

Testarea unei Retele Blockchain

Magazin NFT cu abonamente

Cuprins

Partea 1: Alpha-Alpha Version (1/3)

1. *Obiectiv Alpha-Alpha*
2. *Contextul General al Temei*
3. *Arhitectura Aplicației*
4. *Colectarea Materialelor Esențiale*
5. *Definiții Esențiale*
6. *Analiza Aplicațiilor Existente*
7. *Servicii și Resurse Tehnice*
8. *Funcționalități ale Aplicației Client*
9. *Setup Inițial*

Partea 2: Beta-Beta Version (2/3)

1. *Obiectivul Versiunii Beta-Beta*
2. *Structura Smart Contractelor*
3. *Testare Automatizată*
4. *Testare Performanță*

5. **Rezultatele Testelor Manuale**
6. **Vulnerabilități Specifice Verificate**
7. **Analiză Comparativă**

Partea 3: Final Version (3/3)

1. **Prezentare Generală a Platformei**
2. **Testare Extinsă și Automatizată**
3. **Securitate**
4. **Flux Complet de Utilizare**
5. **Sugestii de Extindere**
6. **Deployment**
7. **Interfețe Suportate**

Alpha-Alpha Version

1. Obiectiv Alpha-Alpha

- Realizarea unei **analize state-of-the-art** pentru temă.
- Colectarea de materiale esențiale (articole, resurse, definiții).
- Identificarea aplicațiilor existente similare.
- Setup inițial pentru dezvoltarea și testarea rețelei Blockchain.

2. Contextul general al temei

De ce un magazin NFT cu abonamente?

- Abonamentele digitale sunt omniprezente (Netflix, Spotify etc.).

- NFT-urile permit un control real asupra abonamentului.
- Pot fi transferabile, revandute, verificate on-chain.

3. Arhitectura aplicației

<u>Componentă</u>	<u>Descriere</u>
Smart Contract (Solidity) ->	Creează și administrează produse sub formă de NFT
Frontend (React + Ethers.js) ->	Interfață pentru utilizatori
Hardhat ->	Framework de testare și deployment
Metamask ->	Wallet pentru conectare și tranzacții

4. Colectarea Materialelor Esențiale (state-of-the-art)

Articole Științifice relevante (PDF):

- [Testing Strategies for NFT Smart Contracts](#)
- [Smart Contract Security Best Practices \(Consensys\)](#)
- [Security Analysis of Non-Fungible Tokens](#)
- [Performance Analysis of NFT Marketplaces](#)
- [ERC-721 Non-Fungible Token Standard](#)
- [NFT-based Subscription Models](#)
- [Ethereum: A Secure Decentralised Generalised Transaction Ledger](#)

Pagini Web și Resurse Relevante:

- [OpenSea – cel mai mare Marketplace NFT](#)
- [Rarible – Marketplace NFT descentralizat](#)
- [Ethereum Smart Contracts Testing Guide](#)
- [ERC-721 și ERC-1155 – Standarde NFT](#)

- OpenZeppelin Docs
- Hardhat Documentation
- Chainlink Docs

5. Definiții Esențiale

<u>Concept</u>	<u>Definiție</u>
NFT (Non-Fungible Token) ->	Token unic pe blockchain care reprezintă un activ digital (în cazul nostru: abonamente).
Smart Contract ->	Cod care rulează automat pe blockchain, executând logică prestabilită.
Testare Unitare ->	Testarea funcțiilor individuale (ex: creare NFT, validare abonament).
Testare de Integrare ->	Testarea interacțiunii dintre contracte și aplicație web/backend.
Testare Performanță ->	Măsurarea vitezei, scalabilității și eficienței rețelei.
Testare Securitate ->	Detectarea vulnerabilităților (ex: reentrancy, overflow, frontrunning).

6. Analiza aplicațiilor existente

Exemple:

1. **Unstoppable Domains** – NFT ca domenii
2. **Audius** – platformă de streaming cu NFT-uri pentru acces
3. **Unlock Protocol** – oferă NFT-uri care acționează ca abonamente

Avantaje:

- Permite control complet utilizatorului
- Posibilitate de revânzare a NFT-urilor (abonamente transferabile)
- Transparent și descentralizat

Dezavantaje:

- Costuri ridicate pe rețelele Layer-1
- Dificultate în gestionarea expirării abonamentelor (necesită cron jobs sau oracole)
- Probleme de scalabilitate

7. Identificarea serviciilor și resurselor disponibile pentru testare

Rețele:

- Ethereum Sepolia, Polygon Mumbai, Base

Framework:

- Hardhat, Truffle

Biblioteci:

- OpenZeppelin, Chainlink

Wallet:

- Metamask, WalletConnect

Testare Unitară și de Integrare

- **Truffle** (Solidity / Ethereum)
- **Hardhat** (Testare rapidă și debugging Ethereum smart contracts)
- **Ganache** (Rețea Ethereum locală pentru teste rapide și izolate)

