



Proyecto #2

Lenguajes de Programación

Profesor Bryan Tomás Hernández Sibaja

Preparado por:

Feng Liu Jimmy

Kuo Liu Francisco

Wu Zhong Danielo

Tecnológico de Costa Rica

23/05/2025

Tabla de contenidos

| | |
|--|----|
| Enlace de GitHub..... | 3 |
| Pasos de Instalación del Programa..... | 3 |
| Manual de Usuario..... | 5 |
| Main.hs..... | 7 |
| Sistema de inicio de sesión..... | 8 |
| Encriptación/Desencriptación..... | 8 |
| Sistema de gestión de usuarios:..... | 9 |
| Ver servicios guardados..... | 10 |
| Agregar nuevos servicios..... | 10 |
| Editar un servicio..... | 11 |
| Borrar un servicio..... | 12 |
| Consultar un servicio..... | 13 |
| ProyectosLenguajes2.cabal..... | 14 |

Enlace de GitHub

[Proyecto 2 de Lenguajes de Programación](#)

Pasos de Instalación del Programa

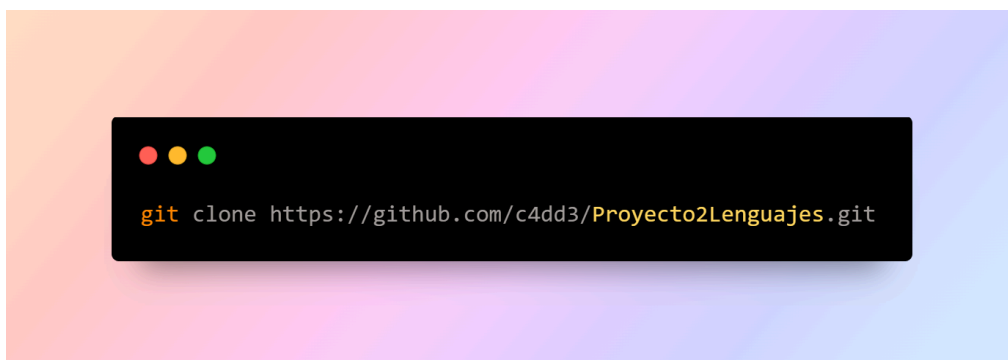
Para poder hacer uso del sistema de gestión de contraseñas es necesario cumplir con ciertos requisitos previos:

- Git: Acceso a GitHub para poder clonar el repositorio.
- Compilador de Haskell: Para el proyecto se hizo uso de GHCup el cual posee compiladores para Haskell y otras herramientas relacionadas con el lenguaje de programación.
- Cabal: Este es el sistema de construcción y gestión de dependencias para Haskell, utilizado para correr el programa.
- GHC: Compilador de Haskell.

