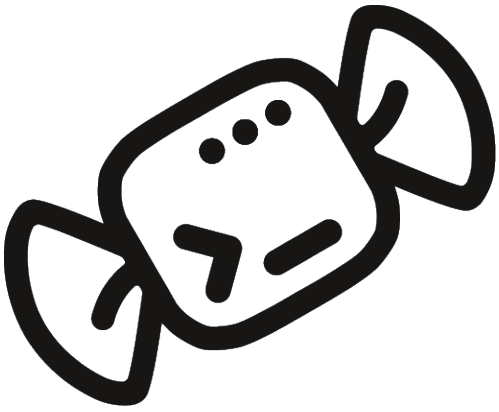

SWEetCode

Specifica tecnica

Componenti del gruppo

Bresolin G.
Campese M.
Ciriolo I.
Dugo A.
Feltrin E.
Michelon R.
Orlandi G.

Università di Padova
Ingegneria del Software

Registro delle versioni

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
v2.0.0(7)	$2024-02-28$	Dugo A. Bresolin G.	Michelon R.	Aggiunta progettazione logica UploadDocuments.
v2.0.0(6)	$2024-02-28$	Feltrin E. Michelon R.	Bresolin G.	Prima stesura architettura di sistema e tecnologie utilizzate.

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Obiettivo del documento

L'obiettivo che ci si pone nella realizzazione di questo documento è descrivere le scelte tecnologiche e l'architettura del prodotto *Knowledge Managment AI*. Verrà seguito un approccio top-down, partendo dall'architettura del sistema passando poi all'architettura delle componenti ed infine alla progettazione di dettaglio.

1.2 Glossario

Per evitare ambiguità ed incomprensioni relative al linguaggio e ai termini utilizzati nella documentazione del progetto viene presentato un Glossario. I termini ambigui o tecnici-specifici presenti nello stesso, vengono identificati nei corrispondenti documenti con un pedice |g| e con una scrittura in corsivo. All'interno dei documenti viene identificata con tale scrittura solo e soltanto la prima occorrenza presente nel testo di un termine definito nel Glossario.

1.3 Riferimenti

1.3.1 Riferimenti normativi

- *(Norme di progetto v2.0.0(0))*;
- *Regolamento del progetto didattico*:
  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf`
  (Ultimo accesso: 2024-02-26);
- *Standard ISO/IEC 9126*:
  `https://it.wikipedia.org/wiki/ISO/IEC_9126`
  (Ultimo accesso: 2024-02-26).

1.3.2 Riferimenti informativi

- *(Analisi dei requisiti v2.0.0(0))*;
- *Capitolato C1*: *Knowledge Management AI*
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C1.pdf`
    (Ultimo accesso: 2024-02-26);
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C1p.pdf`
    (Ultimo accesso: 2024-02-26).
- *Dispense su Dependency Injection*:
  `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Architetturali%20-%20Dependency%20Injection.pdf`
  (Ultimo accesso: 2024-02-26);
- *Dispense su OOP*:
  `https://www.math.unipd.it/~rcardin/swea/2023/Object-Oriented%20Progamming%20Principles%20Revised.pdf`
  (Ultimo accesso: 2024-02-26);

