UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

Parcial II

Docente: Cuadros Valdivia Ana María

Integrantes:

Alva Cornejo, Jose Javier

Diaz Vasquez, Esdras Amado

AREQUIPA-PERÚ

2025

Diagrama de Casos de Uso

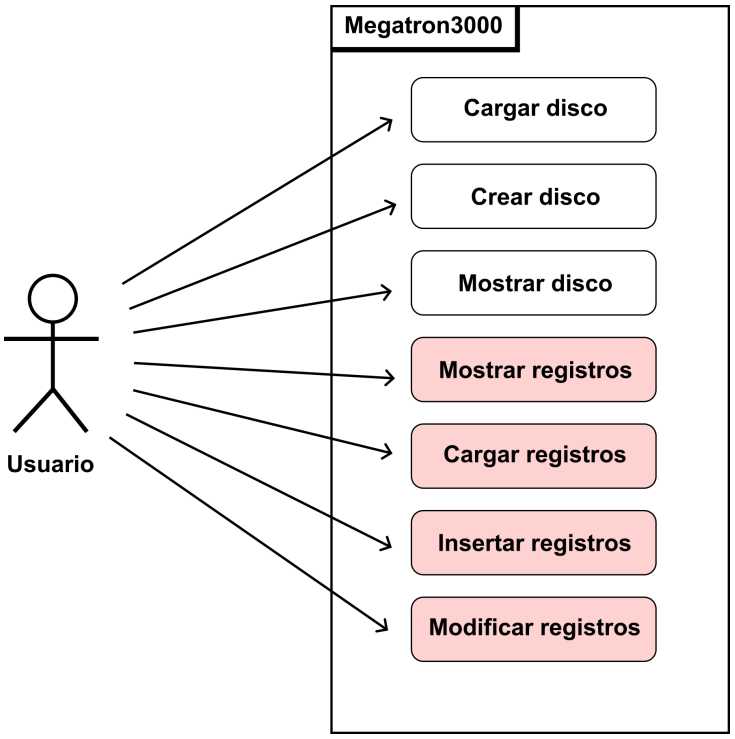
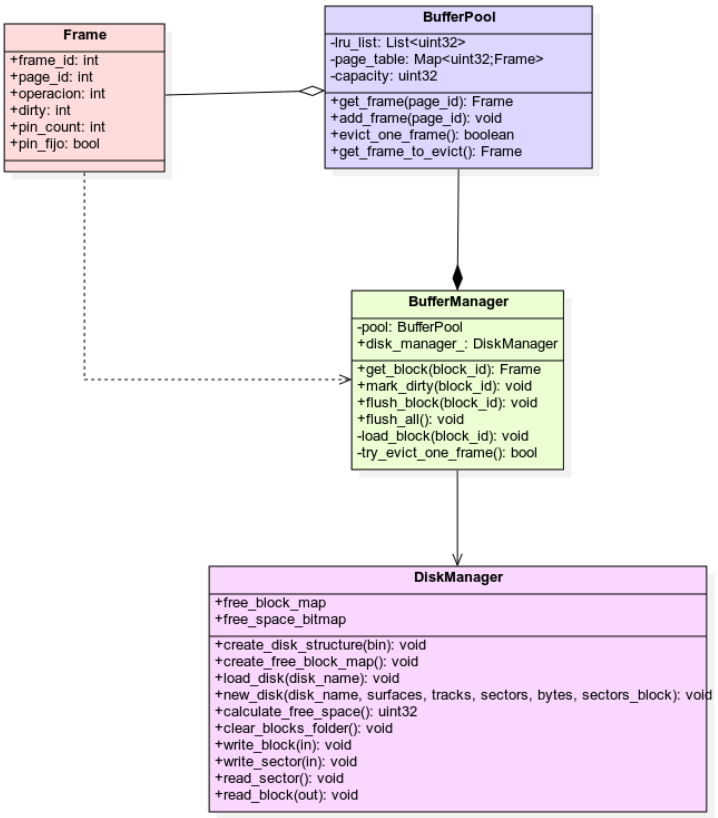


Diagrama de clases



Descripción de clases:

DiskManager

Clase encargada de la creación, el cargado y el control del disco. Toda operación se realiza con el modelo Cilindro-Cabeza-Sector.

