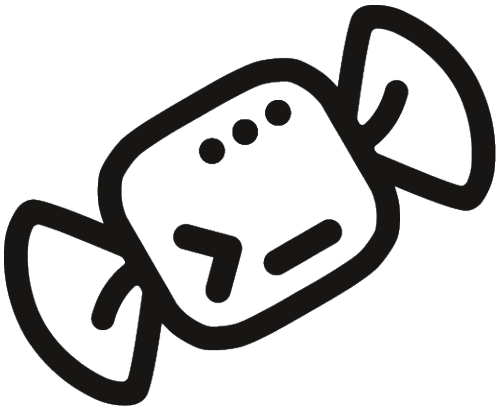

SWEetCode

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
v2.25.0(1)	$2024-03-22$	Campese M.	Bresolin G.	Aggiornamento routing componenti e progettazione SetConfiguration.
v2.22.0(1)	$2024-03-20$	Campese M.	Ciriolo I.	Aggiornamento sezioni di progettazione delle componenti di Configuration.
v2.17.0(1)	$2024-03-15$	Campese M.	Ciriolo I.	Aggiornamento progettazione di dettaglio.
v2.0.0(36)	$2024-03-02$	Michelon R.	Bresolin G.	Aggiunta progettazione di dettaglio ViewDocumentContent.
v2.0.0(35)	$2024-03-02$	Bresolin G.	Campese M.	Aggiunta progettazione di dettaglio UploadDocuments.
v2.0.0(34)	$2024-03-02$	Feltrin E.	Ciriolo I.	Aggiunta progettazione di dettaglio RenameChat.
v2.0.0(33)	$2024-03-02$	Michelon R.	Orlandi G.	Aggiunta progettazione di dettaglio GetDocuments.
v2.0.0(32)	$2024-03-02$	Dugo A.	Bresolin G.	Aggiunta progettazione di dettaglio GetChats.
v2.0.0(31)	$2024-03-02$	Feltrin E.	Michelon R.	Aggiunta progettazione di dettaglio GetChatMessages.
v2.0.0(30)	$2024-03-02$	Ciriolo I.	Bresolin G.	Aggiunta progettazione di dettaglio EnableDocuments.
v2.0.0(29)	$2024-03-02$	Feltrin E.	Dugo A.	Aggiunta progettazione di dettaglio EmbedDocuments.

Continua nella pagina successiva

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
v2.0.0(28)	$2024-03-02$	Campese M.	Feltrin E.	Aggiunta progettazione di dettaglio DeleteDocuments.
v2.0.0(27)	$2024-03-02$	Dugo A.	Orlandi G.	Aggiunta progettazione di dettaglio DeleteChats.
v2.0.0(26)	$2024-03-02$	Bresolin G.	Dugo A.	Aggiunta progettazione di dettaglio ConcealDocuments.
v2.0.0(25)	$2024-03-02$	Michelon R.	Ciriolo I.	Aggiunta progettazione di dettaglio AskChatbot.
v2.0.0(24)	$2024-03-02$	Bresolin G.	Michelon R.	Aggiunta progettazione logica ViewDocumentContent.
v2.0.0(23)	$2024-03-02$	Ciriolo I.	Dugo A.	Aggiunta progettazione logica UploadDocuments.
v2.0.0(22)	$2024-03-02$	Campese M.	Orlandi G.	Aggiunta progettazione logica RenameChat.
v2.0.0(21)	$2024-03-02$	Feltrin E.	Campese M.	Aggiunta progettazione logica GetDocuments.
v2.0.0(20)	$2024-03-02$	Bresolin G.	Michelon R.	Aggiunta progettazione logica GetChats.
v2.0.0(19)	$2024-03-02$	Michelon R.	Bresolin G.	Aggiunta progettazione logica GetChatMessages.
v2.0.0(18)	$2024-03-02$	Orlandi G.	Campese M.	Aggiunta progettazione logica EnableDocuments.
v2.0.0(17)	$2024-03-02$	Dugo A.	Bresolin G.	Aggiunta progettazione logica EmbedDocuments.

Continua nella pagina successiva

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
v2.0.0(16)	$2024-03-02$	Michelon R.	Dugo A.	Aggiunta progettazione logica DeleteDocuments.
v2.0.0(15)	$2024-03-02$	Bresolin G.	Feltrin E.	Aggiunta progettazione logica DeleteChats.
v2.0.0(14)	$2024-03-02$	Dugo A.	Orlandi G.	Aggiunta progettazione logica ConcealDocuments.
v2.0.0(13)	$2024-03-02$	Michelon R.	Campese M.	Aggiunta progettazione logica AskChatbot.
v2.0.0(8)	$2024-02-28$	Feltrin E.	Michelon R.	Aggiunta versioni tecnologie utilizzate.
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 Management 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, a differenza di Django che include molte funzionalità out-of-the-box che potrebbero non essere necessarie;
  - **Flessibilità:** Flask offre una grande flessibilità, permettendo agli sviluppatori di strutturare le loro applicazioni come preferiscono, a differenza di Django che segue un approccio più rigido e strutturato;
  - **Facilità d'uso:** Flask è facile da usare e da imparare, rendendolo ideale per i principianti;
  - **Meno overhead rispetto a Django:** A causa della sua leggerezza e flessibilità, Flask può avere meno overhead rispetto a un framework più pesante come Django;
  - **Maggiore controllo rispetto a Django:** Flask offre agli sviluppatori un maggiore controllo sulle funzionalità delle loro applicazioni, a differenza di Django che fornisce molte funzionalità predefinite che potrebbero non essere necessarie o desiderate.
- **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;
  - **Richiede più configurazione rispetto a Django:** Flask richiede più configurazione rispetto ad altri framework come Django, che hanno più funzionalità integrate;
  - **Supporto della comunità più piccolo rispetto a Django:** Anche se Flask ha una comunità attiva, non è grande come quella di Django. Questo potrebbe significare meno risorse di apprendimento e supporto disponibili.

Versione scelta: Flask 3.0.2.

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;
  - **Librerie ricche rispetto a Java o C++:** Python ha una vasta gamma di librerie e framework, che possono aiutare a semplificare lo sviluppo e a ridurre il tempo di sviluppo, a differenza di altri linguaggi come Java o C++ che potrebbero non avere una gamma così ampia di librerie disponibili;
  - **Sintassi più semplice rispetto a Java o C++:** Python ha una sintassi più semplice e pulita, il che rende il codice più leggibile e mantenibile rispetto a linguaggi come Java o C++.
- **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;
  - **Gestione della memoria:** Python utilizza un garbage collector per la gestione della memoria, che può non essere efficiente come la gestione manuale della memoria in linguaggi come C++;
  - **Non è fortemente tipizzato:** A differenza di linguaggi come Java o C++, Python non è un linguaggio fortemente tipizzato. Questo può portare a errori di runtime che sarebbero stati catturati al momento della compilazione in un linguaggio fortemente tipizzato.

Versione scelta: Python 3.9.

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 rispetto ad Angular:** Next.js è noto per la sua efficienza, il che significa che le applicazioni create con Next.js sono veloci e performanti, a differenza di Angular che può essere più pesante e meno efficiente;
- **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 rispetto ad Angular:** Next.js è facile da usare e da imparare, specialmente per coloro che sono già familiari con React, a differenza di Angular che può avere una curva di apprendimento più ripida;
- **Supporto per TypeScript:** A differenza di molti altri framework, Next.js offre un supporto integrato per TypeScript, il che può migliorare l'affidabilità e la robustezza del codice.

- **Svantaggi:**
  - **Overhead rispetto a React da solo:** 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;
  - **Richiede più risorse rispetto a React da solo:** A causa delle sue funzionalità aggiuntive, Next.js può richiedere più risorse di sistema rispetto a React da solo.

Versione scelta: Next.js 14.1.

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 rispetto a JavaScript:** Typescript offre un controllo dei tipi a tempo di compilazione, il che significa che gli errori possono essere rilevati e corretti prima dell'esecuzione, a differenza di JavaScript che è un linguaggio interpretato e gli errori possono essere rilevati solo a runtime;
  - **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;
  - **Supporto per le annotazioni di tipo:** A differenza di JavaScript, TypeScript supporta le annotazioni di tipo, il che può migliorare la leggibilità del codice e facilitare la manutenzione.
- **Svantaggi:**

- **Curva di apprendimento rispetto a JavaScript:** Typescript può essere più difficile da imparare rispetto a JavaScript a causa delle sue funzionalità aggiuntive;
- **Compilazione:** A differenza di JavaScript, TypeScript deve essere compilato in JavaScript prima di poter essere eseguito, il che può aggiungere un passaggio aggiuntivo nel processo di sviluppo.

Versione scelta: Typescript 5.3.3.

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 rispetto a Vagrant:** Docker consente di eseguire più applicazioni in modo isolato sulla stessa infrastruttura hardware, migliorando l'efficienza e riducendo i costi, a differenza di Vagrant che può richiedere più risorse di sistema;
  - **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;
  - **Supporto per la CI/CD:** Docker può essere facilmente integrato in pipeline di integrazione continua e distribuzione continua (CI/CD), il che può semplificare il processo di sviluppo e distribuzione.
- **Svantaggi:**
  - **Complessità rispetto a Vagrant:** Docker può aggiungere una certa complessità a un progetto a causa della necessità di gestire i container e le loro dipendenze, a differenza di Vagrant che può essere più semplice da configurare e gestire;
  - **Curva di apprendimento:** Docker ha una curva di apprendimento ripida e può richiedere un certo tempo per essere padroneggiato;
  - **Compatibilità:** Non tutti i sistemi operativi supportano Docker nativamente, il che può limitare la sua utilità in alcuni ambienti.

Versione scelta: Docker Desktop 4.28.0.

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 co-

modità 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 rispetto a TensorFlow o PyTorch:** Langchain semplifica l'interazione tra modelli di apprendimento automatico e risorse esterne, fornendo moduli e integrazioni, a differenza di TensorFlow o PyTorch che potrebbero richiedere più codice e sforzo per integrare con risorse esterne;
  - **Flessibilità:** Langchain supporta una varietà di modelli di apprendimento automatico e risorse esterne, rendendolo adatto a una vasta gamma di applicazioni;
  - **Facilità d'uso rispetto a TensorFlow o PyTorch:** Langchain è facile da usare e da imparare, rendendolo ideale per i principianti, a differenza di TensorFlow o PyTorch che possono avere una curva di apprendimento più ripida;
  - **Supporto per una varietà di risorse esterne:** A differenza di molti altri framework, Langchain offre un supporto integrato per una varietà di risorse esterne, il che può semplificare lo sviluppo e l'integrazione.
- **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.

Versione scelta: Langchain 0.1.9.

