PROYECTO FINAL

PROGRAMACIÓN EN PARALELO

Y DISTRIBUIDO

UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE NEGOCIOS CAMPUS IV

INGENIERÍA EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

PROGRAMACIÓN DISTRIBUIDA Y EN PARALELO

PROYECTO FINAL

INTEGRANTES

NOÉ GUILLEN GERARDO

JOSÉ EDUARDO OROZCO CARDENAS

SAMUEL SANCHEZ GUAZMAN

JEANNETTE SANCHEZ HERNANDEZ

EMILIA ZUÑIGA LOSADA

DOCENTE

MCC. VANESSA BENAVIDES GARCIA

GRADO Y GRUPO

4° “D”

TAPACHULA DE CÓRDOVA Y ORDOÑEZ, CHIAPAS A 08 DE MAYO DE 2022

INDICE

2. INTRODUCCIÓN

Considerando el problema al que se nos pidió dar solución podemos decir que, en la actualidad, el hospital ‘Sal vivo si puedes’ cuenta con una escasez sistema, por lo que en el presente proyecto se busca idear un sistema con la capacidad de llevar a cabo la administración de dicho hospital para atender a todo el personal y a los pacientes con una mejor calidad y control proporcionando a estos con información fácil de manejar.

A continuación, se verá cómo fue evolucionando el proyecto en cuanto a la aplicación de sistemas que no solo mejoren las condiciones de trabajo, sino que fomente y reduzca la carga laboral ofreciendo un mejor trabajo y calidad de servicio a los empleadores que asistan y que de cierta forma usen el programa.

Las técnicas ocupadas en el sistema permiten determinar los tiempos de comienzo y fin de cada una de las actividades que realizará el usuario, de forma que el proyecto se lleve a cabo en el menor tiempo posible y que la dificultad sea mínima al emplearlo.

Con este proyecto se pretende obtener más habilidades y destrezas del manejo, uso e implementación de las bases de datos conectadas a un lenguaje de programación para diseñar programas y aplicaciones para los usuarios y resolver la problemática del hospital, sin más que mencionar le invitamos a adentrarse a este proyecto de desarrollo del mundo de la programación, esperemos que sea interesante y de mucha ayuda para los usuarios.

3. SUSTENTO TEÓRICO

Durante la realización del proyecto fuimos utilizando diferentes herramientas que se nos han proporcionado hasta este punto de la carrera, así como algunas herramientas extras que a través de la investigación individual se han podido aprender y aplicar, los lenguajes nos proporcionan diversas funciones que podemos utilizar para resolver las problemáticas que se nos presenten, en la realización de este proyecto se utilizaron varios de esos elementos, por lo que a continuación se enlistan y se conceptualizan para mayor comprensión del proyecto.

3.1 GitHub

Consiste en una herramienta o servicio en la web que permite a los desarrolladores almacenar y administrar sus códigos en la nube, así como llevar un registro y control de los cambios que se realicen al mismo. Específicamente, es un sistema de control de versión distribuida, es decir, el código fuente es posible encontrarlo en la computadora de cualquier desarrollador esto permite un fácil acceso a las ramas y fusiones, es importante comprender los conceptos anteriores pues son base para entendedor el funcionamiento de github, es posible llevar un control del proyecto principal mediante una bifurcación, esto es denominado repositorio que se puede comprender como la rama master la cual contendrá todos los cambios realizados a través de las fusiones, estas son posibles a través de la creación de las ramas de todos los colaboradores del proyecto, por ello se dice que github es una herramienta de desarrollo colaborativo pues permite a más de un desarrollar hacer cambios al código pero usando su rama personal donde los cambios realizados no afectaran a la rama master [imagen 1]. Github posee una interfaz fácil de comprender donde especifica cada una de las acciones que se pueden realizar, una de ellas es ver cuando alguien ha hecho un commit, cuando se hayan realizado cambios en cualquiera de las ramas o solicitudes de extracción de código. Para fines del proyecto esta herramienta se implementó en Visual Studio Code donde proporciona herramientas para subir cambios o extraerlos ya sea de nuestra rama o las de los colaboradores.

3.2 Programación en paralelo (definición)

En los inicios de la programación paralela se encuentra Federico Luigi Menabrea y su “Bosquejo de la Máquina Analítica inventada por Charles Babbage”. De forma general la computación paralela es una forma de cómputo en la que se usan ya sea dos o más procesadores con el fin de poder resolver una tarea. Esta se basa principalmente en una técnica en el principio donde algunas de las tareas se pueden dividir en partes pequeñas, estas tareas se resuelven de forma simultánea y al finalizar la tarea estas se pueden unir y así complementar la tarea principal.

La computación paralela en la actualidad se ha convertido en el paradigma dominante al momento de fabricar procesadores, por lo anterior es fundamental conocer no solo las aplicaciones actuales de esta forma de cómputo, sino también la importancia que tendrá en el futuro y tener presente que este tipo de computación no siempre será como la estamos aprendiendo en estos momentos, sino que todo está susceptible a tener un cambio al paso de los años.

Este tipo de programación nos permite solucionar problemas que no caben en un solo procesador y que no se resuelven en un tiempo considerable, usando estos sistemas de cómputo es posible poder ejecutar problemas con mayor complejidad más rápidamente, esto implica menos tiempo y mayor rendimiento, así como una significativa reducción de gastos económicos.

El rendimiento de las computadoras tradicionales secuenciales está saturándose porque las aplicaciones de hoy en día necesitan realizar trabajo más complejo, por lo cual la solución es tener varios procesadores en sistemas paralelos, al así obtener ganancia de eficiencia, siempre y cuando los algoritmos se diseñen adecuadamente.

3.2.1 Características

- Memoria compartida: La memoria compartida es un modelo en el que los procesos paralelos comparten el mismo acceso al espacio de almacenamiento de datos del ordenador, anterior surge porque todos los procesos del sistema del ordenador están en un mismo nivel de acceso a la

lectura y escritura de los datos almacenados, este modelo facilita el intercambio eficiente de información entre los procesos.

- Interacción de procesos: se refiere a los diferentes mecanismos por los cuales procesos paralelos son capaces de comunicarse entre sí. Las formas más comunes de interacción son la memoria y el paso de mensajes compartidos, pero también puede darse de forma implícita.
- Paso de mensajes: esto permite que los diferentes procesos paralelos puedan intercambiar información pasándose mensajes de un equipo a otro, apoyando su capacidad de trabajar juntos para completar tareas. Estos mensajes pueden ser sincrónicos o asíncronos, donde uno se da al mismo momento y los segundos no.
- Interacción implícita: es una característica de la programación de un ordenador que la hace diferente a todos los demás paradigmas. Un programador puede escribir un programa de ordenador utilizando código únicamente paralelo que proporciona la estructura necesaria para que un programa se pueda comunicar dentro de sí mismo mientras realiza un procesamiento paralelo.
- Paralelismo de tareas: permite a un ordenador distribuir las tareas entre sus procesadores. Funciona ejecutando varias tareas al mismo tiempo utilizando los mismos datos, lo que proporciona la comunicación entre procesadores.
- Resuelve problemas que no se pueden resolver en un tiempo razonable
- Permite ejecutar problemas de un orden y complejidad mayor

3.3 Programación en lenguaje C

Este lenguaje fue desarrollado por Dennis M. Ritchie entre 1969 y 1972 en los laboratorios Bell, este lenguaje es un lenguaje orientado a implementación de los sistemas operativos concretamente Unix.

El lenguaje C es un lenguaje el cual está basado en el paradigma de programación estructurada con el cual se pueden desarrollar una gran variedad de aplicaciones. A la par que se trata de un lenguaje de tipos de datos estáticos, débilmente

tipificado, de medio nivel, pero con muchas características de bajo nivel, y dispone de muchas estructuras típicas de un alto nivel.

Características:

- Lenguaje flexible.
- Conjunto reducido de palabras reservadas.
- Punteros a funciones y variables estáticas.
- Acceso a memoria de bajo nivel mediante los punteros.
- Un núcleo simple.
- Sistema de tipos de datos que impide operaciones sin sentido.

Carencias:

- Soporte para la programación orientada a objetos.
- Funciones anidadas.
- Soporte nativo para programación multihilo.
- Recolección de basura nativa.

3.4 Diseño de base de datos en SQL

El diseño de base de datos en SQL es el diseño de base de datos es un proceso cuyo objetivo es definir la estructura adecuada para nuestro sistema de información. En este artículo analizaremos las fases principales del diseño de una base de datos y veremos los principios de diseño que deberemos seguir para obtener una base de datos bien estructurada y eficiente y que cumpla con los objetivos de nuestro proyecto.

El diseño de base de datos es un proceso fundamental a la hora de modelar nuestros conjuntos de datos y definir las operaciones que queremos realizar sobre ellos. Los datos son el activo más importante de una organización y una base de datos bien diseñada influye de forma directa en la eficiencia que obtendremos a la hora de almacenar, recuperar y analizar nuestros datos.

3.4.1 Diseño de un diagrama ER

Para diseñar un diagrama entidad-relación se requiere de un proceso de diseño para determinar la finalidad de nuestra base de datos, buscando y organizando la información necesaria reuniendo todos los tipos de información que se vaya a desear registrar dentro de nuestra base de datos, por poner un ejemplo, los nombres de los médicos o pacientes en una tabla.

Ahora explicando a profundidad lo que es un diseño de un diagrama entidad relación es aquel modelo de datos que se basa en una colección de herramientas conceptuales para la descripción, relación semántica y restricciones de consistencia que tendrán los datos, el modelo ER es un modelo de datos de alto nivel.

El modelo está basado en una percepción de lo que se tiene del mundo real que consiste en una recolección de objetos básicos que se denominan como entidades, y de relaciones entre estos nombrados objetos, a continuación, una breve imagen describiendo las entidades y sus relaciones con otros objetos**:** [Imagen 2].

En esta imagen podemos apreciar cómo están relacionados 2 entidades o también conocidas como entidades fuertes (Representado como rectángulos) entre sí, en este caso en una relación de uno a muchos entre las entidades y sus respectivos atributos (Representados como óvalos).

El diseño de entidad relación se desarrolló para facilitar el diseño de las bases de datos permitiéndonos la especificación de un esquema que representa la estructura lógica completa de una base de datos, el ER es uno de los modelos de datos semánticos, este aspecto yace en la representación del significado de los datos.

