DOCUMENTO TECNICO COMPLETO — MVP DI ELARA

Architettura, Componenti Operativi, Scelte Tecnologiche e Adattamento Futuro ad App iOS.

1. Visione generale di Elara

Elara è una piattaforma fintech orientata all'intelligenza artificiale, progettata per unire in un'unica vista: portafogli finanziari, beni fisici, esposizioni al rischio, debiti e capacità di simulare ottimizzazioni matematiche su misura. L'obiettivo è fornire all'utente un quadro completo del proprio Net Worth e dare strumenti avanzati tipici della finanza quantitativa, ma resi semplici e accessibili.

2. Struttura dell'MVP

L'MVP si basa su quattro pilastri fondamentali: importazione del portafoglio, stima dei beni fisici, calcolo del Net Worth e modulo iniziale di ottimizzazione quantitativa. Ogni componente è progettato per essere estendibile e migliorabile senza riscrivere parti cruciali dell'infrastruttura.

3. Scelta Tecnologica: Perché Python è il linguaggio ideale

3.1 Python come fondamento del backend

Python è consolidato come standard de facto nella data science, nella finanza quantitativa e nel machine learning. Offre librerie mature, veloci da implementare e orientate alla sperimentazione continua. Questo consente a una startup come Elara di muoversi rapidamente mantenendo robustezza tecnica.

3.2 Librerie chiave di Python utilizzate da Elara

- Pandas e NumPy per manipolazione, aggregazione e normalizzazione dei dati finanziari. - scikit-learn, PyTorch o TensorFlow per classificazione immagini, regressione dei valori e modelli AI. - cvxpy per la costruzione del modello matematico di ottimizzazione. - Pydantic e FastAPI per la definizione rapida e sicura del backend API.

3.3 Vantaggi strategici di Python

Python permette iterazioni rapide su modelli e funzioni: fondamentale per un MVP che deve evolversi con feedback reali. La sua comunità enorme, unita all'integrazione con tutte le tecnologie moderne cloud e AI, lo rende il cuore ideale di un'architettura fintech.

3.4 Come Python si integra perfettamente con Swift in fase successiva

Quando l'app iOS sarà sviluppata, verrà utilizzato SwiftUI per il frontend mobile, mentre Python continuerà a essere il backend principale. L'interazione avverrà tramite API REST. In questo modo non è necessario riscrivere il core del sistema: Swift si occuperà solo della

parte visuale e dell’interazione utente, mentre Python gestirà dati, logiche e AI.

La divisione dei ruoli è chiara: - SwiftUI → Interfaccia utente moderna, animazioni, esperienza nativa. - Python → Elaborazione dati, AI, motore di ottimizzazione. - Comunicazione → Struttura REST in JSON. Questa architettura garantisce longevità, scalabilità e permette di espandere l’infrastruttura senza rework totale.

4. Architettura Backend dell’MVP

L’architettura è modulare, semplice e scalabile: • Frontend Web di base per l’MVP • Backend Python con FastAPI • Database PostgreSQL • Modulo AI (immagini e regressioni) • Servizio di ottimizzazione quantitativa • API per futura app iOS

5. Fase 1 — Implementazione tecnica completa dell’MVP

La Fase 1 rappresenta la costruzione delle fondamenta del prodotto. Di seguito una descrizione operativa completa di ogni componente:

5.1 Implementazione modulo di importazione del portafoglio

L’obiettivo è permettere agli utenti di importare facilmente i dati del proprio portafoglio da broker esterni. Il sistema deve accettare CSV con campi anche non standardizzati e interpretarli in modo intelligente.

Procedure operative: - Utilizzo di pandas per rilevare automaticamente header e colonne. - Normalizzazione dei ticker tramite API esterne. - Conversione delle valute in tempo reale. - Identificazione della tipologia di asset (aziona, ETF, crypto, obbligazione).

5.2 Parsing intelligente con AI

Quando il CSV presenta campi irregolari, viene utilizzata una piccola rete neurale o modello linguistico leggero per comprendere quali colonne corrispondono a quali dati. Questo migliora la precisione e la robustezza dell’importazione.

5.3 Dashboard Portafoglio nel backend

Il backend calcola: - Valore totale del portafoglio. - Distribuzione percentuale. - Rendimento aggregato. - Peso per settore, area geografica, tipo di asset. Questi dati vengono poi inviati al frontend tramite API REST.

5.4 Implementazione modulo beni fisici

Questo è un vantaggio competitivo di Elara. L’utente può aggiungere un oggetto, opzionalmente fornire una foto e ottenere una stima del valore corrente.

Operazioni AI del modulo: - Classificazione immagine per identificare la categoria del bene. - Modello regressivo basato su dataset di riferimento per stimare il valore. - Valore suggerito inserito direttamente nel sistema Net Worth.

Questo permette all’utente di ottenere una valutazione istantanea dei propri oggetti, aumentando l'engagement e la percezione di valore dell’app.

5.5 Calcolo dinamico del Net Worth

Una volta ottenuti: investimenti, beni fisici, liquidità e debiti, il sistema calcola automaticamente il Net Worth aggiornato.

Componenti operative: • Aggregazione di tutte le categorie. • Aggiornamento in tempo reale delle variazioni dei prezzi. • Grafici storici e breakdown dettagliati.

5.6 Modulo di ottimizzazione quantitativa

Questo modulo permette di mostrare una simulazione di portafoglio ottimizzato matematicamente con il metodo di Markowitz. Il backend calcola: - Rendimenti medi storici degli asset. - Matrice delle correlazioni. - Minimizzazione del rischio tramite cvxpy. - Tre portafogli simulati: conservativo, bilanciato, aggressivo. Viene sempre mostrato un disclaimer che specifica che si tratta di simulazioni non costituenti consulenza finanziaria.

6. Fase 2 — Preparazione API future per App iOS

Per assicurare continuità tra MVP web e app mobile, il backend viene strutturato fin da subito con endpoint REST chiari. Questo consente allo sviluppo mobile in SwiftUI di collegarsi direttamente alle funzioni già operative, evitando duplicazioni e riducendo enormemente il lavoro futuro.

6.1 Comunicazione SwiftUI → FastAPI

SwiftUI invierà richieste HTTP alle API Python. I dati verranno restituiti in JSON, rendendo la comunicazione semplice, leggibile e stabile.

6.2 Benefici di questa architettura ibrida

• Nessuna duplicazione della logica. • Python rimane il motore finanziario e AI. • Swift gestisce solo UI e UX, come è corretto. • Riduzione costi e tempo di sviluppo. • Scalabilità verso Android o Web App complete.

7. Conclusione

La struttura delineata rende Elara un prodotto solido, estensibile e moderno. Python fornisce la potenza e flessibilità necessarie per il dominio finanziario e AI, mentre SwiftUI permetterà un’app mobile nativa elegante e fluida. L’MVP costruito su queste basi potrà evolvere senza interruzioni verso un sistema completo e competitivo nel mercato fintech.