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 rispetto a Faiss o Annoy:** Pinecone è progettato per effettuare ricerche di similarità su larga scala in modo efficiente, a differenza di Faiss o Annoy che potrebbero non essere ottimizzati per ricerche su larga scala;
  - **Precisione rispetto a Faiss o Annoy:** Pinecone offre un'alta precisione nelle ricerche di similarità, il che lo rende ideale per applicazioni che richiedono un alto grado di precisione, a differenza di Faiss o Annoy che potrebbero non offrire la stessa precisione;
  - **Facilità d'uso:** Pinecone è facile da usare e da imparare, rendendolo ideale per i principianti;
  - **Scalabilità:** A differenza di molti altri database di vettori, Pinecone è progettato per scalare con le esigenze dell'applicazione, il che può semplificare la gestione delle risorse.
- **Svantaggi:**

- **Costo rispetto a Faiss o Annoy:** Pinecone può essere costoso da utilizzare per applicazioni su larga scala, a differenza di Faiss o Annoy che sono open source e gratuiti da utilizzare;
- **Limitazioni:** Pinecone potrebbe non supportare tutte le funzionalità di ricerca di similarità che potrebbero essere necessarie per alcune applicazioni.

Versione scelta: Pinecone Client 3.1.0.

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 con Langchain rispetto a Faiss o Annoy:** ChromaDB è ben integrato con Langchain, il che facilita l'interazione tra i due, a differenza di Faiss o Annoy che potrebbero richiedere più codice e sforzo per integrare con Langchain;
  - **Popolarità rispetto a Faiss o Annoy:** 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, a differenza di Faiss o Annoy che potrebbero non avere una comunità di sviluppatori così grande;
  - **Facilità d'uso:** ChromaDB è facile da usare e da imparare, rendendolo ideale per i principianti;
  - **Supporto per una varietà di tipi di dati:** A differenza di molti altri database di vettori, ChromaDB supporta una varietà di tipi di dati, il che può semplificare la gestione dei dati.
- **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.

Versione scelta: ChromaDB 0.4.24.

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 rispetto a TensorFlow o PyTorch:** OpenAI è molto conosciuto nel campo dell'apprendimento automatico, il che significa che ha una grande comunità di sviluppatori e molte risorse disponibili, a differenza di TensorFlow o PyTorch che potrebbero non avere una comunità di sviluppatori così grande;
- **Integrazione:** OpenAI ha un'integrazione completa con Langchain, il che facilita l'interazione tra i due, a differenza di TensorFlow o PyTorch che potrebbero richiedere più codice e sforzo per integrare con Langchain;
- **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, a differenza di TensorFlow o PyTorch che sono open source e gratuiti da utilizzare;
  - **Limitazioni:** Non tutte le funzionalità sono disponibili in tutti i modelli offerti da OpenAI.

Versione modello LLM scelto: gpt-3.5-turbo-instruct.
Versione modello di embeddings scelto: text-embedding-3-small.

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, a differenza di TensorFlow o PyTorch che potrebbero richiedere la configurazione e l'addestramento di modelli personalizzati;
  - **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 rispetto a TensorFlow o PyTorch:** HuggingFace è facile da usare e da imparare, rendendolo ideale per i principianti, a differenza di TensorFlow o PyTorch che possono avere una curva di apprendimento più ripida.
- **Svantaggi:**
  - **Risorse rispetto a TensorFlow o PyTorch:** L'esecuzione dei modelli in locale può richiedere molte risorse hardware, il che può non essere ideale per

tutte le macchine, a differenza di TensorFlow o PyTorch che possono essere ottimizzati per l'esecuzione su hardware specifico;

- **Complessità:** A causa della vasta gamma di modelli disponibili, può essere difficile scegliere il modello più adatto alle proprie esigenze.

Versione modello LLM scelto: meta-llama/Llama-2-7b-chat-hf.
Versione modello di embeddings scelto: sentence-transformers/all-mpnet-base-v2.

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 rispetto a Google Cloud Storage o Azure Blob Storage:** Amazon S3 può memorizzare qualsiasi quantità di dati e servire qualsiasi livello di traffico richiesto, a differenza di Google Cloud Storage o Azure Blob Storage che potrebbero avere limiti sulla quantità di dati o sul traffico;
  - **Affidabilità:** Amazon S3 offre una durabilità dell'11 9's, il che significa che i dati sono estremamente sicuri, a differenza di Google Cloud Storage o Azure Blob Storage che potrebbero non offrire lo stesso livello di durabilità;
  - **Sicurezza rispetto a Google Cloud Storage o Azure Blob Storage:** Amazon S3 offre potenti funzionalità per proteggere i dati, tra cui controllo degli accessi, crittografia in transito e a riposo, e altro ancora, a differenza di Google Cloud Storage o Azure Blob Storage che potrebbero non offrire le stesse funzionalità di sicurezza.
- **Svantaggi:**
  - **Costo rispetto a Google Cloud Storage o Azure Blob Storage:** Il costo di Amazon S3 può aumentare rapidamente con l'aumentare dell'uso, a differenza di Google Cloud Storage o Azure Blob Storage che potrebbero avere costi più prevedibili;
  - **Complessità:** Amazon S3 ha molte funzionalità e opzioni, il che può renderlo complesso da configurare e gestire, a differenza di Google Cloud Storage o Azure Blob Storage che potrebbero essere più semplici da configurare e gestire.

Versione scelta: Amazon Simple Storage Service (Amazon S3).

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 rispetto a MySQL o SQLite:** Postgres supporta una vasta gamma di tipi di dati nativi, operatori e funzioni, tra cui JSON, XML e array, a differenza di MySQL o SQLite che potrebbero non supportare tutti questi tipi di dati;
- **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, a differenza di MySQL o SQLite che potrebbero non offrire tutte queste funzionalità;
- **Flessibilità rispetto a MySQL o SQLite:** 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, a differenza di MySQL o SQLite che potrebbero non offrire la stessa flessibilità.

- **Svantaggi:**
  - **Complessità rispetto a MySQL o SQLite:** 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 come MySQL o SQLite;
  - **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.

Versione scelta: PostgreSQL 16.2.

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
        /blueprints
        /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

4.2.1 AskChatbot

4.2.1.1 Descrizione

Questa componente ha il compito di ottenere una risposta ad un messaggio da parte del chatbot. È costituita da:

- **Route API**: /askChatbot;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `message` : messaggio da parte dell'utente;
  - `chatId` : id della chat a cui appartiene il messaggio.
- **Implementazione generale**: La componente viene implementata dal blueprint `askChatbot` che esegue i seguenti controlli:
  1. Il parametro `message` non deve essere nullo nè vuoto;
  2. Il parametro `chatId` deve essere valido.

4.2.1.2 Esiti possibili

Tabella 1: Esiti possibili AskChatbot

Codice	Descrizione	Risposta
200	Risposta ottenuta con successo.	-
400	Il parametro `message` è nullo.	Parametri insufficienti.
400	Il parametro `message` è vuoto.	Filtro 'message' non valido.
400	Il parametro `chatId` è nullo, non un numero o un numero negativo.	Filtro 'message' non valido.
500	Risposta del chatbot fallita.	Errore di generazione della risposta.

4.2.1.3 Lista sottocomponenti

- **AskChatbotController**: classe controller che si occupa del richiedere una risposta al chatbot ad un messaggio appartente ad una chat allo use case AskChatbotUseCase;
- **AskChatbotLangchain**: classe che implementa la porta AskChatbotPort, adattando la chiamata di AskChatbotService a classi offerte dal framework Langchain;
- **AskChatbotPort**: interfaccia che rappresenta la porta in uscita per effettuare la richiesta di una risposta al chatbot ad un messaggio appartente ad una chat;
- **AskChatbotService**: classe service che implementa lo use case AskChatbotUseCase;

- **AskChatbotUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per richiedere una risposta al chatbot ad un messaggio appartente ad una chat;
- **ChatHistoryManager**: classe per interagire con le chat come oggetti lanchain.Memory;
- **ChatId**: classe di business che rappresenta l'id di una chat utilizzato nella ricerca;
- **ChatOperationResponse**: classe che contiene l'esito di un'operazione effettuata su una chat;
- **Message**: classe di business che rappresenta un messaggio di una chat;
- **MessageResponse**: classe che rappresenta un messaggio risposta del chatbot;
- **MessageSender**: classe di business che rappresenta il mittente di un messaggio;
- **PersistChatPort**: interfaccia che rappresenta la porta in uscita verso il database per la storicizzazione delle chat;
- **PostgresChat**: classe di persistence che rappresenta una chat;
- **PostgresChatOperationResponse**: classe che contiene l'esito di un'operazione effettuata su Postgres riguardo una chat;
- **PostgresMessage**: classe di persistence che rappresenta un messaggio;
- **PostgresMessageSenderType**: enumeration che rappresenta i valori che può assumere il campo sender di un PostgresMessage;
- **PostgresPersistChat**: classe adapter che implementa la porta PersistChatPort, adattando la chiamata di persistChat a PostgresChatORM;
- **PostgresChatORM**: classe che si occupa di effettuare le operazioni su Postgres configurato, cioè il sistema di storicizzazione delle chat dell'applicazione.

4.2.2 ChangeConfiguration

4.2.2.1 Descrizione

Questa componente ha il compito di cambiare la configurazione del modello LLM del sistema. È costituita da:

- **Route API**: /changeConfiguration;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `LLMModel` : stringa che rappresenta il modello LLM da configurare.
- **Implementazione generale**: La componente viene implementata dal blueprint `changeConfiguration` che esegue i seguenti controlli:
  1. Il parametro `LLMModel` non deve essere nullo nè vuoto.

4.2.2.2 Esiti possibili

Tabella 2: Esiti possibili ChangeConfiguration

Codice	Descrizione	Risposta
200	Configurazione avvenuta con successo.	-
400	Il parametro `LLMModel` è nullo.	Parametri insufficienti.
400	Il parametro `LLMModel` è vuoto.	Modello LLM 'LLMModel' non valido.
500	Configurazione fallita.	Errore nell'aggiornamento del modello LLM.

4.2.2.3 Lista sottocomponenti

- **ChangeConfigurationController**: classe controller che si occupa del cambio del modello LLM;
- **ChangeConfigurationPort**: interfaccia che rappresenta la porta in uscita verso il database, per effettuare il cambio di modello LLM;
- **ChangeConfigurationPostgres**: classe che implementa ChangeConfigurationPort, adattando la chiamata di ChangeConfigurationService a PostgresConfigurationORM;
- **ChangeConfigurationService**: classe service che implementa lo use case ChangeConfigurationUseCase;
- **ChangeConfigurationUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per eseguire il cambio della configurazione del modello LLM;
- **ConfigurationOperationResponse**: classe che contiene l'esito di un'operazione eseguita sulla configurazione del modello LLM;
- **PostgresConfigurationOperationResponse**: classe che contiene l'esito di un'operazione effettuata su Postgres riguardo la configurazione del modello LLM;
- **PostgresConfigurationORM**: classe che si occupa di effettuare le operazioni su Postgres configurato, cioè dove viene storicizzata la configurazione del sistema.

