



## Descrizione

### PROJECT WORK

CdS Informatica per le Aziende Digitali (L-31)

Settori Scientifico-Disciplinari coinvolti: Informatica (INF/01), Ingegneria Economico-Gestionale (ING-IND/35);

Tema n. 1: La digitalizzazione dell'impresa

Traccia del PW # Sviluppo di una applicazione full-stack API-based per un'impresa del settore finanziario

Descrizione situazione-problema \*

Le imprese del settore finanziario, come banche, assicurazioni e fintech, gestiscono una grande quantità di dati e transazioni, spesso in tempo reale. Per garantire un accesso sicuro, scalabile ed efficiente ai loro servizi, molte di queste aziende adottano architetture API-based, che permettono l'integrazione tra diversi sistemi e l'accesso ai servizi da parte di applicazioni web e mobile.

Si richiede allo studente di sviluppare un'applicazione full-stack basata su API per un'impresa del settore finanziario. L'applicazione dovrà fornire un servizio significativo (es. gestione conti, simulazione investimenti, richiesta prestiti, gestione polizze assicurative) e implementare un'architettura moderna con un backend RESTful e un'interfaccia utente intuitiva.

Obiettivi

Lo studente sarà in grado di:

- Implementare un'applicazione web utilizzando HTML, CSS e JavaScript per la parte front-end (qualsiasi framework va bene)
- Utilizzare Python, Java o qualsiasi altro linguaggio per creare una semplice back-end API che gestisca i casi d'uso in ambito finanziario selezionati, purché object-oriented

Elaborato/Artefatto/prodotto

Selezionare un'impresa del settore finanziario e individuare lo scenario d'uso preferito.

Sviluppare un'applicazione web utilizzando HTML, CSS e JavaScript per la parte front-end e utilizzare il linguaggio Python, Java o qualsiasi altro linguaggio object-oriented per creare un semplice back-end API che gestisca le logiche selezionate.

Produrre un rapporto che contenga:

- Il racconto del contesto in cui opera l'impresa e il tipo di servizio che offre.
- I dettagli degli aspetti di design (UML, ER)
- La documentazione delle API sviluppate (tipo Swagger)
- I codici front-end e back-end su repository Git o similari
- Un resoconto del processo seguito per lo sviluppo dei codici, includendo una descrizione di dettaglio degli snippet ritenuti più interessanti
- Test funzionale dell'applicazione mediante screenshot

Valutazione

Rubrica di valutazione

Questo è il contesto di questo topic. Cosa puoi capirci?

Questo **Project Work** del Corso di Laurea **Informatica per le Aziende Digitali (L-31)** riguarda lo sviluppo concreto di un'applicazione **full-stack basata su API** per un'impresa del settore finanziario, e rappresenta un esercizio integrato di progettazione, sviluppo e documentazione di un sistema informativo moderno. <sup>[1]</sup> <sup>[2]</sup> <sup>[3]</sup>

## Contesto e Finalità

L'obiettivo è simulare un caso reale di **digitalizzazione dell'impresa**, applicando le conoscenze informatiche acquisite al contesto economico e gestionale di banche, assicurazioni o fintech. Il tema della **digitalizzazione** si lega alla necessità per le aziende di integrare i propri sistemi informativi tramite **API RESTful**, per favorire interoperabilità, sicurezza, scalabilità e accesso in tempo reale ai servizi. <sup>[4]</sup> <sup>[2]</sup> <sup>[3]</sup>

## Struttura del lavoro

Il progetto richiede di:

- Scegliere un'impresa reale o ipotetica nel settore finanziario.
- Identificare uno **scenario d'uso** come gestione conti, simulazione di investimenti, richiesta prestiti o gestione polizze.
- Sviluppare una **web app** con un **frontend in HTML, CSS, JavaScript (o framework simile)**.
- Creare un **backend object-oriented** in **Python, Java o altro linguaggio** che esponga API REST. <sup>[2]</sup> <sup>[1]</sup>

## Componenti dell'elaborato

Il Project Work deve includere:

- Descrizione del contesto aziendale e del servizio offerto.
- Progettazione tecnica (diagrammi **UML**, modello **ER** dei dati).
- Documentazione delle API (ad esempio con **Swagger/OpenAPI**).
- Repository Git con i codici frontend e backend.
- Analisi del processo di sviluppo, con commento di snippet significativi.
- **Test funzionale** dell'app con screenshot e risultati. <sup>[3]</sup> <sup>[1]</sup> <sup>[2]</sup>

## Competenze sviluppate

Questo progetto permette di dimostrare:

- Capacità di **progettazione e sviluppo software completo** (front + back).
- Comprensione dei **principi architetturali API-based**.