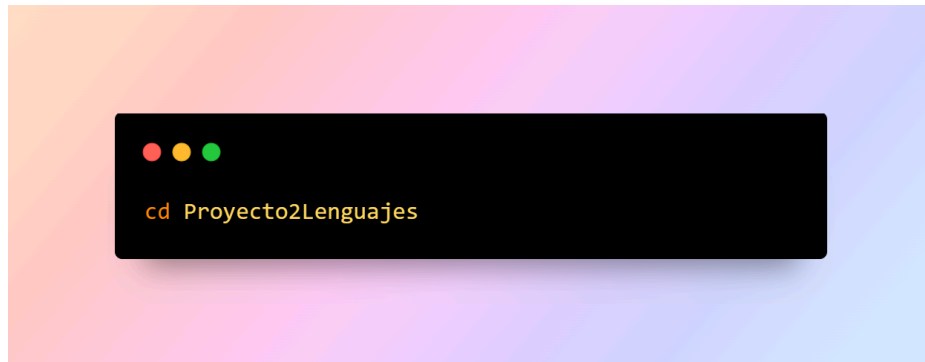
Instrucciones de instalación

1. Clonar el repositorio de GitHub mediante el link.

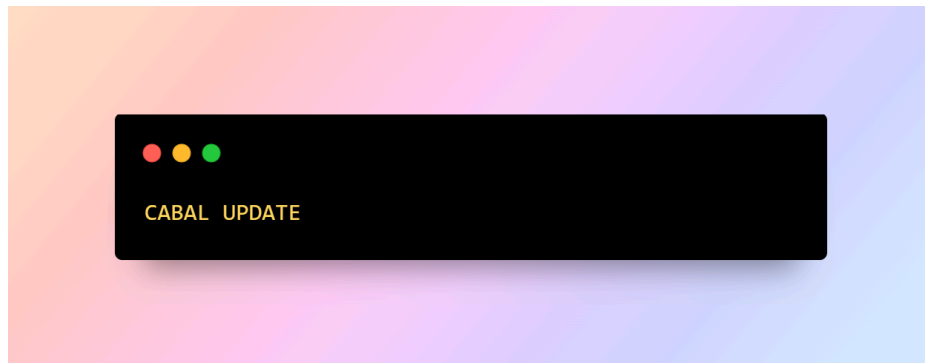
Ya sea mediante el uso de comandos en la terminal utilizando Git o mediante otros programas como GitHub Desktop. En caso de hacerlo en una terminal se debe poner el siguiente comando:



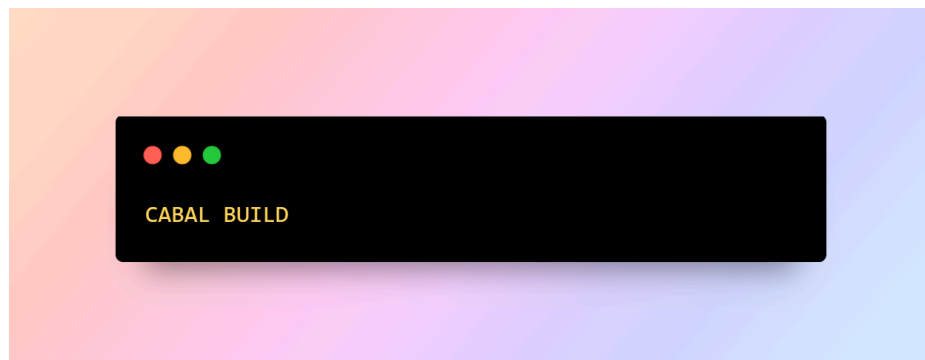
2. Navegar a la raíz del repositorio:



3. Se llama al comando UPDATE para actualizar el índice de paquetes:



4. Se llama al comando de BUILD para la construcción o compilación del proyecto:



5. Se llama al comando de RUN para ejecutar el programa:



Manual de Usuario

Al iniciar el programa con el comando “cabal run” se despliega el menú principal que tiene una serie de opciones de las cuales el usuario puede escoger.

```
=== Administrador de Contraseñas ===  
Seleccione una opción:  
1. Iniciar Sesión  
2. Registrarse  
3. Salir del programa
```

Al darle a opción 1, se muestra un texto que indica al usuario ingresar su nombre de usuario y luego su PIN para poder acceder a las contraseñas. En el caso de que el nombre de usuario no existe o el PIN esté incorrecto, se indica el respectivo error al usuario y se regresa al menú principal.

```
--- Iniciar Sesión ---  
Ingrese nombre de usuario:  
franciscokuoliu  
Ingrese su PIN:  
101010  
Inicio de sesión exitoso. Bienvenido, franciscokuoliu.
```

Si la sesión se inició correctamente, se muestra el siguiente menú para el manejo de contraseñas.

```
--- Menú de franciscokuoliu ---  
1. Ver servicios guardados  
2. Agregar nuevo servicio  
3. Editar un servicio  
4. Borrar un servicio  
5. Consultar un servicio  
6. Cerrar sesión
```

La opción 1 muestra todos los servicios guardados indicando el nombre del servicio, el usuario o correo de dicho servicio y su respectiva contraseña. Para el identificador

se muestra solamente los primeros dos y los últimos dos caracteres. En el caso de la contraseña está ocultada completamente.

```
--- Tus servicios guardados ---
Servicio      Identificador      Contraseña
-----
1  gmail      fr*****om*****
```