4.2.3 ConcealDocuments

4.2.3.1 Descrizione

Questa componente ha il compito di occultare gli embeddings dei documenti indicati dall'utente. È costituita da:

- **Route API**: /concealDocuments;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `documentIds` : una lista di stringhe che rappresentano gli id dei documenti da occultare.
- **Implementazione generale**: La componente viene implementata dal blueprint `concealDocuments` che esegue i seguenti controlli:

1. Il parametro `documentIds` non deve essere nullo nè vuoto.

4.2.3.2 Esiti possibili

Tabella 3: Esiti possibili ConcealDocuments

Codice	Descrizione	Risposta
200	Occultamento avvenuto con successo.	-
400	Il parametro `documentIds` è nullo.	Parametri insufficienti.
400	La lunghezza del parametro `documentIds` è 0.	Nessun id di documento specificato.
400	Un id di `documentIds` è vuoto.	Id di documento 'id' non valido.
500	Occultamento fallito.	Errore nell'occultamento dei documenti.

4.2.3.3 Lista sottocomponenti

- **ConcealDocumentsController**: classe controller che si occupa del passaggio di una lista di DocumentId allo use case ConcealDocumentsUseCase a partire da una lista di stringhe che rappresentano l'id dei documenti da occultare;
- **ConcealDocumentsPort**: interfaccia che rappresenta la porta per effettuare l'occultamento dei documenti nel vector store;
- **ConcealDocumentsService**: classe service che implementa lo use case ConcealDocumentsUseCase;
- **ConcealDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare l'occultamento dei documenti;
- **ConcealDocumentsVectorStore**: classe adapter che implementa la porta ConcealDocumentsPort, adattando la chiamata di ConcealDocumentsService a VectorStoreManager;
- **DocumentId**: classe di business che rappresenta l'id di un documento;
- **DocumentOperationResponse**: classe che contiene l'esito di un'operazione effettuata su un documento;
- **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;
- **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.

4.2.4 DeleteChats

4.2.4.1 Descrizione

Questa componente ha il compito di eliminare una lista di chat, aggiornando di conseguenza il database utilizzato per la storicizzazione delle chat. È costituita da:

- **Route API**: /deleteChats;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `chatIds` : lista di Id utilizzati per identificare univocamente le chat.
- **Implementazione generale**: La componente viene implementata dal blueprint `deleteChats` che esegue i seguenti controlli:
  1. Il parametro `chatIds` non deve essere nullo nè vuoto.

4.2.4.2 Esiti possibili

Tabella 4: Esiti possibili DeleteChats

Codice	Descrizione	Risposta
200	Eliminazione avvenuta con successo.	-
400	Il parametro `chatIds` è nullo.	Parametri insufficienti.
400	La lunghezza del parametro `chatIds` è 0.	Nessun chat id specificato.
400	Un id di `chatIds` è vuoto.	Chat id 'Id' non valido.
500	Eliminazione fallita.	Errore nell' eliminazione delle chat.

4.2.4.3 Lista sottocomponenti

- **ChatId**: Vedi (§4.2.1.3)
- **ChatOperationResponse**: vedi (§4.2.1.3) ;
- **DeleteChatsController**: classe controller che si occupa del passaggio allo use case CreateChatUseCase di interi rappresentanti gli id di una lista di chat per eseguire la loro eliminazione;
- **DeleteChatsPort**: interfaccia che rappresenta la porta in uscita per effettuare l'eliminazione di una lista di chat verso il database per la storicizzazione delle chat;
- **DeleteChatsPostgres**: classe che implementa la porta DeleteChatPort, adattando la chiamata di DeleteChatsService a PostgresChatORM;
- **DeleteChatsService**: classe service che implementa lo use case DeleteChatsUseCase;
- **DeleteChatsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare l'eliminazione di una lista di chat;

- **PostgresChatOperationResponse**: vedi (§4.2.1.3) ;
- **PostgresChatORM**: vedi (§4.2.1.3) .

4.2.5 ConfigurationManager

4.2.5.1 Descrizione

Questa componente ha il compito di gestire la configurazione del sistema.

4.2.5.2 Lista sottocomponenti

- **ConfigurationManager**: classe che espone metodi per ottenere le porte in uscita delle componenti del sistema;
- **PostgresConfiguration**: classe di persistence che rappresenta la configurazione del sistema di userId, vector store, document store, LLM ed embedding model su Postgres;
- **PostgresConfigurationORM**: vedi (§4.2.2.3) ;
- **PostgresDocumentStoreConfiguration**: classe di persistence che rappresenta la configurazione di un document store;
- **PostgresDocumentStoreType**: enumeration che rappresenta i valori che può assumere il campo `name` in PostgresDocumentStoreConfiguration;
- **PostgresEmbeddingModelConfiguration**: classe di persistence che rappresenta la configurazione di un embedding model;
- **PostgresEmbeddingModelType**: enumeration che rappresenta i valori che può assumere il campo `name` in PostgresEmbeddingModelConfiguration;
- **PostgresLLMModelConfiguration**: classe di persistence che rappresenta la configurazione di un modello LLM;
- **PostgresLLMModelType**: enumeration che rappresenta i valori che può assumere il campo `name` in PostgresLLMModelConfiguration;
- **PostgresVectorStoreConfiguration**: classe di persistence che rappresenta la configurazione di un vector store su Postgres;
- **PostgresVectorStoreType**: enumeration che rappresenta i valori che può assumere il campo `name` in PostgresVectorStoreConfiguration.

4.2.6 DeleteDocuments

4.2.6.1 Descrizione

Questa componente ha il compito di eliminare una lista di documenti e i loro rispettivi embeddings. È costituita da:

- **Route API**: /deleteDocuments;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `documentIds` : una lista di stringhe che rappresentano gli id dei documenti da eliminare.

- **Implementazione generale**: La componente viene implementata dal blueprint `deleteDocuments` che esegue i seguenti controlli:
  1. Il parametro `documentIds` non deve essere nullo nè vuoto.

4.2.6.2 Esiti possibili

Tabella 5: Esiti possibili DeleteDocuments

Codice	Descrizione	Risposta
200	Eliminazione avvenuta con successo.	-
400	Il parametro `documentIds` è nullo.	Parametri insufficienti.
400	La lunghezza del parametro `documentIds` è 0.	Nessun id di documento specificato.
400	Un id di `documentIds` è vuoto.	Id di documento 'id' non valido.
500	Eliminazione fallita.	Errore nell'eliminazione dei documenti.

4.2.6.3 Lista sottocomponenti

- **AWSDocumentOperationResponse**: classe che contiene l'esito di un'operazione effettuata su un AWS S3 riguardo un documento;
- **AWSS3Manager**: classe che si occupa di effettuare le operazioni sul bucket di Amazon S3 configurato, cioè il sistema di archiviazione documenti dell'applicazione;
- **DeleteDocuments**: classe che si occupa di inoltrare la richiesta di eliminare una lista di documenti nella porta esterna DeleteDocumentsPort diretta verso il sistema di archiviazione;
- **DeleteDocumentsAWSS3**: classe adapter che implementa la porta DeleteDocumentsPort, adattando la chiamata di DeleteDocuments a AWSS3Manager;
- **DeleteDocumentsController**: classe controller che si occupa del passaggio di una lista di DocumentId allo use case DeleteDocumentsUseCase a partire da una lista di stringhe che rappresentano gli id dei documenti da eliminare;
- **DeleteDocumentsEmbeddings**: classe che si occupa di eliminare gli embeddings di una lista di documenti nella porta esterna DeleteEmbeddingsPort diretta verso il vector store;
- **DeleteDocumentsPort**: interfaccia che rappresenta la porta in uscita per eliminare una lista di documenti dal sistema di archiviazione;
- **DeleteDocumentsService**: classe service che implementa lo use case DeleteDocumentsUseCase;
- **DeleteDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per eliminare i documenti;

- **DeleteEmbeddingsPort**: interfaccia che rappresenta la porta in uscita per eliminare gli embeddings di una lista di documenti dal vector store;
- **DeleteEmbeddingsVectorStore**: classe adapter che implementa la porta DeleteEmbeddingsPort, adattando la chiamata di DeleteDocumentsEmbeddings a VectorStoreManager;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.7 EmbedDocuments

4.2.7.1 Descrizione

Questa componente ha il compito di generare e memorizzare gli embeddings dei documenti indicati dall'utente. È costituita da:

- **Route API**: /embedDocuments;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `documentIds` : una lista di stringhe che rappresentano gli id dei documenti di cui generare gli embeddings.
- **Implementazione generale**: La componente viene implementata dal blueprint `embedDocuments` che esegue i seguenti controlli:
  1. Il parametro `documentIds` non deve essere nullo nè vuoto.

4.2.7.2 Esiti possibili

Tabella 6: Esiti possibili EmbedDocuments

Codice	Descrizione	Risposta
200	Generazione embeddings avvenuta con successo.	-
400	Il parametro `documentIds` è nullo.	Parametri insufficienti.
400	La lunghezza del parametro `documentIds` è 0.	Nessun id di documento specificato.
400	Un id di `documentIds` è vuoto.	Id di documento 'id' non valido.
500	Generazione embeddings fallita.	Errore nella generazione degli embeddings.

4.2.7.3 Lista sottocomponenti

- **AWSDocument**: classe che rappresenta i documenti gestibili da AWSS3Manager;
- **AWSS3Manager**: vedi (§4.2.6.3) ;
- **Chunkerizer**: classe che crea chunks a partire da testo;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentOperationResponse**: vedi (§4.2.3.3) ;
- **DocumentStatus**: classe di business che rappresenta lo status di un documento;
- **DOCXTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti docx;
- **EmbeddingsCreator**: classe che dialoga con il modello di embeddings per creare gli embeddings a partire dai chunk forniti in input;
- **EmbeddingsUploader**: classe che si occupa di effettuare la chiamata dell'upload degli embeddings nella porta esterna diretta verso il vector store;
- **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;
- **EmbeddingsUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload degli embeddings verso i vector store;
- **EmbeddingsUploaderVectorStore**: classe che offre un metodo per eseguire l'upload degli embeddings nel vector store;
- **EmbedDocumentsController**: classe controller che si occupa del passaggio di una lista di DocumentId allo use case EmbedDocumentsUseCase a partire da una lista di stringhe che rappresentano l'id dei documenti di cui generare gli embeddings;
- **EmbedDocumentsService**: classe service che implementa lo use case EmbedDocumentsUseCase;
- **EmbedDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per generare gli embeddings dei documenti;
- **GetDocumentsContent**: classe che si occupa di inoltrare la richiesta di recuperare i documenti nella porta esterna diretta verso il sistema di archiviazione;
- **GetDocumentsContentAWSS3**: classe adapter che implementa la porta GetDocumentsContentPort, adattando la chiamata di GetDocumentsContent a AWSS3Manager;
- **GetDocumentsContentPort**: interfaccia che rappresenta la porta in uscita per recuperare il contenuto dei documenti dal sistema di archiviazione;
- **GetDocumentsStatus**: classe che si occupa di inoltrare la richiesta di recuperare gli status di una lista di documenti nella porta esterna GetDocumentsStatusPort diretta verso il vector store;