- *Dispense su Diagrammi delle classi*:
  `https://www.math.unipd.it/~rcardin/swea/2023/Diagrammi%20delle%20Classi.pdf`
  (Ultimo accesso: 2024-02-26);
- *Dispense su Pattern architetturali*:
  `https://www.math.unipd.it/~rcardin/swea/2022/Software%20Architecture%20Patterns.pdf`
  (Ultimo accesso: 2024-02-26);
- *Dispense su Pattern creazionali*:
  `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Creazionali.pdf`
  (Ultimo accesso: 2024-02-26);
- *Dispense su Pattern strutturali*:
  `https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Strutturali.pdf`
  (Ultimo accesso: 2024-02-26);
- *Dispense su Principi SOLID*:
  `https://www.math.unipd.it/~rcardin/swea/2021/SOLID%20Principles%20of%20Object-Oriented%20Design_4x4.pdf`
  (Ultimo accesso: 2024-02-26);
- *Dispense sulla Progettazione software (argomento T6)*:
  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T6.pdf`
  (Ultimo accesso: 2024-02-26);
- *Dispense sulla Qualità del software (argomento T7)*:
  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T7.pdf`
  (Ultimo accesso: 2024-02-26);
- *Repository su Architettura esagonale*:
  `https://github.com/rcardin/hexagonal`
  (Ultimo accesso: 2024-02-26); `https://github.com/rcardin/hexagonal-java/`
  (Ultimo accesso: 2024-02-26);
- *Repository Ingegneria del software professor Cardin*:
  `https://github.com/rcardin/swe-imdb`
  (Ultimo accesso: 2024-02-26);
- Riferimenti a scelte tecnologiche:
  - *AWS S3*:
    `https://aws.amazon.com/it/s3/`
    (Ultimo accesso: 2024-02-26);
  - *ChromaDB*:
    `https://www.trychroma.com/`
    (Ultimo accesso: 2024-02-26);
  - *Docker*:
    `https://www.docker.com/`
    (Ultimo accesso: 2024-02-26);
  - *Flask*:
    `https://flask.palletsprojects.com/en/3.0.x/`
    (Ultimo accesso: 2024-02-26);

- *HuggingFace*:
  `https://huggingface.co/`
  (Ultimo accesso: 2024-02-26);
- *Langchain*:
  `https://python.langchain.com/docs/get_started/introduction`
  (Ultimo accesso: 2024-02-26);
- *Next.js*:
  `https://nextjs.org/`
  (Ultimo accesso: 2024-02-26);
- *OpenAI*:
  `https://openai.com/`
  (Ultimo accesso: 2024-02-26);
- *Pinecone*:
  `https://www.pinecone.io/`
  (Ultimo accesso: 2024-02-26);
- *Postgres*:
  `https://www.postgresql.org/`
  (Ultimo accesso: 2024-02-26);
- *Python*:
  `https://www.python.org/`
  (Ultimo accesso: 2024-02-26);
- *Typescript*:
  `https://www.typescriptlang.org/`
  (Ultimo accesso: 2024-02-26);

- *(Glossario v2.0.0(0))*;
- *(Piano di progetto v2.0.0(0))*;
- *(Piano di qualifica v2.0.0(0))*;
- *Verbali esterni ed interni*.

2 Tecnologie utilizzate

In questa sezione vengono elencate e descritte le tecnologie utilizzate nello sviluppo, illustrando le motivazioni a sostegno di ogni scelta e le alternative scartate.

2.1 Flask

Flask è un micro framework per applicazioni web. È stato scelto per la sua leggerezza e la sua flessibilità, rispetto ad altri framework come Django che potrebbero risultare troppo pesanti per le esigenze del progetto.

- **Vantaggi:**
  - **Leggerezza:** Flask è noto per la sua leggerezza, il che significa che ha poche dipendenze. Questo lo rende perfetto per progetti più piccoli dove non è necessario un carico pesante di funzionalità;
  - **Flessibilità:** Flask offre una grande flessibilità, permettendo agli sviluppatori di strutturare le loro applicazioni come preferiscono;
  - **Facilità d'uso:** Flask è facile da usare e da imparare, rendendolo ideale per i principianti.
- **Svantaggi:**
  - **Manca di alcune funzionalità out-of-the-box:** Flask è un microframework, il che significa che potrebbe non avere tutte le funzionalità che potrebbero essere necessarie per un'applicazione più complessa;
  - **Potrebbe richiedere più tempo per sviluppare applicazioni complesse:** A causa della sua natura minimalista, gli sviluppatori potrebbero dover scrivere più codice per realizzare funzionalità che in altri framework potrebbero essere disponibili out-of-the-box.

2.1.1 Python

Python è un linguaggio di programmazione ad alto livello, interpretato, interattivo, orientato agli oggetti e di script. È progettato per essere altamente leggibile. Rispetto ad altri linguaggi come Java o C++, Python offre una sintassi più semplice e pulita, rendendo il codice più leggibile e mantenibile.

