Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Sistemas Operativos 2

Manual de Técnico
Proyecto

Carlos Ojani NG Valladares - 201801434

Cada día se busca que el numero de usuarios del sistema operativo Linux crezca, pero esto es difícil ya que la mayoría de las personas son usuarios finales y desconocen como ejecutar ciertos comandos para ver la información que están acostumbrados a ver fácilmente en Windows, por ejemplo, el administrador de tareas. Es por eso, que se presenta una solución amigable para el usuario que consiste en proveer una representación gráfica de lo que es el uso que le está dando a su CPU y la cantidad libre de memoria en disco.

Tecnologías Utilizadas

Go
Golang o Go (Go Programming Language) es un lenguaje de programación open source bastante reciente, creado en 2007 por los programadores de Google Ken Thompson y Rob Pike, quienes ya contaban con una reconocida trayectoria por la creación de los lenguajes B y Limbo, respectivamente.

GO cada vez gana más popularidad entre los desarrolladores alrededor del mundo, de hecho, es el lenguaje de programación que más aceleradamente está creciendo.

Wails
Es un framework que nos permite escribir aplicaciones de escritorio utilizando Go. La diferencia entre Wails y otras herramientas es que expone el código Go al frontend como funciones que devuelven promesas. Wails logra hacer esto por medio de un mecanismo de ligadura, el cual explicaremos más adelante.

El código front end se puede desarrollar utilizando cualquier framework de Javascript, como por ejemplo Angular, React, Vue.js y Vuetify.

Wails cuenta con la capacidad de envolver código Go así como frontend web en un único binario. La interfaz de línea de comandos de Wails hace que este proceso sea más fácil ya que se encarga de la creación, compilación y empaquetamiento de proyectos.

Reactjs
React es una librería Javascript focalizada en el desarrollo de interfaces de usuario. Así se define la propia librería y evidentemente, esa es su principal área de trabajo. Sin embargo, lo cierto es que en React encontramos un excelente aliado para hacer todo tipo de aplicaciones web, SPA (Single Page Application) o incluso aplicaciones para móviles. Para ello, alrededor de React existe un completo ecosistema de módulos, herramientas y componentes capaces de ayudar al desarrollador a cubrir objetivos avanzados con relativamente poco esfuerzo.

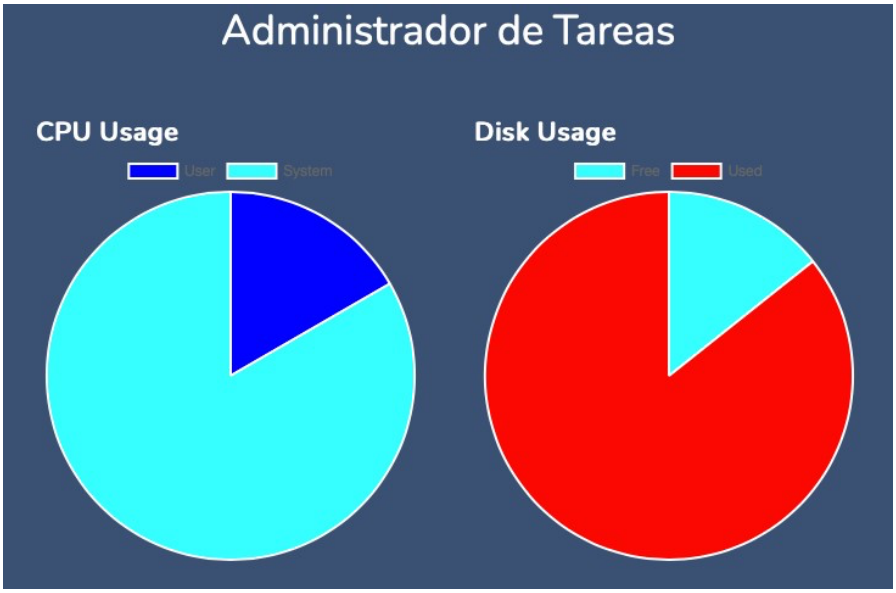
Por tanto, React representa una base sólida sobre la cual se puede construir casi cualquier cosa con Javascript. Además facilita mucho el desarrollo, ya que nos ofrece muchas cosas ya listas, en las que no necesitamos invertir tiempo de trabajo.

