**Relazione finale progetto Aconsy’s e-commerce**

**Introduzione**

L’E-Commerce di Aconsy mira alla vendita di abbigliamento vario integrando la trasparenza, privatezza e velocità della Blockchain, integrando tag NFC per una migliore autenticità e tracciabilità del prodotto.

Uno degli aspetti innovativi del sistema sarà l’integrazione con la blockchain di Solana, scelta per la sua velocità e basso costo delle transazioni. Gli utenti potranno autenticarsi in modo sicuro attraverso il proprio wallet decentralizzato, ottenendo così un’esperienza di login sicura, senza credenziali tradizionali.

Il sistema permetterà di effettuare il pagamento dei prodotti, oltre ai vari metodi di pagamento standard, in stablecoins (USDT, USDC) e Solana offrendo transazioni rapide, trasparenti e resistenti alla censura.

L’obbiettivo principale è il contrasto a qualsiasi forma di contraffazione del prodotto che porta manca guadagni alle grandi aziende ogni anno per importi di miliardi di euro. Inoltre un massiccio utilizzo della blockchain Solana e quindi una maggiore adozione, in particolare ai principianti grazie alle funzionalità user-friendly che Solana offre.

**Specifica dei requisiti**

**Descrizione generale**

1. **Funzionalità del prodotto**

Il sito consentirà agli utenti di:

- Navigare un catalogo di prodotti di abbigliamento

- Visualizzare i dettagli di autenticità

- Registrarsi/autenticarsi tramite wallet Solana

- Aggiungere articoli al carrello e completare acquisti

- Effettuare pagamenti tradizionali o in stablecoin (USDT/USDC) e Solana

- Tracciare l’ordine e visualizzare lo storico delle transazioni blockchain

**2. Caratteristiche degli utenti**

Gli utenti target sono clienti interessati all’abbigliamento digitalmente autenticato, con una conoscenza base e non dell’uso dei wallet crypto. Il sito sarà progettato per essere user-friendly.

**3. Vincoli**

* La blockchain di riferimento sarà **Solana**, su Network Devnet
* Il pagamento crypto avverrà tramite stablecoin compatibili (USDT, USDC) e Solana
* Il sistema userà **React** per il frontend e **Node.js, JavaScript e TypeScript** per il backend

**Requisiti specifici**

1. **Requisiti dell’interfaccia utente**

L’interfaccia utente deve essere moderna ed elegante, ma allo stesso tempo user-friendly.

Sarà presente una Navbar per accedere alle pagine “Novità, uomo, donna, contatti” oltre che una barra di ricerca, il pulsante per connettere il wallet, il carrello e la scelta della lingua del sito.

1. **Requisiti funzionali**

**Per l’utente:**

* + Navigare, cercare e visualizzare i prodotti presenti sul sito, inizialmente senza necessità di connessione del proprio wallet, necessario invece per l’aggiunta di un prodotto nel carrello
  + Visualizzazione dei dettagli di un prodotto, cliccandoci su di esso si aprirà la pagina che mostrerà sia i dettagli fisici che dell’NFT associato, compreso il public address
  + Effettuare la connessione del proprio wallet su Solana tramite estensione del browser, oppure tramite l’icona di Phantom Embedded per la creazione o accesso del wallet tramite mail per i meno esperti
  + Modificare la quantità di prodotti o la rimozione nel carrello
  + Effettuare Checkout e pagamento dei prodotti presenti nel carrello tramite stablecoin (USDT-USDC su rete Solana) e la stessa Solana
  + Ricevere conferma d’ordine e pagina di tracciamento ordine
  + Visualizzare storico ordine e transazione su blockchain in caso di pagamento tramite crypto

**Per amministratore:**

* + Aggiunta, modifica o rimozione di prodotti nel catalogo
  + Visualizzazione e aggiornamento dello stato degli ordini
  + Invio di NFT associato al prodotto al compratore

1. **Vincoli di progettazione**

Tecnologie obbligatorie:

* + Il frontend deve essere sviluppato utilizzando React.js
  + Il backend dovrà essere realizzato in Node.js, utilizzando JavaScript e TypeScript
  + Le comunicazioni tra frontend e backend dovranno avvenire tramite RESTful API.