La opción 2 permite al usuario agregar un nuevo servicio ingresando el nombre del servicio, el nombre de usuario o correo respectivo y por último la contraseña.

```
--- Agregar nuevo servicio ---
Nombre del servicio (ej: Gmail, Netflix, etc.):
gmail
Nombre de usuario o correo del servicio:
francisco.kuoliu@gmail.com
Contraseña del servicio:
asdfasdf
```

La opción 3 permite al usuario editar un servicio que tiene guardado en el sistema por medio de su ID. Puede cambiar todos los datos si eso desea, si aprieta ENTER, puede mantener los datos que tiene actualmente.

```
--- Tus servicios guardados ---
ID  Servicio      Identificador      Contraseña
-----
1  gmail      fr*****om*****

Servicios disponibles:
1. gmail (fr*****om)
Ingrese el ID del servicio que desea editar:
1
Editando servicio: gmail
Nuevo nombre del servicio (Enter para mantener: gmail):
x
Nuevo identificador (Enter para mantener: francisco.kuoliu@gmail.com):
Nueva contraseña (Enter para mantener):
Servicio editado exitosamente.
```

La opción 4 permite al usuario borrar un servicio que había guardado en el sistema, indicando el ID respectivo.

```
--- Tus servicios guardados ---
ID  Servicio      Identificador      Contraseña
-----
1  x              fr*****om*****

Ingrese el ID del servicio que desea borrar:
```

La opción 5 es la que permite al usuario consultar un servicio. Se ingresa el nombre del servicio que desea consultar y al encontrar el servicio, se despliegan los datos

de dicho servicio. El usuario puede copiar el nombre de usuario o correo y la contraseña a su portapapeles.

```
Ingresa el nombre del servicio a consultar:
x
Servicio: x
Email/Usuario: francisco.kuoliu@gmail.com
Contraseña: asdfasdf
¿Desea copiar al portapapeles?
1. Copiar email/usuario
2. Copiar contraseña
3. No copiar
```

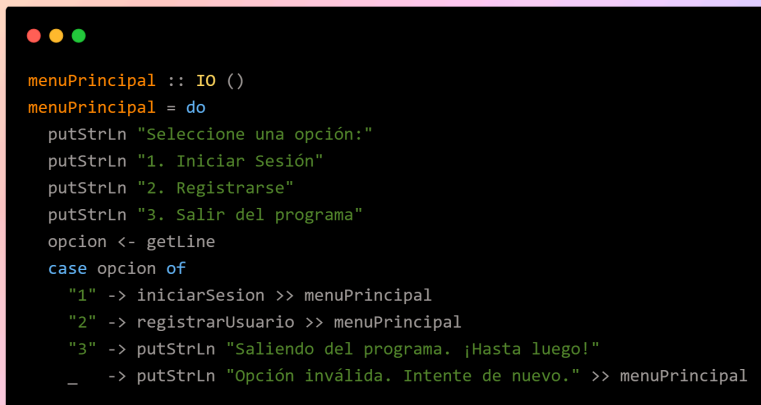
La opción 6 de cerrar sesión como indica el nombre, cierra la sesión y regresa el usuario al menú principal.

Main.hs

Dentro del “main” primero se imprime el título de la aplicación, posteriormente llama a la función de “menuPrincipal”. Dado que es de tipo IO (entrada y salida) no retorna nada.

```
main :: IO ()
main = do
    putStrLn "=== Administrador de Contraseñas ==="
    menuPrincipal
```

Una vez llamada la función “menuPrincipal” se mostrará un menú dentro del cual el usuario es capaz de seleccionar entre 3 opciones, esto mediante el uso de “case”, donde se compara lo que el usuario ingresa mediante el “getLine” y se guarda en “opcion”:



```
menuPrincipal :: IO ()
menuPrincipal = do
  putStrLn "Seleccione una opción:"
  putStrLn "1. Iniciar Sesión"
  putStrLn "2. Registrarse"
  putStrLn "3. Salir del programa"
  opcion <- getLine
  case opcion of
    "1" -> iniciarSesion >> menuPrincipal
    "2" -> registrarUsuario >> menuPrincipal
    "3" -> putStrLn "Saliendo del programa. ¡Hasta luego!"
    _   -> putStrLn "Opción inválida. Intente de nuevo." >> menuPrincipal
```