- **GetDocumentsStatusPort**: interfaccia che rappresenta la porta in uscita per recuperare gli status di una lista di documenti dal vector store;
- **GetDocumentsStatusVectorStore**: classe adapter che implementa la porta GetDocumentsStatusPort, adattando la chiamata di GetDocumentsStatus a VectorStoreManager;
- **HuggingFaceEmbeddingModel**: classe che implementa LangchainEmbeddingModel offrendo la possibilità di creare embeddings attraverso un embedding model di HuggingFace;
- **LangchainDocument**: classe che rappresenta un documento e i suoi embeddings;
- **LangchainEmbeddingModel**: classe astratta che permette di interagire con un oggetto langhchain.Embeddings;
- **OpenAIEmbeddingModel**: classe che implementa la classe astratta LangchainEmbeddingModel per interagire con un modello di generazione di embeddings di OpenAI;
- **PDFTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti PDF;
- **Status**: enumeration che rappresenta i valori che può assumere lo status di un documento;
- **TextExtractor**: interfaccia che espone il metodo astratto di estrazione di testo da un documento;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentStatusResponse**: classe che rappresenta la risposta generata da un vector store, contenente lo status del documento richiesto;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.8 EnableDocuments

4.2.8.1 Descrizione

Questa componente ha il compito di riabilitare gli embeddings dei documenti indicati dall'utente. È costituita da:

- **Route API**: /enableDocuments;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `documentIds` : una lista di stringhe che rappresentano gli id dei documenti da riabilitare.
- **Implementazione generale**: La componente viene implementata dal blueprint `enableDocuments` che esegue i seguenti controlli:

1. Il parametro `documentIds` non deve essere nullo nè vuoto.

4.2.8.2 Esiti possibili

Tabella 7: Esiti possibili EnableDocuments

Codice	Descrizione	Risposta
200	Riabilitazione avvenuta con successo.	-
400	Il parametro `documentIds` è nullo.	Parametri insufficienti.
400	La lunghezza del parametro `documentIds` è 0.	Nessun id di documento specificato.
400	Un id di `documentIds` è vuoto.	Id di documento 'id' non valido.
500	Riabilitazione fallita.	Errore nella riabilitazione dei documenti.

4.2.8.3 Lista sottocomponenti

- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentOperationResponse**: vedi (§4.2.3.3) ;
- **EnableDocumentsController**: classe controller che si occupa del passaggio di una lista di DocumentId allo use case EnableDocumentsUseCase a partire da una lista di stringhe che rappresentano l'id dei documenti da riabilitare;
- **EnableDocumentsPort**: interfaccia che rappresenta la porta in uscita per effettuare la riabilitazione dei documenti nel vector store;
- **EnableDocumentsService**: classe service che implementa lo use case EnableDocumentsUseCase;
- **EnableDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la riabilitazione dei documenti;
- **EnableDocumentsVectorStore**: classe adapter che implementa la porta EnableDocumentsPort, adattando la chiamata di EnableDocumentsService a VectorStoreManager;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.9 GetChatMessages

4.2.9.1 Descrizione

Questa componente ha il compito di ottenere una chat completa, ovvero che comprende sia le sue informazioni che i suoi messaggi. È costituita da:

- **Route API**: /getChatMessages;
- **Metodo**: GET;
- **Lista parametri HTTP**:
  - `chatId` : id della chat da recuperare.
- **Implementazione generale**: La componente viene implementata dal blueprint `getChatMessages` che esegue i seguenti controlli:
  1. Il parametro `chatId` non deve essere nullo nè minore di 0.

4.2.9.2 Esiti possibili

Tabella 8: Esiti possibili GetChatMessages

Codice	Descrizione	Risposta
200	Chat recuperata con succcesso.	-
400	Il parametro `chatId` è nullo.	Parametri insufficienti.
400	Il parametro `chatId` è minore di 0.	Chat id 'chatId' non valido.
500	Recupero della Chat fallito.	Errore nel recupero dei messaggi.

4.2.9.3 Lista sottocomponenti

- **Chat**: classe di business che rappresenta una chat completa;
- **ChatId**: vedi (§4.2.1.3) ;
- **GetChatMessagesController**: classe controller che si occupa del richiedere una chat allo use case GetChatMessagesUseCase;
- **GetChatMessagesPort**: interfaccia che rappresenta la porta in uscita per effettuare la richiesta di una chat verso il database per la storicizzazione delle chat;
- **GetChatMessagesPostgres**: classe che implementa la porta GetChatMessagesPort, adattando la chiamata di GetChatMessagesService a PostgresChatORM;
- **GetChatMessagesService**: classe service che implementa lo use case GetChatMessagesUseCase;
- **GetChatMessagesUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per richiedere una chat;
- **Message**: vedi (§4.2.1.3) ;
- **MessageSender**: vedi (§4.2.1.3) ;
- **PostgresChat**: vedi (§4.2.1.3) ;
- **PostgresChatORM**: vedi (§4.2.1.3) ;
- **PostgresMessage**: vedi (§4.2.1.3) ;
- **PostgresMessageSenderType**: vedi (§4.2.1.3) .

4.2.10 GetChats

4.2.10.1 Descrizione

Questa componente ha il compito di ottenere una lista di preview di chat, eventualmente filtrando la ricerca. È costituita da:

- **Route API**: /getChats;
- **Metodo**: GET;
- **Lista parametri HTTP**:
  - `filter` : filtro da applicare per la ricerca tra le chat.
- **Implementazione generale**: La componente viene implementata dal blueprint `getChats` che esegue i seguenti controlli:
  1. Il parametro `filter` non deve essere nullo.

4.2.10.2 Esiti possibili

Tabella 9: Esiti possibili GetChats

Codice	Descrizione	Risposta
200	Ricerca avvenuta con successo.	-
400	Il parametro `filter` è nullo.	Parametri insufficienti
404	Non sono state trovate chat	-

4.2.10.3 Lista sottocomponenti

- **ChatFilter**: classe di business che rappresenta il filtro utilizzato nella ricerca;
- **ChatId**: vedi (§4.2.1.3) ;
- **ChatPreview**: classe di business che rappresenta la preview di una chat;
- **GetChatsController**: classe controller che si occupa del richiedere le preview delle chat allo use case GetChatsUseCase, con eventuale passaggio di un filtro per eseguire una ricerca;
- **GetChatsPort**: interfaccia che rappresenta la porta in uscita per effettuare la richiesta delle chat eventualmente filtrate verso il database per la storicizzazione delle chat;
- **GetChatsPostgres**: classe che implementa la porta GetChatsPort, adattando la chiamata di GetChatsService a PostgresChatORM;
- **GetChatsService**: classe service che implementa lo use case GetChatsUseCase;
- **GetChatsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per richiedere le chat, eventualmente filtrate;
- **Message**: vedi (§4.2.1.3) ;

- **MessageSender**: vedi (§4.2.1.3) ;
- **PostgresChatORM**: vedi (§4.2.1.3) ;
- **PostgresChatPreview**: classe di persistence che rappresenta la preview di una chat;
- **PostgresMessage**: vedi (§4.2.1.3) ;
- **PostgresMessageSenderType**: vedi (§4.2.1.3) .

4.2.11 GetConfiguration

4.2.11.1 Descrizione

Questa componente ha il compito di ricavare la configurazione del sistema. È costituita da:

- **Route API**: /getConfiguration;
- **Metodo**: GET;
- **Lista parametri HTTP**: -
- **Implementazione generale**: La componente viene implementata dal blueprint `getConfiguration` che esegue i seguenti controlli:
  1. L'oggetto configurazione non deve essere nullo;
  2. Nessun campo dell'oggetto configurazione deve essere nullo.

4.2.11.2 Esiti possibili

Tabella 10: Esiti possibili GetConfiguration

Codice	Descrizione	Risposta
200	Configurazione ricavata con successo.	-
404	Almeno un campo della configurazione è nullo.	Configurazione inesistente.
400	Recupero configurazione fallito.	Errore nel recupero della configurazione.

4.2.11.3 Lista sottocomponenti

- **Configuration**: classe che contiene la configurazione e le informazioni riguardo a vector store, document store, LLM ed embedding model;
- **DocumentStoreConfiguration**: classe che rappresenta la configurazione di un document store;
- **DocumentStoreType**: enumeration che rappresenta i valori che può assumere il campo `name` in DocumentStoreConfiguration;

- **EmbeddingModelConfiguration**: classe che rappresenta la configurazione di un modello di embedding;
- **EmbeddingModelType**: enumeration che rappresenta i valori che può assumere il campo `name` in EmbeddingModelConfiguration;
- **GetConfigurationController**: classe che si occupa di ritornare la configurazione di vector store, document store, embedding model e LLM;
- **GetConfigurationPort**: interfaccia che rappresenta la porta in uscita per ottenere la configurazione del sistema;
- **GetConfigurationPostgres**: classe che implementa la porta GetConfigurationPort, adattando la chiamata di GetConfigurationService a PostgresConfigurationORM;
- **GetConfigurationService**: classe service che implementa lo use case GetConfigurationUseCase;
- **GetConfigurationUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare il recupero della configurazione;
- **LLMModelConfiguration**: classe che rappresenta la configurazione di un modello LLM;
- **LLMModelType**: enumeration che rappresenta il tipo di valori del campo `name` di LLMModelConfiguration;
- **PostgresConfiguration**: vedi (§4.2.5.2) ;
- **PostgresConfigurationORM**: vedi (§4.2.2.3) ;
- **PostgresDocumentStoreConfiguration**: vedi (§4.2.5.2) ;
- **PostgresDocumentStoreType**: vedi (§4.2.5.2) ;
- **PostgresEmbeddingModelConfiguration**: vedi (§4.2.5.2) ;
- **PostgresEmbeddingModelType**: vedi (§4.2.5.2) ;
- **PostgresLLMModelConfiguration**: vedi (§4.2.5.2) ;
- **PostgresLLMModelType**:vedi (§4.2.5.2) ;
- **PostgresVectorStoreConfiguration**: vedi (§4.2.5.2) ;
- **PostgresVectorStoreType**: vedi (§4.2.5.2) ;
- **VectorStoreConfiguration**: classe che rappresenta la configurazione di un vector store;
- **VectorStoreType**: enumeration che rappresenta i valori che può assumere il campo `name` in VectorStoreConfiguration.

4.2.12 GetConfigurationOptions

4.2.12.1 Descrizione

Questa componente ha il compito di ottenere le possibili opzioni delle configurazioni di modello di embedding, LLM, vector store e document store. È costituita da:

- **Route API**: /getConfigurationOptions;

- **Metodo**: GET;
- **Lista parametri HTTP**: -
- **Implementazione generale**: La componente viene implementata dal blueprint `getConfigurationOptions` che esegue i seguenti controlli:
  1. L'oggetto opzioni di configurazione non deve essere nullo.

