Testarea unei Retele Blockchain - Final Version 3/3

Magazin NFT cu abonamente

1.Prezentare Generală

Această platformă bazată pe Solidity permite comercianților să creeze planuri de abonament în blockchain, unde fiecare abonament este reprezentat printr-un token NFT unic (ERC-721). Utilizatorii pot achiziționa, primi refund-uri și vizualiza deținerea NFT-urilor care simbolizează abonamente.

Componente:

1. `AbonamentNFT.sol` — contract NFT ERC-721 pentru abonamente
2. `PlanAbonament.sol` — logica vânzării, refund-urilor și gestionarea planului
3. `RetailerFactory.sol` — fabrică de planuri care poate crea instanțe de `PlanAbonament`
4. `Lock.sol` — exemplu de contract auxiliar pentru blocarea fondurilor
5. Teste Hardhat complete: unitare, de integrare, proprietăți și gaze

AbonamentNFT.sol

Descriere

Implementă ERC721 cu extensiile `URIStorage` și `Enumerable`, permite emiterea de NFT-uri ca dovadă a achiziției unui abonament.

Variabile

- `pretAbonament`: prețul plătit pentru abonament
- `abonamenteVandute`: contor de NFT-uri emise
- `abonamentPlanContract`: adresa planului asociat
- `metadataURI`: link IPFS pentru metadatele NFT-ului

Funcții

- `cumparaAbonament(address)` — emite NFT și trimite ETH planului
- `refundTransfer(from, to, id)` — mută NFT-ul către plan pentru refund
- `tokenURI(id)` — returnează URI-ul NFT-ului

Modificatori

- `nonReentrant` (pentru protecție contra reentrancy attack)

Plan Abonament.sol

Descriere

Contractul responsabil pentru vânzarea NFT-urilor (abonamentelor), refund-uri și administrarea logicii comerciale.

Variabile

- `pretSubscriptie`: prețul unui abonament
- `durata`: durata abonamentului (în zile)
- `abonamenteDisp`: abonamente rămase
- `metadataURI`: IPFS link
- `cancelled`: stare a planului
- `retrageri`: mapping pentru sume refundate per adresă

Funcții

- `cumparaSubscriptie()` — validează ETH și emitere NFT
- `cancelPlan()` — anulează vânzările
- `refundSubscriptie(tokenId)` — mută NFT-ul înapoi și salvează suma refund
- `withdraw()` — utilizatorul împretează refundul

- `calculeazaDiscount()` — calculează reduceri pe bază de volum
- `testCallRefund()` — apel de test pentru refund

Loguri:

- `RefundRequested`
- `AbonamentCumparat`

RetailFactory.sol

Descriere

Contract care permite crearea planurilor de abonament. Numai deținătorul (owner) poate crea planuri.

Funcții

- `createSubscriptionPlan(...)` — creează un nou contract `PlanAbonament` și emite un eveniment `PlanCreated`
- `abonamente(uint256)` — mapping pentru adresele planurilor create

Lock.sol (Auxiliar)

- Contract simplu pentru blocarea fondurilor ETH până la o anumită dată
- Exemplu pentru `time lock`
- Utilizează `withdraw()` protejat de timestamp și ownership

2. Testare Extinsă și Automatizată

Testarea este o componentă esențială a platformei. Include:

Teste Unitare (AbonamentNFT, PlanAbonament, RetailerFactory)

- Validarea parametrilor constructorului (`price > 0`, `URI != empty` etc.)
- Emitere corectă NFT și transfer ETH
- Verificarea interfețelor ERC721, Metadata, Enumerable

- Acces controlat la funcții critice (refund doar de la contractul plan)

Teste de Integrare

- Cumpărarea abonamentelor din planuri create prin `RetailerFactory`
- Confirmare NFT mint + reducere `abonamenteDisp`
- Verificarea transferului de ownership NFT
- Refund și retragere ETH după `cancelPlan`

Property-Based Testing (cu `fast-check`)