- Abilità di integrare competenze tecniche con la **gestione economico-aziendale**, coerentemente con la missione del CdS L-31, che fonde informatica e business management digitale. <sup>[2]</sup> <sup>[3]</sup>

In sintesi, si tratta di un esercizio interdisciplinare volto a realizzare un servizio digitale reale per un contesto aziendale finanziario, con un approccio che unisce **programmazione, progettazione dei dati e gestione digitale d'impresa**.



## Realizzare un portafoglio crypto centralizzato sarebbe in linea con le guidelines di questa tesi?

Sì, **realizzare un portafoglio crypto centralizzato** sarebbe perfettamente **in linea con le linee guida del Project Work** del CdS *Informatica per le Aziende Digitali (L-31)*, a condizione che venga trattato come un **caso d'uso API-based nel settore finanziario** e rispetti i principi di sicurezza, usabilità e architettura moderna richiesti. <sup>[9]</sup> <sup>[10]</sup> <sup>[11]</sup>

### Coerenza con il tema "digitalizzazione dell'impresa"

Il portafoglio crypto rientra nella digitalizzazione dei servizi finanziari, uno dei settori più rappresentativi dell'innovazione digitale. Le tecnologie blockchain e i crypto-asset vengono oggi integrate da banche e fintech per offrire **nuovi servizi di gestione patrimoniale, pagamento e investimento**. Quindi, un'applicazione che gestisce un **wallet centralizzato** può essere vista come una piattaforma fintech che digitalizza un servizio tradizionale (custodia e movimentazione di asset). <sup>[10]</sup> <sup>[11]</sup>

### Allineamento con requisiti progettuali

Il tema risponde a tutti i punti della traccia:

- **Ambito finanziario:** la gestione di crypto-asset è parte dei servizi fintech.
- **Architettura API-based:** il backend può esporre API RESTful per operazioni come creazione account, visualizzazione saldi, transazioni e gestione chiavi.
- **Applicazione full-stack:** un frontend web (React, Vue o simile) può interfacciarsi al backend (in Python Flask/FastAPI o Java Spring).
- **Sicurezza e gestione dei dati:** include concetti di autenticazione, crittografia e auditing coerenti con le best practice fintech e indicazioni Assogestioni 2024 (standard CCSS, ISO 27001). <sup>[12]</sup> <sup>[11]</sup>

### Aspetti accademici e gestionali

Dal punto di vista economico-gestionale (ING-IND/35), il progetto permette di analizzare:

- Modelli di business di custody centralizzata.
- Rischi e governance nella gestione di chiavi crittografiche.

- Conformità a normative come **MiCAR (Regolamento UE 2023/1114)**.<sup>[11]</sup>

## Conclusione

Un **crypto wallet centralizzato** è del tutto compatibile con la linea del Project Work perché:

- È un servizio finanziario digitale.
- Usa un'architettura API-based full-stack.
- Coinvolge problematiche tecniche e organizzative rilevanti.

Si consiglia di posizionarlo come un **“servizio fintech di gestione risorse digitali”**, evidenziando non tanto la speculazione su crypto, ma **la digitalizzazione sicura della gestione patrimoniale** con API scalabili e frontend user-friendly.



## I pro e i contro nella realizzazione di questa tesi, per una figura con competenze prevalentemente in ambito IT, quali sono? I tempi di sviluppo previsti sono di circa 30 giorni

Per una figura con competenze prevalentemente IT, la scelta di sviluppare un **portafoglio crypto centralizzato** come Project Work per il CdS *Informatica per le Aziende Digitali (L-31)* presenta vantaggi e svantaggi ben definiti in termini di **fattibilità tecnica, valore accademico, e gestione del tempo (30 giorni)**.<sup>[18] [19] [20] [21] [22]</sup>

### Vantaggi

#### 1. Aderenza al tema del corso

Il progetto si colloca pienamente nel settore fintech, combinando digitalizzazione, sicurezza e gestione di asset finanziari digitali. È coerente con le finalità del CdS, che mira a sviluppare profili in grado di integrare tecnologie IT nei processi aziendali.<sup>[19]</sup>

#### 2. Elevato valore tecnico

Un crypto wallet richiede l'integrazione di:

- Backend API REST per la gestione di utenti, saldi e transazioni.
- Frontend interattivo per dashboard e visualizzazione portafoglio.
- Elementi di sicurezza come hashing, session management e crittografia.