4.2.12.2 Esiti possibili

Tabella 11: Esiti possibili GetConfigurationOptions

Codice	Descrizione	Risposta
200	Opzioni di configurazione ricavate con successo.	-
404	Opzioni di configurazione non esistenti.	-

4.2.12.3 Lista sottocomponenti

- **ConfigurationOptions**: classe che contiene liste delle possibili configurazioni per vector store, document store, embedding model e LLM;
- **DocumentStoreConfiguration**: vedi (§4.2.11.3) ;
- **DocumentStoreType**: vedi (§4.2.11.3) ;
- **EmbeddingModelConfiguration**: vedi (§4.2.11.3) ;
- **EmbeddingModelType**: vedi (§4.2.11.3) ;
- **GetConfigurationOptions**: classe che si occupa di ottenere le possibili configurazioni di vector store, document store, embedding model e LLM;
- **GetConfigurationOptionsPort**: interfaccia che rappresenta la porta in uscita per ottenere le possibili configurazione di vector store, document store, embedding model e LLM;
- **GetConfigurationOptionsPostgres**: classe che implementa la porta GetConfigurationOptionsPort, adattando la chiamata di GetConfigurationOptionsService a PostgresConfigurationORM;
- **GetConfigurationOptionsService**: classe service che implementa lo use case GetConfigurationOptionsUseCase;
- **GetConfigurationOptionsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare il recupero dellopzioni di configurazione per vector store, document store, embedding model e LLM;
- **LLMModelConfiguration**: vedi (§4.2.11.3) ;
- **LLMModelType**: vedi (§4.2.11.3) ;
- **PostgresConfigurationORM**: vedi (§4.2.2.3) ;

- **PostgresDocumentStoreConfiguration**: vedi (§4.2.5.2) ;
- **PostgresDocumentStoreType**: vedi (§4.2.5.2) ;
- **PostgresEmbeddingModelConfiguration**: vedi (§4.2.5.2) ;
- **PostgresEmbeddingModelType**: vedi (§4.2.5.2) ;
- **PostgresLLMModelConfiguration**: vedi (§4.2.5.2) ;
- **PostgresLLMModelType**: vedi (§4.2.5.2) ;
- **PostgresVectorStoreConfiguration**: vedi (§4.2.5.2) ;
- **PostgresVectorStoreType**: vedi (§4.2.5.2) ;
- **VectorStoreConfiguration**: vedi (§4.2.11.3) ;
- **VectorStoreType**: vedi (§4.2.11.3) .

4.2.13 GetDocuments

4.2.13.1 Descrizione

Questa componente ha il compito di ricavare la lista di tutti i documenti presenti nel sistema e di rispondere a ricerche basate sull'id dei documenti. È costituita da:

- **Route API**: /getDocuments;
- **Metodo**: GET;
- **Lista parametri HTTP**:
  - `filter` : stringa che rappresenta il filtro opzionale da applicare alla ricerca dei documenti.
- **Implementazione generale**: La componente viene implementata dal blueprint `getDocuments` che esegue i seguenti controlli:
  1. Il parametro `filter` non deve essere nullo.

4.2.13.2 Esiti possibili

Tabella 12: Esiti possibili GetDocuments

Codice	Descrizione	Risposta
200	Ricerca avvenuta con successo.	-
400	Il parametro `filter` è nullo.	Parametri insufficienti.
404	Ricerca fallita.	Errore nella ricerca dei documenti.

4.2.13.3 Lista sottocomponenti

- **AWSDocumentMetadata**: classe che rappresenta i metadati di un documento ricavato da AWS;
- **AWSS3Manager**: vedi (§4.2.6.3) ;

- **DocumentFilter**: classe di business che rappresenta il filtro applicabile alla ricerca di documenti;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentMetadata**: classe di business che rappresenta i metadati di un documento;
- **DocumentStatus**: vedi (§4.2.7.3) ;
- **DocumentType**: enumeration che rappresenta i valori che può assumere il tipo di un documento;
- **ElaborationException**: classe che rappresenta il messaggio di eccezione;
- **GetDocumentsController**: classe controller che si occupa del passaggio di un oggetto DocumentFilter allo use case GetDocumentsUseCase a partire da una stringa che rappresenta il filtro della ricerca;
- **GetDocumentsFacadeService**: classe service che implementa lo use case GetDocumentsUseCase;
- **GetDocumentsListAWSS3**: classe adapter che implementa la porta GetDocumentsMetadataPort, adattando la chiamata di GetDocumentsMetadata a AWSS3Manager;
- **GetDocumentsMetadata**: classe che si occupa di ricavare i metadati di una lista di documenti nella porta esterna GetDocumentsMetadataPort diretta verso il sistema di archiviazione documenti a partire da un filtro di ricerca;
- **GetDocumentsMetadataPort**: interfaccia che rappresenta la porta in uscita per ricavare i metadati di una lista di documenti dal sistema di archiviazione a partire da un filtro di ricerca;
- **GetDocumentsStatus**: vedi (§4.2.7.3) ;
- **GetDocumentsStatusPort**: vedi (§4.2.7.3) ;
- **GetDocumentsStatusVectorStore**: (§4.2.7.3) ;
- **GetDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per ricercare dei documenti;
- **LightDocument**: classe di business che fornisce una rappresentazione leggera dei documenti, esculdendo il contenuto;
- **Status**: vedi (§4.2.7.3) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentStatusResponse**: (§4.2.7.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.14 RenameChat

4.2.14.1 Descrizione

Questa componente ha il compito di rinominare una chat, aggiornando di conseguenza il database utilizzato per la storicizzazione delle chat. È costituita da:

- **Route API**: /renameChat;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `chatId` : l'Id utilizzato per identificare univocamente una chat;
  - `title` : nuovo titolo da assegnare alla chat.
- **Implementazione generale**: La componente viene implementata dal blueprint `renameChat` che esegue i seguenti controlli:
  1. Il parametro `chatId` non deve essere nullo;
  2. Il parametro `chatId` non deve essere vuoto, diverso da un numero o un numero negativo;
  3. Il parametro `title` non può essere vuoto.

4.2.14.2 Esiti possibili

Tabella 13: Esiti possibili RenameChat

Codice	Descrizione	Risposta
200	Rinomina avventua con successo.	-
400	Il parametro `chatId` o il parametro `title` sono nulli.	Parametri insufficienti.
400	Il parametro `chatId` non è valido.	Chat id 'chatId' non valido.
400	Il parametro `title` è vuoto.	Il titolo della chat non può essere vuoto.
500	Rinomina fallita.	Errore nella rinomina della chat.

4.2.14.3 Lista sottocomponenti

- **ChatId**: vedi (§4.2.1.3) ;
- **ChatOperationResponse**: vedi (§4.2.1.3) ;
- **RenameChatController**: classe controller che si occupa di effettuare la rinomina di una chat;
- **RenameChatPort**: interfaccia che rappresenta la porta in uscita per effettuare la rinomina di una chat verso il database per la storicizzazione delle chat;
- **RenameChatPostgres**: classe che implementa la porta RenameChatPort, adattando la chiamata di RenameChatService a PostgresChatORM;

- **RenameChatService**: classe service che implementa lo use case RenameChatUseCase;
- **RenameChatUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la rinomina di una chat;
- **PostgresChatOperationResponse**: vedi (§4.2.1.3) ;
- **PostgresChatORM**: vedi (§4.2.1.3) .

4.2.15 SetConfiguration

4.2.15.1 Descrizione

Questa componente ha il compito di impostare le configurazioni di vector store, documents store, modelli LLM e di embedding al primo avvio del sistema. È costituita da:

- **Route API**: /setConfiguration;
- **Metodo**: POST;
- **Lista parametri HTTP**:
  - `embeddingModel` : modello di embedding scelto;
  - `documentStore` : metodo di storicizzazione dei documenti scelto;
  - `LLMModel` : modello LLM scelto;
  - `vectorStore` : vector store scelto.
- **Implementazione generale**: La componente viene implementata dal blueprint `setConfiguration` che esegue i seguenti controlli:
  1. Il parametro `embeddingModel` non deve essere vuoto o nullo;
  2. Il parametro `documentStore` non deve essere vuoto o nullo;
  3. Il parametro `LLMModel` non deve essere vuoto o nullo;
  4. Il parametro `vectorStore` non deve essere vuoto o nullo.

4.2.15.2 Esiti possibili

Tabella 14: Esiti possibili SetConfiguration

Codice	Descrizione	Risposta
200	Configurazione impostata con successo.	-
400	Uno o più dei parametri (`embeddingModel`, `documentStore`, `LLMModel`, `vectorStore`) è nullo.	Parametri insufficienti.
400	Il parametro `LLMModel` è vuoto.	Modello LLM 'LLMModel' non valido.

Continua nella pagina successiva

Tabella 14: Esiti possibili SetConfiguration (cont)

Codice	Descrizione	Risposta
400	Il parametro `embeddingModel` è vuoto.	Modello di embedding 'embeddingModel' non valido.
400	Il parametro `documentStore` è vuoto.	Document Store 'documentStore' non valido.
400	Il parametro `vectorStore` è vuoto.	Vector Store 'vectorStore' non valido.
500	Configurazione fallita.	Errore nella inizializzazione della configurazione.

4.2.15.3 Lista sottocomponenti

- **ConfigurationOperationResponse**: vedi (§4.2.2.3) ;
- **PostgresConfigurationOperationResponse**: vedi (§4.2.2.3) ;
- **PostgresConfigurationORM**: vedi (§4.2.2.3) ;
- **SetConfigurationController**: classe controller che si occupa del passaggio allo use case SetConfigurationUseCase di stringhe che rappresentano rispettivamente LLModel, document store, vector store ed embedding model, per inizializzare la configurazione del sistema;
- **SetConfigurationPort**: interfaccia che rappresenta la porta in uscita per effettuare la inizializzazione della configurazione di sistema;
- **SetConfigurationPostgres**: classe che implementa la porta SetConfigurationPort, adattando la chiamata di SetConfigurationService a PostgresConfigurationORM;
- **SetConfigurationService**:classe service che implementa lo use case SetConfigurationUseCase;
- **SetConfigurationUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la inizializzazione della configurazione di sistema.

4.2.16 UploadDocuments

4.2.16.1 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.

- **Implementazione generale**: La componente viene implementata dal blueprint `uploadDocuments` che esegue i seguenti controlli:
  1. Ogni documento deve avere un titolo;
  2. Ogni documento deve essere di tipo *PDF* o *DOC*;
  3. Devono esserci dei documenti nell'area di staging.

4.2.16.2 Esiti possibili

Tabella 15: Esiti possibili UploadDocuments

Codice	Descrizione	Risposta
200	Upload avvenuto con successo.	-
400	Non ci sono documenti di cui fare l'upload.	Parametri insufficienti.
422	Un documento ha titolo vuoto.	L'upload di documenti senza titolo non è supportato.
422	Formato non accettato.	Documento filename non supportato.
500	Upload fallito.	Errore nell'upload dei documenti.