- **Vantaggi:**
  - **Facilità d'uso:** Python ha una sintassi molto pulita e facile da leggere, il che rende il linguaggio molto facile da imparare per i nuovi programmatori;
  - **Versatilità:** Python può essere utilizzato per una vasta gamma di applicazioni, tra cui sviluppo web, data analysis, machine learning, intelligenza artificiale, creazione di GUI e scripting di sistema;
  - **Grande comunità:** Python ha una grande comunità di sviluppatori che contribuiscono attivamente alla sua manutenzione e miglioramento. Ciò significa che ci sono molte risorse disponibili per l'apprendimento e la risoluzione dei problemi.
- **Svantaggi:**

- **Velocità:** Python non è il linguaggio più veloce a causa della sua natura interpretata e può non essere la scelta migliore per le applicazioni che richiedono prestazioni elevate.

2.2 Next.js

Next.js è un framework per applicazioni web basato su React. È stato scelto per la sua efficienza e per le sue funzionalità di rendering lato server. Rispetto ad altri framework come Angular, Next.js offre una maggiore efficienza e facilità d'uso, rendendolo ideale per questo progetto.

- **Vantaggi:**
  - **Efficienza:** Next.js è noto per la sua efficienza, il che significa che le applicazioni create con Next.js sono veloci e performanti;
  - **Rendering lato server:** Next.js offre funzionalità di rendering lato server, il che significa che può migliorare le prestazioni dell'applicazione e l'ottimizzazione dei motori di ricerca;
  - **Facilità d'uso:** Next.js è facile da usare e da imparare, specialmente per coloro che sono già familiari con React.
- **Svantaggi:**
  - **Overhead:** Next.js può aggiungere un certo overhead a un'applicazione a causa delle sue funzionalità aggiuntive, il che può non essere necessario per le applicazioni più semplici;
  - **Complessità:** A causa delle sue funzionalità aggiuntive, Next.js può essere più complesso da configurare e gestire rispetto a React da solo.

2.2.1 Typescript

Typescript è un super-set di JavaScript che aggiunge tipi statici e oggetti orientati alla programmazione. È stato scelto per la sua affidabilità e robustezza. Rispetto a JavaScript, TypeScript offre un controllo dei tipi più rigoroso, il che può aiutare a prevenire errori di runtime.

- **Vantaggi:**
  - **Affidabilità:** Typescript offre un controllo dei tipi a tempo di compilazione, il che significa che gli errori possono essere rilevati e corretti prima dell'esecuzione;
  - **Robustezza:** Typescript supporta le funzionalità di programmazione orientata agli oggetti, il che può rendere il codice più robusto e facile da gestire;
  - **Interoperabilità:** Typescript è un super-set di JavaScript, il che significa che qualsiasi codice JavaScript valido può essere utilizzato in Typescript.
- **Svantaggi:**
  - **Curva di apprendimento:** Typescript può essere più difficile da imparare rispetto a JavaScript a causa delle sue funzionalità aggiuntive.

2.3 Docker

Docker è una piattaforma open source che automatizza la distribuzione, la scalabilità e l'isolamento delle applicazioni utilizzando la virtualizzazione a livello di sistema operativo. È stato scelto per la sua efficienza e portabilità. Rispetto ad altre soluzioni come Vagrant, Docker offre una maggiore efficienza e facilità d'uso.

- **Vantaggi:**
  - **Efficienza:** Docker consente di eseguire più applicazioni in modo isolato sulla stessa infrastruttura hardware, migliorando l'efficienza e riducendo i costi;
  - **Portabilità:** Con Docker, le applicazioni e le loro dipendenze possono essere confezionate come un'unità portatile chiamata container, che può essere eseguita su qualsiasi macchina che supporti Docker;
  - **Isolamento:** Docker isola le applicazioni in container separati, il che significa che ogni applicazione può avere le proprie dipendenze e non interferire con le altre applicazioni.
- **Svantaggi:**
  - **Complessità:** Docker può aggiungere una certa complessità a un progetto a causa della necessità di gestire i container e le loro dipendenze;
  - **Curva di apprendimento:** Docker ha una curva di apprendimento ripida e può richiedere un certo tempo per essere padroneggiato.

2.4 Langchain

Langchain è un framework che facilita l'interazione tra modelli di apprendimento automatico e risorse esterne come database o altri servizi web. È stato scelto per la sua comodità e flessibilità. Rispetto ad altri framework come TensorFlow o PyTorch, Langchain offre una maggiore facilità d'uso e una migliore integrazione con le risorse esterne.