Questi aspetti consentono di dimostrare padronanza di architetture web e di programmazione orientata agli oggetti.<sup>[23] [20]</sup>

#### 3. Scalabilità del progetto

Può essere implementato in modo progressivo: autenticazione, gestione delle criptovalute simulate, cronologia transazioni e API modulari. Ciò lo rende gestibile in 30 giorni con un buon piano di sviluppo agile (MVP).<sup>[23]</sup>

#### 4. Innovatività accademica

Il tema si allinea con i trend più recenti della **finanza digitale e blockchain**, garantendo un impatto positivo in sede di valutazione per originalità e attinenza a tecnologie emergenti. <sup>[22]</sup>  
<sup>[24]</sup>

### Svantaggi

#### 1. Complessità della sicurezza

Anche in un wallet "centralizzato", occorre considerare la gestione sicura di dati sensibili (chiavi private, credenziali, token). Gli errori d'integrazione API o di configurazione possono creare vulnerabilità gravi, come indicato dalle linee guida AGID sulla sicurezza del software. <sup>[23]</sup>

#### 2. Richiesta di conoscenze interdisciplinari

Servono nozioni base di **finanza digitale e gestione transazioni** (non solo programmazione). Bisogna comprendere i flussi tipici di un servizio finanziario, anche se simulato, per mantenere coerenza con l'ambito economico-gestionale. <sup>[24]</sup>

#### 3. Gestione dei tempi

Il ciclo completo (design + sviluppo + test + documentazione) in 30 giorni può risultare impegnativo. I tempi si riducono significativamente se si punta a un prototipo funzionale (MVP) con funzioni simulate, senza interazione diretta con blockchain reali. <sup>[20]</sup> <sup>[23]</sup>

#### 4. Limitato valore gestionale se troppo tecnico

Essendo un corso che fonde IT e management, un approccio esclusivamente tecnico potrebbe penalizzare la parte gestionale. È importante includere una riflessione su modello di business, governance, e rischi operativi del servizio. <sup>[19]</sup>

### Considerazioni temporali (30 giorni)

Con una pianificazione agile, il progetto è **realizzabile in 30 giorni** se:

- **Settimana 1-2:** progettazione e setup API-backend.
- **Settimana 3:** sviluppo interfaccia web e test funzionali.
- **Settimana 4:** documentazione, UML, Swagger, screenshot e relazione finale.

Si consiglia di simulare le operazioni senza connessione blockchain reale, concentrandosi sull'**architettura API, sicurezza e model-view separation**, pienamente sufficienti per il livello accademico richiesto.

In sintesi, per una figura con solide competenze informatiche, questo progetto è **coerente, formativo e realizzabile**, purché venga gestito con un piano di sviluppo semplice e un occhio di riguardo agli aspetti di **sicurezza, modellazione e documentazione API**.



# <q>Considerazioni temporali (30 giorni)

Con una pianificazione agile, il progetto è realizzabile in 30 giorni se:

Settimana 1-2: progettazione e setup API-backend.

Settimana 3: sviluppo interfaccia web e test funzionali.

Settimana 4: documentazione, UML, Swagger, screenshot e relazione finale.

Si consiglia di simulare le operazioni senza connessione blockchain reale, concentrandosi sull'architettura API, sicurezza e model-view separation, pienamente sufficienti per il livello accademico richiesto.

In sintesi, per una figura con solide competenze informatiche, questo progetto è coerente, formativo e realizzabile, purché venga gestito con un piano di sviluppo semplice e un occhio di riguardo agli aspetti di sicurezza, modellazione e documentazione API.</q> Approfondiamo questo punto

Approfondendo la pianificazione proposta (30 giorni con approccio agile), il progetto del portafoglio crypto centralizzato può essere gestito in modo efficace tramite la **metodologia MVP (Minimum Viable Product)** e strumenti moderni come **FastAPI**, **React** o **Vue**, garantendo un equilibrio tra sviluppo tecnico, documentazione e valore accademico.<sup>[38] [39] [40] [41]</sup>

## Settimana 1–2: Progettazione e setup del backend API

L'obiettivo iniziale è creare le fondamenta del sistema.

- **Fase 1: Analisi e design**
  - Definire gli *use case* e il modello dei dati (utenti, asset, transazioni).
  - Produrre **diagrammi UML** (Use Case, Class Diagram, Sequence Diagram).
  - Impostare l'**architettura API RESTful**, separando le risorse (es. /users, /wallets, /transactions).
- **Fase 2: Implementazione backend**
  - Usare **FastAPI (Python)** o **Spring Boot (Java)** per la logica business.
  - Implementare endpoint CRUD simulando dati in database SQLite o PostgreSQL.
  - Inserire funzioni di autenticazione, validazione input e gestione errori.
  - Testare con **Swagger UI** o **Postman** per generare la documentazione automatica.<sup>[42]</sup>  
<sup>[38]</sup>

## Settimana 3: Frontend e test funzionali

Questa fase introduce l'integrazione tra API e interfaccia utente.