Sistema de inicio de sesión

El programa sigue un flujo clásico de inicio de sesión primero solicita al usuario su nombre de usuario, en caso de existir, procederá a ingresar su pin de 6 dígitos. Para verificar el pin, se extrae a partir de archivo de usuarios.txt y se descripta la contraseña guardada, en caso de coincidencia se le permitirá su ingreso. Este sistema funciona mediante el uso de funciones auxiliares para el inicio de sesión. Para ello hace uso de funciones auxiliares como “verificarCredenciales”, “verificarUsuarioExistente” y “desencriptarAES”. La primera verifica que el usuario haya ingresado las credenciales correctas, es decir, el pin, durante esta se llama a “desencriptarAES” para obtener el pin guardado en usuarios.txt. La segunda función se encarga de verificar la existencia del usuario en el txt.

Encriptación/Desencriptación

La función encriptarAES recibe un string y lo encripta utilizando el algoritmo AES-128 en modo CBC. Para ello, convierte la clave secreta (que debe tener exactamente 16 bytes) en un objeto de cifrado mediante cipherInit. Si la conversión falla, lanza un error. Si tiene éxito, se utiliza un vector de inicialización (IV) fijo compuesto por 16 bytes con valor cero. Luego, el texto plano se convierte en ByteString, se le aplica un relleno PKCS7 con la función pad (para que su longitud sea múltiplo del tamaño de bloque de AES), y finalmente se cifra con cbcEncrypt. El resultado se codifica en base64 para facilitar su almacenamiento o transmisión, y se devuelve como String.

La función `desencriptarAES` hace lo contrario: toma un texto encriptado codificado en base64, lo decodifica y lo desencripta usando la misma clave y el mismo IV fijo. Primero intenta inicializar el cifrador igual que antes; si falla, lanza un error. Luego decodifica el texto base64 a bytes, y lo descifra con `cbcDecrypt` usando el mismo IV. El resultado es el texto en claro con padding, por lo que se utiliza la función `unpad` para quitarlo antes de convertirlo nuevamente en String.

En cuanto a las funciones auxiliares, `pad` se encarga de aplicar el padding estilo PKCS7 al mensaje antes de cifrarlo. Lo hace calculando cuántos bytes faltan para completar un bloque de 16 bytes (que es el tamaño de bloque de AES), y luego añade ese número como valor repetido en los bytes finales. Por ejemplo, si faltan 3 bytes, se agregan tres veces el valor `\3`. Finalmente, la función `unpad` hace lo inverso: elimina ese padding después de la desencriptación, tomando la longitud original menos el valor del último byte.

Con respecto al registro de usuarios, el flujo comienza con `registrarUsuario`, el cual solicita al usuario que introduzca un nombre. Posteriormente, se verifica si ese nombre ya está registrado en el archivo `usuarios.txt`. Si el nombre ya existe, el sistema informa que debe intentarse con otro nombre. Si no existe, se procede a la solicitud de un PIN mediante la función `solicitarPIN`.

La función `solicitarPIN` exige que el PIN tenga exactamente seis dígitos numéricos. El sistema solicita al usuario que lo introduzca dos veces para confirmar su validez. Si los valores ingresados no coinciden o no cumplen con el formato, se vuelve a solicitar el PIN. Una vez validado, el PIN es cifrado utilizando la función `encriptarAES`, y junto con el nombre de usuario, se guarda en el archivo `usuarios.txt` mediante la función `guardarUsuario`. El registro exitoso concluye con un mensaje de confirmación para el usuario.

Sistema de gestión de usuarios:

De igual forma se implementa un menú para usuario, éste funciona igualmente con el uso de un `case` para determinar qué opción quiere realizar el usuario, entre las cuales posee las siguientes opciones:

1. Ver servicios guardados
2. Agregar nuevos servicios
3. Editar un servicio

4. Borrar un servicio
5. Consultar un servicio
6. Salir

Cada opción llamará a una función auxiliar para cumplir con lo solicitado, para ello es necesario el nombre del usuario.

Ver servicios guardados

Esta función se encuentra compuesta por tres funciones principales: “verServicios”, “mostrarServicio” y “ocultarUsuario”.