- **Vantaggi:**
  - **Comodità:** Langchain semplifica l'interazione tra modelli di apprendimento automatico e risorse esterne, fornendo moduli e integrazioni;
  - **Flessibilità:** Langchain supporta una varietà di modelli di apprendimento automatico e risorse esterne, rendendolo adatto a una vasta gamma di applicazioni;
  - **Facilità d'uso:** Langchain è facile da usare e da imparare, rendendolo ideale per i principianti.
- **Svantaggi:**
  - **Limitazioni:** Non tutte le funzionalità sono disponibili in tutte le lingue supportate da Langchain;
  - **Documentazione:** La documentazione di Langchain potrebbe non essere così completa o aggiornata come quella di altri framework.

2.4.1 Pinecone

Pinecone è un database di vettori che consente di effettuare ricerche di similarità su larga scala. È stato scelto per la sua efficienza e precisione. Rispetto ad altri database di vettori come Faiss o Annoy, Pinecone offre una maggiore efficienza e una migliore precisione nelle ricerche di similarità.

- **Vantaggi:**
  - **Efficienza:** Pinecone è progettato per effettuare ricerche di similarità su larga scala in modo efficiente;
  - **Precisione:** Pinecone offre un'alta precisione nelle ricerche di similarità, il che lo rende ideale per applicazioni che richiedono un alto grado di precisione;
  - **Facilità d'uso:** Pinecone è facile da usare e da imparare, rendendolo ideale per i principianti.
- **Svantaggi:**
  - **Costo:** Pinecone può essere costoso da utilizzare per applicazioni su larga scala;
  - **Limitazioni:** Pinecone potrebbe non supportare tutte le funzionalità di ricerca di similarità che potrebbero essere necessarie per alcune applicazioni.

2.4.2 ChromaDB

ChromaDB è un database di vettori locale open-source. È stato scelto per la sua integrazione con Langchain e la sua popolarità tra i database di vettori locali. Rispetto ad altri database di vettori locali come Faiss o Annoy, ChromaDB offre una migliore integrazione con Langchain e una maggiore popolarità.

- **Vantaggi:**
  - **Integrazione:** ChromaDB è ben integrato con Langchain, il che facilita l'interazione tra i due;
  - **Popolarità:** ChromaDB è il database di vettori locale open-source più popolare, il che significa che ha una grande comunità di sviluppatori e molte risorse disponibili;
  - **Facilità d'uso:** ChromaDB è facile da usare e da imparare, rendendolo ideale per i principianti.
- **Svantaggi:**
  - **Limitazioni:** Non tutte le funzionalità sono disponibili in tutte le lingue supportate da ChromaDB;
  - **Documentazione:** La documentazione di ChromaDB potrebbe non essere così completa o aggiornata come quella di altri database.

2.4.3 OpenAI

OpenAI è una piattaforma di apprendimento automatico che offre una varietà di modelli, tra cui GPT-3 e GPT-4. La scelta di utilizzare OpenAI sia per il Large Language Model (LLM) che per gli embeddings è motivata dalla sua reputazione consolidata nel

campo dell'apprendimento automatico e dalla sua completa integrazione con Langchain. Rispetto all'addestramento di modelli personalizzati con TensorFlow o PyTorch, l'utilizzo dei modelli pre-addestrati di OpenAI offre una maggiore facilità d'uso.

- **Vantaggi:**
  - **Popolarità:** OpenAI è molto conosciuto nel campo dell'apprendimento automatico, il che significa che ha una grande comunità di sviluppatori e molte risorse disponibili;
  - **Integrazione:** OpenAI ha un'integrazione completa con Langchain, il che facilita l'interazione tra i due;
  - **Flessibilità:** OpenAI offre la possibilità di scegliere tra diversi modelli semplicemente cambiando un parametro, a seconda delle esigenze e della disponibilità dell'utente finale.
- **Svantaggi:**
  - **Costo:** L'utilizzo di OpenAI può essere costoso, soprattutto per le applicazioni su larga scala;
  - **Limitazioni:** Non tutte le funzionalità sono disponibili in tutti i modelli offerti da OpenAI.

