



AlmaStreet - Tecnologie Web

Anno accademico 2025/2026

Porcheddu Alessandro
alessandro.porcheddu@studio.unibo.it

Matteo Todeschi
matteo.todeschi2@studio.unibo.it

Visione

AlmaStreet è applicativo web gestionale che simula un vero e proprio "Crypto Stock Market" ma con la particolarità di essere indirizzato alla community universitaria. Gli studenti qui non usano soldi reali, ma "CFU virtuali" come valuta per speculare sulla popolarità e sull'andamento virale di Meme riguardo la vita universitaria. L'obiettivo degli utenti è cercare di massimizzare il valore del proprio portafoglio attraverso investimenti, vendite e proposte.

In particolare, i **Meme** sono il centro della UX: introducono un fattore "imprevedibile", cioè sapere se la propria battuta verrà apprezzata più di altre. Questo sprona gli utenti a impegnarsi nel creare scherzi innovativi e originali per attirare l'attenzione degli altri giocatori in veste di **investitori**.

A differenza dei mercati azionari tradizionali AlmaStreet implementa un protocollo **AMM (Automated Market Maker)**, ovvero un sistema che (a differenza della borsa reale) fornisce agli utenti una controparte che compra quando loro vendono e vende quando vogliono comprare. Questo meccanismo permette un gioco quasi frenetico, cambiamenti improvvisi di prezzo e premia fortemente gli i primi investitori.

Architettura

Lo Stack Tecnologico è stato selezionato per permettere di sviluppare un'applicazione con molte funzionalità interconnesse tra di loro pur rispettando le richieste del bando.

Componente	Tecnologia	Ruolo
Backend	Laravel	Logica di business, transazioni atomiche DB, supporto a migrations e seeders. Utilizza Eloquent ORM per la gestione del database e delle relazioni. Facilita la gestione API RESTful. Si basa sulla divisione dei vari componenti dell'applicativo in base ai ruoli (architettura MVC)
Frontend	HTML5 Javascript "Vanilla"	Interazione utente dinamica e manipolazione DOM quando necessario.
Styling	CSS Tailwind/Flowbite	UI più responsive e riutilizzo di stile per componenti diversi. Definizione di un tema globale di colori.

Extension	TradingView Lightweight Charts, Charts.js	Librerie per la visualizzazione in tempo reale dell'andamento dei grafici dei meme e il grafico a torta del portafoglio.
------------------	---	--

Funzioni Implementative

1. Trading Core:

- **Pricing:** Utilizzo di un algoritmo matematico per il calcolo del prezzo:

$$P = P_{base} + M \cdot S$$
 - P: prezzo finale
 - P_{base} : prezzo attuale
 - M: volatilità
 - S: numero di azioni
- **Gestione Slippage:** Per garantire performance elevate, il calcolo dello slippage (differenza di prezzo tra acquisto desiderato ed effettivo) su grandi volumi è implementato con una formula integrale, eliminando la necessità di cicli iterativi.

2. Il Rettorato (Admin):

- **Dashboard CRUD Completa:** Interfaccia amministrativa per la gestione totale delle operazioni chiave:
 - Approvazione dei Meme
 - Monitoraggio e rilevamento delle anomalie
 - Visualizzazione delle notifiche
 - Creazione e gestione di eventi globali

3. Profilo Utente

- **Visualizzazione Finanziaria Completa:** Dashboard dedicata che offre:
 - Calcolo in tempo reale del Net Worth (Patrimonio Netto).
 - Grafici dettagliati di Asset Allocation.

4. Gamification

- **Distribuzione CFU (Dividendi):** Erogazione automatica giornaliera di CFU agli azionisti, basata su uno snapshot periodico delle partecipazioni.
- **Riconoscimenti e Premi:** Sistema automatico di assegnazione di badge (es. "Diamond Hands") che premiano il comportamento di trading degli utenti.

5. Accessibilità

- **Assistenza AI:** Essendo il nostro servizio basato sui meme sotto forma di immagini, obbligare gli utenti a fornire manualmente un'alternativa testuale sarebbe risultato un modo per scaricare su soggetti terzi la responsabilità di fornire un servizio accessibile.
Si è dunque optato per un sistema che si occupa di:
 - Verificare che il contenuto non sia offensivo o inadeguato
 - Creare un'alternativa testuale coerente e completa del meme
- Come da linee guida inoltre il sito è stato pensato per rispettare le norme WCAG

Interfaccia (UI / UX)

L'interfaccia utente è stata progettata seguendo rigorosamente un approccio **Mobile First**, ottimizzato per l'utilizzo a una mano (thumb-driven design), unendo i canoni estetici del **Modern Fintech** (pulizia, contrasto elevato) all'estetica **Social**.