- La función “verServicios” se encarga de verificar si existe el archivo que contiene los servicios guardados (servicios.txt). En caso de no existir, notifica al usuario que no hay servicios almacenados. Si el archivo existe, filtra todas aquellas líneas que pertenecen al usuario indicado y, si encuentra resultados, los despliega en formato de tabla con encabezados alineados. Para cada servicio encontrado, se utiliza la función “mostrarServicio” para presentar la información de forma legible.
- La función “mostrarServicio” toma un índice y una línea del archivo como entrada, y se encarga de dividir para separar el usuario del bloque de datos cifrados. Luego, descifra estos datos y extrae los campos correspondientes: nombre del servicio, identificador y contraseña. El identificador se presenta parcialmente oculto usando la función “ocultarUsuario”, mientras que la contraseña se reemplaza por una cadena de asteriscos de la misma longitud para preservar la privacidad del usuario.
- Por último, la función “ocultarUsuario” se utiliza para ocultar parcialmente el identificador (como un correo o nombre de usuario), mostrando solo los dos primeros y dos últimos caracteres, y reemplazando los intermedios con asteriscos. Si el identificador tiene una longitud menor o igual a cuatro caracteres, se oculta completamente.

Agregar nuevos servicios

La función agregarServicio toma como argumento el nombre de usuario general (nombreUsuario), y luego ejecuta una secuencia de acciones en el contexto de entrada/salida (IO). Primero, muestra un encabezado por pantalla indicando que se va a agregar un nuevo servicio. Luego, solicita al usuario que introduzca el nombre

del servicio (por ejemplo, "Gmail"), seguido del nombre de usuario o correo asociado a ese servicio, y finalmente la contraseña correspondiente. Todos estos datos se capturan con `getLine`. Posteriormente, se combinan en un solo texto (`datosPlano`), concatenando los tres valores separados por dos puntos, es decir, formando una cadena como "Gmail:usuario@gmail.com:miClave123".

Este texto plano se pasa a la función `encriptarAES`, que lo transforma en un texto encriptado (generalmente una cadena de caracteres codificada en base64). A continuación, se forma una línea (entrada) que une el `nombreUsuario` con los datos cifrados, separados también por dos puntos, y se le añade un salto de línea (`\n`) al final. Esta línea se guarda en el archivo `servicios.txt` usando `appendFile`, que añade la información al final del archivo sin borrar lo anterior. Finalmente, la función imprime un mensaje de confirmación al usuario indicando que el servicio fue guardado exitosamente.

Editar un servicio

Esta función está compuesta por varias subfunciones que permiten al usuario editar un servicio guardado de forma segura y sencilla.

- La función principal "editarServicio" comienza listando los servicios asociados al usuario mediante la función "listarServiciosConIndice", que retorna una lista de tuplas con el índice y la línea original cifrada de cada servicio. Si no hay servicios, se informa al usuario. En caso contrario, se muestran los servicios disponibles con su índice, nombre descifrado y el identificador parcialmente oculto, para que el usuario pueda elegir cuál editar.
- Luego, se solicita al usuario ingresar el ID del servicio que desea modificar. Si el ID es válido, se extraen los datos descifrados correspondientes a ese servicio y se solicita al usuario ingresar nuevos valores para el nombre del servicio, identificador y contraseña, permitiendo dejar campos en blanco para mantener los valores actuales.
- Los datos actualizados se vuelven a cifrar con "encriptarAES" y se reemplaza la línea correspondiente en el archivo usando la función "actualizarLineaArchivo", que lee todo el archivo, reemplaza la línea vieja por la nueva y vuelve a escribir el archivo completo.

Las funciones auxiliares que facilitan esta tarea son:

- “listarServiciosConIndice”: verifica si existe el archivo, filtra los servicios del usuario, los muestra en tabla con formato y devuelve la lista enumerada para manipulación.
- “formatoServicioTabla”: recibe una línea cifrada, la descifra y devuelve una cadena formateada con el nombre del servicio, identificador oculto y contraseña oculta con asteriscos.
- “actualizarLineaArchivo”: realiza la actualización efectiva del archivo de servicios, reemplazando la línea antigua por la nueva.

