1. تعريف الكلاس Block

الكلاس Block يمثل كتلة في سلسلة البلوكتشين. تحتوي الكتلة على:

- **timestamp: الوقت الزمني يمثل تاريخ إنشاء الكتلة.**
- **transactions: المعاملات قائمة بالمعاملات في الكتلة.**
- **previousHash: الهاش السابق يربط الكتلة بالكتلة السابقة لتشكيل سلسلة.**
- **hash: الهاش يتم حسابه باستخدام خوارزمية SHA-256 لتمييز الكتلة.**
- **nonce: الـ رقم يستخدم في عملية التعدين.**

الكلاس يحتوي على دالتين رئيسيتين:

- **calculateHash() لحساب الهاش باستخدام crypto.**
- **mineBlock(difficulty) لمحاولة إيجاد هاش يبدأ بعدد معين من الأصفار (حسب مستوى الصعوبة) باستخدام brute force عبر تعديل الـ nonce.**

---

2. تعريف الكلاس Blockchain

الكلاس Blockchain يُستخدم لإنشاء سلسلة الكتل وإدارتها. يحتوي على:

- **chain: المصفوفة التي تحتوي على جميع الكتل.**
- **difficulty: مدى صعوبة التعدين (عدد الأصفار المطلوبة في بداية الهاش).**
- **pendingTransactions: المعاملات التي لم تُدرج بعد في كتلة.**
- **miningReward: مكافأة التعدين للمُعدّن.**

ويحتوي على وظائف:

- **createGenesisBlock(): ينشئ الكتلة الأولى (الكتلة الأصلية).**
- **getLatestBlock(): يرجع آخر كتلة في السلسلة.**
- **minePendingTransactions(miningRewardAddress): ينشئ كتلة جديدة من المعاملات المعلقة ويكافئ المعدّن.**
- **addTransaction(transaction): يضيف معاملة جديدة إلى قائمة المعاملات المعلقة.**
- **getBalance(address): يحسب الرصيد الخاص بعنوان معين عن طريق تتبع المعاملات.**

---

3. إعداد شبكة P2P باستخدام WebSocket

يتم إعداد خادم WebSocket على المنفذ WS_PORT في الشبكة (nodes) ، ويُستخدم لتبادل الكتل الجديدة بين العقد:

- يتم حفظ كل اتصال WebSocket في مصفوفة sockets.
- عند استقبال رسالة من عقدة أخرى من نوع NEW_BLOCK ، يتم إضافة الكتلة إلى سلسلة البلوكتشين المحلية ثم يتم بثها إلى باقي العقد باستخدام broadcast.

---

4. وظائف المساعدة (Helper Functions)

وظيفتان للتعامل مع ملفات الإعداد:

- loadConfig(): يقرأ الإعدادات من ملف config.json.
- saveConfig(config): يحفظ التعديلات على الملف نفسه.

---

5. التوثيق باستخدام JWT

دالة authenticateToken تُستخدم كوسيط Middleware لحماية المسارات:

- تقوم بالتحقق من صحة التوكن المُرسل في الهيدر Authorization.
- في حال عدم وجود توكن أو وجود توكن غير صالح، يتم رفض الطلب.
- في حالة النجاح، يتم إرفاق بيانات المستخدم مع الطلب للسماح له بتنفيذ العمليات المحمية.

---

6. مسار تسجيل الدخول /auth/login

يُستخدم username و apiKey الموجودين في ملف الإعداد لتسجيل الدخول باستخدام:

- يتم البحث عن المستخدم في قائمة users.
- إذا تم العثور عليه، يتم إنشاء JWT Token صالح لمدة ساعة وإرساله للمستخدم.
- إذا لم يتم العثور عليه، يتم الرد بخطأ "Invalid credentials".

---

7. استعلام عن الرصيد /balance/:username

يقوم بحساب الرصيد الكلي للمستخدم من ملف الإعداد بالإضافة إلى الرصيد الناتج عن المعاملات الموجودة في البلوكتشين:

- يتم استخدام blockchain.getBalance() لحساب المعاملات المسجلة.
- الرصيد الكلي هو مجموع الرصيد المحفوظ في ملف الإعداد + رصيد البلوكتشين.

---

8. معرفة المعاملات المعلقة/pending-transactions

يرجع للمستخدم جميع المعاملات التي لم يتم تعدينها بعد ضمن الكتل.

---

9. إضافة معاملة جديدة/transaction

يُستخدم هذا المسار لإرسال مبلغ إلى مستخدم آخر:

- **يتم التحقق من صحة المستخدمين.**
- **يتم التأكد من أن المرسل لديه رصيد كافٍ.**
- **تتم إضافة المعاملة إلى pendingTransactions.**
- **يتم تحديث أرصدة المستخدمين في config.json.**

---

10. تعدين كتلة/mine

يسمح للمستخدم بتعدين المعاملات المعلقة:

- **يتم إنشاء كتلة جديدة تحتوي على جميع المعاملات المعلقة.**
- **بعد التعدين، يتم مكافأة المعدّن بإضافة معاملة له في قائمة المعاملات التالية.**
- **يتم بث الكتلة الجديدة إلى جميع العقد في الشبكة.**

---

11. عرض سلسلة البلوكتشين/blockchain

يعرض جميع الكتل في السلسلة للمستخدم بعد التوثيق، بحيث يستطيع التحقق من جميع المعاملات والتعدين.

---

12. تشغيل الخادم

يبدأ الخادم في الاستماع على المنفذ PORT الخاص بـ HTTP وعلى المنفذ WS_PORT الخاص بـ WebSocket.