3.4.2 Base de datos en SQL

Las bases de datos SQL consisten en bases de datos relacionales que utilizan el llamado lenguaje SQL, es decir, el lenguaje de consulta estructurado, o llamado en inglés como Structured Query Language.

Una base de datos en SQL está formada por tablas con filas y columnas, en las filas figuran los diferentes registros de toda la tabla, mientras que en las columnas responden a los campos.

Ahora, mencionando que el lenguaje SQL es aquel que nos permite la creación de tipos de tablas relacionales, Dicho lenguaje cuenta con unas características que serán nombradas a continuación:

- Se trata de un lenguaje de definición de datos que contienen comandos que le permiten al usuario establecer, modificar o borrar esquemas de relación.
- SQL es un lenguaje interactivo que permite realizar consultas gracias al álgebra y el cálculo relacional.
- Este lenguaje de base de datos nos permite establecer restricciones de integridad a la información que se almacena en una base de datos.
- El lenguaje en SQL facilita la creación de bases de datos mediante tablas, las cuáles a su vez están formadas por registros (filas) y campos (columnas).
- El lenguaje SQL es compatible con otros lenguajes de programación como Java, C++ o PHP.

A continuación, se muestra cómo se escribe en el lenguaje SQL la creación de una base de datos con sus tablas que contienen sus respectivos campos que contendrán dicha tabla: [Imagen 3].

3.5 Administración de bases de datos en SQL

Estas son un conjunto de diferentes actividades que permiten que se pueda seguir manteniendo la continuidad o disponibilidad en el servicio que proporcionan las bases de datos, de esta forma se garantiza la seguridad de los datos y ayudar a tener una mejor organización de los datos. Estas tareas son realizadas por el encargado de la administración de las bases de datos de las que estén en su poder y a su alcance, esta es la persona responsable de instalar el software de la base de datos con mecanismos para hacer cumplir una política de seguridad para un site. Es posible instalar más de una base de datos en una misma máquina, este término hace referencia de forma más precisa, cualquier conjunto en específico de programas binarios y bases de datos instaladas. Entre las actividades que realiza un administrador de una base de datos, se encuentran el respaldo y recuperación, administración y autenticación de usuarios.

3.5.1 Consultas SQL

Las consultas son estructuras del lenguaje de base de datos las cuales nos permiten consultar y modificar la información que se encuentra almacenada en una base de datos. Estas nos permiten poder manejar grandes cantidades de datos, seleccionando únicamente las que se necesitan, de esta forma se puede consultar la información, crear, modificar, actualizar el contenido o bien añadiendo o eliminando información de cada tabla. Las consultas se manejan a través de combinación de comandos, consultas, operadores y funciones que son distintivas de este lenguaje, estas son algunas de las herramientas que nos ayudan a completar las consultas para el manejo de datos [Imagen 4].

Dentro de estas consultas podemos encontrar los tipos de comandos uno de ellos es el DDL, estos son los que permiten que se puedan crear nuevas bases de datos, añadir y eliminar elementos, a estos se les denomina como Data Definition Language o Lenguaje de Definición de Datos, en estos entra la creación, borrado completo y modificación de ciertos elementos de la base de datos, por otro lado están los DML, se denominan Data Manipulation Language o Lenguaje de Manipulación de Datos, estas son algunas operaciones básicas y las más comunes que se hacen con los datos que tienen las tablas de una base de datos,, estas también son llamadas CRUD donde se involucra Create, Read, Update and Delete. Decierta forma las consultas son las que ayudan a darle forma a los datos, siempre y cuando se tenga claro que es lo que se desea obtener de la base de datos y cuál será el manejo que se le dé.

3.5.2 Procedimientos almacenados

Los procedimientos almacenados son un grupo o conjunto de diferentes instrucciones, estas son compiladas con un único llamado, un ejemplo de cómo activarlas es esperando parámetros en el procedimiento almacenado, donde la momento de hacer la inserción del registro y haciendo todas las validaciones al llegar a la validación final se realizan más de una acción, estas pueden ser hacer inserción a una nueva tabla, borrar registros que tienen relación con el dato que se está ingresando, así mismo se pueden realizar operaciones dentro del

procedimiento, estas operaciones pueden ser sumas, restas entre otras almacenando los resultados en una variable así mismo se pueden agregar datos de forma automática sin tener que pedírselos al usuario. [Imagen 5]

Por lo anterior se puede observar que se pueden agrupar diferentes instrucciones que se deseen realizar en SQL y cualquier tipo de lógica que se quiera aplicar, cuando el procedimiento se crea y se compila se guarda automáticamente en la base de datos, no como un dato sino como el procedimiento que es y se puede volver a llamar las veces que sea necesarias, nos ayudan poder acceder de forma rápida a los registros sin la necesidad de conocer a fondo la estructura de la base de datos. Los procedimientos almacenados ofrecen ventajas importantes:

- Rendimiento: al ser ejecutados por el motor de base de datos ofrecen un rendimiento inmejorable ya que no es necesario transportar datos a ninguna parte. Cualquier proceso externo tiene una penalidad de tiempo adicional dada por el transporte de datos.
- Potencia: el lenguaje para procedimientos almacenados es muy potente. Permiten ejecutar operaciones complejas en pocos pasos ya que poseen un conjunto de instrucciones avanzado.
- Centralización: al formar parte de la base de datos los procedimientos almacenados están en un lugar centralizado y pueden ser ejecutados por cualquier aplicación que tenga acceso a la misma.

3.6 MPI (definición)

(Message Passing Interface) es una especificación para programación de comunicación o paso de mensajes entre procesos que proporciona una librería para el lenguaje de programación C y otros más. Para poder hacer uso de esta librería se debe de tener descargada esta misma y en el programa en C, se debe de agregar la librería, la cual es: “#include <mpi.h>”. [Imagen 6]

Características de MPI:

1. Estandarización
2. Amplia funcionalidad
3. Buenas prestaciones
4. Existencia de implementaciones libres.
5. Portabilidad.

Características básicas de la programación con MPI:

1. Iniciar, gestionar y finalizar procesos MPI.
2. Comunicar datos entre dos procesos.
3. Operaciones de comunicación entre grupos de procesos.
4. Crear tipos arbitrarios de datos.

3.6.1 Funciones de MPI

Cualquier programa que tenga MPI debe de tener unas cuantas funciones las cuales le indican al programa que se hará uso del MPI [Imagen 7], dichas funciones son las siguientes:

1. MPI_Init(): Esta función inicializa el entorno de ejecución e indica al sistema que se deben de hacer uso del MPI.
2. MPI_Finazlize(): Esta función finaliza el entorno de ejecución MPI.
3. MPI_Comm_rank(): Esta es la función que identifica los procesos uno de otro.
4. MPI_Comm_size(): Esta es la función que determina el número de procesos corriendo asociados al comunicador.
5. MPI_Wtime(): Guarda el tiempo en el cual se ejecuta un proceso, se debe de tener una variable de tiempo inicial y una variable de tiempo final.

3.7 Procesos

Los procesos(hijos) en este lenguaje de programación son algo bastante curioso, ya que literal es cualquier programa en ejecución y es totalmente independiente de otros procesos ya que cada uno de ellos puede tener diferente funcionalidad y en nuestro código podemos tener más de un proceso.

Algo indispensable que se debe de tener en cuenta es que es 100% requerido que para hacer uso de los procesos se debe de emplear la función “fork()” ya que esta

función es la que se encarga de crear los hijos, pero, una de las posibles o supuestas desventajas de hacer uso de los hijos es que al principio uno no tiene control sobre estos mismos y literal se ejecutan de manera no controlada y en la mayoría de los casos se ejecuta el padre antes que los hijos, y eso es un grave error que se debe evitar.

Para llegar a evitar ese tipo de situaciones se emplea la función “wait()”, esta función wait es la que hace que el padre espere a que todos los hijos se ejecuten, haciendo que el padre sea el último en ejecutarse. A todo esto, un proceso cuenta con varios estados, los cuales son los siguientes:

1. Listo para ejecutarse: En este estado el proceso o hijo está listo para que al llegar a cierta parte del código el usuario lo ejecute, claro, este mismo no sabe que eso acaba de ocurrir, ya que es algo que el usuario no observa, y al provocar la ejecución de este mismo es donde se pasa al siguiente estado.
2. En ejecución: en este estado se puede observar al proceso siendo ejecutado, y realizando los comandos que se le designó desde un principio, dada la decisión que el usuario llegue a tomar a la hora de que está interactuando con la ampliación o sistema es donde se verá a cuál de los siguientes estados se pasa.
3. Bloqueado: en este estado se entra cuando no hay un respuesta de parte del usuario o del propio sistema hacia el hijo, es decir; cuando el sistema no loe indica al hijo si debe de seguir ejecutándose o debe de morir(o en su caso se le debe de pasar el microprocesador a otro proceso), por ejemplo: cuando en una transacción, después de restarle el saldo a una cuenta(cobrar un cheque) no se le indica al sistema que si se hizo el cobro con éxitos le debe de dar un “commit transaction” y si no se hizo bien, se le debe decir al sistema que le dé un “rollback transaction”.
4. Zombi: este estado es el que se define como un proceso que ha sido ejecutado, pero aún no tiene entrada en la tabla de procesos, es decir, que el proceso se ha ejecutado y finalizado, pero todos los datos referentes a el y la memoria usada no han sido desreferenciados de este mismo.

3.8 Funciones

Las funciones en este lenguaje de programación son una de las cosas más importantes, ya que son estas mismas las que nos ayudan a llevar un mejor control sobre nuestro código ya que se crean bajo un principio: “Divide y vencerás”, lo que esto significa es que al dividir nuestro código en partes, podemos llegar identificar de manera más sencilla nuestros posibles errores, ya que por ejemplo, podemos tener cuatro funciones, las cuales llevan a cabo las operaciones matemáticas básicas(suma, resta, multiplicación y división) y a la hora de usar la función de multiplicación, nos da un resultado erróneo, pues muy fácil, buscamos la función de multiplicación de multiplicación y corregimos los errores [Imagen 8]. Sintaxis:

```
tipo nombre_funcion(entrada_de_parametros){
    //code
}
```

Donde:

Tipo: Esta parte de la función es la cual nos indica el tipo de retorno que tendrá nuestra función, los tipos de retornos que existen son dos, los cuales son:

1. Void: Este tipo de retorno es el que no devuelve nada y por ende no hay necesidad de agregar la palabra reservada return al final de la función.
2. Int: En este tipo de retorno, es lo contrario, ya que devuelve un entero, y el programador debe de saber cómo tomar este return, ya que se puede utilizar de diferentes formas, como, por ejemplo, una bandera, o dependiendo el resultado que el programador establezca, se puede evaluar ese retorno para saber si se hizo alguna operación o no.