Blockchain:

* + L’unica blockchain supportata sarà Solana, utilizzando la rete Devnet per sviluppo e testing
  + I pagamenti in criptovaluta saranno limitati alle stablecoin USDT e USDC e Solana nativa, compatibili con Solana

Autenticazione:

- Gli utenti si autenticheranno esclusivamente tramite wallet decentralizzato Solana.

- Non sarà implementato un sistema tradizionale con username/password.

Deployment:

* + Verrà utilizzato come sistema di deployment e build del progetto Vercel, utilizzato maggiormente per la pubblicazione di web app su blockchain

Database:

* + I dati verranno immagazzinati nel database PostgreSQL Supabase, grazie alla sua potenza, scalabilità e alta adozione.

1. **Requisiti logici del database**

Il database conterrà le seguenti entità principali:

* + user ( wallet\_address)
  + product ( name, description, image\_url, price, size, color, gender, nft\_address, isNew, isAvailable )
  + order ( nft\_address, id\_transaction, status, wallet\_buyer, creation\_date, shipping\_address )

1. **Requisiti relativi alle prestazione**

- Il sistema dovrà garantire un tempo medio di risposta inferiore a 2 secondi per ogni richiesta HTTP

- Il sito dovrà supportare almeno 5 utenti simultanei nella fase di testing su Devnet

- Le transazioni su blockchain dovranno essere confermate in un tempo massimo di 5 secondi (media Solana Devnet)

**6. Attributi del sistema**

- Affidabilità: il sistema sarà progettato per gestire correttamente errori di rete o problemi di autenticazione con il wallet.

- Sicurezza: non verranno memorizzate password o dati sensibili, solo wallet address pubblici.

- Manutenibilità: la separazione tra frontend e backend facilita aggiornamenti indipendenti.

- Scalabilità: grazie all’uso di Supabase e Vercel, il sistema è facilmente scalabile per un numero maggiore di utenti.

**Progettazione del sistema**

1. **Architettura generale**

Il sistema comprenderà un’architettura client-server a tre livelli:

* Frontend: sviluppato in React.js, si occupa della UI e interagisce con gli utenti
* Backend: in Node.js, esposto tramite RESTful API, gestisce la logica applicativa e la comunicazione con il database e la blockchain
* Database: Supabase (PostgreSQL), per memorizzare utenti, ordini e prodotti

1. **Componenti principali**

* Client web (React): interfaccia grafica, connessione wallet, interazione con catalogo e carrello
* API server (Node.js):

1. Riceve richieste http
2. Valida l’identità dell’utente tramite il wallet
3. Crea/modifica/cancella ordini
4. Interagisce con Supabase
5. Crea transazioni di pagamento
6. Invia l’NFT corrispondente al prodotto comprato

* Solana Blockchain (Devnet): verifica e registra la transazione dei prodotti autentici venduti. Inoltre, verranno utilizzate librerie “Helper” fornite da Phantom e altri provider (Metaplex ecc) per la connessione wallet e transazione
* Supabase (DB): memorizza dati strutturati (prodotti, ordini, utenti) e ne restituisce le informazioni

1. **Flusso di un acquisto**
2. L’utente apre il sito, potendo scegliere dall’icona presente nella Navbar la lingua preferita
3. Connette il proprio wallet al sito in anticipo, oppure dopo aver visualizzato il catalogo
4. Seleziona il prodotto e lo aggiunge nel carrello specificando le quantità
5. Modifica il contenuto del carrello rimuovendo o decrementando le quantità
6. Effettua il checkout e sceglie di effettuare un pagamento in crypto

5,1. L’utente cambia idea, non effettua il pagamento e vengono ripristinati i prodotti presenti nel carrello

1. Paga in crypto e la transazione risulta “confirmed” su blockchain:
   * La transazione viene firmata dal wallet Solana
   * Viene inviato un NFT al wallet dell’utente, registrato nella blockchain
2. Il backend salva l’ordine e i dettagli su Supabase, inoltre il prodotto dopo l’acquisto diventa non più disponibile.
3. L’utente può visualizzare l’ordine e lo storico delle transazioni.
4. **Struttura del database**

