;
        printf("Dame tu ID de la empresa porfavor: ");
        scanf("%d",&id);
        int existe=comprobacion_existencia_empleado(empleados,num_empleados,id);
        if(existe==1){
                for(i=0;i<num_empleados;i++){
                                if(empleados[i].id==id){
                                printf("\nCargando los datos del empleado con ID %d\
                                printf("-------------------------------------------
                printf("|Nombre del empleado\t |  %s\n",empleados[i].nombre);
                printf("-------------------------------------------------------------\
                printf("|Area de trabajo\t |  %s\n",empleados[i].area);
                printf("-------------------------------------------------------------\
                printf("|ID del empleado\t |  %d\n",empleados[i].id);
                printf("-------------------------------------------------------------\
                printf("|Sueldo percibido\t |  %.2f\n",empleados[i].sueldo);
                printf("-------------------------------------------------------------\
                printf("|Numero de telefono \t |  %s\n",empleados[i].num_cel);
                printf("-------------------------------------------------------------\
                printf("|Horas por semana\t |  %d\n",empleados[i].hrs_sem_trbj);
                printf("-------------------------------------------------------------\
                printf("\n");

        }
        }
```

```
                    }
        else{
            printf("\nID␣no␣recnocido.␣Este␣empleado␣no␣esta␣registrado\n\n");
        }
        printf("Cualquier␣duda␣o␣aclaracion,␣favor␣de␣comunicarse␣con␣el␣administrad
}

void modificar_area(struct empleado empleados[100],int num_empleados,int producc,int
    int id;
    int i;
    printf("\tIntroduzca␣el␣ID␣del␣empleado␣del␣cual␣quiere␣modificar␣su␣numero␣de␣c
    scanf("%d",&id);
    int existe=comprobacion_existencia_empleado(empleados,num_empleados,id);
    if(existe==1){
        char nueva_area[200];
        printf("\tIntroduzca␣la␣nueva␣area␣de␣trabajo␣del␣empleado:␣");
        fflush(stdin);
        gets(nueva_area);
        for(i=0;i<num_empleados;i++){
            if(empleados[i].id==id){
                if(strcmp(empleados[i].area,"Produccion")==0){
                    producc--;
                }
                if(strcmp(empleados[i].area,"Finanzas")==0){
                    finan--;
                }
                if(strcmp(empleados[i].area,"Ingenieria")==0){
                    ingen--;
                }
                if(strcmp(empleados[i].area,"R.␣Humanos")==0){
                    rechum--;
                }
                if(strcmp(empleados[i].area,"Limpieza")==0){
                    limp--;
                }
                if(strcmp(empleados[i].area,"Gerencia")==0){
                    geren--;
                }
                if(strcmp(empleados[i].area,"Transporte")==0){
                    transp--;
                }

                strcpy(empleados[i].area,nueva_area);
```

```
                if(strcmp(empleados[i].area,"Produccion")==0){
                    producc++;
                }
                if(strcmp(empleados[i].area,"Finanzas")==0){
                    finan++;
                }
                if(strcmp(empleados[i].area,"Ingenieria")==0){
                    ingen++;
                }
                if(strcmp(empleados[i].area,"R. Humanos")==0){
                    rechum++;
                }
                if(strcmp(empleados[i].area,"Limpieza")==0){
                    limp++;
                }
                if(strcmp(empleados[i].area,"Gerencia")==0){
                    geren++;
                }
                if(strcmp(empleados[i].area,"Transporte")==0){
                    transp++;
                }
                printf("Area de trabajo actualizada.\n");

                *produccion=producc;
                *finanzas=finan;
                *ingenieria=ingen;
                *rechumanos=rechum;
                *limpieza=limp;
                *gerencia=geren;
                *transporte=transp;
            }
        }
    }
    else{
        printf("ID no reconocido.EMPLEADO INEXISTENTE\n\n");
    }
}

void modificar_sueldo(struct empleado empleados [100],int num_empleados){
    int id;
    int i;
    printf("\tIntroduzca el ID del empleado del cual quiere modificar su sueldo: ");
    scanf("%d",&id);
    int existe=comprobacion_existencia_empleado(empleados,num_empleados,id);
```