3.9 Funciones de la librería Time

Es una librería de la biblioteca estándar del lenguaje de programación C. Nos permite el tratamiento, conversión y operaciones entre formatos de fecha y hora, por medio de funciones que contiene time.h. Esta librería cuenta con las siguientes funciones: asctime, clock, ctime, difftime, gmtime, localtime, mktime, strftime, time. A continuación, una tabla con el nombre y descripción de estas funciones.

Nombre	**Descripción**
char * asctime(struct tm *)	Recibe una variable de tipo puntero a estructura tm (struct tm*) y devuelve una cadena de caracteres cuyo formato es: "Www Mmm dd hh:mm:ss yyyy\n" (ej: Tue May 15 19:07:04 2008\n)
clock_t clock (void)	Devuelve el número de pulsos de reloj desde que se inició el proceso
char * ctime(time_t *)	Recibe una variable de tipo puntero a time_t (time t*) y devuelve una cadena con el mismo formato que asctime()
double difftime(time_t, time t)	Recibe dos variables de tipo time_t, calcula su diferencia y devuelve el resultado (double) expresado en segundos.
struct tm *gmtime(time_t *)	Recibe un puntero a una variable de tiempo (time_t*) y devuelve su conversión como fecha/hora UTC a struct tm a través de un puntero.

A esta estructura se le puede llamar tipo de dato, he aquí una tabla con ellos

Nombre	**Descripción**
clock_t	tipo de dato devuelto por clock(), generalmente un long int
time t	tipo de dato devuelto por time(), generalmente un long int
struct tm	representación del tiempo en formato de calendario (fecha/hora)

3.10 Estructuras condicionales

Las estructuras condicionales comparan una variable contra otros valores para que, en base al resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Es importante recordar que la condición debe dar un resultado booleano, por lo que lo más normal es usar operadores relacionales y condicionales. Las estructuras de control son las siguientes:

3.10.1 if

Esta estructura permite ejecutar un conjunto de sentencias en función del valor que tenga la expresión de comparación (se ejecuta si la expresión de comparación tiene valor true). Las llaves {} sirven para agrupar en un bloque las sentencias que se han de ejecutar, y no son necesarias si sólo hay una sentencia dentro del if [Imagen 9].

3.10.2 if-else

Análoga a la anterior, de la cual es una ampliación. Las sentencias incluidas en el else se ejecutan en el caso de no cumplirse la expresión de comparación (false), Permite introducir más de una expresión de comparación. Si la primera condición no se cumple, se compara la segunda y así sucesivamente. En el caso de que no se cumpla ninguna de las comparaciones se ejecutan las sentencias correspondientes al else. Se trata de una alternativa a la bifurcación if elseif else cuando se compara la misma expresión con distintos valores [Imagen 10].

3.10.3 Switch

Las características más relevantes de switch [Imagen 11] son las siguientes:

- Cada sentencia case se corresponde con un único valor de expression. No se pueden establecer rangos o condiciones, sino que se debe comparar con valores concretos. El ejemplo del Apartado 2.3.3.3 no se podría realizar utilizando switch.
- Los valores no comprendidos en ninguna sentencia casen se pueden gestionar en default, que es opcional.

- En ausencia de break, cuando se ejecuta una sentencia case se ejecutan también todas las case que van a continuación, hasta que se llega a un break o hasta que se termina el switch.

3.11 Estructuras repetitivas

Las estructuras repetitivas son las encargadas de repetir varias veces un conjunto de instrucciones de código de forma cíclica, es decir de manera repetitiva, en Java existen 3 tipos de ciclos.

3.11.1 For

El ciclo For es el que permite ejecutar iteraciones un número determinado de veces, empleándose principalmente cuando ya se tiene el conocimiento de cuantas veces queremos que se ejecute la instrucción, siendo alimentada por e elementos indispensables para su funcionamiento la inicialización, la terminación y por último el incremento [Imagen 12].

3.11.2 While

El ciclo While es aquel que ejecuta un conjunto de sentencias mientras se cumpla una determinada condición, en otras palabras, mientras la condición sea verdadera se seguirán ejecutando instrucciones de manera cíclica.

3.11.3 Do – While

El ciclo Do-while trabaja muy similar al ciclo While, lo que lo diferencia de este es que en este ciclo la condición se evaluara hasta el final de este, permitiendo que las sentencias se ejecuten de manera cíclica por lo menos una vez, a diferencia del While que puede que no se ejecuten ni una sola vez [Imagen 13].

3.12 Operaciones de cadenas (Strcpy, strtol, strcmp)

- Strcpy: La función strcpy funciona para copiar la cadena a la cual señala s2 (incluyendo el carácter nulo de terminación) al arreglo al cual señala s1. Si la copia se lleva a cabo entre objetos que se superponen, el comportamiento queda indefinido.

La función strcpy devuelve el valor de s1, en otras palabras, copia el valor de los parámetros de s2 en la variable s1 para así devolver una única cadena de texto, ya implementado en código en C se vería de la siguiente manera**:**

En el código se implementa la función strcpy para copiar los parámetros de la variable “ch” en la variable char o cadena llamada fecha.

- Strtol: La función strtol() convierte la parte inicial de la cadena de entrada nptr en un valor entero de tipo long de acuerdo a la base dada, que debe estar entre 2 y 36 ambos incluidos o ser el valor especial 0 [Imagen 14].

A continuación, un ejemplo de cómo se declara esta función:

long int strtol(const char *str, char **endptr, int base)

-Strcmp: Es una función cuya utilidad radica en comparar dos cadenas diferentes, para determinar si son iguales o si hay diferencia.

si las cadenas son iguales te devolverá un “0”, si la primera cadena es menor que la segunda devolverá un número negativo y finalmente si la primera cadena es mayor que la segunda devolverá un numero positivo. La sintaxis es la siguiente:

---strcmp(cadena1, cadena2);

3.13 Funciones de SQL en C

Estas funciones se pueden utilizar para consultar el estado de un objeto de conexión de base de datos existente mediante código en C, también sirven para insertar, modificar o eliminar datos en la base de datos.

Las siguientes funciones devuelven valores de parámetros establecidos en la conexión. Estos valores son fijos durante la vida de la conexión. Si se usa una cadena de conexión de varios hosts, los valores de PQhost, PQport y PQpass pueden cambiar si se establece una nueva conexión usando el mismo objeto.

3.13.1 PQdb

Devuelve el nombre de la base de datos de la conexión.

```
char *PQdb(const PGconn *conn);
```

3.13.2 PQuser

Devuelve el nombre de usuario de la conexión.

```
char *PQuser(const PGconn *conn);
```

3.13.3 PQpass

Devuelve la contraseña de la conexión.

```
char *PQpass(const PGconn *conn);
```

3.13.4 PQoptions

Devuelve las opciones de la línea de comandos pasadas en la solicitud de conexión.

```
char *PQoptions(const PGconn *conn);
```

Las siguientes funciones devuelven datos de estado que pueden cambiar a medida que se ejecutan operaciones en el objeto PGconn.

3.13.5PQstatus

Devuelve el estado de la conexión.

```
ConnStatusType PQstatus(const PGconn *conn);
```

El estado puede ser uno de varios valores. Sin embargo, solo dos de estos se ven fuera de un procedimiento de conexión asíncrona: CONNECTION_OK y CONNECTION_BAD.

3.13.6 PQtransactionStatus

Devuelve el estado actual de la transacción del servidor. PGTransactionStatusType PQtransactionStatus(const PGconn *conn);

El estado puede ser PQTRANS_IDLE (actualmente inactivo), PQTRANS_ACTIVE (un comando está en curso), PQTRANS_INTRANS (inactivo, en un bloque de transacción válido) o PQTRANS_INERROR (inactivo, en un bloque de transacción fallido). Se informa PQTRANS_UNKNOWN si la conexión es mala.

PQTRANS_ACTIVE se informa solo cuando se ha enviado una consulta al servidor y aún no se ha completado.

3.13.7 PQerrorMessage

Devuelve el mensaje de error generado más recientemente por una operación en la conexión.

char *PQerrorMessage(const PGconn *conn);

3.13.8 PQexec

Envía un comando al servidor y espera el resultado [Imagen 15]. PGresult *PQexec(PGconn *conn, const char *comando);

Devuelve un puntero PGresult o posiblemente un puntero nulo. Por lo general, se devolverá un puntero no nulo, excepto en condiciones de falta de memoria o errores graves, como la imposibilidad de enviar el comando al servidor. Si se devuelve un puntero nulo, debe tratarse como un resultado PGRES_FATAL_ERROR. Utilice PQerrorMessage para obtener más información sobre dichos errores.

3.13.9 PQresultStatus

Devuelve el estado del resultado del comando [Imagen 16]. ExecStatusType PQresultStatus(const PGresult *res); PQresultStatus puede devolver uno de los siguientes valores:

- PGRES_EMPTY_QUERY

  La cadena enviada al servidor estaba vacía.

- PGRES_COMMAND_OK

  Finalización exitosa de un comando que no devuelve datos.

- PGRES_TUPLES_OK

  Finalización exitosa de un comando que devuelve datos (como SELECCIONAR o MOSTRAR).

- PGRES_COPY_OUT

  Se inició la transferencia de datos de copia (desde el servidor).

- PGRES_COPY_IN

  Se inició la transferencia de datos Copiar en (al servidor).

- PGRES_BAD_RESPONSE

  La respuesta del servidor no se entendió.

- PGRES_NO FATAL_ERROR

  Ocurrió un error no fatal (un aviso o advertencia).

- PGRES_FATAL_ERROR

  Ocurrió un error fatal.

3.13.10 PQresStatus

Convierte el tipo enumerado devuelto por PQresultStatus en una constante de cadena que describe el código de estado. La persona que llama no debe liberar el resultado.

```
char *PQresStatus(ExecStatusType estado);
```

3.13.11 PQntuples

Devuelve el número de filas (tuplas) en el resultado de la consulta [Imagen 17].

```
int PQntuples(const PGresult *res);
```

3.13.12 PQnfields

Devuelve el número de columnas (campos) en cada fila del resultado de la consulta [Imagen 18].

```
int PQnfields(const PGresult *res);
```

3.13.13 PQfnombre

Devuelve el nombre de columna asociado con el número de columna dado. Los números de columna comienzan en 0. La persona que llama no debe liberar el resultado directamente. Se liberará cuando el identificador de PGresult asociado se pase a PQclear.

```
char *PQfname(const PResultado *res,
        int numero_columna);
```

Se devuelve NULL si el número de columna está fuera de rango.