User: wallet\_address VARCHAR (44)

Product: nft\_address VARCHAR (44), name VARCHAR, description TEXT, image\_url TEXT, price INT4, size VARCHAR, color VARCHAR, gender TEXT, isNew BOOL, isAvailable BOOL

Order: nft\_address (PK + FK) TEXT[], id\_transaction TEXT, status VARCHAR, wallet\_buyer VARCHAR(44), creation\_date DATE, shipping\_address TEXT

1. **API Endpoints**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metodo** | **Percorso** | **Funzione** | **Autorizzazione** |
| GET | /service/ProductService | getProducts | Pubblico |
| GET | /service/ProductService | getProductByAddress | Pubblico |
| GET | /service/ProductService | getNewProducts | Pubblico |
| GET | /service/ProductService | searchProducts | Pubblico |
| GET | /service/ProductService | getProductsBySize | Pubblico |
| GET | /service/ProductService | getProductsByColor | Pubblico |
| GET | /service/productService | getProductsByGender | Pubblico |
| POST | /service/userService | createUser | Wallet connesso |
| PUT | /service/userService | updateUser | Wallet connesso |
| POST | /service/userService | createOrder | Wallet connesso |
| GET | /service/userService | getUserOrders | Wallet connesso |

**Implementazione del sistema**

**Ambienti e strumenti**

* **Versionamento:** Git + GitHub (branch main → deploy Vercel)
* **IDE:** VS Code
* **Linguaggi:** TypeScript, JavaScript (frontend e backend)
* **Package manager:** npm
* **Dipendenze principali:**
  + Frontend: Next.js, React.js, CSS Modules
  + Web3: @solana/web3.js, @metaplex-foundation/umi, mpl-toolbox, umi-bundle-defaults, umi-signer-wallet-adapters,
  + DB: @supabase/supabase-js

**Struttura del progetto**

/

├─ src/app/

│ ├─ layout.tsx

│ ├─ page.tsx

│ ├─ new/page.tsx

│ ├─ men/page.tsx

│ ├─ women/page.tsx

│ ├─ history/page.tsx

│ └─ checkout/page.tsx

│ └─ product/[nft\_address]/page.tsx

├─ components/

│ ├─ Search.tsx

│ ├─ CartContext.tsx

│ ├─ Cart.tsx

│ └─ AppWalletProvider.tsx

├─ service/

│ ├─ productService.ts

│ └─ userService.ts

├─ styles/

│ ├─ page.module.css

│ ├─ new.module.css

│ └─ checkout.module.css

│ └─ cart.module.css

│ └─ history.module.css

│ └─ product.module.css

│ └─ search.module.css

├─ utils/

├─ supabaseClient.ts

├─ next.config.js

└─ package.json

**Frontend**

* **Layout e routing:** app/layout.tsx per font, Navbar e Context Providers.
* **Stato e Context:**
  + CartContext per carrello e persistenza (localStorage)
  + AppWalletProvider per adapting wallet Solana
* **Componenti chiave:**
  + ProductCard: immagine, nome, prezzo, badge NFT, pulsante “Add to Cart”
  + Search: fetch dinamico
  + Cart: overlay, controllo stock, pulsante checkout disabilitato senza wallet

**Backend / API**

* **Next.js API Routes**
* **CRUD Supabase:**
  + createOrUpdateUser, createOrder, getUserOrders in userService.ts
  + getProducts, searchProducts in productService.ts
* **Business logic:**
  + Generazione id\_transaction (randomId())
  + Validazione stock prima di inviare la transazione

**Integrazione Blockchain**

* **Umi + mpl-toolbox:**
  + createUmi(rpcUrl).use(mplToolbox()) + walletAdapterIdentity(wallet)
* **Transfer SPL-Token:** transferTokens({ source, destination, mint, amount })
* **Transfer SOL:** transferSol({ source, destination, amount }) con sol(cartTotal/price)
* **Gestione risultato:** estrazione res.result.value.confirmationStatus e signature via base58.deserialize

