# Planificación

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tarea | Fecha inicio | Fecha final | Descripción | Estado |
| Manejo archivos .xlsx | 28/05 | 29/05 | Leer, Imprimir, Filtrar | Listo |
| Manejo argumentos | 28/05 | 29/05 | Ingresar archivos como argumentos para trabajar con ellos posteriormente | En proceso |
| Montar cluster | 29/05 | 30/05 |  | Listo |
| Estudio como desarrollar algoritmo de forma secuencial | 30/05 | 01/06 |  | Listo |
| Desarrollo secuencial del algoritmo | 01/06 | 06/06 |  | En proceso |
| Estudio algoritmo y forma de abordar el problema general | 06/06 | 07/06 |  |  |
| Estudio que módulos paralelizar | 07/06 | 10/06 |  |  |
| Paralelizar módulos | 10/06 | 25/06 |  |  |

# Desarrollo diario (secuencial)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fecha | Hora | Integrante | Descripción | Estado |
| 28/05 | 10:00 am - 12:30 pm | Vale | Inicio módulo leer e imprimir archivos | Finalizado |
| 29/05 | 10:00 pm - 1:30 am | Vale | Estructuras de archivos (filtrar) | Finalizado |
| 29/05 | 6:00 pm – 8:00 pm | Vale | Termino estructuras de archivos | Finalizado |
| 01/06 | 05:30 pm - | Vale | Desarrollo secuencial del algoritmo | En proceso |
| 04/06 |  | Vale | Generar archivo.xlsx completo de salida (nueva librería xlsxio) |  |
| 04/06 | 11:30 am - 12:30 pm | Vale | Try-catch para verificar carga de archivos | Finalizado |
| 04/06 | 3:00 pm -  3:10 pm | Vale | Cambiar estructuras a singular | Finalizado |
| 04/06 | 10:40 am – 11:30 am | Vale | Funciones para cargar los archivo.xlsx | Finalizado |
| 04/06 | 11:00 pm -  12:15 am  10:00 am – 12:30 pm | Vale | Crear vector salas que contenga los datos del archivo xlsx | En proceso |
|  |  | Vale | Crear vector de bloques y asignar una sala a cada uno |  |
|  |  | Vale | Ingreso a estructura profesor y crear vector con los disponibles según bloque y día |  |
|  |  | Vale | Ordenar vector profesores por prioridad (cantidad de bloques que requieren a la semana) |  |
|  |  | Vale | Asignación de salas y verificación |  |
|  |  |  | Crear función imprimir bloque, que me imprima el Excel de salida |  |

# Repositorio

EL programa será escrito en C++ y se usará una carpeta local.

\*\* Cambio a IDE Netbeans a Visual Studio Code --→28/05

Enlace git: https://github.com/vfaurec/Proyecto\_Horario

# Tareas/actividades

A continuación se detalla la forma de trabajo para el proyecto de PARALELAS:

* Leer e imprimir archivos .xlsx
* Filtrar por campo algún archivo .xlsx
* Escribir y guardar nuevo archivo (formato .xlsx)
* Manejo de argumentos en la ejecución (ingreso de archivos por parámetros)
* Estudiar forma de abordar problema de salas con algún algoritmo, de manera secuencial
* Paralelizar por módulos
* Estudio forma de abordar el problema paralizando módulos
* Estudiar hostfile
* Crear un cluster (manual paso a paso)

# Código

A continuación se detallan convenciones del código, para que los integrantes del equipo trabajen de manera estandarizada.

* Los nombres de las **variables** inician con mayúscula (ejemplo. Nombre\_Usuario)
* Uso de **librería xlnt**

Instalación:

$ git clone<https://github.com/tfussell/xlnt.git>

$ cd xlnt

$ cmake .

$ make -j 2

$ sudo make install

$ sudo ldconfig (Se reemplaza la carpeta xlnt que se subió vacía)

Compilación y ejecución:

$ g++ main.cpp -o ejecutable -std=c++14 -xlnt/include -lxlnt

$ ./tarea -c Cursos.xlsx -d Docentes.xlsx -s Salas.xlsx

* Uso de librería **xlsxio** (xlsx actualizada)

Instalación:

$ sudo apt install libexpat1-dev

$ sudo apt install libzip-dev

$ sudo apt install libminizip-dev

$ sudo apt install doxygen

$ sudo apt install graphviz

Clonar repositorio:

$ git clone https://github.com/brechtsanders/xlsxio

$ cd xlsxio/

Instalar librería xlsxio:

$ cmake -G"Unix Makefiles"

$ make

$ sudo make install

Compilación y ejecución: (ejemplo main.cpp en carpeta de la librería xlsxio)

$ sudo sh -c "echo '/usr/local/lib' >> /etc/ld.so.conf"

$ sudo ldconfig

$ g++ main.cpp -lxlsxio\_write -o excel

$ ./excel

# Ideas

* A la estructura de docentes en vez de dejar como string la disponibilidad de bloques, hacer un vector de bool según disponibilidad.
* Modularizar ingreso a archivos en en main.cpp

# Resolución secuencial

Se inicia con un vector de 39 bloques que representan los días de la semana (de lunes a sábado) con 7 bloques disponibles en la semana y 4 el día sábado. → **vector bloques[38]** (39 bloques semanales)

Dentro de cada espacio de vector bloques tenemos las 42 salas disponibles. → **vector salas[41]** (42 salas disponibles)

\* el vector salas debe tener asignado un campo fijo (sala) y queda completo cuando se le asigna un profesor.

Se recorrerá por columnas iniciando el día lunes con el bloque 1 y finaliza el día sábado con el bloque 4.

Al ingresar al vector de salas se realizará en cada celda lo siguiente:

1) verificar que profesores tengo disponibles en ese bloque de ese día. → **vector disponible[dinámico]**

2) A los profesores disponibles los ordenaré por prioridad, dejando en primera posición a aquellos que imparten bloques con mayor cantidad de horas semanales. → plantear uso de **cola o pila o vector de prioridad**

3) **Restricción**: si el **ramo** comienza con el id: INF se le asigna una sala de laboratorio, si no, da igual.

4) **Restricción**: un **profesor** solo puede ser asignado a un bloque, una sala y un ramo por verificación.

Continúo así llenando sala a sala, una vez estén completas, paso al bloque siguiente. Lo que se va guardando se escribe en un archivo .xlsx de salida. Este tiene formato de:

* Una hoja por sala
* fila : bloques
* columna : días de la semana

En cada celda se escribe el **código\_ramo - id\_profesor**.

\*) **Restricción**: si el **profesor** lleva asignado a 4 bloques ese día, debe continuar el día siguiente.