- **Fase 3: Sviluppo UI**
  - Realizzare il frontend in **HTML**, **CSS** e **JavaScript** oppure con **React/Vue**.
  - Implementare dashboard per login, saldo, transazioni e storico.

- **Fase 4: Integrazione e test**

- Collegare il frontend agli endpoint API tramite chiamate `fetch()` o Axios.
- Effettuare test funzionali sulle funzionalità principali usando dati mockati.
- Applicare controlli di sicurezza minimi (es. autenticazione basata su token).

## Settimana 4: Documentazione, UML e rifinitura

L'ultima parte consolida il lavoro tecnico e produce il materiale per la valutazione accademica.

- **Fase 5: Documentazione tecnica**

- Redigere il report secondo il modello PW, includendo UML e motivazioni di design.
- Allegare la **documentazione Swagger/OpenAPI** e screenshot dell'app.
- Commentare gli snippet di codice principali e il processo di sviluppo guidato dalle iterazioni (*agile sprints*). [\[43\]](#) [\[44\]](#)

- **Fase 6: Validazione MVP**

- Il prototipo deve essere dimostrabile e stabile, anche senza blockchain reale.
- Obiettivo: mostrare la piena comprensione di architetture API, flussi di dati e separazione tra logica e presentazione. [\[45\]](#) [\[40\]](#) [\[41\]](#)

## Benefici dell'approccio agile in 30 giorni

1. **Incrementalità:** permette di avere un prototipo funzionante fin dalla seconda settimana, migliorabile con sprint successivi.
2. **Chiarezza didattica:** il docente può valutare progressi e coerenza delle iterazioni.
3. **Efficienza:** si ottiene un risultato concreto senza eccessivo carico teorico, mantenendo focus su architettura e sicurezza. [\[40\]](#) [\[43\]](#) [\[45\]](#)

## Consiglio operativo finale

Concentrarsi su un MVP che includa autenticazione, visualizzazione del saldo e gestione delle transazioni simulate. Questo approccio risponde pienamente ai criteri di una tesi L-31 — **digitale, integrata, funzionale e ben documentata**, completabile realisticamente in 30 giorni.



## Quali design pattern individui in questo progetto?

Un progetto di **portafoglio crypto centralizzato full-stack API-based** incorpora diversi **design pattern architetturali e di progettazione software** comunemente adottati nelle applicazioni fintech moderne. I principali pattern identificabili si organizzano lungo tre livelli: **architettura generale**, **pattern di backend**, e **pattern strutturali e comportamentali**. [\[58\]](#) [\[59\]](#) [\[60\]](#) [\[61\]](#) [\[62\]](#)

## Architettura generale

### 1. Layered (N-Tier) Architecture

L'applicazione è organizzata in più livelli indipendenti:

- **Presentation layer:** interfaccia web (React, HTML/CSS/JS) che interagisce con l'utente.
- **Business logic layer:** servizi backend che implementano regole del dominio (es. gestione transazioni, autenticazione).
- **Data access layer:** repository o adattatori per salvare dati (wallet, utenti, transazioni) nel database.

Questo pattern garantisce separazione delle responsabilità, testabilità e manutenzione semplificata. [\[63\]](#) [\[58\]](#)

### 2. Client-Server Pattern

Il progetto segue una chiara separazione tra client web (frontend) e server (API REST backend). Il client invia richieste HTTP e riceve risposte JSON dal backend tramite endpoint RESTful, modello standard per soluzioni fintech moderne. [\[61\]](#)

### 3. Model-View-Controller (MVC)

L'architettura può essere intesa come variante MVC:

- **Model:** entità di dominio (User, Transaction, Wallet).
- **View:** pagine o componenti React che mostrano i dati.
- **Controller:** endpoint API che gestiscono logica applicativa e orchestrano servizi. [\[62\]](#)

## Pattern di backend

### 1. Repository Pattern

Ogni entità del dominio (es. UserRepository, TransactionRepository) ha una classe dedicata per la gestione di query al database. Questo incapsula l'accesso ai dati e semplifica la sostituzione del DB o l'uso di ORM (SQLAlchemy, SQLModel) senza modificare la logica core. [\[60\]](#)