```
    if(existe==1){
        float nuevo_sueldo;
        printf("\tIntroduzca el nuevo sueldo del empleado: ");
        scanf("%f",&nuevo_sueldo);
        for(i=0;i<num_empleados;i++){
            if(empleados[i].id==id){
                empleados[i].sueldo=nuevo_sueldo;
                printf("Sueldo actualizado.\n");
            }
        }
    }
    else{
        printf("ID no reconocido.No se reconoce al empleado.\n\n");
    }
}

void modificar_num_cel(struct empleado empleados[100],int num_empleados){
    int id;
    int i;
    printf("\tIntroduzca el ID del empleado del cual quiere modificar su numero de c
    scanf("%d",&id);
    int existe=comprobacion_existencia_empleado(empleados,num_empleados,id);
    if(existe==1){
        char nuevo_num_cel[200];
        printf("\tIntroduzca el nuevo numero de celular del empleado: ");
        fflush(stdin);
        gets(nuevo_num_cel);
        for(i=0;i<num_empleados;i++){
            if(empleados[i].id==id){
                strcpy(empleados[i].num_cel,nuevo_num_cel);
                printf("Numero de celular actualizado.\n");
            }
        }
    }
    else{
        printf("ID no reconocido.No se reconoce al empleado.\n\n");
    }
}

void modificar_hrs_sem_trbj(struct empleado empleados [100],int num_empleados){
    int id;
    int i;
    printf("\tIntroduzca el ID del empleado del cual quiere modificar su sueldo: ");
    scanf("%d",&id);
```

```
    int existe=comprobacion_existencia_empleado(empleados,num_empleados,id);
    if(existe==1){
        int nuevo_hrs_sem_trbj;
        printf("\tIntroduzca␣las␣nuevas␣horas␣semanales␣de␣trabajo␣:␣");
        scanf("%d",&nuevo_hrs_sem_trbj);
        for(i=0;i<num_empleados;i++){
            if(empleados[i].id==id){
                empleados[i].hrs_sem_trbj=nuevo_hrs_sem_trbj;
                printf("Horas␣de␣trabajo␣actualizadas.\n");
            }
        }
    }
    else{
        printf("ID␣no␣reconocido.␣No␣se␣reconoce␣al␣empleado.\n\n");
    }
}

int comprobacion_existencia_empleado(struct empleado empleados[100], int num_emplead
    int resultado=0;
    int i;
    for(i=0;i<num_empleados;i++){
        if(empleados[i].id==id){
            resultado=1;
        }
    }
    return resultado;
}

int claveUsuario(void){
    char usuAdm[20], claveAdm[20], claveAdmconf[20], iniusuAdm[20], iniclaveAdm[20];
        char nombre[20];
    int n, i, j, cc=0, lenUsu=0, lenClave=0, flagadmsesion=0;

    printf("\t*****␣Registro␣de␣datos␣del␣administrador␣******\n\nIngrese␣su␣nombre␣
        while (cc==0){
        printf("\n\tNombre␣de␣usuario:␣");
    scanf("%s", usuAdm);
                if (strlen(usuAdm)<5){
                        printf("\nNombre␣de␣usuario␣demasiado␣corto,␣debe␣ser␣de␣al␣
                printf("\nNombre␣de␣usuario:␣");
                scanf("%s", usuAdm);
                } else{
                        cc++;
                }
```

```
    }

cc = 0;
while (cc==0){
    printf("\n\tNueva␣contrasenia␣(Debe␣ser␣de␣8␣-␣15␣caracteres):␣");
    scanf("%s", claveAdm);
    n = strlen(claveAdm);
    while (lenClave==0){
        if (n<8){
            system("cls");
            printf("\nContrasenia␣demasiado␣corta,␣debe␣ser␣de␣8␣-␣15␣caracteres
            printf("\tNueva␣contrasenia:␣");
            scanf("%s", claveAdm);
            n = strlen(claveAdm);
        }
        else if (n>15){
            system("cls");
            printf("\nContrasenia␣demasiado␣larga,␣debe␣ser␣de␣8␣-␣15␣caracteres
            printf("\n\tNueva␣contrasenna:␣");
            scanf("%s", claveAdm);
            n = strlen(claveAdm);
        }
        else if (n>=8 && n<=15){
            lenClave++;
        }
    }
    printf("\tConfirmar␣constrasenia:␣");
    scanf("%s", claveAdmconf);
    for (i=0; i<n ;i++){
        if (claveAdm[i]!=claveAdmconf[i]){
            system("cls");
            printf("\nxx␣Contrasenia␣no␣es␣la␣misma␣xx");
            lenClave=0;
            break;
        }
    }
    if (i==n){
        cc++;
    }
}