- **Navigazione Ergonomica:** su mobile è implementata una bottom navigation bar fissa che contiene le sezioni principali (Market, Portafoglio, Classifica, Profilo). Il pulsante centrale di creazione meme è sopraelevato per dare enfasi all'azione di listing e invogliare l'utente.
- **Trade Station:** per prevenire errori cognitivi e garantire rapidità, la pagina operativa utilizza una bottom bar fissa con i pulsanti **Vendi** e **Compra** sempre visibili. L'interazione finale per l'acquisto avviene tramite pannello modale in sovrapposizione dal basso, mantenendo l'utente nel contesto senza cambi pagina bruschi.
- **Skeleton Loading:** durante le chiamate API, il sistema mostra "scheletri" della struttura invece di spinner generici, migliorando la percezione di velocità.
- **Toast Notifications:** gli esiti delle transazioni (es. "ordine eseguito") appaiono come popup non invasivi a scomparsa automatica.

Struttura Implementativa

Struttura dei Packages

```
uni-meme-stock-market/
├── app/
│   ├── Http/
│   │   ├── Controllers/
│   │   │   ├── AdminController.php
│   │   │   ├── AuthController.php
│   │   │   ...
│   │   ├── Middleware/
│   │   └── Requests/
│   ├── Models/
│   │   ├── Admin/
│   │   ├── Financial/
│   │   ...
│   └── Services/
│       ├── TradingService.php
│       ├── GeminiService.php
│       ...
├── database/
│   ├── migrations/
│   └── seeders/
└── public/
└── resources/
    ├── js/
    │   ├── components/
    │   ├── pages/
    │   ...
    ├── css/
    └── views/
        ├── components/
        ├── pages/
        ...
└── routes/
└── tests/
```

Architettura Dettagliata

Controllers

I controller gestiscono la logica di instradamento delle richieste e la comunicazione tra il frontend e i servizi.

- AdminController.php: gestione amministrativa del marketplace e degli utenti.
- AuthController.php: autenticazione, registrazione e gestione sessioni.
- CreateController.php: logica per la proposta e creazione di nuovi "Meme".
- MarketplaceController.php: Visualizzazione dei meme quotati e dati di mercato.
- NotificationController.php: Gestione delle notifiche utente.
- ProfileController.php: Gestione del profilo utente e del portfolio.
- TradingController.php: Esecuzione e anteprima degli ordini di acquisto/vendita.

Services

Contengono la logica di business pura, mantenendo i controller snelli.

- TradingService.php: Core logic per il calcolo dei prezzi, esecuzione ordini e slippage.
- MarketService.php: Gestione dello stato del mercato (sospensioni, trend).
- GeminiService.php: Integrazione con AI per l'analisi e la validazione dei meme.
- NotificationService.php & NotificationDispatcher.php: Gestione notifiche.
- OtpService.php: Gestione della verifica a due fattori via email.
- UserService.php & AdminService.php: Operazioni CRUD e logica specifica per ruoli.

Models

Rappresentano le entità del database e le loro relazioni, organizzati per dominio.

- User.php: Entità utente principale.
- Admin: AdminAction, GlobalSetting, MarketCommunication.
- Financial: Portfolio, PriceHistory, DividendHistory, Transaction.
- Market: Meme, Category, Watchlist.
- Gamification: Badge, UserBadge.

Database

- Migrations: definizione della struttura delle tabelle (Users, Memes, Badges, etc.).
- Seeders: popolamento iniziale del database con dati di test e configurazioni globali.
- Factories: generatori di dati fittizi per testing, sviluppo e presentazione.

Resources

Il cuore del frontend (Blade + AJAX).

- JS:
 - Components: componenti UI riutilizzabili (pulsanti, grafici, card).
 - Pages: schermate principali (Dashboard, Marketplace, TradeStation).
 - Services: client API per comunicare con il backend.
 - Core & utils: logica condivisa e helper lato client.
- CSS: stili globali e configurazioni Tailwind/CSS.
- Views:
 - Components: template dei componenti riutilizzabili.
 - Pages: pagine dell'applicazione
 - Errors: pagine di default per errori HTTP (404, 403, etc)
 - Emails: struttura per l'email di verifica OTP

Storage

- app/public/data: Contiene una folder per ogni utente con al suo interno i meme caricati e l'avatar del profilo
- app/public/badges: Contiene gli asset grafici dei Badges

Gestione dei Dati

Di seguito è rappresentato il modello logico-relazionale del database.

Per la visualizzazione completa e dettagliata si faccia riferimento alla [cartella docs](#) del repository Github.