3.13.14 PQgetResult

Espera el siguiente resultado de una llamada anterior de PQsendQuery, PQsendQueryParams, PQsendPrepare o PQsendQueryPrepared y lo devuelve. Se devuelve un puntero nulo cuando se completa el comando y no habrá más resultados.

```
PGresult *PQgetResult(PGconn *conn);
```

4.CONCLUSIÓN

Como conclusión al presente proyecto presentado, se buscó dar solución a la problemática planteada por el hospital “Sal vivo si puedes” en la cual se logró elaborar y plantear un programa solucionando el problema dado, para ello se realizaron investigaciones referentes a la estructura tanto del programa como de la base de datos implementada para ayudar a mejorar la eficiencia del programa.

Estas investigaciones llevadas al trabajo práctico de la codificación en base de datos se hicieron para tener un mejor control de usuarios y datos que se ingresaran en esta, esto a su vez se creó para la facilidad del usuario al emplear el programa, donde el usuario tiene la posibilidad de guardar, modificar, eliminar y ver todos los registros.

Es relevante mencionar la importancia de entender cómo funcionan y se aplican los procesos y funciones en un programa, estos son por tanto un conjunto de instrucciones que ejecutan una tarea determinada y que hemos encapsulado en un formato estándar para que nos sea muy sencillo de manipular y reutilizar, ejemplificando y haciendo más fácil su uso en el código fuente.

Según lo visto en el programa empleando la librería MPI ejemplificamos el paso de mensajes entre procesos para tener un registro y manipulación entre los diferentes grupos de procesos declarándose los almacenamientos del identificador de procesos y del número de procesos, así mismo el conteo del tiempo del menú principal y la impresión de todos los tiempos para el control del usuario hacia el programa y el correcto uso que le dará.

Por último, mencionar una característica importante del trabajo en equipo es establecer espacios de creatividad e innovación, que permita la participación activa y dinámica de los miembros del equipo, implementando un ambiente laboral de escucha mutua, sin tener en cuenta los niveles de jerarquía, valorando sin distinción todas las opiniones, facilitando la estructuración de todo el proyecto mutuo provocando más rendimiento en nuestras actividades.

5. REFERENCIAS

Cáceres, C. Q. (14 de Marzo de 2012). *usmp*. Obtenido de usmp: https://www.usmp.edu.pe/publicaciones/boletin/fia/info41/procedimiento.html

Dell, E. (08 de Julio de 2020). *historiadelaempresa*. Obtenido de historiadelaempresa: https://historiadelaempresa.com/programacion-paralela

Ortiz, J. (11 de Agosto de 2021). *teldat*. Obtenido de teldat: https://www.teldat.com/blog/es/computacion-paralela-capacidad-procesamiento/

UNAM. (8 de Noviembre de 2021). *programas.cuaed*. Obtenido de programas.cuaed: https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/992/mod_resource/content/2/contenido/index.html

UNIR. (17 de Julio de 2021). *unir*. Obtenido de unir: https://www.unir.net/marketing-comunicacion/revista/consultas-sql/

Desconocido. (2020). Funciones en Lenguaje C. Mayo 03, 2022, de officeappas Sitio web: https://view.officeapps.live.com/op/view.aspx?src=http%3A%2F%2Fwww.inf.udec.cl%2F~jlopez%2FFUNDPRO%2FLAMINASCurso%2F07_Funciones.ppt&wdOrigin=BROWSELINK

Desconocido. (2020). ¿Qué es el lenguaje de programación C?. Mayo 03, 2022, de W-ictea Sitio web: https://www.ictea.com/cs/index.php?rp=%2Fknowledgebase%2F8834%2FiQue-es-el-lenguaje-de-programacion-C.html

ayudaley. (Desconocido de Desconocido de 2018). *ayudaleyprotecciondatos.com*. Obtenido de ayudaleyprotecciondatos.com: https://ayudaleyprotecciondatos.es/bases-de-datos/sql/

Desconocido. (24 de 11 de 2005). *codigomaldito.blogspot.com*. Obtenido de codigomaldito.blogspot.com: http://codigomaldito.blogspot.com/2005/11/funciones-de-cstring-strcpy-y-strncpy.html

Desconocido. (Desconocido de Desconocido de 2017). *itpn.mx*. Obtenido de itpn.mx: http://itpn.mx/recursositics/5semestre/fundamentosderedes/Unidad%20II.pdf

Desconocido. (Desconocido de Desconocido de 2021). *solucioningenieril.com*. Obtenido de solucioningenieril.com: http://solucioningenieril.com/programacion_en_c/comparacion_de_cadenas_con_funcion_strcmp

Desconocido. (2007). Procesos e Hilos en C de Unix/Linux. Mayo 05, 2022, de chuidiang Sitio web: http://www.chuidiang.org/clinux/procesos/procesoshilos.php

Desconocido. (2005). Introducción a MPI. Mayo 05, 2022, de Informatica.uv Sitio web: http://informatica.uv.es/iiguia/ALP/materiales2005/2_2_introMPI.htm

6. ANEXOS

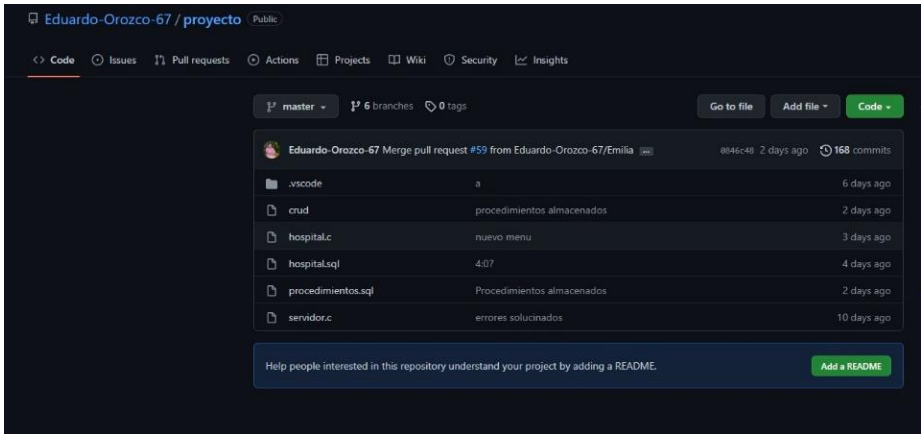


Imagen 1. Repositorio del proyecto en GitHub

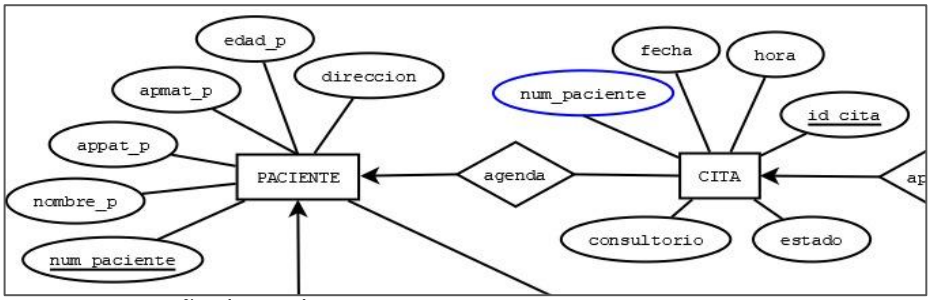


Imagen 2. Diseño de un diagrama ER

```
--creacion de la base
create database Hospital;

--Conectarse a la base de datos
psql Hospital

--cambiarse a la base datos
\c Hospital

--creacion de la tablas
create table Paciente
(
        num_paciente integer not NULL GENERATED ALWAYS AS IDENTITY,
        nombre_p varchar NOT NULL,
        appat_p varchar NOT NULL,
        apmat_p varchar NOT NULL,
        edad_p integer NOT NULL,
        direccion varchar NOT NULL,
        CONSTRAINT Paciente_pkey PRIMARY KEY(num_paciente)
);
```

Imagen 3. Diseño de una base de datos

```
sprintf(busced, "Select * from medico where cedula = '%s';", cedula);
ress = PQexec(bd, busced); // Ejecuta linea postgres
```

Imagen 4. Consulta básica de SQL.

```
CREATE PROCEDURE AgendarCita ( IN vpaciente INTEGER, IN vcon VARCHAR, IN vfecha DATE, IN vhora TIME, INOUT pRes INTEGER) AS $$
	DECLARE
		vpaciente alias for $1;
		vcon alias for $2;
		vfecha alias for $3;
		vhora alias for $4;
		vestado alias for $5;
		vreg record;
		vnuevo varchar;
		vban integer;
	BEGIN
		-- Selecciona el paciente
		Select into vreg * from Paciente where num_paciente = vpaciente for update;
		--Entra si el paciente existe
		IF FOUND THEN
			-- Si el paciente existe buscar la cita
		IF vfecha > now() THEN
		    --buscar una cita con ese id y que este activa
		    select into vreg * from cita where num_paciente = vpaciente and estado = 'activo' for update;
		    IF FOUND THEN
		        vban:= 0;
		        Rollback;
		    ELSE
		        --Realiza la insercion de la cita
		        vnuevo := 'activo';
		        Insert into Cita (num_paciente,consultorio,fecha,hora,estado) values ( vpaciente, vcon, vfecha, vhora, vnuevo);
```

Imagen 5. Parte inicial de un procedimiento almacenado

```
// tiempos de los procesos del menu principal y general
double tmpinic = 0.0; /*TIEMPO INICIO DE LA EJECUCION*/
double tmpfin;        /*TIEMPO FINAL DE LA EJECUCION*/
double tmp1s = 0.0;
double tmp1fs;
double tmp2s = 0.0;
double tmp2fs;
double tmp3m = 0.0;
double tmp3fm;
```

Imagen 6. Declarar los tiempos iniciales y finales del MPI

```
/*ALMACENAMOS EL IDENTIFICADOR DEL PROCESO*/
MPI_Comm_rank(MPI_COMM_WORLD, &id);

/*ALMACENAMOS EL NUMERO DE PROCESOS*/
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
```