Borrar un servicio

La función `borrarServicio` toma como argumento el `nombreUsuario`, y comienza obteniendo una lista de servicios asociados a ese usuario mediante la función `listarServiciosConIndice`. Si no hay servicios registrados, la función simplemente termina. En caso contrario, le pide al usuario que introduzca el ID del servicio que desea borrar. Luego convierte ese input a un número usando `reads`. Si la conversión tiene éxito y el ID está dentro del rango válido (es decir, entre 1 y la cantidad de servicios listados), se extrae la línea específica que representa ese servicio (`lineaABorrar`). Antes de proceder, se le pide al usuario una confirmación ("s" o "S"). Si el usuario confirma, se llama a la función `eliminarLineaArchivo` para quitar esa línea del archivo `servicios.txt`, y luego se muestra un mensaje confirmando que el servicio fue eliminado. Si el ID no es válido o el input no es un número, se muestra un mensaje de error apropiado.

La función auxiliar `eliminarLineaArchivo` se encarga de eliminar una línea específica de un archivo. Primero lee todo el contenido del archivo con `readFile`, luego utiliza `evaluate (length contenido)` para forzar la evaluación estricta del contenido antes de modificar el archivo (esto es necesario en Haskell debido a su naturaleza perezosa, y para asegurar que el archivo esté completamente leído y liberado antes de escribir). Luego, divide el contenido en líneas con `lines`, y filtra aquellas que no sean iguales a `lineaABorrar`. Finalmente, vuelve a escribir el archivo con las líneas resultantes usando `writeFile`, uniéndolas nuevamente con saltos de línea (`unlines`).

Consultar un servicio

La función `consultarServicio` permite buscar un servicio específico que haya sido previamente guardado por un usuario determinado (`nombreUsuario`). Primero solicita al usuario que ingrese el nombre del servicio a buscar. Luego lee todo el contenido del archivo `servicios.txt`, lo divide en líneas y filtra únicamente aquellas que pertenecen al usuario actual, lo cual se hace comprobando que el prefijo de cada línea (antes del primer :) coincida con `nombreUsuario`. A continuación, la función recorre esas líneas del usuario, descripta la parte cifrada de cada una utilizando `descriptarAES`, y separa el texto plano resultante por los dos puntos (:), que en principio contiene tres partes: nombre del servicio, identificador (correo o nombre de usuario), y contraseña. Si el nombre del servicio descriptado coincide con el buscado, esa línea se considera una coincidencia. Si no se encuentra ninguna coincidencia, se informa que no se encontró el servicio. Si hay coincidencias, se toma la primera, se extraen sus componentes y se muestran en pantalla. Además, se ofrece al usuario la opción de copiar al portapapeles el email/usuario o la contraseña. Según la opción ingresada (1, 2 o 3), se llama a la función `copiarAlPortapapeles` para hacer la copia correspondiente o no se copia nada.

La función `copiarAlPortapapeles` detecta el sistema operativo y usa el comando adecuado para copiar el texto recibido como argumento al portapapeles del sistema. En Linux usa `xclip`, en Windows (mingw32) usa `clip`, y en macOS (darwin) utiliza `pbcopy`. Si el sistema no es reconocido, muestra un mensaje de error indicando que no es compatible. Esta función permite al usuario recuperar sus credenciales de manera práctica sin tener que escribirlas manualmente.

ProyectosLenguajes2.cabal

El archivo `.cabal` es el archivo de configuración principal en un proyecto Haskell que utiliza el sistema de construcción Cabal. Su función es describir todos los aspectos necesarios para compilar, construir y gestionar el proyecto correctamente.

- Información del paquete: nombre, versión, autor, licencia, etc.
- Dependencias: lista de bibliotecas externas necesarias para compilar el proyecto.
- Punto de entrada: el archivo Haskell que contiene la función "main" cuando se trata de un ejecutable.
- Módulos expuestos: en caso de ser una biblioteca, se indican los módulos que pueden ser utilizados por otros proyectos.

Gracias a este archivo, Cabal puede construir el proyecto correctamente, instalar las dependencias necesarias y ejecutar tareas como pruebas o instalación. Es, en esencia, una guía completa que le dice a Cabal cómo manejar el proyecto.