

## ## Présentation du projet

**FortyTwo42 (F42)** est un token ERC-20 déployé sur la blockchain Ethereum.

### ### Informations du token

- **Nom** : FortyTwo42
- **Symbole** : F42
- **Supply initiale** : 1,000,000 F42
- **Blockchain** : Ethereum
- **Réseau de test** : Sepolia
- **Standard** : ERC-20

### ### Objectif du projet

Créer un token fonctionnel respectant les contraintes imposées : inclure "42" dans le nom.

## ## Architecture technique

### ### Stack technologique

Composant	Technologie
<b>Langage smart contract</b>	Solidity
<b>Framework de développement</b>	Hardhat
<b>Bibliothèques</b>	OpenZeppelin
<b>Langage de scripting</b>	TypeScript
<b>Provider blockchain</b>	Alchemy
<b>Réseau de test</b>	Ethereum Sepolia
<b>Explorateur</b>	Etherscan Sepolia

## ## Explication des technologies utilisées

### ### Alchemy

#### **Qu'est-ce que c'est ?**

Alchemy est une plateforme qui permet aux développeurs d'accéder facilement à la blockchain sans devoir gérer leur propre nœud Ethereum.

#### **Pourquoi l'utiliser ?**

- Évite de synchroniser un nœud complet (plusieurs centaines de Go)
- Fournit une API fiable et rapide
- Offre des outils de monitoring et de debugging

#### **Comment ça marche ?**

1. On crée une App sur le dashboard Alchemy
2. On récupère une clé API et une URL RPC
3. Hardhat utilise cette URL pour envoyer les transactions

### ### Hardhat

#### **Qu'est-ce que c'est ?**

Hardhat est un framework de développement Ethereum qui permet de compiler, tester, débbugger et déployer des smart contracts.

#### **Fonctionnalités utilisées :**

- Compilation des contrats Solidity
- Exécution de tests automatisés
- Déploiement sur différents réseaux
- Génération de types TypeScript (via TypeChain)

### ### OpenZeppelin

#### **Qu'est-ce que c'est ?**

OpenZeppelin est une bibliothèque de smart contracts audités et sécurisés. Elle fournit des implémentations standards des tokens (ERC-20) et des mécanismes de sécurité.

#### **Contrats utilisés dans FortyTwo42 :**

Contrat	Rôle	Justification
<b>ERC20</b>	Implémentation standard du token	Garantit la compatibilité avec tous les wallets et exchanges
<b>Ownable</b>	Gestion de la propriété	Permet de restreindre certaines actions au créateur du contrat
<b>Pausable</b>	Système de pause	Permet de suspendre les transferts en cas d'urgence
<b>ReentrancyGuard</b>	Protection contre les attaques	Empêche les appels récursifs malveillants

### ### 🌐 Sepolia Etherscan

**\*\*Qu'est-ce que c'est ?\*\***

Sepolia Etherscan est un explorateur de blockchain pour le réseau de test Sepolia. Il permet de visualiser toutes les activités on-chain.

**\*\*URL :\*\*** <https://sepolia.etherscan.io/>

**\*\*Ce qu'on peut y voir :\*\***

- ✅ Transactions (statut, gas utilisé, timestamp)
- ✅ Adresses (solde, historique des transactions)
- ✅ Smart contracts (code source, ABI, événements)
- ✅ Blocs (mineurs, transactions incluses)

**\*\*Exemple pour notre token :\*\***

Une fois déployé, on peut consulter :

- L'adresse du contrat
- Le code source vérifié
- Les holders du token
- L'historique des transferts

### ### 📖 Comprendre le Gas

Terme	Définition
<b>**Gas**</b>	Unité de mesure du travail effectué sur Ethereum (comme du carburant)
<b>**Gas Price**</b>	Prix par unité de gas (en Gwei)
<b>**Transaction Fee**</b>	Coût total = <code>`Gas utilisé × Gas Price`</code>

## ## Choix techniques et justifications

### ### 1. Pourquoi Ethereum et pas une autre blockchain ?

**\*\*Avantages d'Ethereum :\*\***

- ✅ Blockchain la plus mature et sécurisée
- ✅ Écosystème le plus développé (wallets, outils, documentation)
- ✅ Standard ERC-20 universellement reconnu
- ✅ Large communauté de développeurs

**\*\*Pourquoi Sepolia (testnet) ?\*\***

- Pas de coût réel (ETH gratuit via faucets)
- Conditions similaires au mainnet
- Explorateur Etherscan disponible
- Permet de tester en toute sécurité

### ### 2. Pourquoi Solidity 0.8.22 ?

**\*\*Avantages de cette version :\*\***

- Protection automatique contre les overflow/underflow
- Compatible avec OpenZeppelin
- Stable et largement adoptée
- Optimisations du compilateur

### ### 3. Pourquoi OpenZeppelin ?

**\*\*Sécurité avant tout :\*\***

- Contrats audités par des experts
- Utilisés par des milliers de projets
- Mises à jour régulières
- Documentation complète

### ### 4. Pourquoi Hardhat ?

- **Expérience Développeur :** Il offre des tests et un débogage supérieurs avec des stack traces claires.

- **Modernité :** Support natif de TypeScript, essentiel pour les grands projets.