Funcionamiento de la Aplicación

La aplicación está construida sobre Wails, que es básicamente una solución similar a Electronjs, pero para el lenguaje Go. Que nos provee una conexión entre la aplicación de go con nuestro frontend, y al también nos permite generar una aplicación de escritorio con un ejecutable dependiendo del sistema operativo.

Para el build de la aplicación, Wails nos facilita la tarea con un simple comando: wails build. Nos genera un ejecutable (binario, en MacOS o Linux) en la carpeta build que también genera.

Se muestra a continuación el entorno gráfico de la aplicación.



Uso del CPU
Para este requerimiento se utiliza la librería del usuario ***shirou*** que específicamente ofrece una con respecto al rendimiento del CPU del sistema operativo. Muy práctica de utilizar. La librería contiene una interfaz que directamente interactúa con la data del CPU actual.

La librería nos ofrece junto con la interfaz de CPU una función de porcentaje, que nos trae esa información, ya sea por el porcentaje de CPU utilizado por el usuario, el sistema o que no esté en uso. Esto se envía a través de un struct hacia el frontend.

```go
func (a *App) GetCPUPercentage() CPUData{
    perc, err := cpu.Percent(1*time.Second, false)
    if err != nil {
        return CPUData{}
    }

    return CPUData{perc[cpu.CPUser], 0, 0}
}
```

Disco Utilizado

En este caso se utiliza una librería propia de go. Una que realiza llamados al sistema anfitrión, ***syscall***. Básicamente nos puede traer los datos estadísticos del sistema operativo y justamente nos provee una función, que dependiendo de la ubicación, nos trae la memoria utilizada, así como la memoria total del sistema. En este caso, la memoria usada, no es más que la diferencia de la memoria total con la memoria libre.

El frontend recibe a través de un struct, que llega como un objeto, lo que es la memoria total, memoria disponible y la utilizada. Ya listas para ser capturadas y utilizadas.

```go
func (a *App) DiskUsage() DiskStatus {
    disk := DiskStatus{}
    fs := syscall.Statfs_t{}
    err := syscall.Statfs("/", &fs)
    if err != nil {
        return disk
    }
    all := fs.Blocks * uint64(fs.Bsize)
    free := fs.Bfree * uint64(fs.Bsize)
    used := disk.All - disk.Free
    disk.All = float64(all)/float64(GB)
    disk.Free = float64(used)/float64(GB)
    disk.Used = float64(free)/float64(GB)
    return disk
}
```

Memoria Utilizada

Para obtener la información de memoria se utilizó la siguiente librería: "github.com/pbnjay/memory". El cual nos facilita los datos de memoria libre y el total de memoria del sistema que lo esté utilizando. Para obtener la memoria usada, se realiza la diferencia del total con lo que se tiene libre. De este forma se utiliza a nivel de código.

```
You, a week ago | 1 author (You)
type Memory struct {
    MemTotal     uint64
    MemFree      uint64
    MemUsed uint64
}

func (a *App) ReadMemoryStats() Memory {
    memTotal := memory.TotalMemory()/1000000
    memFree := memory.FreeMemory()/100000
    memUsed := memTotal - memFree
    mem := Memory{
        MemTotal: memTotal,
        MemFree: memFree,
        MemUsed: memUsed,
    }
    return mem
}
```

Bloqueo y Desbloqueo de Puertos USB

Para estas funcionalidades se utilizan comandos en la terminal, tanto para Linux como para MacOS. Ya que la aplicación esta desarrollada principalmente para Mac.

Se maneja de la siguiente forma:

```
// Bloquear todos los dispositivos USB.
func (a *App) BlockAllDevices()  {
    // Ejecutar el comando adecuado para bloquear todos los dispositivos USB en función del sistema operativo.
    var cmd *exec.Cmd

    fmt.Printf("Sistema Operativo: %v\n", runtime.GOOS)
    if runtime.GOOS == "darwin" {
        cmd = exec.Command("sudo", "pmset", "-a", "hibernatemode", "0")
    } else if runtime.GOOS == "linux" {
        //cmd = exec.Command("sudo", "sh", "-c", "echo '0' > /sys/bus/usb/drivers/usb/unbind")
        cmd = exec.Command("sh", "-c", "echo 'Xunix..unix' | sudo -S chmod 0000 /media")
    } else {
        fmt.Errorf("Sistema operativo no soportado")
    }

    // Ejecutar el comando y comprobar si se ha producido algún error.
    a.WriteLog("Puertos USB bloqueados.")
    err := cmd.Run()
    if err != nil {
        return
    }
}
```