2.4.4 HuggingFace

HuggingFace è una piattaforma di apprendimento automatico che offre migliaia di modelli open-source, tra cui modelli di linguaggio e modelli di embeddings. La decisione di utilizzare HuggingFace sia per il Large Language Model (LLM) che per gli embeddings è motivata dalla sua flessibilità e dalla possibilità di scaricare i modelli localmente per l'esecuzione offline. Questo offre una maggiore flessibilità rispetto all'addestramento di modelli personalizzati con TensorFlow o PyTorch, che può richiedere risorse computazionali significative e competenze specialistiche.

- **Vantaggi:**
  - **Flessibilità:** HuggingFace offre una vasta gamma di modelli, il che significa che è possibile scegliere il modello più adatto alle proprie esigenze;
  - **Località:** HuggingFace offre la possibilità di scaricare i modelli in locale, il che significa che possono essere eseguiti sulle proprie macchine senza la necessità di una connessione internet;
  - **Facilità d'uso:** HuggingFace è facile da usare e da imparare, rendendolo ideale per i principianti.
- **Svantaggi:**
  - **Risorse:** L'esecuzione dei modelli in locale può richiedere molte risorse hardware, il che può non essere ideale per tutte le macchine;
  - **Complessità:** A causa della vasta gamma di modelli disponibili, può essere difficile scegliere il modello più adatto alle proprie esigenze.

2.5 AWS S3

Amazon S3 (Simple Storage Service) è un servizio di storage di oggetti offerto da Amazon Web Services. È stato scelto per la sua scalabilità, affidabilità, e sicurezza. Rispetto ad altre soluzioni di storage come Google Cloud Storage o Azure Blob Storage, AWS S3 offre una maggiore scalabilità e una migliore integrazione con altri servizi AWS.

- **Vantaggi:**
  - **Scalabilità:** Amazon S3 può memorizzare qualsiasi quantità di dati e servire qualsiasi livello di traffico richiesto;
  - **Affidabilità:** Amazon S3 offre una durabilità dell'11 9's, il che significa che i dati sono estremamente sicuri;
  - **Sicurezza:** Amazon S3 offre potenti funzionalità per proteggere i dati, tra cui controllo degli accessi, crittografia in transito e a riposo, e altro ancora.
- **Svantaggi:**
  - **Costo:** Il costo di Amazon S3 può aumentare rapidamente con l'aumentare dell'uso;
  - **Complessità:** Amazon S3 ha molte funzionalità e opzioni, il che può renderlo complesso da configurare e gestire.

2.6 Postgres

Postgres, o PostgreSQL, è un potente sistema di gestione di database relazionali ad oggetti open source. È stato scelto per la sua robustezza, affidabilità e flessibilità. Rispetto ad altri DBMS come MySQL o SQLite, Postgres offre una maggiore robustezza e una migliore supporto per le funzionalità di programmazione orientata agli oggetti.

- **Vantaggi:**
  - **Robustezza:** Postgres supporta una vasta gamma di tipi di dati nativi, operatori e funzioni, tra cui JSON, XML e array;
  - **Affidabilità:** Postgres è noto per la sua affidabilità e integrità dei dati. Offre transazioni atomiche, commit multi-versione (MVCC), punti di controllo, logging di scrittura anticipata (WAL) e una serie di meccanismi di replica;
  - **Flessibilità:** Postgres è estensibile, il che significa che gli sviluppatori possono definire i propri tipi di dati, operatori e funzioni. Inoltre, può essere utilizzato sia come un database SQL tradizionale che come una soluzione NoSQL per la memorizzazione di documenti.
- **Svantaggi:**
  - **Complessità:** A causa della sua vasta gamma di funzionalità, Postgres può essere più complesso da configurare e gestire rispetto ad altri sistemi di gestione di database;
  - **Prestazioni:** Sebbene Postgres sia altamente ottimizzato, le sue prestazioni potrebbero non essere all'altezza di altri database per alcune applicazioni, in particolare quelle che richiedono letture ad alta velocità di grandi quantità di dati.

3 Architettura di sistema

3.1 Modello architetturale

Il sistema è progettato seguendo l'**architettura esagonale**, un modello architetturale che mira a creare una separazione netta tra la business logic dell'applicazione e i servizi esterni, le fonti di dati e le interfacce utente con cui interagisce. Questa struttura organizzativa posiziona il nucleo al centro, circondato da "porte" che fungono da interfaccia tra il nucleo e il mondo esterno.