Imagen 7. Algunas funciones del MPI

```
247  void GuardarCita()
248  {
249      int num_paciente, resp;
250      char consultorio[10], fechac[20], hora[20], nombre[20], appat[20], apmat[20], buspac[200], alta[200];
251
252      time_t t;
253      char fecha[100];
254      t = time(NULL);
255      char *ch;
256      ch = ctime(&t);
257      strcpy(fecha, ch);
258      // printf("%s",fecha);
259
```

Imagen 8. Parte inicial de un método void

```
if (fila == 0)
{
    printf("ATENCION: La tabla está vacía\n");
}
```

Imagen 9. Estructura condicional if

```
fila = PQntuples(resultado); //filas de la tabla
if (fila == 0)
{
    printf ("La tabla está vacía\n");
}
else
{
    columna = PQnfields(resultado); //Columnas de la tabla
```

Imagen 10. Estructura condicional if - else

```
switch (opcMod)
{
case 1:
    setbuf(stdin, NULL);
    printf("Introduzca el nuevo nombre del cliente:");
    fgets(nombre, 30, stdin);
    setbuf(stdin, NULL);
```

Imagen 11. Parte inicial de un switch

```
for (i = 0; i < fila; i++)
{
    printf("----------
    for (j = 0; j < columna; j++)
    {
```

Imagen 12. Estructura repetitiva for

```
do
{
    printf("\t\t
    printf("\t\t
    printf("\nBienvenido al menu de secretarios\n");
    printf("\n 1.- Catalogos \n 2.- procesos \n 3.- Reportes \n 4.- Salir \n");
    printf("Elija una opcion: ");
    scanf("%i", &opcS);

    } while (opcS != 4); // Fin del ciclo secretaria
  }
```

Imagen 13. Estructura repetitiva do - while

```
prespl = strtol(PQgetvalue(resultado, 0, 0), NULL, 10);
```

Imagen 14. Operaciones en cadena: Strtol

```
resultado = PQexec(bd, alta); // Ejecuta linea postgres
```

Imagen 15. Funciones de SLQ en C

```
if (PQresultStatus(resultado) == PGRES_COMMAND_OK)
{
```

Imagen 16. Funciones de SLQ en C

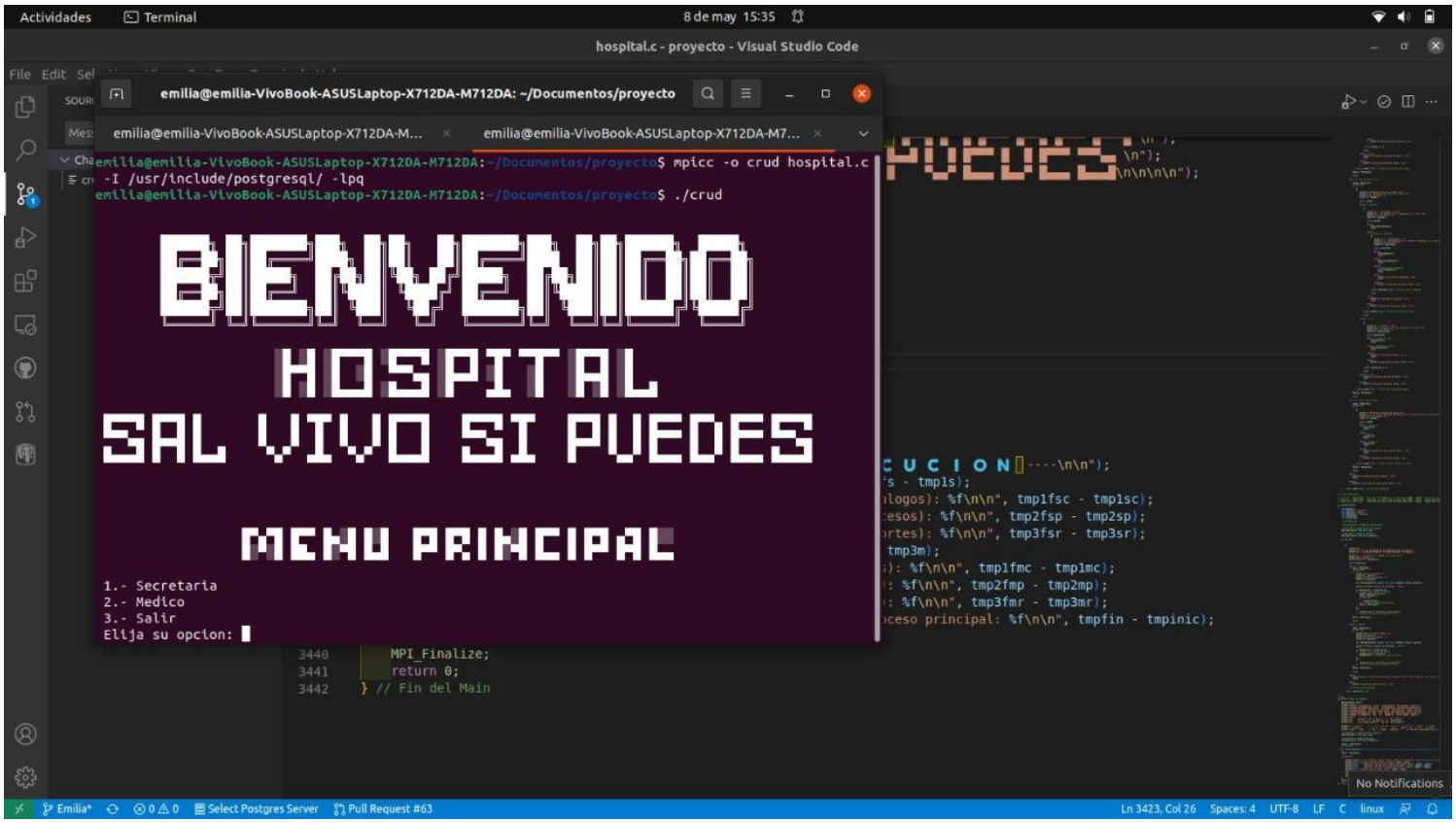
```
aFila = PQntuples(resultado);
```

Imagen 17. Funciones de SLQ en C

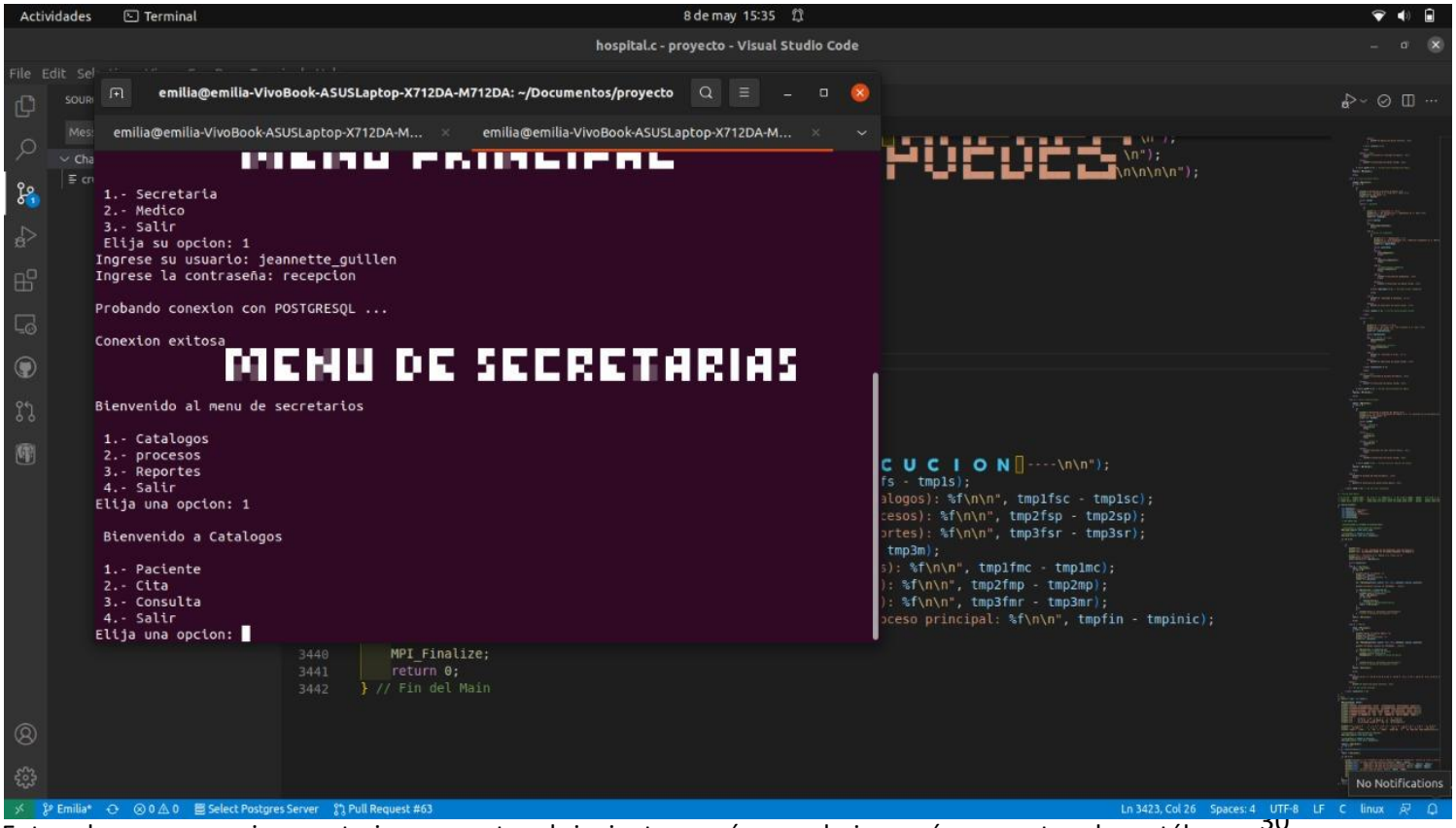
```
aColumna = PQnfields(resultado);
```