4.2.16.3 Lista sottocomponenti

- **AWSDocument**: vedi (§4.2.7.3) ;
- **AWSDocumentOperationResponse**: vedi (§4.2.6.3) ;
- **AWSS3Manager**: vedi (§4.2.6.3) ;
- **Chunkerizer**: vedi (§4.2.7.3) ;
- **Document**: classe di business che rappresenta i documenti completi;
- **DocumentContent**: classe di business che rappresenta il contenuto di un documento;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentMetadata**: vedi (§4.2.13.3) ;
- **DocumentOperationResponse**: vedi (§4.2.3.3) ;
- **DocumentStatus**: vedi (§4.2.7.3) ;
- **DocumentType**: vedi (§4.2.13.3) ;
- **DocumentUploader**: classe che si occupa di effettuare la chiamata dell'upload dei documenti nella porta esterna diretta verso il sistema di archiviazione documenti;
- **DocumentUploaderAWSS3**: classe adapter che implementa la porta DocumentUploaderPort, adattando la chiamata di DocumentUploader a AWSS3Manager;

- **DocumentUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload dei documenti verso il sistema di archiviazione documenti;
- **DOCXTextExtractor**: vedi (§4.2.7.3) ;
- **EmbeddingsCreator**: vedi (§4.2.7.3) ;
- **EmbeddingsUploader**: vedi (§4.2.7.3) ;
- **EmbeddingsUploaderFacadeLangchain**: vedi (§4.2.7.3) ;
- **EmbeddingsUploaderPort**: vedi (§4.2.7.3) ;
- **EmbeddingsUploaderVectorStore**: vedi (§4.2.7.3) ;
- **HuggingFaceEmbeddingModel**: vedi (§4.2.7.3) ;
- **LangchainDocument**: vedi (§4.2.7.3) ;
- **LangchainEmbeddingModel**: vedi (§4.2.7.3) ;
- **NewDocument**: classe di presentation che contiene le informazioni relative ai documenti appena inseriti dall'utente, presenti nella richiesta HTTP;
- **OpenAIEmbeddingModel**: vedi (§4.2.7.3) ;
- **PDFTextExtractor**: vedi (§4.2.7.3) ;
- **PlainDocument**: classe di business che rappresenta i documenti, compresi i metadati e il contenuto;
- **Status**: vedi (§4.2.7.3) ;
- **TextExtractor**: vedi (§4.2.7.3) ;
- **UploadDocumentsController**: classe controller che si occupa del passaggio di Documents allo use case UploadDocumentsUseCase a partire da NewDocuments;
- **UploadDocumentsService**: classe service che implementa lo use case UploadDocumentsUseCase;
- **UploadDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare l'upload dei documenti;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.17 ViewDocumentContent

4.2.17.1 Descrizione

Questa componente ha il compito di recuperare tutte le informazioni di un documento, compreso il suo contenuto. È costituita da:

- **Route API**: /viewDocumentContent;
- **Metodo**: GET;

- **Lista parametri HTTP**:
  - documentId: una stringa che rappresenta l'id del documento da recuperare.
- **Implementazione generale**: La componente viene implementata dal blueprint `getDocumentContent` che esegue i seguenti controlli:
  1. Il parametro documentId non deve essere vuoto o nullo.

4.2.17.2 Esiti possibili

Tabella 16: Esiti possibili ViewDocumentContent

Codice	Descrizione	Risposta
200	Documento recuperato con successo.	-
400	Il parametro documentId è nullo.	Parametri insufficienti.
400	Il parametro documentId è vuoto.	Id di documento 'documentId' non valido.
404	Recupero documento fallito.	Errore nel recupero del documento.

4.2.17.3 Lista sottocomponenti

- **AWSDocument**: vedi (§4.2.7.3) ;
- **AWSS3Manager**: vedi (§4.2.6.3) ;
- **Document**: vedi (§4.2.16.3) ;
- **DocumentContent**: vedi (§4.2.16.3) ;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentMetadata**: vedi (§4.2.13.3) ;
- **DocumentStatus**: vedi (§4.2.7.3) ;
- **DocumentType**: vedi (§4.2.13.3) ;
- **GetDocumentController**: classe controller che si occupa del passaggio di un DocumentId allo use case GetDocumentUseCase a partire da una stringa che rappresenta l'id del documento da recuperare;
- **GetDocumentFacadeService**: classe service che implementa lo use case GetDocumentUseCase;
- **GetDocumentsContent**: vedi (§4.2.7.3) ;
- **GetDocumentsContentAWSS3**: vedi (§4.2.7.3) ;
- **GetDocumentsContentPort**: vedi (§4.2.7.3) ;
- **GetDocumentsStatus**: vedi (§4.2.7.3) ;
- **GetDocumentsStatusPort**: (§4.2.7.3) ;
- **GetDocumentsStatusVectorStore**: vedi (§4.2.7.3) ;

- **GetDocumentUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per recuperare un documento;
- **PlainDocument**: vedi (§4.2.16.3) ;
- **Status**: vedi (§4.2.7.3) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentStatusResponse**: vedi (§4.2.7.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.3 Database

5 Progettazione di dettaglio

5.1 AskChatbot

5.1.1 Diagramma delle classi

5.1.2 Lista delle sottocomponenti

5.1.2.1 AskChatbotController

- **Attributi**:
  - `useCase: AskChatbotUseCase.`
- **Metodi**:
  - `askChatbot(message:string, ChatId:int): MessageResponse`
    Metodo che ritorna un MessageResponse rappresentante la risposta da parte del chatbot ad un messaggio dell'utente sotto forma di string, appartenente alla chat identificata dall'intero chatId. Nel caso in cui il chatId sia vuoto e venga inoltre generata correttamente una risposta, viene creata una nuova chat;

5.1.2.2 AskChatbotLangchain

- **Implementa**: AskChatbotPort;
- **Attributi**:
  - `chain: langchain.Chain;`
  - `chatHistoryManager: ChatHistoryManager.`
- **Metodi**:
  - `askChatbot(message:Message, chatId:ChatId): MessageResponse`
    Implementazione del metodo astratto di AskChatbotPort per ottenere una risposta dal chatbot ad un messaggio appartenente ad una chat identificata da un ChatId.

5.1.2.3 AskChatbotPort

- **Metodi**:
  - `askChatbot(message: Message, chatId: ChatId): MessageResponse`
    Metodo astratto per ottenere una risposta dal chatbot ad un messaggio appartenente ad una chat identificata da un ChatId. Nel caso in cui il chatId sia vuoto e venga inoltre generata correttamente una risposta, viene creata una nuova chat.

5.1.2.4 AskChatbotService

- **Implementa**: AskChatbotUseCase;
- **Attributi**:
  - `askChatbotoutPort: AskChatbotPort;`
  - `persistChatOutPort: PersistChatPort.`

- **Metodi**:
  - `askChatbot(message:Message, chatId:ChatId): MessageResponse`
    Implementazione del metodo astratto di AskChatbotUseCase per ottenere tramite outPort una risposta dal chatbot ad un messaggio appartenente ad una chat identificata da un ChatId. Nel caso in cui il chatId sia vuoto e venga inoltre generata correttamente una risposta, viene creata una nuova chat.

5.1.2.5 AskChatbotUseCase

- **Metodi**:
  - `askChatbot(message:Message, chatId:ChatId): MessageResponse`
    Metodo astratto per ottenere una risposta dal chatbot ad un messaggio appartenente ad una chat identificata da un ChatId. Nel caso in cui il chatId sia vuoto e venga inoltre generata correttamente una risposta, viene creata una nuova chat.

5.1.2.6 ChatHistoryManager

- **Metodi**:
  - `getChatHistory(chatId:int): langchain.PostgresChatMessageHistory`
    Metodo per ottenere una langchain.PostgresChatMessageHistory rappresentante il contesto di una chat identificata da un ChatId;

5.1.2.7 ChatId

- **Attributi**:
  - `id: int.`

5.1.2.8 ChatOperationResponse

- **Attributi**:
  - `chatId: ChatId;`
  - `message: string;`
  - `status: boolean.`

5.1.2.9 Message

- **Attributi**:
  - `content: string;`
  - `relevantDocument: List<DocumentId>;`
  - `sender: MessageSender;`
  - `timestamp: datetime.`

5.1.2.10 MessageResponse

- **Attributi**:
  - `chatId: ChatId;`
  - `message: Message;`
  - `status: boolean.`

5.1.2.11 MessageSender (Enumeration)

- **Valori**:
  - `Chatbot;`
  - `User.`

5.1.2.12 PersistChatPort

- **Metodi**:
  - `persistChat( messages:List<Message>, chatId:ChatId): ChatOperationResponse`
    Metodo astratto che prende una lista di Messages e la rende persistente, restituendo un ChatOperationResponse, che rappresenta l'esito della operazione.

5.1.2.13 PostgresChat

- **Attributi**:
  - `id: int;`
  - `messages: List<PostgresMessage>;`
  - `title: string.`
- **Metodi**:
  - `toChat():Chat`
    Metodo che ritorna un oggetto Chat.

5.1.2.14 PostgresChatOperationResponse

- **Attributi**:
  - `chatId: int;`
  - `message: string;`
  - `status: boolean.`
- **Metodi**:
  - `toChatOperationResponse(): ChatOperationResponse`
    Metodo che ritorna l'esito dell'operazione effettuata sottoforma di ChatOperationResponse.

5.1.2.15 PostgresMessage

- **Attributi**:
  - `content: string;`
  - `relevantDocument: List<string>;`
  - `sender: PostgresMessageSenderType;`
  - `timestamp: datetime.`
- **Metodi**:
  - `toMessage(): Message`
    Metodo che permette di passare da PostgresMessage a un oggetto Message.

5.1.2.16 PostgresMessageSenderType(Enumeration)

- **Valori**:
  - `AI;`
  - `human.`

5.1.2.17 PostgresPersistChat

- **Implementa**: PersistChatPort;
- **Attributi**:
  - `postgresChatORM: PostgresChatORM.`
- **Metodi**:
  - `persistChat(messages:List<Message>, chatId: ChatId): ChatOperationResponse`
    Metodo che implementa il metodo persistChat di PersistChatPort; prende una lista di Messages e la rende persistente, restituendo un ChatOperationResponse, che rappresenta l'esito della operazione.
  - `toPostgresMessageFrom(message:Message): PostgresMessage`
    Metodo che permette di passare da un oggetto Message ad un oggetto persistente PostgresMessage.