### 2. Service Layer Pattern

La logica applicativa (es. validare transazioni, calcolare bilanci) viene messa in servizi dedicati (WalletService, AuthService, TransactionService) che interagiscono con i repository. In questo modo, il controller REST resta snello e riusabile. [\[59\]](#) [\[64\]](#)

### 3. Dependency Injection Pattern

Per migliorare testabilità e modularità, il backend può iniettare dipendenze (repository, logger, gestore sicurezza) nei servizi o controller. Framework come FastAPI supportano questa pratica nativamente. [\[65\]](#) [\[58\]](#)

### 4. DTO (Data Transfer Object)

Gli oggetti DTO standardizzano lo scambio dati tra backend e frontend, tramite schemi Pydantic in FastAPI. Questo pattern migliora la validazione e separa il modello interno da quello esposto via API. [\[66\]](#) [\[65\]](#)

## Pattern strutturali e comportamentali

### 1. Observer/Event Pattern

Può essere adottato per notifiche interne (es. invio email o aggiornamento bilancio dopo una transazione) o per logging asincrono. Soluzioni come RabbitMQ/Kafka lo rendono scalabile. [\[59\]](#)

### 2. Factory Pattern

Utile nella creazione di oggetti Wallet o Transaction differenziando tra tipi di transazioni o wallet senza cambiare la logica client. [\[62\]](#)

### 3. Strategy Pattern

Applicabile per algoritmi intercambiabili, ad esempio per diversi modelli di validazione o politiche di calcolo delle commissioni, rendendo la logica più estendibile. [\[67\]](#) [\[58\]](#)

### 4. Singleton Pattern

Il gestore di connessioni database o il logger applicativo possono essere implementati come singleton, garantendo una singola istanza condivisa tra moduli. [\[63\]](#) [\[62\]](#)

## Sintesi dei pattern individuati

Categoria	Pattern	Scopo principale
Architetturale	Layered / N-Tier	Separazione tra logica, dati e UI <a href="#">[58]</a>
Architetturale	Client-Server	Comunicazione REST tra frontend e backend <a href="#">[61]</a>
Strutturale	Repository	Gestione centralizzata della persistenza <a href="#">[60]</a>
Strutturale	Service Layer	Organizzazione della logica di business <a href="#">[64]</a>
Strutturale	DTO	Validazione e trasporto dati tra livelli <a href="#">[66]</a>
Comportamentale	Observer	Esecuzione asincrona di eventi (notifiche) <a href="#">[59]</a>
Creazionale	Factory	Creazione flessibile di entità o servizi <a href="#">[62]</a>
Comportamentale	Strategy	Selezione dinamica di algoritmi <a href="#">[58]</a>
Strutturale	Singleton	Gestione risorse condivise (DB, configurazioni) <a href="#">[63]</a>

Nel complesso, questa combinazione di design pattern riflette le migliori pratiche di sviluppo **API-first fintech application**, garantendo **scalabilità, modularità e chiarezza architetturale**, perfettamente coerenti con un **Project Work universitario full-stack**.



# Esportami il project work in .md con tutte le osservazioni effettuate

Ecco l'esportazione in formato Markdown (.md) che include tutte le osservazioni effettuate sul Project Work per lo sviluppo di un portafoglio crypto centralizzato API-based nel contesto del CdS Informatica per le Aziende Digitali (L-31):

```
# Project Work - Informatica per le Aziende Digitali (L-31)
```

```
## Titolo
```

```
Sviluppo di un'applicazione full-stack API-based per un'impresa del settore finanziario:
```

```
---
```

```
## Contesto e obiettivi
```

```
Le imprese del settore finanziario gestiscono dati e transazioni in tempo reale. L'obietti
```

```
L'applicazione dovrà integrare:
```

- Backend RESTful API (es. Python FastAPI)
- Frontend web (React, HTML/CSS/JavaScript)
- Architettura API-based moderna
- Simulazione di operazioni crypto senza blockchain reale

```
---
```

```
## Vantaggi del progetto
```

- Coerenza con la digitalizzazione fintech e i trend in ambito finanziario moderno.
- Uso di tecnologie attuali (API REST, frontend dinamico).
- Dimostrazione di competenze in architettura software, sicurezza e modellazione dati.
- Scalabilità tramite sviluppo incrementale (MVP).
- Impatto accademico elevato per innovazione e attualità [web:4][web:21][web:27].

```
---
```

```
## Svantaggi e criticità
```

- Complessità nella gestione della sicurezza e dati sensibili.
- Richiesta di competenze interdisciplinari IT-finanza.
- Sfida nel completare un ciclo completo in 30 giorni.
- Possibile carenza nella parte gestionale senza adeguato focus su modello business e noi

```
---
```

```
## Piano di sviluppo (30 giorni)
```

```
### Settimane 1-2: Progettazione e backend API
```

- Definizione requisiti e use case.
- Realizzazione diagrammi UML (Use Case, Class Diagram, Sequence Diagram).
- Sviluppo backend con FastAPI o equivalente: gestione utenti, wallet, transazioni.
- Testing e documentazione API con Swagger/Postman.

```
### Settimana 3: Frontend e test
```

- Creazione interfaccia utente React o JavaScript.
- Integrazione chiamate API.
- Test funzionali con dati simulati.