Imagen 18. Funciones de SLQ en C

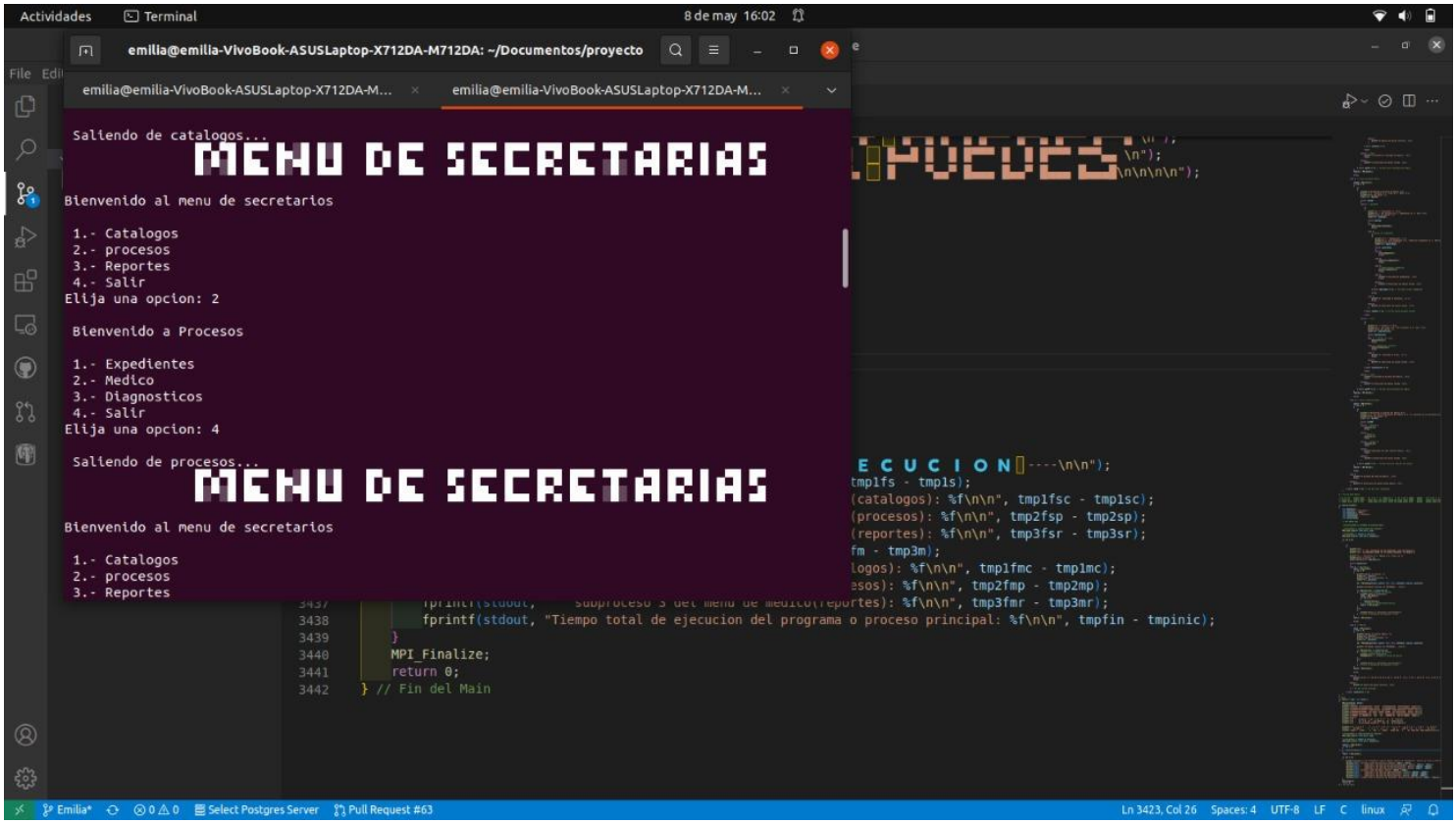
7. CAPTURAS DEL CÓDIGO EN EJECUCIÓN



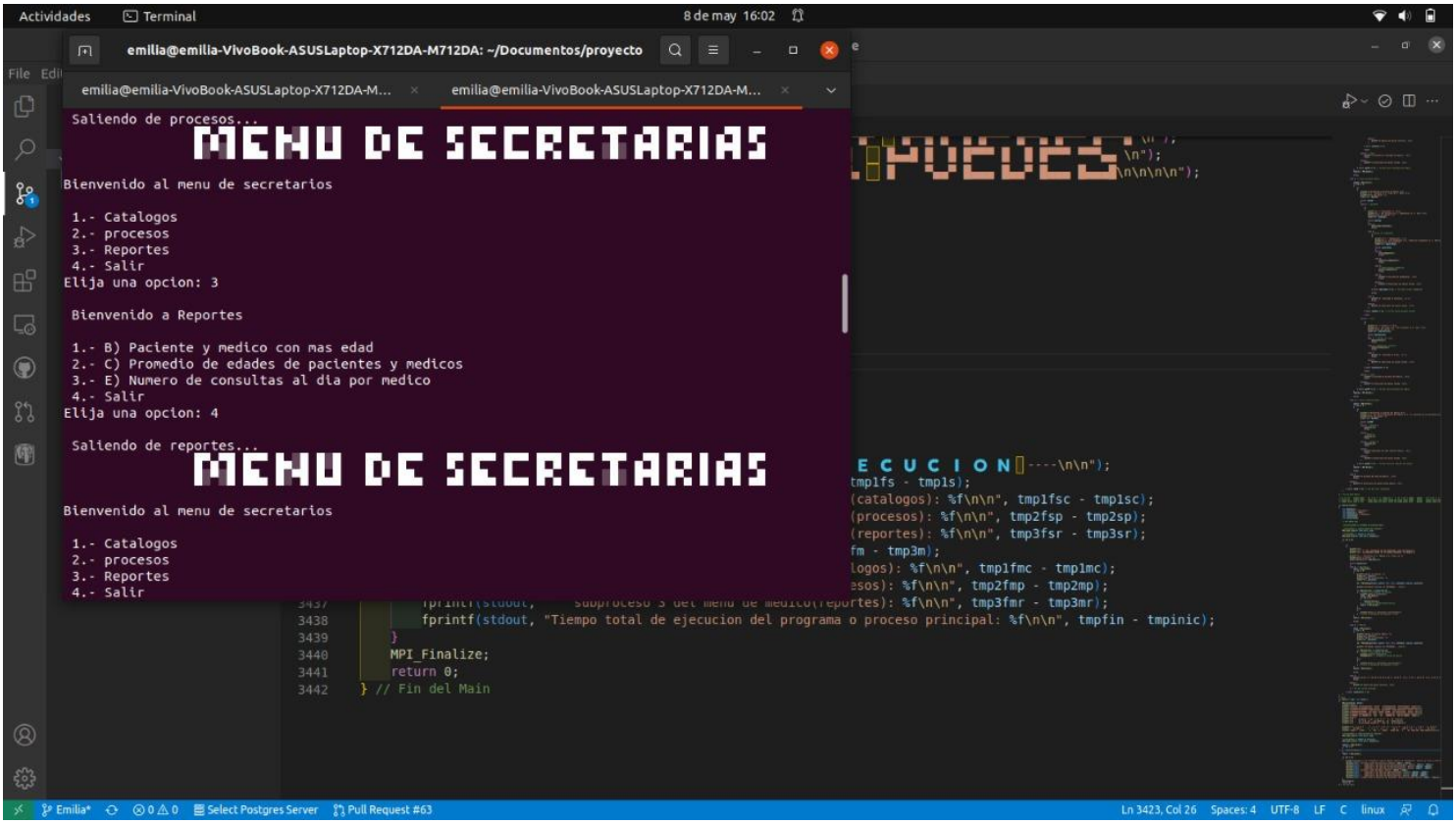
Pantalla inicial del programa, se empieza pidiendo el usuario y la contraseña, puede ser secretaria o médico.



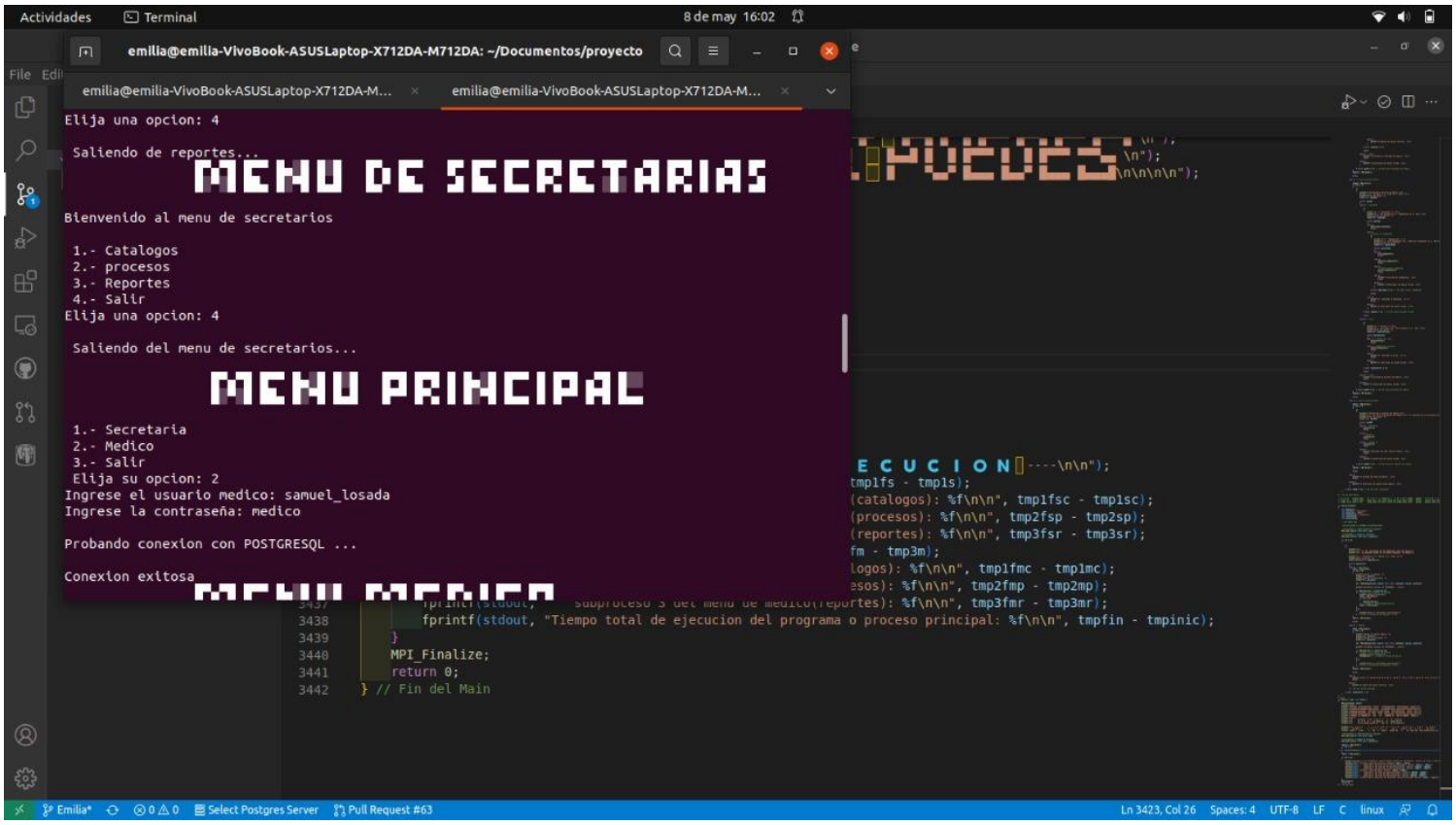
Entrando como usuario secretaria se muestra el siguiente menú secundario, aquí se muestran los catálogos.