Il **nucleo** dell'applicazione è il fulcro del sistema, contenente la logica di dominio e le regole di business. La sua progettazione mira a evitare riferimenti diretti a dettagli tecnologici specifici, promuovendo l'indipendenza dal contesto esterno.

Le **porte** costituiscono il confine tra il nucleo dell'applicazione e il mondo esterno, consentendo una comunicazione strutturata. Esistono due tipi principali di porte:

- Inbound Port (o **Use Case**): consentono al nucleo di essere invocato da componenti esterni attraverso un'interfaccia definita. Rappresentano i punti di accesso al nucleo e isolano la logica di dominio da implementazioni specifiche;
- Outbound Port: consentono al nucleo di accedere a funzionalità esterne, come l'interazione con librerie esterne o sistemi di persistenza. Forniscono un'astrazione che preserva l'indipendenza del nucleo da dettagli tecnologici specifici.

I **services** implementano le inbound port dell'applicazione e fanno parte della business logic. La loro implementazione è concentrata sulla logica di dominio, senza preoccuparsi degli aspetti tecnologici specifici.

Gli **adapters** costituiscono il livello più esterno dell'applicazione. Esistono due tipi di adapters:

- Input Adapters (o **Controllers**): sono responsabili di invocare operazioni sulle porte in ingresso. Traducono le azioni provenienti dall'esterno in chiamate alle porte in ingresso del nucleo, facilitando la traduzione delle richieste esterne in operazioni comprensibili per il nucleo;
- Output Adapters: gestiscono le porte in uscita, traducendo le azioni del nucleo in operazioni comprensibili per il mondo esterno.

3.2 Descrizione delle componenti

L'architettura generale del sistema è composta da due componenti: frontend e backend.

3.2.1 Frontend

Il frontend si occupa di fornire un'interfaccia grafica all'utente per dialogare con il sistema. Inoltra le richieste dell'utente al backend e mostra i risultati ottenuti.

3.2.2 Backend

Il backend si occupa di elaborare le richieste degli utenti, interagendo con i sistemi di persistenza e i servizi esterni. In particolare, il backend dialoga con il sistema di archi-

viazione documenti, il vector store, il database delle chat e con i modelli di intelligenza artificiale necessari per il corretto funzionamento dell'applicazione.

3.3 Assemblaggio delle componenti

Le componenti sono assemblate insieme utilizzando Docker Compose. In particolare sono prodotti i seguenti container Docker:

- **db**: espone l'istanza del database chat nella porta 3000, abilitando il dialogo con il backend;
- **backend**: espone la componente backend nella porta 4000, dando al frontend la possibilità di chiamare le API offerte;
- **frontend**: espone il frontend dell'applicazione web nella porta 80, dando la possibilità all'utente di connettersi e interagire con il sistema.

3.4 Struttura del sistema

3.4.1 Frontend

La struttura organizzativa del frontend segue la struttura standard definita dal framework Next.js.

3.4.2 Backend

La struttura organizzativa del backend segue la seguente struttura:

```
/backend
        /adapter
                /in
                   /web -- controllers
                /out -- implementazioni di Outbound Port
        /application
                /port
                   /in -- Inbound Ports (Use Cases)
                   /out -- Outbound Ports
                   /service -- implementazioni di Inbound Port
        /domain -- classi di business
```

Questa struttura riflette il modello architetturale scelto, facilitando il passaggio da progettazione a codifica.

4 Architettura delle componenti

4.1 Frontend

4.2 Backend

4.3 Database

4.4 Libreria per la persistenza

4.4.1 UploadDocuments

4.4.1.1 Descrizione

Questa componente ha il compito di memorizzare i documenti inseriti dall'utente nel sistema di archiviazione e calcolare gli embeddings, memorizzandoli nel vector store configurato. È costituita da:

- **Route API**: /uploadDocuments;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - "documents": una lista di documenti.

4.4.1.2 Esiti possibili

Codice	Descrizione	Risposta
200	Upload avvenuto con successo.	-
500	Upload fallito.	Errore nell'upload dei documenti.
500	Upload fallito.	Errore nel calcolo degli embeddings dei documenti.