**Database**

CREATE TABLE users (

wallet\_address VARCHAR(44) PRIMARY KEY,);

CREATE TABLE products (

nft\_address VARCHAR(44) PRIMARY KEY,

name TEXT, description TEXT, image\_url TEXT,

price INTEGER, size VARCHAR, color VARCHAR,

gender TEXT, isNew BOOLEAN, isAvailable BOOLEAN

);

CREATE TABLE orders (

id SERIAL PRIMARY KEY,

nft\_address TEXT[],

id\_transaction VARCHAR PRIMARY KEY,

status VARCHAR,

wallet\_buyer VARCHAR(44) REFERENCES users(wallet\_address),

creation\_date DATE

);

**Deployment & CI/CD**

* **Frontend:** Vercel (auto-deploy da GitHub)
* **Database:** Supabase (project Dev)
* **Env vars:**
  + NEXT\_PUBLIC\_SUPABASE\_URL
  + NEXT\_PUBLIC\_SUPABASE\_ANON\_KEY

**Testing**

**Ambiente di test**

* **Browser: Brave**
* **Risoluzione: 1920 × 1080 con schermo 150%**
* **Device: desktop**
* **Network: rete locale, Solana Devnet**

| **#** | **Scenario** | **Passi** | **Risultato atteso** | **Esito** |
| --- | --- | --- | --- | --- |
| **1** | **Caricamento home page** | **1. Aprire / 2. Verificare navbar, hero, featured products, footer** | **Tutti gli elementi visibili, nessun errore JavaScript** | **✅** |
| **2** | **Cambio lingua** | **1. Click sul pulsante lingua 2. Verificare che URL passi tra / e /it** | **URL e contenuti cambiano lingua correttamente** | **✅** |
| **3** | **Navigazione “New Arrivals, Man, Women”** | **1. Click su “New Arrivals, Men, Women” in navbar 2. Verificare lista prodotti nuovi, uomo, donna** | **URL /new, prodotti filtrati** | **✅** |
| **4** | **Ricerca prodotti (> 3 caratteri)** | **1. Click sul tasto Search 2. Digitare “tank” 3. Verificare risultati dinamici** | **Appare lista prodotti il cui nome contiene “tank”** | **✅** |
| **5** | **Aggiunta al carrello** | **1. Su un ProductCard click “Add to Cart” 2. Aprire overlay carrello** | **Item mostrato, badge conta item aggiornato** | **✅** |
| **6** | **Modifica quantità** | **1. Aprire carrello 2. Click “+” e “–” su un item 3. Verificare totali** | **Quantità e prezzo nel summary si adattano correttamente** | **✅** |
| **7** | **Out of stock** | **1. Simulare isAvailable = false 2. Verificare “Out of Stock” e stile disabled hover** | **Pulsante disabilitato, al hover appare icona “🚫”** | **✅** |
| **8** | **Checkout USDC** | **1. Selezionare radio USDC 2. Click “Complete Purchase” 3. Confermare firma wallet** | **Transazione SPL USDC su Devnet, conferma “confirmed”, ordine salvato** | **✅** |
| **9** | **Checkout USDT** | **Stessi passi dello USDC** | **Transazione SPL USDT, conferma e salvataggio ordine** | **✅** |
| **10** | **Checkout SOL** | **1. Selezionare radio SOL 2. Click “Complete Purchase” 3. Firma SOL** | **Transazione SOL su Devnet, conferma “confirmed”, ordine salvato** | **✅** |
| **11** | **Storico ordini** | **1. Collega wallet 2. Visita /orders** | **Lista ordini precedenti con link a Solana Explorer** | **✅** |

**Esiti e osservazioni**

* **Tutti i casi di test manuale sono passati senza errori critici.**
* **Performance: tempo medio di caricamento pagina < 2 s in locale; transazioni blockchain confermate in ~3 s con ritardi o in casi particolari errati dopo test ripetuti a causa del flood-blocking della rete.**
* **Limiti: in Devnet occasionali ritardi rete; nessun test su ambiente mobile reale.**