5.1.2.18 PostgresChatORM

- **Metodi**:
  - `createChat(): PostgresChatOperationResponse`
    Metodo per creare una nuova PostgresChat sul database Postgres utilizzato per la storicizzazione della chat;
  - `deleteChats(chatIds:List<int>): List<PostgresChatOperationResponse>`
    Metodo per eliminare chat sul database Postgres, utilizzato per la storicizzazione delle chat. Ritorna l'esito dell'operazione con una lista di PostgresChatOperationResponse;
  - `getChatMessages(chatId:int): PostgresChat`
    Metodo per ottenere una PostgresChat dal database Postgres utilizzato per

la storicizzazione delle chat a partire da un intero rappresentante l'id della chat;

- `getChats(chatFilter:string): List<PostgresChatPreview>`
  Metodo per ottenere una lista di PostgresChatPreview dal database Postgres utilizzato per la storicizzazione delle chat, eventualmente filtrando la ricerca attraverso chatFilter;
- `persistChat(messages:List<PostgresMessage>, chatId:int): PostgresChatOperationResponse`
  Metodo che prende una lista di PostgresMessages e la rende persistente utilizzando il metodo saveMessages. Restituisce un PostgresChatOperationResponse, che rappresenta l'esito della operazione.
- `renameChat(chatId:int, title:string): PostgresChatOperationResponse`
  Metodo per rinominare con title una chat identificata tramite chatId sul database Postgres utilizzato per la storicizzazione delle chat. Ritorna l'esito dell'operazione con un PostgresChatOperationResponse;
- `saveMessages(messages:List<PostgresMessage>, chatId:int): PostgresChatOperationResponse`
  Metodo per salvare una lista di PostgresMessages. Restituisce l'esito dell'operazione come PostgresChatOperationResponse.

5.2 ChangeConfiguration

5.2.1 Diagramma delle classi

5.2.2 Lista delle sottocomponenti

5.2.2.1 ChangeConfigurationController

- **Attributi**:
  - `useCase: ChangeConfigurationUseCase.`
- **Metodi**:
  - `changeLLMModel(LLMModel:string): ConfigurationOperationResponse`
    Metodo che si occupa di effettuare il cambio di configurazione del LLMModel identificato da una stringa e ritorna l'esito della operazione come ConfigurationOperationResponse.

5.2.2.2 ChangeConfigurationPort

- **Metodi**:
  - `changeLLMModel(LLMModel:LLMModelType): ConfigurationOperationResponse`
    Metodo astratto che si occupa di effettuare il cambio di configurazione del modello LLM con LLMModelType e ritorna l'esito della operazione come ConfigurationOperationResponse.

5.2.2.3 ChangeConfigurationPostgres

- **Implementa**: ChangeConfigurationPort;
- **Attributi**:

- `postgresConfigurationORM: PostgresConfigurationORM.`

- **Metodi**:
  - `changeLLMModel(LLMModel:LLMModelType): ConfigurationOperationResponse` Implementazione del metodo astratto di ChangeConfigurationPort per eseguire il cambio di configurazione del LLMModel. Ritorna l'esito della operazione come ConfigurationOperationResponse;
  - `toPostgresLLMModelTypeFrom(LLMModel:LLMModelType): PostgresLLMModelType` Metodo che ottiene un PostgresLLMModelType a partire da un LLMModelType.

5.2.2.4 ChangeConfigurationService

- **Implementa**: ChangeConfigurationUseCase;
- **Attributi**:
  - `outPort: ChangeConfigurationPort.`
- **Metodi**:
  - `changeLLMModel(LLMModel:LLMModelType): ConfigurationOperationResponse` Implementazione del metodo astratto di ChangeConfigurationUseCase per eseguire il cambio di configurazione del LLMModel. Ritorna l'esito della operazione come ConfigurationOperationResponse.

5.2.2.5 ChangeConfigurationUseCase

- **Metodi**:
  - `changeLLMModel(LLMModel:LLMModelType): ConfigurationOperationResponse` Metodo astratto per eseguire il cambio di configurazione del LLMModel. Ritorna l'esito della operazione come ConfigurationOperationResponse.

5.2.2.6 ConfigurationOperationResponse

- **Attributi**:
  - `message: string;`
  - `status: boolean.`

5.2.2.7 PostgresConfigurationOperationResponse

- **Attributi**:
  - `message: string;`
  - `status: boolean.`
- **Metodi**:
  - `toConfigurationOperationResponse(): ConfigurationOperationResponse` Metodo per ottenere un ConfigurationOperationResponse a partire dal PostgresChatOperationResponse.

5.2.2.8 PostgresConfigurationORM

- **Metodi**:
  - `changeLLMModel(userId:int, LLMModel:PostgresLLMModelType): PostgresConfigurationOperationResponse`
    Metodo che esegue il cambio di configurazione del LLMModel con un PostgresLLMModelType. Ritorna l'esito della operazione come PostgresConfigurationOperationResponse;
  - `getConfiguration(userId:int): PostgresConfiguration`
    Metodo per ottenere la configurazione del sistema; ricava le configurazioni di userId, vector store, modello di embedding, LLM e metodo di storicizzazione dei documenti. Ritorna un oggetto PostgresConfiguration;
  - `getConfigurationChoices(userId:int): PostgresConfigurationChoice`
    Metodo che restituisce la scelta di configurazione dell'utente come oggetto PostgresConfigurationChoice;
  - `getDocumentStoreOptions(): List<PostgresDocumentStoreConfiguration>`
    Metodo che restituisce una lista delle possibili alternative di configurazione di storage per i documenti sottoforma di lista di PostgresDocumentStoreConfiguration;
  - `getEmbeddingModelOptions(): List<PostgresEmbeddingModelConfiguration>`
    Metodo che restituisce le possibili scelte di configurazione per il modello di embeddingm sottoforma di una lista di PostgresEmbeddingModelConfiguration;
  - `getLLMModelOptions(): List<PostgresLLMModelConfiguration>`
    Metodo che restituisce le possibili scelte di modelli LLM sottoforma di lista di PostgresLLMModelConfiguration;
  - `getVectorStoreOptions(): List<PostgresVectorStoreConfiguration>:`
    Metodo che restituisce la lista delle possibili alternative di configurazione di vector store sottoforma di lista di PostgresVectorStoreConfiguration.
  - `setConfiguration(userId:int, LLMModel:PostgresLLMModelType, DocumentStore:PostgresVectorStore:PostgresVectorStoreType, EmbeddingModel:PostgresEmbeddingModelType): PostgresConfigurationOperationResponse`
    Metodo per impostare le configurazioni di vector store, modello di embedding, LLM e metodo di storicizzazione dei documenti.

5.3 ConcealDocuments

5.3.1 Diagramma delle classi

5.3.2 Lista delle sottocomponenti

5.3.2.1 ConcealDocumentsController

- **Attributi**:
  - `useCase: ConcealDocumentsUseCase.`
- **Metodi**:

- `concealDocuments(documentIds:List<string>): List<DocumentOperationResponse>`
  Metodo che si occupa di trasformare le stringhe di id in DocumentId e inoltrare l'occultamento dei documenti a ConcealDocumentsUseCase;

5.3.2.2 ConcealDocumentsPort

- **Metodi**:
  - `concealDocuments(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo astratto per occultare una lista di documenti a partire dal loro DocumentId, dialogando con il vector store. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.3.2.3 ConcealDocumentsService

- **Implementa**: ConcealDocumentsUseCase;
- **Attributi**:
  - `outPort: ConcealDocumentsPort.`
- **Metodi**:
  - `concealDocuments(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di ConcealDocumentsUseCase per occultare una lista di documenti a partire dal loro DocumentId. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.3.2.4 ConcealDocumentsUseCase

- **Metodi**:
  - `concealDocuments(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo astratto per occultare una lista di documenti a partire dal loro DocumentId. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.3.2.5 ConcealDocumentsVectorStore

- **Implementa**: ConcealDocumentsPort;
- **Attributi**:
  - `vectorStoreManager: VectorStoreManager.`
- **Metodi**:
  - `concealDocuments(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di ConcealDocumentsPort per occultare una lista di documenti a partire dal loro DocumentId, dialogando con

il vector store. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.3.2.6 DocumentId

- **Attributi**:
  - `id: string.`

5.3.2.7 DocumentOperationResponse

- **Attributi**:
  - `documentId: DocumentId;`
  - `message: string;`
  - `status: boolean.`

5.3.2.8 VectorStoreChromaDBManager

- **Attributi**:
  - `chroma: chromadb.PersistentClient;`
  - `collection: chromadbCollection.`
- **Implementa**: VectorStoreManager;
- **Metodi**:
  - `concealDocuments(documentsId:List<string>): List<VectorStoreDocumentOperationResponse>`
    Implementazione del metodo astratto di VectorStoreManager per occultare in ChromaDB gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `deleteDocumentsEmbeddings(documentsIds:List<string>): List<VectorStoreDocumentOperationResponse>`
    Implementazione del metodo astratto di VectorStoreManager per eliminare da ChromaDB gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `enableDocuments(documentsId:List<string>): List<VectorStoreDocumentOperationResponse>`
    Implementazione del metodo astratto di VectorStoreManager per abilitare in ChromaDB gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `getDocumentsStatus(documentIds:List<string>): List<VectorStoreDocumentStatusResponse>`
    Implementazione del metodo astratto di VectorStoreManager per ottenere

da ChromaDB gli status di una lista di documenti dai loro metadati a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornandoli in una lista di VectorStoreDocumentStatusResponse;

- `uploadEmbeddings(documentsIds:List<string>, documentsChunks: List<List<LangchainCoreDocument>>, documentEmbeddings: List<List<List<float>>>): List<VectorStoreDocumentOperationResponse>`
  Implementazione del metodo astratto di VectorStoreManager per effettuare in ChromaDB l'upload degli embeddings di un documento. Ritorna un VectorStoreDocumentOperationResponse;
- `getRetriever(embeddingModel:LangchainEmbeddingModel): langchain.BaseRetriever`
  Implementazione del metodo astratto di VectorStoreManager per ottenere un langchain.BaseRetriever da un vector store ChromaDB.

5.3.2.9 VectorStoreDocumentOperationResponse

- **Attributi**:
  - `documentId: string;`
  - `message: string;`
  - `status: boolean.`
- **Metodi**:
  - `toDocumentOperationResponse(): DocumentOperationResponse`
    Metodo che crea e ritorna un DocumentOperationResponse.

5.3.2.10 VectorStoreManager

- **Metodi**:
  - `concealDocuments(documentsId:List<string>): List<VectorStoreDocumentOperationResponse>`
    Metodo astratto per occultare gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `deleteDocumentsEmbeddings(documentsIds:List<string>): List<VectorStoreDocumentOperationResponse>`
    Metodo astratto per eliminare gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `enableDocuments(documentsId:List<string>): List<VectorStoreDocumentOperationResponse>`
    Metodo astratto per abilitare gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `getDocumentsStatus(documentIds:List<string>): List<VectorStoreDocumentStatusResponse>`

Metodo astratto per ottenere gli status di una lista di documenti dai loro metadati a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornandoli in una lista di VectorStoreDocumentStatusResponse;

- `uploadEmbeddings(documentsIds:List<string>, documentsChunks: List<List<LangchainCoreDocument>>,documentEmbeddings: List<List<List<float>>>):List<VectorStoreDocumentOperationResponse>`
  Metodo astratto per effettuare l'upload degli embeddings di un documento. Ritorna un VectorStoreDocumentOperationResponse;
- `getRetriever(embeddingModel:LangchainEmbeddingModel): langchain.BaseRetriever`
  Metodo astratto per ottenere un langchain.BaseRetriever.