4.4.1.3 Tracciamento dei requisiti

- RF.O.21
- RF.O.21.1
- RF.O.21.3
- RF.O.21.4
- RF.O.21.6

4.4.1.4 Lista sottocomponenti

- **NewDocument**: classe di presentation che contiene le informazioni relative ai documenti appena inseriti dall'utente, presenti nella richiesta HTTP;
- **DocumentMetadata**: classe di business che rappresenta i metadati di un documento;
- **DocumentType**: enumeration che rappresenta i valori che può assumere il tipo di un documento;

- **DocumentId**: classe di business che rappresenta l'id di un documento;
- **DocumentStatus**: classe di business che rappresenta lo status di un documento;
- **Status**: enumeration che rappresenta i valori che può assumere lo status di un documento;
- **DocumentContent**: classe di business che rappresenta il contenuto di un documento;
- **Document**: classe di business che rappresenta i documenti completi;
- **PlainDocument**: classe di business che rappresenta i documenti, compresi i metadati e il contenuto;
- **UploadDocumentController**: classe controller che si occupa del passaggio di Documents allo use case UploadDocumentUseCase a partire da NewDocuments;
- **UploadDocumentUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare l'upload dei documenti;
- **UploadDocumentService**: classe service che implementa lo use case UploadDocumentUseCase;
- **DocumentUploader**: classe che si occupa di effettuare la chiamata dell'upload dei documenti nella porta esterna diretta verso il sistema di archiviazione documenti;
- **EmbeddingsUploader**: classe che si occupa di effettuare la chiamata dell'upload degli embeddings nella porta esterna diretta verso il vector database;
- **DocumentUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload dei documenti verso il sistema di archiviazione doucumenti;
- **EmbeddingsUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload degli embeddings verso i vector store;
- **DocumentUploaderAWSS3**: classe adapter che implementa la porta DocumentUploaderPort, adattando la chiamata di DocumentUploader a AWSS3Manager;
- **AWSDocument**: classe che rappresenta i documenti gestibili da AWSS3Manager;
- **AWSDocumentOperationResponse**: classe che contiene l'esito di un'operazione effettuata su un AWS S3 riguardo un documento;
- **DocumentOperationResponse**: classe che contiene l'esito di un'operazione effettuata su un documento;
- **EmbeddingsUploaderFacadeLangchain**: classe adapter che implementa la porta EmbeddingsUploaderPort, adattando la chiamata di EmbeddingsUploader alla sequenza di operazioni necessarie per il calcolo degli embeddings e il loro upload nel vector store;
- **LangchainDocument**: classe che rappresenta un documento e i suoi embeddings;

- **AWSS3Manager**: classe che si occupa di effettuare le operazioni sul bucket di Amazon S3 configurato, cioè il sistema di archiviazione documenti dell'applicazione;
- **DocumentToText**: classe che si occupa di convertire documenti;
- **TextExtractor**: interfaccia che espone il metodo astratto di estrazione di testo da un documento;
- **PDFTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti PDF;
- **DOCXTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti docx;
- **Chunkerizer**: classe che crea chunks a partire da testo;
- **EmbeddingsCreator**: classe che dialoga con il modello di embeddings per creare gli embeddings a partire dai chunk forniti in input;
- **LangchainEmbeddingModel**: interfaccia che rappresenta il modello di embeddings;
- **OpenAIEmbeddingModel**: classe che implementa LangchainEmbeddingModel offrendo la possibilità di creare embeddings attraverso l'embedding model di OpenAI;
- **HuggingFaceEmbeddingModel**: classe che implementa LangchainEmbeddingModel offrendo la possibilità di creare embeddings attraverso un embedding model di HuggingFace;
- **EmbeddingsUploaderVectorStore**: classe che offre un metodo per eseguire l'upload degli embeddings nel vector store;
- **VectorStoreManager**: interfaccia che rende disponibile metodi per dialogare con i vector store;
- **VectorStorePineconeManager**: classe che implementa VectorStoreManager, offrendo la possibilità di dialogare con il vector store Pinecone;
- **VectorStoreChromaDBManager**: classe che implementa VectorStoreManager, offrendo la possibilità di dialogare con il vector store Chroma;
- **VectorStoreDocumentOperationResponse**: classe che contiene l'esito di un'operazione effettuata su un vector store riguardo un documento.

5 Progettazione di dettaglio