Testare Performanță

- **JMeter** (Simulare cereri simultane pentru marketplace NFT)
- **Hyperledger Caliper** (benchmarking blockchain)
- **Apache Benchmark** (testarea performanței API-urilor marketplace)

Testare Securitate

- **Mythril** (vulnerabilități smart-contracts Ethereum)
- **Slither** (analiză statică smart-contracts NFT)

- **Echidna** (fuzzing pentru detectarea vulnerabilităților NFT)

8. Client App – NFT Marketplace / Product Subscription DApp

Features:

1. **Create Product** – creează un NFT ce reprezintă un produs / abonament
2. **Buy Product** – cumpără un NFT (abonament activ)
3. **Delete Product** – doar deținătorul îl poate șterge
4. **Add Review** – utilizatorii care au cumpărat pot lăsa review-uri

9. Setup Inițial:

1. Precondiții:

- Node.js (v18+ recomandat)
- npm (sau yarn)
- Git (opțional, dar util)

2. Inițializare proiect:

```
mkdir my-blockchain-project

cd my-blockchain-project

npm init -y

npm install --save-dev hardhat

npx hardhat
```

3. Structura proiectului:

```
my-blockchain-project/
|
|
| — contracts/      # Smart contracts în Solidity
|
|   └─ Lock.sol      # Contract de exemplu
```

```

|
|— scripts/          # Scripturi de interacțiune/testare
|   └─ deploy.js
|
|
|— test/             # Teste (Mocha/Chai)
|   └─ Lock.js
|
|
|— hardhat.config.js  # Configurație Hardhat
|
|— package.json
|
└─ README.md

```

4. Rulare blockchain local

npx hardhat test

npx hardhat run scripts/deploy.js --network localhost

npx hardhat node

Beta-Beta Version

1. Obiectivul versiunii Beta-Beta

- Integrarea contractelor inteligente finale: EventFactory, Event, BiletNFT.
- Testarea completa: unitara, integrare, performanta, securitate.
- Evaluare concreta a costurilor si scalabilitatii pe retele de test.
- Pregatire pentru versiunea finala.

2. Structura Smart Contractelor

EventFactory.sol:

- Creeaza noi evenimente (Event) cu parametrii personalizati.

- Tine evidenta adreselor evenimentelor create.

Event.sol:

- Gestioneaza vanzarea de bilete pentru un eveniment.
- Permite anularea evenimentului si cererea de refunduri.
- Interactiuneaza cu contractul BiletNFT pentru mintare si refund.

BiletNFT.sol:

- ERC721 NFT care reprezinta biletele.
- Asigura logica de emitere (cumparaBilet) si transfer (refundTransfer).
- Integreaza storage de metadata (URIStorage) si enumerare (Enumerable).

3. Testare Automatizata

Framework: Hardhat

Teste principale:

- Cumpărare bilet reusita.
- Refuz cumparare daca evenimentul e anulat.
- Refund cerut corect dupa anulare.
- Retragerere refund cu verificarea balantei.
- Verificare ca nu se pot cumpara mai multe bilete decat disponibilitatea.
- Verificare erori la withdraw fara fonduri.

4. Testare Performanta

Setup:

- Ganache Local Network

Simulare:

- 100 de utilizatori cumpara bilete simultan.
- Măsurarea timpului mediu de procesare a tranzactiilor.

Rezultate:

- Timp mediu tranzactie: 0.5 secunde.
- Rata success: 100%.

5. Rezultatele Testelor Manuale (Hardhat)

- Refund permis doar după anularea evenimentului.
- Refund respins dacă biletul nu aparține utilizatorului.
- Cumpărare respinsă dacă evenimentul este anulat.
- Retragerere fonduri respinsă fără sold disponibil.

5. Vulnerabilități Specifice Verificate

- Controlul de Acces: Funcția `anuleazaEvent` este protejată prin `onlyOwner`.
- Protecție împotriva Reentrancy: Tranzacțiile ETH folosesc `.call` și verifică succesul.
- Lipsa Overflow/Underflow: Solidity ^0.8.0 oferă protecție implicită.
- DoS prin Transfer Eșuat: Toate transferurile sunt verificate prin success/failure.
- Ownership Safe: Contractele folosesc corect OpenZeppelin Ownable.

6. Analiza Comparativa Contracte

Proiectul implementat, bazat pe contractele Event, BiletNFT și EventFactory, oferă o structură modulară și scalabilă, asigurând o separare eficientă a funcțiilor de creare, gestionare și vânzare de bilete NFT. În comparație cu soluțiile alternative, avantajele principale sunt securitatea ridicată și extensibilitatea. Astfel, soluția actuală reprezintă o bază solidă și sigură pentru un marketplace NFT de evenimente.

Final Version 3/3

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.

Variable

- `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.

Variable

- `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 împrêtează 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.