`void create_free_block_map()`
Se itera por todo el disco de forma vertical para agrupar bloques, En caso de que al menos un sector este usado, se asume el bloque como ocupado.

`void create_disk_structure(bool bin)`
Esta función crea la estructura física del disco simulando superficies, pistas y sectores, eliminando cualquier estructura previa (excepto metadatos) y generando archivos en formato binario o texto según se indique.

`void load_disk(std::string load_disk_name)`
Carga y calcula specs disco basado en carpetas/files. Se carga `free_space_bitmap`

`void new_disk(std::string new_disk_name, size_t surfaces, size_t tracks_per_surf, size_t sectors_per_track, size_t sector_size, size_t sectors_per_block)`
Crea disco con cierto nombre, implica crear estructura completa, `free_space_bitmap` y guardar datos en sector especial 0.

`void load_disk(std::string load_disk_name);`
Carga y calcula las especificaciones del disco basado en carpetas/files. Se carga `free_space_bitmap`.

BufferManager

Clase que gestiona el acceso eficiente a bloques del disco mediante una memoria intermedia limitada (buffer pool) compuesta por frames, cargando, marcando como modificados ("dirty") y escribiendo al disco solo cuando es necesario.

`Frame &get_block(size_t block_id)`
Esta función retorna una referencia al bloque pedido, cargándolo desde el disco si no está en memoria (buffer pool).

`void mark_dirty(size_t block_id)`
Marca como "modificado" (dirty) el bloque con el ID dado, indicando que su contenido en memoria ya no coincide con el que está en disco y debe ser escrito de vuelta en algún momento.

`void flush_block(size_t block_id)`
Escribe al disco el bloque con ID block_id si está cargado y marcado como modificado (dirty), y luego lo marca como limpio.

`void flush_all()`

Recorre todos los bloques del buffer que pueden ser desalojados y, si alguno fue modificado (dirty), lo escribe en disco.

`void load_block(size_t block_id)`
Carga un bloque desde el disco al buffer pool, haciendo espacio si es necesario al desalojar otros bloques.

`bool try_evict_one_frame`
Intenta desalojar un bloque del buffer pool y, si está modificado (`dirty`), lo escribe en disco antes de retirarlo.

BufferPool

Clase que administra en memoria un conjunto limitado de bloques (Frames) usando una política LRU para decidir qué eliminar cuando el buffer está lleno.

`Frame &get_frame(size_t page_id)`
Función que devuelve el bloque con page_id desde el buffer y actualiza su posición en la lista LRU para marcarlo como el más recientemente usado.

`void add_frame(size_t page_id, std::unique_ptr<Frame> frame)`
Inserta un nuevo bloque en el buffer, y si ya está lleno, primero elimina el menos usado (según LRU) para hacer espacio.

`bool evict_one_frame()`
Elimina del buffer el bloque menos recientemente usado (al final de la LRU), pero no lo guarda si estaba modificado: eso debe hacerlo quien llama.

`std::unique_ptr<Frame> get_frame_to_evict()`
Elimina y retorna el bloque menos recientemente usado del buffer pool, permitiendo que el llamador decida qué hacer con él (por ejemplo, escribirlo en disco si está dirty).

Frame

Representa un bloque de memoria con su identificador, el ID de la página que contiene, su tipo de operación, si fue modificado (dirty), cuántas veces está siendo usado simultáneamente (pin_count) y si está fijado para no ser desalojado (pin_fijo).

`dirty` permite decidir si hay que escribir antes de eliminar.
`pin_count` ayuda a proteger páginas en uso activo.
`pin_fijo` garantiza que ciertas páginas nunca se desalojen, sin importar la política.
`operacion` puede ser útil para análisis, estadísticas o visualización del uso del buffer.