 **MIAGE IF Apprentissage**

Projet Composant

**Spécifications du Composant 3**

**“Mineur”**

Réalisé par: **Achraf LAABASSI**

**Antoine COUFLEAU**

**Mohamed KHARRAZ**

Version: **1.2**

**Description**

**Contexte**

Dans le cadre de développement d’un logiciel Blockchain, ce dernier sera divisé en plusieurs modules, indépendants et interagissent entre afin de fournir la prestation globale du logiciel.

L’ensemble des modules du logiciel est comme suit:

Module n°1: Interface fichier.

Module n°2: Portefeuille/Wallet.

Module n°3: Mineur.

Module n°4: Hacheur SHA-256.

Module n°5: Vérificateur du bloc.

Module n°6: Générateur de clé.

Ce document concerne la spécification du module 3 du logiciel, à savoir “Mineur”.

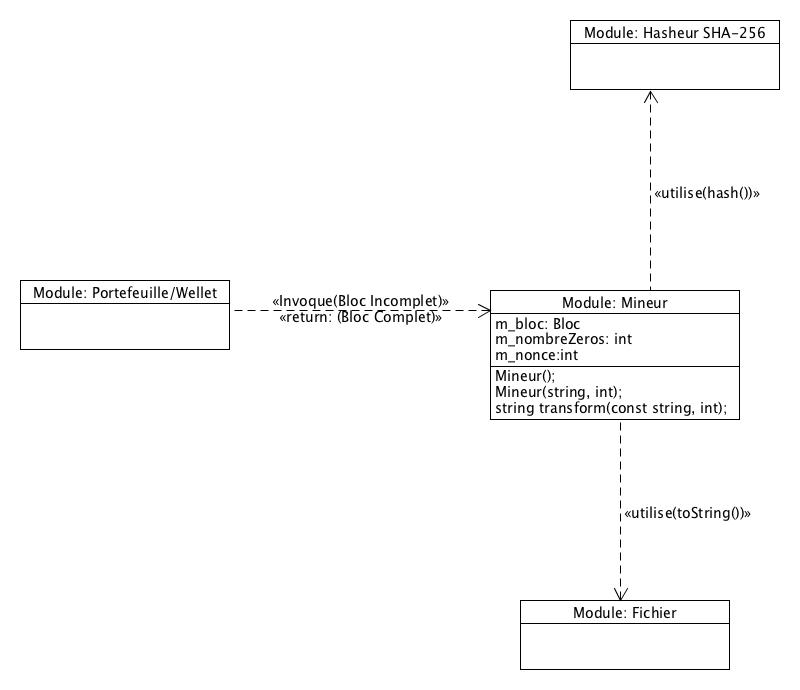
**Fonctionnalités du module 3**

Le module 3 reçoit un bloc, et un nombre de zéros, et alors il doit utiliser la fonctionnalité du hashage fournit par le module 4, et le nonce, afin de générer un hash ayant la particularité de commencer par le nombre de zéros indiqués.

À noter que, le nonce doit être incrémenté jusqu’à trouver le hash qui correspond à la forme voulu, tout en dépassant pas un nombre maximal, faute de quoi il faut signaler une erreur.

Le module retourne un bloc complet(contenant le hash et le nonce) au Wallet.

**Interface et interaction avec les autres composants**



Le module utilise la fonction toString() du module fichier afin de passer une chaîne de caractères au Module SHA-256 pour effectuer le hash.

**Résumé: Headers**

|  |
| --- |
| #ifndef MINEUR\_H  #define MINEUR\_H  #include <string>  class Mineur  {  protected:  // Le nombre de 0 dont doit commencer la châine hachée  int m\_nombreZeros;  // Le nonce utilisé pour générer la bonne forme du hash  int m\_nonce;  //Le bloc incomplet reçu par le module Wallet  Bloc m\_bloc;  //La chaîne hachée  str::string m\_hash;  public:  Mineur(Bloc bloc, int nombreZeros);  /\*  \* utilise le bloc et le nombre de 0 pour effectuer le hash  \* et le mettre dans l’attribut hash  \* @Return: boolean(true en cas de succès et false sinon)  \*/  boolean transform();  /\*  \* retourne le bloc complet  \* @Return: Bloc  \*/  Bloc getM\_bloc();  };  #endif |

**Description:** LA classe Mineur est responsable des fonctionnalités du module.

Elle contient trois attributs membres: nombre de zéros, bloc et nonce.

|  |
| --- |
| **Mineur**(**Bloc bloc, int nombreZeros**) |

constructeur à partir du bloc et nombre de zéros.

|  |
| --- |
| **int getNonce**() |

constructeur à partir du bloc et nombre de zéros.

|  |
| --- |
| **boolean** **transform**() |

utilise le bloc incomplet reçu, et le nombre de zéros demandés, afin de créer une chaine hachée avec la difficulté demande. En fin elle met le nonce, et le hash dans le bloc.

Afin de générer, la bonne forme, il faut utiliser le Nonce, et l’incrémenter jusqu’à trouver la chaîne hashé respectant la difficulté indiqué.

Il faut également utiliser la fonction **toString(false)** développée par le Module Fichier.

**Remarque**: Il y’a un nombre de Nonce maximale à ne pas dépasser, en cas contraire, il faut signaler une erreur.

|  |
| --- |
| **Bloc** **getM\_bloc**() |

Retourne le bloc complet

**Cas d’erreurs**

* Nombre maximal de Nonce dépassé.

**Test**

**Plan de test**

Le test devrait se dérouler selon le processus suivant:

* Constituer un bloc.
* Appelez le mineur.
* Vérifier le Hash.
* Vérifier le Nonce.

**Programme de test: mode d’emploi**

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Résultat attendu | OK/KO | Commentaire |
| Tester le programme en ligne de commande, avec en paramètre un bloc et nombre de zéros | Bloc hashé débutant avec un nombre de zéros correct |  |  |
| Même test | Vérifier que le Nonce était incrémenté |  |  |