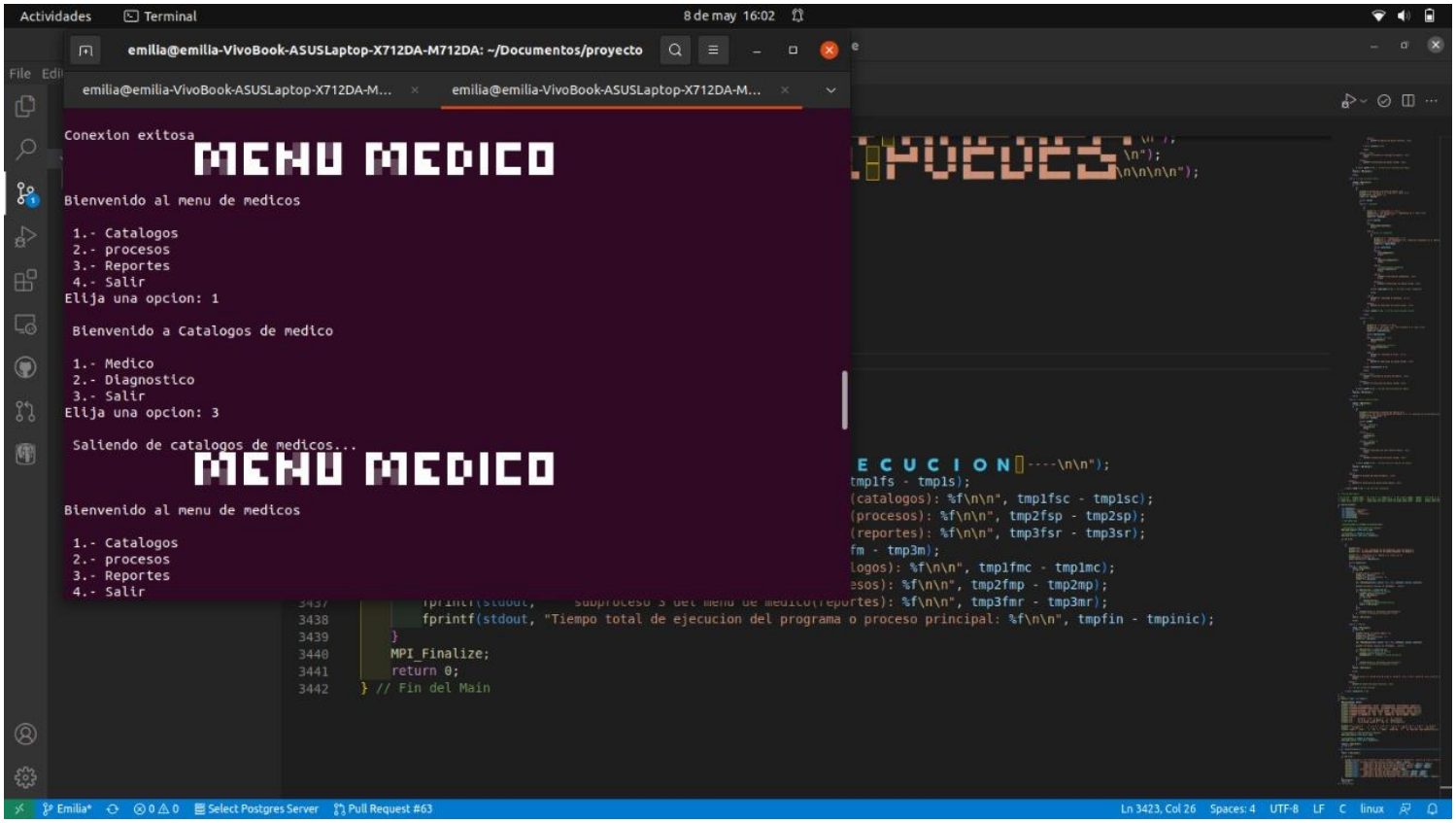
Siguiendo como usuario secretaria se muestran las opciones de la opción 2 que son los procesos.



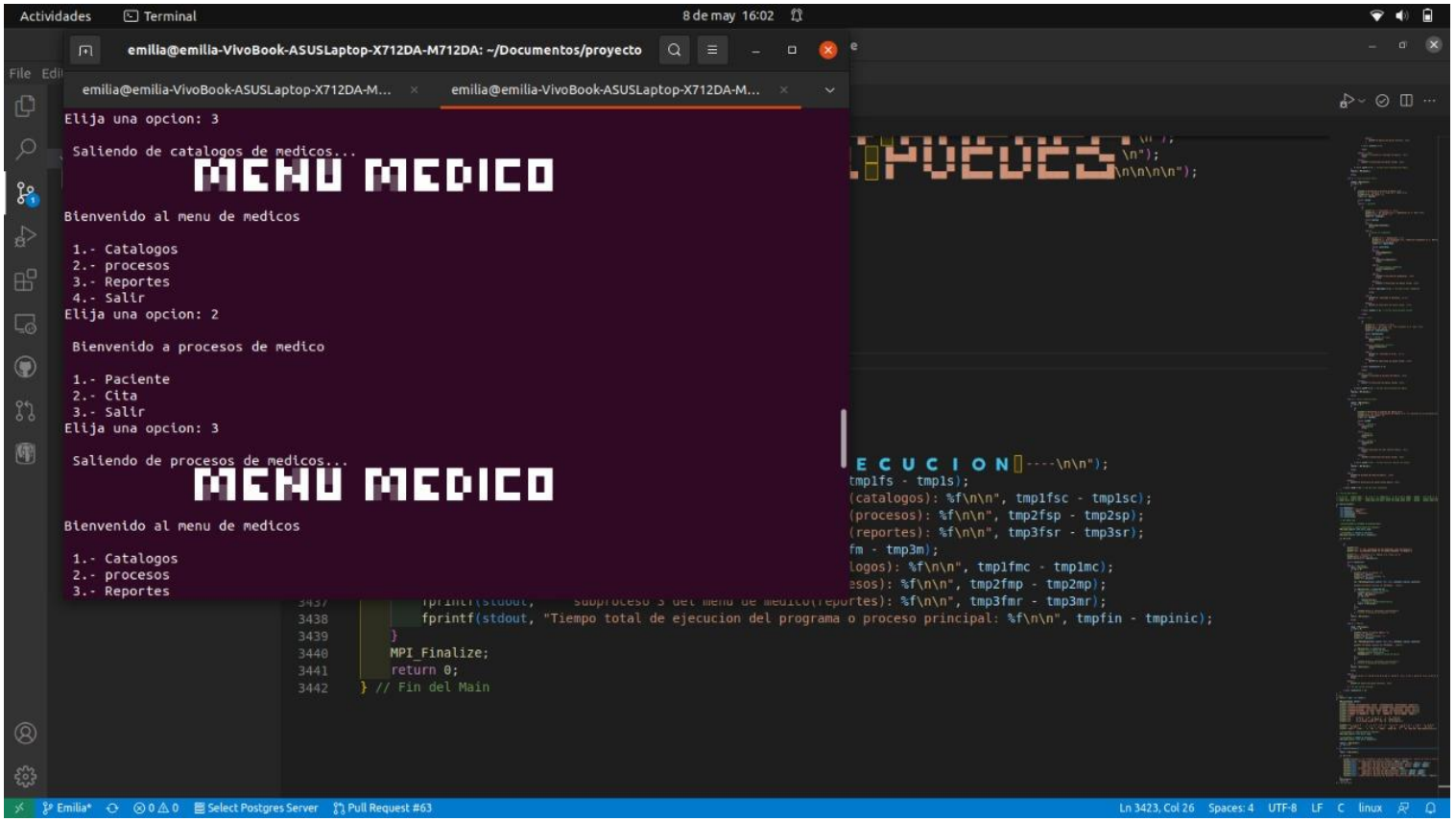
Finalmente en el usuario de Secretaria se muestran los reportes B, C, E.