#### ### Settimana 4: Documentazione e rifinitura

- Redazione report dettagliato con UML, API doc, screenshot.
- Commento snippet di codice e descrizione processo agile.
- Verifica stabilità MVP senza blockchain reale.

---

#### ## Design Pattern individuati

Categoria	Pattern	Scopo principale
-----	-----	-----
Architetturale	Layered / N-Tier	Separazione livelli UI, logica e persistenza
Architetturale	Client-Server	Separazione frontend da backend tramite API
Architetturale	MVC	Organizzazione in Model, View e Controller
Strutturale	Repository	Accesso dati centralizzato, astratto dalla logica
Strutturale	Service Layer	Logica di business incapsulata in servizi
Strutturale	DTO	Validazione e trasporto dati tra backend e frontend
Comportamentale	Observer/Event	Gestione eventi asincroni (es. notifiche)
Creazionale	Factory	Creazione flessibile di oggetti wallet o transazioni
Comportamentale	Strategy	Algoritmi intercambiabili (validazione, comm
Strutturale	Singleton	Gestione univoca di risorse condivise

---

#### ## Considerazioni finali

Il progetto è coerente, formativo e realizzabile con una pianificazione agile di 30 giorni. L'approccio MVP consente di creare un prototipo dimostrabile senza dipendere da una blockchain reale.

---

#### \*\*Fonti e approfondimenti\*\*:

Le informazioni e indicazioni si basano su linee guida universitarie ufficiali, best practices e risorse open source.

Questo template può essere ulteriormente adattato e integrato con codice, diagrammi, e documentazione specifica sviluppata nel progetto.

✱

1. <https://www.studocu.com/it/document/universita-telematica-pegaso/informatica-per-aziende-digitali/project-work-template-criteri-valutazione/100040246>
2. <https://www.unipegaso.it/lauree-triennali/informatica-per-le-aziende-digitali>
3. <https://www.unipegasoroma.com/l-31>
4. <https://webthesis.biblio.polito.it/26381/1/tesi.pdf>
5. <https://www.studocu.com/it/course/universita-telematica-pegaso/informatica-per-aziende-digitali/5874894>
6. <https://www.irpet.it/wp-content/uploads/2023/06/fabbrica-4-0-lombardi.pdf>

