Justificação da alteração feita à classe CloudByte

Na classe Cloudbyte adicionei uma variável global do tipo *Semaphore* para conseguir implementar a sincronização do byte quando está a ser corrigido(para que não fosse corrigido várias vezes).

Optei por utilizar um semáforo incializado a um, como se fosse um cadeado, porque com semáforos é possível diferentes threads atualizarem o seu número de *permits*, optimizando assim, a meu ver, o meu código e uso de recursos.

Outras soluções e breve justificação pela qual não foram escolhidas:

-

```
while (true) {
    cb = this.storageNode.getFilePosition(i);
    if(!cb.isParityOk() && cb.tryLock()) {
        this.storageNode.repairByte(i);
        cb.unlock();
    }
    i++;
    if (i == StorageNode.FILE_SIZE)
        i = 0;
}
```

Não implementei esta solução porque bloqueava a thread inspetora e a impossibilitava de continuar a verificar se outros *Cloud Bytes* estavam corrompidos ou não, enquanto não tivesse acabado de corrigir o que detetou naquele instante.

- A outra solução era tentar "trancar" o *Cloud Byte* dentro da própria thread já lançada para o corrigir, se desse para "trancar" prosseguia-se com a correção do *Cloud Byte*, se não, a thread acabava. o problema que com esta solução, considerei um "abuso" de recursos estar sempre a lançar novas threads só para morrerem nanosegundos depois sem terem servido nenhum propósito.