Y para el desbloqueo de puertos de esta forma:

```go
// Desbloquear todos los dispositivos USB.
func (a *App) UnblockAllDevices() {
    // Ejecutar el comando adecuado para desbloquear todos los dispositivos USB en función del sistema operativo.
    var cmd *exec.Cmd

    fmt.Printf("Sistema Operativo: %v\n", runtime.GOOS)

    if runtime.GOOS == "darwin" {
        cmd = exec.Command("sudo", "kextload", "-b", "com.apple.driver.AppleUSBFTDI")
    } else if runtime.GOOS == "linux" {
        cmd = exec.Command("sh", "-c", "echo 'Xunix..unix' | sudo -S chmod 0777 /media")

    } else {
        fmt.Errorf("Sistema operativo no soportado")
    }

    a.WriteLog("Puertos USB desbloqueados.")
    // Ejecutar el comando y comprobar si se ha producido algún error      You, seconds ago • Uncommitted changes
    err := cmd.Run()
    if err != nil {

        return
    }

}
```

Bitácora

La bitácora es generada en el lado del backend utilizando Go. Cada vez que se inicia la aplicación, este borra el archivo de bitácora viejo e inicializa uno nuevo. Para no acumular datos.

Este se escribe cada que se realiza una operación de bloqueo y desbloqueo de puertos USB. También cuando constantemente se analiza la carpeta local y la de USB para obtener la información de nuevos archivos copiados. Lleva la siguiente estructura.

bitacora.log

You, 2 hours ago | 1 author (You)

```
2023/04/23 11:50:59 2023-04-23 11:50:59.844208 -0600 CST m=+1.260549276 || Archivo $RECYCLE.BIN ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.844586 -0600 CST m=+1.260927696 || Archivo .TemporaryItems ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.844756 -0600 CST m=+1.261097364 || Archivo .Trashes ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.844831 -0600 CST m=+1.261171832 || Archivo .VolumeIcon.icns ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.844893 -0600 CST m=+1.261233786 || Archivo .VolumeIcon.ico ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.844947 -0600 CST m=+1.261288370 || Archivo ._. ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.845016 -0600 CST m=+1.261357628 || Archivo .fseventsd ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.845126 -0600 CST m=+1.261467337 || Archivo 2do Proyecto SO2.pdf ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.84524 -0600 CST m=+1.261581368 || Archivo 8o. semestre ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.845421 -0600 CST m=+1.261761817 || Archivo 8o. semestre amari Seminario ya se encuentra en el U
2023/04/23 11:50:59 2023-04-23 11:50:59.845498 -0600 CST m=+1.261839263 || Archivo Autorun.inf ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.846639 -0600 CST m=+1.262979847 || Archivo CV Amari.pdf ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.846801 -0600 CST m=+1.263141959 || Archivo Captures ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.846901 -0600 CST m=+1.263242085 || Archivo Carlos Ng ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.846987 -0600 CST m=+1.263327802 || Archivo EJERCICIO 2- CADENA DE MARKOV.docx ya se encuentra e
2023/04/23 11:50:59 2023-04-23 11:50:59.84705 -0600 CST m=+1.263391432 || Archivo EjerciciosBDG.rar ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.847112 -0600 CST m=+1.263453005 || Archivo MODELO DE SIMLACION.docx ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.847175 -0600 CST m=+1.263515923 || Archivo Mama ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.84729 -0600 CST m=+1.263630778 || Archivo Nathalia ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.847374 -0600 CST m=+1.263715015 || Archivo PAULA ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.847443 -0600 CST m=+1.263783924 || Archivo Recetas reposteria ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.84751 -0600 CST m=+1.263850998 || Archivo Seagate ya se encuentra en el USB
2023/04/23 11:50:59 2023-04-23 11:50:59.847574 -0600 CST m=+1.263914761 || Archivo Start_Here_Mac.app ya se encuentra en el USB
```

Se conforma de la fecha y hora formateada seguido de la descripción de las operaciones que se llevan a cabo.

Frontend

Wails nos provee variedad de opciones para utilizar como frontend, en este caso se utilizó React, por conveniencia. Así mismo, se utiliza lo que es la librearía Chartjs para la generación de gráficas, así como react-chartjs-2 que es otra librería derivada de Chartjs, pero adaptaba a React. Como se mostró al principio, se utilizaron dos gráficas de Pie o Dona, con el fin de comparar los datos y ver cómo estos de distribuyen con base a su uso.