1 - Componentes

5g_auth_platform.c	
validate_settings	Verifica se os parâmetros do ficheiro de configuração estão dentro dos limites aceitáveis.
system_panic	Interrompe todas as operações, pode ser chamado por qualquer processo.
close_system_manager	Mata todos os processos filho, fecha *a message queue* e liberta os semáforos e *shared memory*.
close_monitor_engine	Fecha o *monitor_engine*
close_authorization_request_manager	Mata todos os processos filho, dá *unlink/close* às *named/unnamed pipes*
close_authorization_engine	Fecha o *authorization_engine* após concluir a tarefa naquele momento
create_sharedmem	Criação da *shared memory*
append_logfile	Anexa uma string ao *logfile*
read_configfile	Faz a leitura do ficheiro de configuração e define as configurações globais
parallel_AuthorizationRequestManager	Cria um *Authorization Request Manager*
parallel_AuthorizationEngine	Cria um *Authorization Engine*
parallel_MonitorEngine	Cria um *Monitor Engine*
receiver_ARM	Aguarda para que algo seja escrito na *mobile_pipe* ou *backend_pipe*, e escrever para a queue correta, *vídeo_queue* ou *other_queue*, sendo o pedido descartado caso esteja cheia.
sender_ARM	Lê das *queues vídeo_queue* e *other_queue* até estas estarem vazias, e escreve para a *pipe* do próximo *authorization engine* disponível.
check_requesttype	Verifica se uma *request* vai para a *vídeo_queue* ou *other_queue*
kill_allchildren	Mata todos os processos filho, caso passada a flag *ARM_flag,* é feita uma *request* exclusivamente para esvaziar as *pipes* do *authorization engines*
handle_request	Aceita ou rejeita o pedido conforme o plafond do cliente.
check_plafond	Verifica se o cliente tem plafond suficiente
create_client	Criação de um cliente
delete_client	Remoção de um cliente
backoffice_user.c	
auth5g_request	Cria uma request que é escrita na *backend_pipe*
user_input	O user pode apresentar as estatísticas referentes aos consumos dos dados nos vários serviços (0) ou limpar as estatísticas relacionadas calculadas até ao momento pelo sistema (1)
messagequeue_response	Espera pela mensagem, e dá display quando a recebe
mobile_user.c	
validate_settings	Verifica se os argumentos passados estão dentro dos limites permitidos
timed_request	É feita uma request, a cada *n* milisegundos
messagequeue_response	Leitura da message queue
auth5g_register	É feito o registo do *user* no formato: *pid#plafond*
auth5g_request	É feita uma request no formato: *pid#type#size*
response_handler	Trata da resposta do sistema
Message_struct.c	
create_queue	Cria a *queue*
count_queue	Retorna o número de mensagens na *queue*
write_queue	Escreve uma mensagem para a *queue*
read_queue	Lê uma mensagem da *queue*

2 - Descrição

5g_auth_plataform

O programa começa por dar handle aos sinais e fazer leitura e validação do ficheiro de configuração (“*5gconfig.config”*), atribuindo os parâmetros às variáveis correspondentes. É depois criada a *message queue* e feita a inicialização dos semáforos e da memoria partilhada, e inicializados os processos para o A*uthorization Request Manager* (ARM) e monitor engine.

Dentro do ARM, vão ser criados os *Authorization Engines* (AE), as threads *sender* e *receiver,* bem como as respetivas pipes.

O Monitor Engine (ME) cria uma thread para periodicamente (30s) enviar notificações para o backoffice, e monitoriza os clients, enviando avisos e fechando-os.

ARM

O *receiver* vai criar (caso ainda não existam) a BACKEND pipe e MOBILE pipe, destinadas ao *backoffice user* e *mobile users*, respetivamente. Espera que seja escrita uma mensagem numa destas pipes, e depois escreve para a devida queue, *video* ou *other*. O *sender* fica à espera do sinal para ler das queues, e espera por AEs_sem para escrever para o próximo AE livre. Caso as queues encham, ARM cria um novo AE, e destroi-o quando este já não for necessário.

Authorization Engine

O AE lê da unnamed pipe, dá handle ao pedido feito, espera durante um tempo estipulado e envia resposta para o cliente, através da message queue.

Monitor Engine

O ME, associado a um semaphore iniciado com valor 1, espera com sem_wait por outro processo que o informe através de sem_post que há uma mudança relevante num cliente. Este analisa-os, enviado notificações, e terminando sessões conforme o necessário.

Mobile_User

Quando é criado um mobile user, são verificados os parâmetros, é aberta a *message_queue*, e este tem como função ir periodicamente enviando pedidos através da MOBILE pipe, com 3 threads, para música, vídeo e social. Ainda utiliza uma outra thread para tratar as mensagens da *message_queue*, que é destruída quando esta estiver vazia.

BackOffice_User

O BackOffice User utiliza duas threads, uma para escrever o input do utilizador na BACKEND pipe e outra para ler as mensagens na *message queue*. Tem como função apresentar a estatísticas referentes aos consumos nos vários serviços, e para dar *reset* a estes, na memória partilhada.

Encerramento do programa

Quando o sistema recebe o sinal SIGINT, é ignorado em todos os processos com excepção do system manager, que vai replicar o sinal SIGQUIT que vai proceder à limpeza dos restantes componentes. O ARM vai fechar as named e unnamed pipes, e dar delete memoria partilhada dos AE, bem como unlink ao sem. O AE vai acabar o processo a decorrer, e é terminado depois. Por fim, o system manager faz a limpeza da message queue, dos semáforos e da memória partilhada.