- Se testează crearea de planuri și cumpărarea de abonamente folosind date aleatorii
- Asigură robustețea sistemului în fața unor intrări neprevăzute

Testare de Gaz

- Măsurare costuri de tranzacții pentru cumpărare abonament
- Optimizare: verificare unde se consumă cel mai mult gas

Exemple de test:

```
expect(await nft.balanceOf(other.address)).to.equal(0);
await nft.connect(other).cumparaAbonament(other.address, { value: price });
expect(await nft.balanceOf(other.address)).to.equal(1);
```

Coduri de revert validate

- `"Price cannot be zero"`
- `"Invalid contract address"`
- `"Metadata URI cannot be empty"`
- `"Suma ETH gresita"`
- `"Abonament anulat"`
- `"No subscriptii available"`
- `"Incorrect ETH amount"`
- `"Plan is not cancelled"`
- `"Not the owner"`
- `"Neautorizat"`

3. Securitate

- `nonReentrant` pe funcțiile critice

- Validări stricte la creare

- Doar owner-ul poate anula planul sau crea planuri
- Transfer ETH doar cu `require(success)`

4. Flux de Utilizare

- **Creează planul**: factory.createSubscriptionPlan("Netflix", 0.1 ether, 30, 100, "descriere", "ipfs://metadata")
- **User cumpără abonament**: plan.cumparaSubscriptie{value: 0.1 ether}();
- **Retailer anulează planul**: plan.cancelPlan();
- **User cere refund**: plan.refundSubscriptie(tokenId);
- **User retrage ETH**: plan.withdraw();

5. Sugestii de Extindere

- Integrare Chainlink pentru metadata aleatorie NFT
- Reînnoire automata (cu Chainlink Keepers)
- Plăți ERC-20 sau stablecoin
- UI React + Wagmi/Viem pentru interacțiune frontend

6. Deployment

```
const [deployer] = await ethers.getSigners();
const Factory = await ethers.getContractFactory("RetailerFactory");
const factory = await Factory.deploy(deployer.address);
await factory.createSubscriptionPlan(...);
```

7. Interfețe Suportate

- ERC721: `0x80ac58cd`
- Metadata: `0x5b5e139f`

- Enumerable: `0x780e9d63`
- ERC721Receiver: `0x150b7a02`

Raport privind Utilizarea Inteligenței Artificiale în Dezvoltarea Platformei de Abonamente NFT

Scopul utilizării AI

Inteligența Artificială a fost folosită în acest proiect ca **instrument de suport** pentru accelerarea procesului de dezvoltare, validare și documentare, fără a înlocui gândirea critică, deciziile de design sau validările manuale.

Zone în care AI-ul a fost util

1. Asistență în scrierea contractelor

AI-ul a ajutat în:

- structurarea logicii de bază a contractelor (`AbonamentNFT`, `PlanAbonament`, `RetailerFactory`);
- integrarea interfețelor și pattern-urilor OpenZeppelin;
- generarea rapidă a codului repetitiv (constructori, validări simple).

Codul final a fost adaptat și ajustat manual pentru logica specifică proiectului.

2. Propuneri pentru testare

AI-ul a oferit:

- șabloane de testare unitare;
- sugestii pentru scenarii de integrare;
- idei pentru testare property-based cu `fast-check`.

Testele au fost revizuite, extinse și rafinate ulterior.

3. Redactarea documentației

AI-ul a fost util în:

- structurarea documentației tehnice;
- explicarea fluxurilor și interacțiunilor între contracte;
- redactarea rapidă a descrierilor și exemplelor de utilizare.

Conținutul a fost verificat și adaptat pentru a reflecta exact implementarea reală.

Limitări ale AI-ului

- Nu a putut anticipa toate cazurile de edge-case sau probleme logice specifice.
- A necesitat ajustări manuale pentru cod optimizat și securizat.
- Nu a înlocuit niciodată testarea efectivă în rețea sau simulări complexe.