7. <https://universitadigitalepegaso.it/corso-di-laurea/informatica-per-le-aziende-digitali/>
8. [https://www.anitec-assinform.it/kdocs/2172877/ricerca\\_e\\_innovazione\\_ict\\_in\\_italia\\_2025\\_def.pdf](https://www.anitec-assinform.it/kdocs/2172877/ricerca_e_innovazione_ict_in_italia_2025_def.pdf)
9. <https://www.unipegaso.it/ateneo/assicurazione-qualita/documenti-sua/l-31---informatica-per-le-aziende-digitali>
10. [https://tesi.luiss.it/32685/1/144733\\_BARMANN\\_ENRICO.pdf](https://tesi.luiss.it/32685/1/144733_BARMANN_ENRICO.pdf)
11. <https://pavesioassociati.it/wp-content/uploads/2023/06/Blockchain-new-markets--Futurew3b-con-il-supperto-di-PwC.pdf>
12. [https://www.assogestioni.it/sites/default/files/docs/assogestioni\\_fondiitalianidigitali\\_lineeguida\\_maggio2024\\_ita.pdf](https://www.assogestioni.it/sites/default/files/docs/assogestioni_fondiitalianidigitali_lineeguida_maggio2024_ita.pdf)
13. [https://www.bancaditalia.it/pubblicazioni/qef/2024-0863/QEF\\_863\\_24\\_IT.pdf](https://www.bancaditalia.it/pubblicazioni/qef/2024-0863/QEF_863_24_IT.pdf)
14. [https://www.tupponi-demarinis.it/kdocs/2032072/guida\\_tech\\_law.pdf](https://www.tupponi-demarinis.it/kdocs/2032072/guida_tech_law.pdf)
15. <https://www.ice.it/it/sites/default/files/inline-files/RPI - IL RUOLO DELLA DIGITALIZZAZIONE.pdf>
16. <https://air.unimi.it/retrieve/278954d1-ccfa-4132-b597-8b086221ef48/9791221161892.pdf>
17. [https://www.oecd.org/content/dam/oecd/it/publications/reports/2020/09/blockchain-for-smes-and-entrepreneurs-in-italy\\_d69d5014/bdbbb4ea-it.pdf](https://www.oecd.org/content/dam/oecd/it/publications/reports/2020/09/blockchain-for-smes-and-entrepreneurs-in-italy_d69d5014/bdbbb4ea-it.pdf)
18. <https://www.studocu.com/it/document/universita-telematica-pegaso/informatica-per-aziende-digitali/project-work-template-criteri-valutazione/100040246>
19. <https://www.unipegaso.it/lauree-triennali/informatica-per-le-aziende-digitali>
20. <https://hexn.io/it/blog/vantaggi-e-svantaggi-della-tecnologia-blockchain-zbltcrffzih4f3cjkwzrwwm>
21. <https://www.ictsecuritymagazine.com/articoli/blockchain-cosa-cosa-non/>
22. <https://www.binance.com/it/academy/articles/Positives-And-Negatives-Of-Blockchain>
23. [https://www.agid.gov.it/sites/agid/files/2024-06/Linee\\_guida\\_per\\_ladozione\\_di\\_un\\_ciclo\\_di\\_sviluppo\\_di\\_software\\_sicuro.pdf](https://www.agid.gov.it/sites/agid/files/2024-06/Linee_guida_per_ladozione_di_un_ciclo_di_sviluppo_di_software_sicuro.pdf)
24. <https://unitesi.unive.it/bitstream/20.500.14247/11661/1/863726-1263666.pdf>
25. [https://tesi.luiss.it/31897/1/227001\\_BRIGANTI\\_NICCOLO'.pdf](https://tesi.luiss.it/31897/1/227001_BRIGANTI_NICCOLO'.pdf)
26. <https://uniscientia.it/laurea-in-informatica-per-le-aziende-digitali/>
27. <https://webthesis.biblio.polito.it/24316/1/tesi.pdf>
28. <https://webthesis.biblio.polito.it/24186/1/tesi.pdf>
29. [https://tesi.univpm.it/retrieve/e7d36e7c-7701-4669-8418-fc2e66c91aa6/tesi\\_alessio\\_clementi\\_28.09-2.pdf](https://tesi.univpm.it/retrieve/e7d36e7c-7701-4669-8418-fc2e66c91aa6/tesi_alessio_clementi_28.09-2.pdf)
30. [https://www.anitec-assinform.it/kdocs/2172877/ricerca\\_e\\_innovazione\\_ict\\_in\\_italia\\_2025\\_def.pdf](https://www.anitec-assinform.it/kdocs/2172877/ricerca_e_innovazione_ict_in_italia_2025_def.pdf)
31. <https://www.eticasgr.com/storie/approfondimenti/criptovalute>
32. <https://www.unipegaso.it/pdf/programma/0602509SECSP11INM>
33. <https://www.agendadigitale.eu/cittadinanza-digitale/pagamenti-digitali/criptovalute-e-finanza-decentralizzata-e-tutto-oro-quel-che-luccica-vantaggi-e-rischi/>
34. <https://disaq.uniparthenope.it/fintechlab/selezioni/>
35. <https://college.h-farm.com/percorsi-universitari/laurea-triennale-digital-economics-finance/>
36. <https://www.studocu.com/it/course/universita-telematica-pegaso/informatica-per-le-aziende-digitali/6585969>
37. <https://www.facebook.com/groups/472752960893584/posts/ciao-a-tuttivolevo-chiedervi-dal-superamento-antiplagio-del-project-work-in-piat/1332143328287872/>