- **Flexibilité :** Un écosystème de plugins riche qui permet d'adapter l'environnement à tous les besoins.

### ### 5. Pourquoi Alchemy ?

Alchemy est l'infrastructure Web3 de choix pour les applications qui exigent performance et fiabilité.

- Fiabilité Supérieure : Connexion ultra-stable et rapide aux blockchains (Ethereum, etc.).
- Outils Avancés : Dashboard puissant et monitoring détaillé pour un débogage et une analyse des requêtes inégalés.
- Expérience Développeur : Niveau gratuit généreux et support de haute qualité, simplifiant l'ensemble du cycle de vie de l'application.

### #### 4. Constructor

```
``solidity
constructor() ERC20("FortyTwo42", "F42") Ownable(msg.sender) {
    _mint(msg.sender, 1000000 * 10**decimals());
}
```

#### \*\*Décomposition :\*\*

1. **ERC20("FortyTwo42", "F42")**
  - Définit le nom et le symbole du token
  - Appelé automatiquement au déploiement
2. **Ownable(msg.sender)**
  - Définit le déployeur comme propriétaire
  - `msg.sender` = adresse qui déploie le contrat
3. **\_mint(msg.sender, 1000000 \* 10\*\*decimals())**
  - Crée 1 million de tokens
  - `decimals()` retourne 18 (standard ERC-20)
  - `10**18` = 1 token avec 18 décimales
  - Total :  $1,000,000 \times 10^{18} = 1,000,000.000000000000000000$  tokens

### ## Déploiement et utilisation

#### ### Étape 1 : Compilation

```
``bash
npx hardhat compile
```

#### ### Étape 2 : Tests locaux

```
``bash
npx hardhat test
```

#### ### Étape 3 : Déploiement sur Sepolia

```
``bash
npx hardhat run scripts/deploy.ts --network sepolia
```

#### \*\*Output attendu :\*\*

```
🚀 Début du déploiement FortyTwo42...
📍 Déploiement avec le compte: 0xYOUR_ADDRESS
💰 Balance du compte: 0.5 ETH
✅ Déploiement terminé avec succès!
📍 Contrat déployé à: 0xCONTRACT_ADDRESS
📦 Nom du token: FortyTwo42
🎯 Symbole: F42
📊 Supply totale: 1000000.0
💰 Balance du déployeur: 1000000.0
🔍 Voir sur Etherscan: https://sepolia.etherscan.io/address/0xCONTRACT_ADDRESS
```

#### ### Étape 4 : Vérification sur Etherscan

```
``bash
npx hardhat verify --network sepolia 0xCONTRACT_ADDRESS
```

#### ### Étape 5 : Ajout à MetaMask

1. Ouvrir MetaMask sur le réseau Sepolia
2. Cliquer sur "Import tokens"
3. Entrer l'adresse du contrat : `0xCONTRACT_ADDRESS`
4. Le symbole F42 et les décimales 18 s'ajoutent automatiquement
5. Confirmer

## **\*\*Résultat :\*\***

The top screenshot shows the Etherscan 'Contract' page for the address 0xD7aa0B2126c4F91c516dc701eb8fb76989C2737E. It features three main sections: Overview (ETH Balance: 0 ETH), More Info (Contract Creator: 0x08d5A60c...7E6495465, 25 secs ago), and Multichain Info (N/A). The bottom screenshot shows the 'Token FortyTwo42 (F42)' page. It includes Overview (Max Total Supply: 1,000,000 F42, 1 Holder, 1 Total Transfer), Market (Onchain Market Cap, Circulating Supply Market Cap), and Other Info (Token Contract: 0xD7aa0B2126c4F91c516dc701eb8fb76989C2737E). Below these is a 'Transfers' table with one entry: Transaction Hash 0x29c4921a6d..., Method 0x0806040, Block 9312476, Age 32 secs ago, From 0x00000000...00000000, To 0x08d5A60c...7E6495465, Amount 1,000,000.

## **## Cas d'usage et démonstration**

**\*\*Scénario :\*\*** Alice envoie 100 F42 à Bob

**\*\*Code :\*\***

```
``typescript
await fortyTwo42.transfer(bobAddress, ethers.parseEther("100"));
``
```

**\*\*Ce qui se passe on-chain :\*\***

1. Vérification : Alice a-t-elle 100 F42 ?
2. Déduction : Balance d'Alice - 100
3. Ajout : Balance de Bob + 100
4. Événement : `Transfer(Alice, Bob, 100)`

### **### Commandes utiles**

```
``bash
# Compilation
npx hardhat compile

# Tests
npx hardhat test
npx hardhat test --coverage

# Déploiement
npx hardhat run deployment/deploy.ts --network sepolia

# Vérification
npx hardhat verify --network sepolia <CONTRACT_ADDRESS>

# Console interactive
npx hardhat console --network sepolia

# Nettoyage
npx hardhat clean
``
```

### **### Variables d'environnement (.env)**

```
``bash
# Alchemy
ALCHEMY_API_KEY=your_alchemy_key

# Wallet
PRIVATE_KEY=0xYOUR_PRIVATE_KEY

# Etherscan (optionnel)
ETHERSCAN_API_KEY=your_etherscan_key
``
```