FILE *archivo = fopen("credenciales.txt", "w");
if (archivo == NULL) {
    printf("No␣se␣pudo␣abrir␣el␣archivo.");
```

```
        return 1;
    }

    for (i=0; usuAdm[i]!='\0'; i++){
        char caracterCodificado = usuAdm[i] + 5;
        fputc(caracterCodificado, archivo);
    }

    fputc(';'+5, archivo);

    for (i=0; claveAdm[i]!='\0'; i++){
        char caracterCodificado = claveAdm[i] + 5;
        fputc(caracterCodificado, archivo);
    }

    fclose(archivo);
}

int lecturaCredenciales(char *iniusuAdm, char *iniclaveAdm){
        int caracterCodificado, i=0;
        char *token, cadena[100], caracterOriginal;
    FILE *archivo = fopen("credenciales.txt", "r");
    if (archivo == NULL) {
        printf("Error al abrir archivo");
        return 0;
    }
        while ((caracterCodificado = fgetc(archivo)) != EOF){
                caracterOriginal = caracterCodificado - 5;
                cadena[i++]=caracterOriginal;
        }
        token = strtok(cadena, ";");
        if (token != NULL){
                strcpy(iniusuAdm, token);
                token = strtok(NULL, ";");
                if (token != NULL){
                        strcpy(iniclaveAdm, token);
                }
        }

            fclose(archivo);
        return 0;
}

void inicioSesion(char *iniusuAdm, char *iniclaveAdm){
```

```
        int i, j, flagadmsesion=0;
        char usuAdm[50], claveAdm[50];

        system("cls");
    while (flagadmsesion==0){
        printf("\t--- Inicio de sesion ---\n");
                printf("\n\tNombre de usuario: ");
        scanf("%s", usuAdm);
        printf("\n\tConstrasenia: ");
                scanf("%s", claveAdm);
                for (j=0; j<strlen(usuAdm) ;j++){
                if (iniusuAdm[j]!=usuAdm[j]){
                    break;
                }
            }
                for (i=0; i<strlen(claveAdm) ;i++){
                        if (iniclaveAdm[i]!=claveAdm[i]){
                                break;
                }
            }
                if (i!=strlen(claveAdm)||j!=strlen(usuAdm)){
                        system("cls");
                        printf("Error: Nombre de usuario o contrasenia incorrecto\n\
            }
                else if (i==strlen(claveAdm) && j==strlen(usuAdm)){
                    flagadmsesion++;
            }
    }
}

void graficadora(int grafica[][49],int produccion,int finanzas,int rechumanos,int in
    int i, j;
        for(i=0;i<50;i++)
        for(j=0;j<49;j++)
            grafica[i][j]=0;
    for(i=49;i>=0;i--){
            for(j=1;j<49;j++){
                if(j%7!=0){
                    if(j>0 && j<7 && produccion!=0 && ((49-i)<produccion)){
                       grafica[i][j]=1;
                    }else if (j>7 && j<14 && finanzas!=0 && ((49-i)<finanzas)){
                       grafica[i][j]=1;
                    }else if (j>14 && j<21 && rechumanos!=0 && ((49-i)<rechumanos)){
                       grafica[i][j]=1;
```

```
                    }else if (j>21 && j<28 && ingenieria!=0 && ((49-i)<ingenieria)){
                        grafica[i][j]=1;
                    }else if (j>28 && j<35 && gerencia!=0 && ((49-i)<gerencia)){
                        grafica[i][j]=1;
                    }else if (j>35 && j<42 && limpieza!=0 && ((49-i)<limpieza)){
                        grafica[i][j]=1;
                    }else if (j>42 && j<49 && transporte!=0 && ((49-i)<transporte)){
                        grafica[i][j]=1;
                    }
                }
            }
        }
        for(i=0;i<50;i++){
            if(50-i>=10)
                printf("\n\t␣%d|\t",50-i);
            else
                printf("\n\t␣␣%d|\t",50-i);
            for(j=0;j<49;j++){
                if(grafica[i][j]!=0)
                    printf("%c",219);
                else
                    printf("␣");
            }
        }
    printf("\n\t␣␣␣␣----------------------------------------------------");
    printf("\n\t␣␣␣␣␣␣␣␣␣␣Prod.␣␣Fnzs.␣R.Hum.␣␣␣Ing.␣␣Grnc.␣␣Limp.␣␣Trans.\n");
    return;
}
```