38. <https://www.azionadigitale.com/sviluppiamo-una-restful-api-con-python-e-fastapi/>
39. <https://lucaterribili.it/come-creare-tuo-primo-progetto-software-guida-passo-passo>
40. <https://larin.it/mvp-development-cose-e-come-svilupparlo/>
41. <https://www.startupgeeks.it/mvp-minimum-viable-product/>
42. <https://scienzadeidati.com/blog/category/web-application/>
43. [https://www.philmark.it/wp-content/uploads/2018/01/CATALOGO-ACADEMY\\_2017-2018-1.pdf](https://www.philmark.it/wp-content/uploads/2018/01/CATALOGO-ACADEMY_2017-2018-1.pdf)
44. <https://www.skillfactory.it/corsi/agile-full-stack-developer>
45. <https://www.lunover.com/it/services/mvp/>
46. <https://docenti.ing.unipi.it/~a009435/issw/isw2122.pdf>
47. [https://thesis.unipd.it/retrieve/6eef0e27-1b20-46b7-a9c0-a89b91b0504d/Corbu\\_TeodorMihail.pdf](https://thesis.unipd.it/retrieve/6eef0e27-1b20-46b7-a9c0-a89b91b0504d/Corbu_TeodorMihail.pdf)
48. <https://techpulsion.com/it/idee-di-progetti-di-sviluppo-web/>
49. <https://www.opstart.it/it/documenti/progetti/5463-DFF5FF06B58E71A66C36381F4C908814>
50. <http://moltiplytech.com/it/chi-siamo/metodo-di-sviluppo>
51. <https://www.freelancer.co.it/freelancers/skills/artificial-intelligence>
52. <https://www.youtube.com/watch?v=OvOSOO6PR3k>
53. <https://innowise.com/it/servizi/sviluppo-mvp/>
54. <https://techpulsion.com/it/roadmap-per-sviluppatori-full-stack/>
55. <https://www.freelancer.co.it/freelancers/skills/penetration-testing>
56. <https://www.freelancer.co.it/freelancers/skills/wikipedia>
57. [https://thesis.unipd.it/retrieve/b2161d20-d0ef-4535-964e-8016d68a96f1/Barutta\\_Andrea.pdf](https://thesis.unipd.it/retrieve/b2161d20-d0ef-4535-964e-8016d68a96f1/Barutta_Andrea.pdf)
58. <https://www.clustox.com/blog/software-architecture-patterns/>
59. <https://codemia.io/system-design/design-a-digital-wallet/solutions/sg72vs/My-Solution-for-Design-a-Digital-Wallet>
60. <https://learn.microsoft.com/it-it/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastucture-persistence-layer-design>
61. <https://tecnovy.com/en/top-10-software-architecture-patterns>
62. <https://www.mindinventory.com/blog/software-architecture-patterns/>
63. <https://www.moontechnolabs.com/blog/software-architecture-patterns/>
64. <https://white-label-digital-wallet.qulix.com/blog/api-first-digital-wallet-architecture>
65. <https://fastapi.tiangolo.com/project-generation/>
66. <https://dev.to/yagnesh97/building-a-modern-web-app-fastapi-react-typescript-template-5d88>
67. <https://iamjeremie.me/post/2025-08/the-design-patterns-and-architecture-i-encountered/>
68. <https://github.com/fastapi/full-stack-fastapi-template>
69. <https://www.youtube.com/watch?v=13tMEW8r6C0>
70. <https://www.datacamp.com/tutorial/farm-stack-guide>
71. <https://testdriven.io/blog/fastapi-react/>
72. [https://www.reddit.com/r/FastAPI/comments/1dhzdif/fullstack\\_developers\\_using\\_fastapi\\_whats\\_your/](https://www.reddit.com/r/FastAPI/comments/1dhzdif/fullstack_developers_using_fastapi_whats_your/)
73. <https://softwaremind.com/blog/guide-to-creating-digital-wallets/>
74. <https://www.youtube.com/watch?v=aSdVU9-SxH4>

75. <https://www.wallarm.com/what/digital-wallet-apis-a-guide-to-integrating-into-your-app>
76. <https://www.digipay.guru/blog/how-digital-wallet-api-solution-make-financial-services-efficient/>
77. <https://www.linkedin.com/pulse/building-high-performance-e-wallet-backend-service-using-otun>
78. [https://github.com/Guray00/template\\_lectures](https://github.com/Guray00/template_lectures)
79. <https://aulab.it/blog/scrivilo-con-markdown-un-coltellino-svizzero-per-la-tua-documentazione>
80. [https://roberto-marcolin.canoprof.fr/eleve/tutorial/Scrivere\\_con\\_Pandoc\\_e\\_Markdown.pdf@1](https://roberto-marcolin.canoprof.fr/eleve/tutorial/Scrivere_con_Pandoc_e_Markdown.pdf@1)
81. <https://www.markdownlang.com/it/advanced/>
82. <http://prmppt.onl/a01>
83. <https://developer.pagopa.it/app-io/guides/io-guida-tecnica/v7.1/risorse-utili/guida-al-markdown>
84. <https://www.markdownlang.com/it/intro/use-cases.html>
85. <https://lifemilch.eu/it/markdown.html>
86. <https://googledocs.readthedocs.io/it/latest/sintassi-md.html>