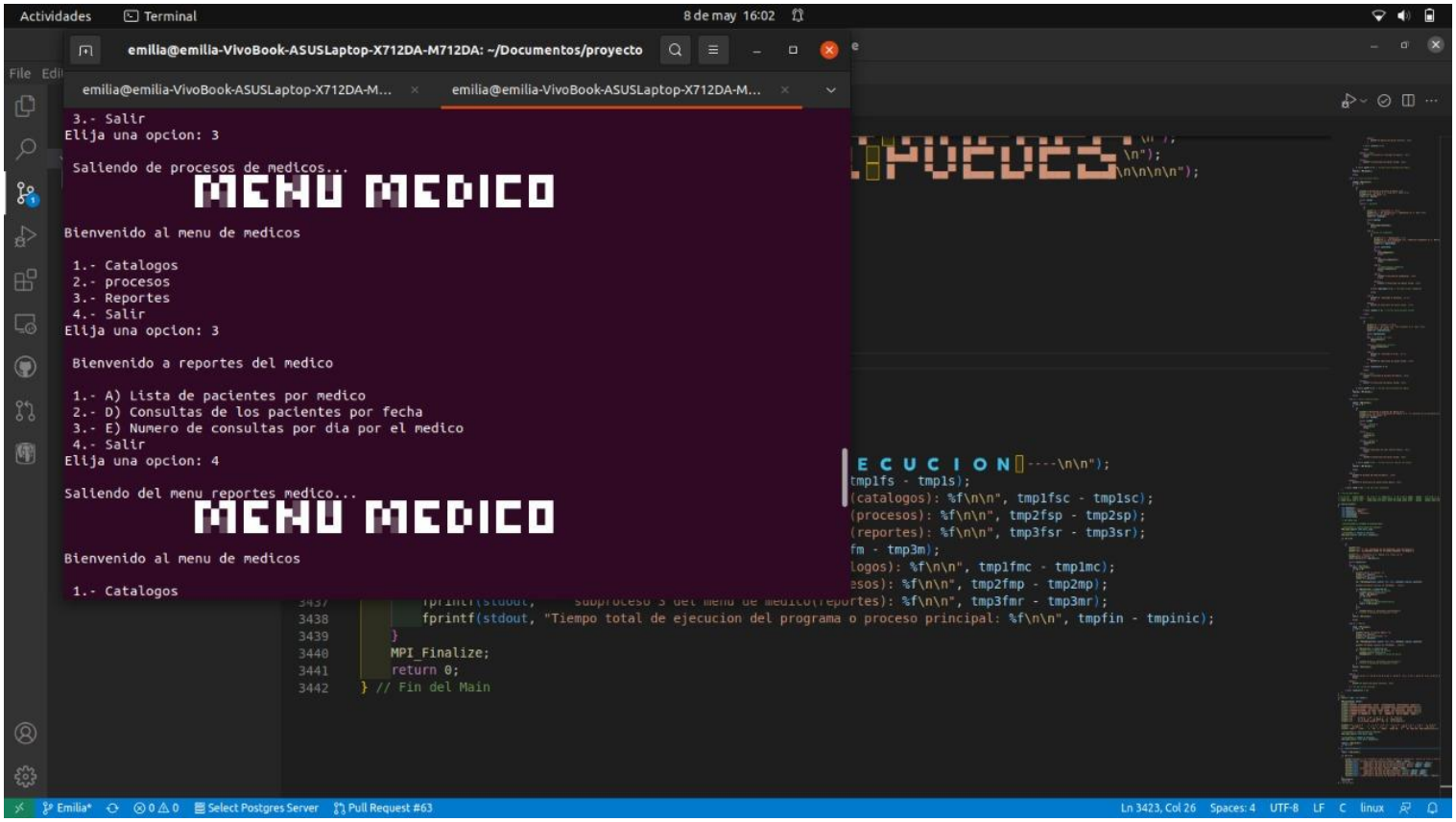
Saliendo del menú de secretaria y entrando como un usuario médico.



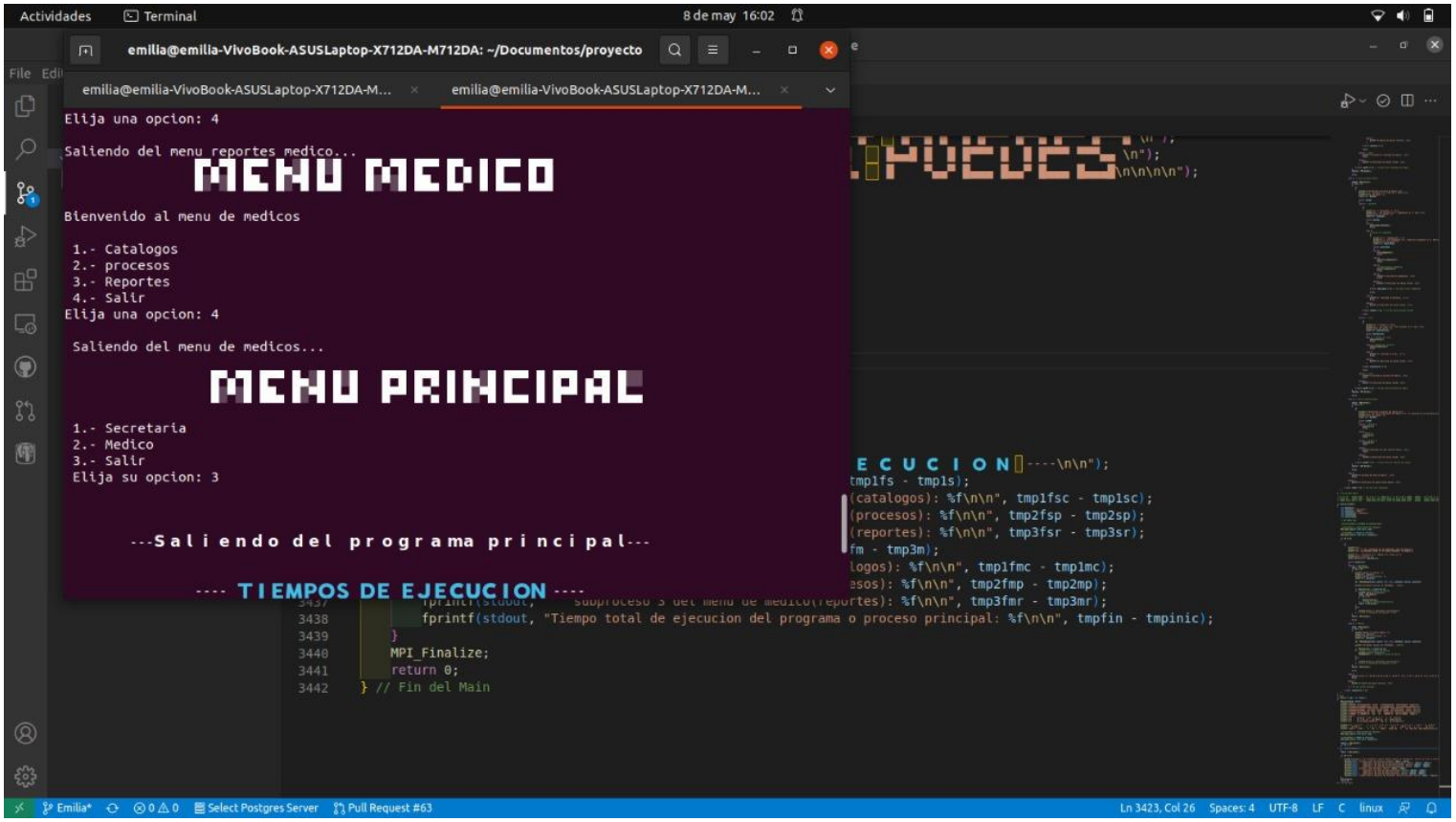
Se muestra el menú secundario del médico y se muestran las opciones al entrar a la opción de catálogos.



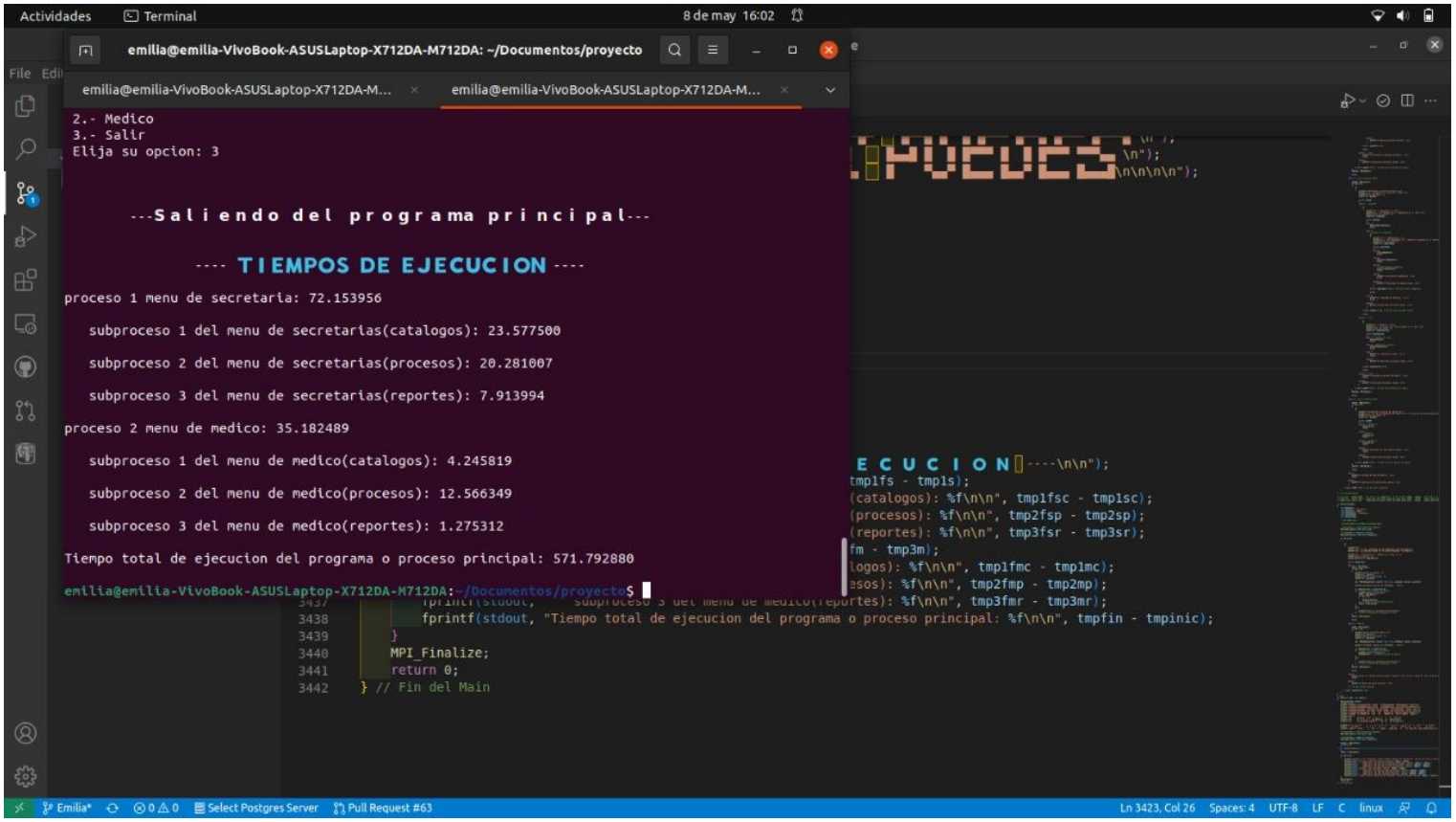
Con el mismo usuario de médico se muestran las opciones de la opción de procesos.



En la última opción se muestran los reportes A, D, C.



Saliendo del menú de médicos y regresando al menú principal.



Saliendo por completo del programan se muestra el mensaje final y se imprimen los tiempos de ejecución de cada opción.