5.3.2.11 VectorStorePineconeManager

- **Attributi**:
  - `dimension: int;`
  - `index: string;`
  - `pinecone: pinecone.Pinecone.`
- **Implementa**: VectorStoreManager;
- **Metodi**:
  - `concealDocuments(documentsId:List<string>): List<VectorStoreDocumentOperationResponse>`
    Implementazione del metodo astratto di VectorStoreManager per occultare in Pinecone gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `deleteDocumentsEmbeddings(documentsIds:List<string>): List<VectorStoreDocumentOperationResponse>`
    Implementazione del metodo astratto di VectorStoreManager per eliminare da Pinecone gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `enableDocuments(documentsId:List<string>): List<VectorStoreDocumentOperationResponse>`
    Implementazione del metodo astratto di VectorStoreManager per abilitare in Pinecone gli embeddings di una lista di documenti a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornando una lista di VectorStoreDocumentOperationResponse;
  - `getDocumentsStatus(documentIds:List<string>): List<VectorStoreDocumentStatusResponse>`
    Implementazione del metodo astratto di VectorStoreManager per ottenere da Pinecone gli status di una lista di documenti dai loro metadati a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornandoli in una lista di VectorStoreDocumentStatusResponse;

- `uploadEmbeddings(documentsIds:List<string>, documentsChunks: List<List<LangchainCoreDocument>>,documentEmbeddings: List<List<List<float>>>):List<VectorStoreDocumentOperationResponse>`
  Implementazione del metodo astratto di VectorStoreManager per effettuare in Pinecone l'upload degli embeddings di un documento. Ritorna un VectorStoreDocumentOperationResponse;
- `getRetriever(embeddingModel:LangchainEmbeddingModel): langchain.BaseRetriever`
  Implementazione del metodo astratto di VectorStoreManager per ottenere un langchain.BaseRetriever da un vector store Pinecone.

5.4 DeleteChats

5.4.1 Diagramma delle classi

5.4.2 Lista delle sottocomponenti

5.4.2.1 ChatId

Vedi (§5.1.2.7) .

5.4.2.2 ChatOperationResponse

Vedi (§5.1.2.8) ;

5.4.2.3 DeleteChatsController

- **Attributi**:
  - `useCase: DeleteChatsUseCase.`
- **Metodi**:
  - `deleteChats(chatIdsList:List<int>): List<ChatOperationResponse>`
    Metodo che elimina tramite useCase una lista di chat identificate da una lista di interi, ritornando una lista di ChatOperationResponse.

5.4.2.4 DeleteChatsPort

- **Metodi**:
  - `deleteChats(chatIdsList:List<ChatId>): List<ChatOperationResponse>`
    Metodo astratto per per eliminare una lista di chat identificate da una lista di ChatId, ritornando una lista di ChatOperationResponse.

5.4.2.5 DeleteChatsPostgres

- **Implementa**: DeleteChatsPort;
- **Attributi**:
  - `postgresORM: PostgresChatORM.`
- **Metodi**:

- `deleteChats(chatIdsList:List<ChatId>): List<ChatOperationResponse>`
  Implementazione del metodo astratto di DeleteChatsPort per eliminare una lista di chat identificate da una lista di ChatId, ritornando una lista di ChatOperationResponse.

5.4.2.6 DeleteChatsService

- **Implementa**: DeleteChatsUseCase;
- **Attributi**:
  - `outPort: DeleteChatsPort.`
- **Metodi**:
  - `deleteChats(chatIdsList:List<ChatId>): List<ChatOperationResponse>`
    Implementazione del metodo astratto di DeleteChatsUseCase per eliminare tramite outPort una lista di chat identificate da una lista di ChatId, ritornando una lista di ChatOperationResponse.

5.4.2.7 DeleteChatsUseCase

- **Metodi**:
  - `deleteChats(chatIdsList:List<ChatId>): List<ChatOperationResponse>`
    Metodo astratto per eliminare una lista di chat identificate da una lista di ChatId, ritornando una lista di ChatOperationResponse.

5.4.2.8 PostgresChatOperationResponse

Vedi (§5.1.2.14) ;

5.4.2.9 PostgresChatORM

Vedi (§5.1.2.18) .

5.5 ConfigurationManager

5.5.1 Diagramma delle classi

5.5.2 Lista delle sottocomponenti

5.5.2.1 ConfigurationManager

- **Attributi**:
  - `ORM: PostgresConfigurationORM.`
- **Metodi**:
  - `getAskChatbotPort(): AskChatbotPort`
    Metodo per ottenere la porta in uscita della componente AskChatbot;

- `getConcealDocumentsPort(): ConcealDocumentsPort`
  Metodo per ottenere la porta in uscita della componente ConcealDocuments;
- `getDeleteDocumentsPort(): DeleteDocumentsPort`
  Metodo per ottenere una delle porte in uscita della componente DeleteDocuments;
- `getDeleteEmbeddingsPort(): DeleteEmbeddingsPort`
  Metodo per ottenere una delle porte in uscita della componente DeleteDocuments;
- `getDocumentsContentPort(): GetDocumentsContentPort`
  Metodo per ottenere la porta in uscita delle componenti EmbedDocuments e ViewDocumentContent;
- `getDocumentsUploaderPort(): DocumentsUploaderPort`
  Metodo per ottenere la porta in uscita della componente UploadDocuments;
- `getEmbeddingsUploaderPort(): EmbeddingsUploaderPort` Metodo per ottenere la porta in uscita della componente EmbedDocuments e UploadDocuments;
- `getEnableDocumentsPort(): EnableDocumentsPort`
  Metodo per ottenere la porta in uscita della componente EnableDocuments;
- `getGetDocumentsMetadataPort(): GetDocumentsMetadataPort`
  Metodo per ottenere una delle porte in uscita della componente GetDocuments;
- `getGetDocumentsStatusPort(): GetDocumentsStatusPort`
  Metodo per ottenere una delle porte in uscita delle componenti EmbedDocuments, GetDocuments e ViewDocumentContent.

5.5.2.2 PostgresConfiguration

- **Attributi**:
  - `documentStore: PostgresDocumentStoreConfiguration;`
  - `embeddingModel: PostgresEmbeddingModelConfiguration;`
  - `id: int;`
  - `LLMModel: PostgresLLMModelConfiguration;`
  - `vectorStore: PostgresVectorStoreConfiguration.`

5.5.2.3 PostgresConfigurationORM

Vedi (§5.2.2.8) ;

5.5.2.4 PostgresDocumentStoreConfiguration

- **Attributi**:
  - `costIndicator: string;`
  - `description: string;`

- `name: PostgresDocumentStoreType;`
- `organization: string;`
- `type: string.`

- **Metodi**:
  - `toDocumentStoreConfiguration(): DocumentStoreConfiguration`
    Metodo che permette di ottenere un DocumentStoreConfiguration a partire dal PostgresDocumentStoreConfiguration.

5.5.2.5 PostgresDocumentStoreType (Enumeration)

- **Valori**:
  - `AWS.`

5.5.2.6 PostgresEmbeddingModelConfiguration

- **Attributi**:
  - `costIndicator: string;`
  - `description: string;`
  - `name: PostgresEmbeddingModelType;`
  - `organization: string;`
  - `type: string.`
- **Metodi**:
  - `toEmbeddingModelConfiguration(): EmbeddingModelConfiguration`
    Metodo che permette di ottenere un EmbeddingModelConfiguration a partire dal PostgresEmbeddingModelConfiguration.

5.5.2.7 PostgresEmbeddingModelType (Enumeration)

- **Valori**:
  - `HUGGINGFACE;`
  - `OPENAI.`

5.5.2.8 PostgresLLMModelConfiguration

- **Attributi**:
  - `costIndicator: string;`
  - `description: string;`
  - `name: PostgresLLMModelType;`
  - `organization: string;`
  - `type: string.`
- **Metodi**:

- `toLLMModelConfiguration(): LLMModelConfiguration`
  Metodo che permette di ottenere un LLMModelConfiguration a partire dal PostgresLLMModelConfiguration.

5.5.2.9 PostgresLLMModelType (Enumeration)

- **Valori**:
  - `HUGGINGFACE;`
  - `OPENAI.`

5.5.2.10 PostgresVectorStoreConfiguration

- **Attributi**:
  - `costIndicator: string;`
  - `description: string;`
  - `name: PostgresVectorStoreType;`
  - `organization: string;`
  - `type: string.`
- **Metodi**:
  - `toVectorStoreConfiguration(): VectorStoreConfiguration`
    Metodo che permette di ottenere un VectorStoreConfiguration a partire dal PostgresVectorStoreConfiguration.

5.5.2.11 PostgresVectorStoreType (Enumeration)

- **Valori**:
  - `CHROMA DB;`
  - `PINECONE.`

5.6 DeleteDocuments

5.6.1 Diagramma delle classi

5.6.2 Lista delle sottocomponenti

5.6.2.1 AWSDocumentOperationResponse

- **Attributi**:
  - `documentId: string;`
  - `message: string;`
  - `status: boolean.`
- **Metodi**:
  -

- `toDocumentOperationResponse(): DocumentOperationResponse:`
  Metodo per ottenere un oggetto DocumentOperationResponse dal AWSDocumentOperationResponse.

5.6.2.2 AWSS3Manager

- **Attributi**:
  - `awsBucketName: string;`
  - `boto3: boto3.session.Session.`
- **Metodi**:
  - `deleteDocuments(documentsId:List<string>): List<AWSDocumentOperationResponse>`
    Metodo per eliminare un documento a partire da una stringa corrispondente al documentId del documento tramite boto3, ritornando un AWSDocumentOperationResponse;
  - `getDocumentById(documentId:string): AWSDocument`
    Metodo per ottenere un AWSDocument a partire da una stringa corrispondente al documentId del documento tramite boto3;
  - `getDocumentsMetadata(documentFilter:string): List<AWSDocumentMetadata>`
    Metodo per ottenere una lista di AWSDocumentMetaData a partire da una eventuale filtro stringa tramite boto3;
  - `uploadDocuments(documents:List<AWSDocument>, forceUpload:boolean): List<AWSDocumentOperationResponse>`
    Metodo per caricare una lista di AWSDocument tramite boto3, con flag forceUpload per forzare il caricamento di documenti già presenti nel sistema attraverso sostituzione, ritornando una lista di AWSDocumentOperationResponse.

5.6.2.3 DeleteDocuments

- **Attributi**:
  - `outPort: DeleteDocumentsPort.`
- **Metodi**:
  - `deleteDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo per eliminare una lista di documenti tramite outPort dal sistema di archiviazione a partire dai loro id.

5.6.2.4 DeleteDocumentsAWSS3

- **Implementa**: DeleteDocumentsPort;
- **Attributi**:
  - awsS3Manager: AWSS3Manager.
- **Metodi**:

- `deleteDocuments(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
  Implementazione del metodo astratto di DeleteDocumentsPort per eliminare una lista di documenti da AWS S3 a partire dai loro id.

5.6.2.5 DeleteDocumentsController

- **Attributi**:
  - `useCase: DeleteDocumentsUseCase.`
- **Metodi**:
  - `deleteDocuments(documentIds:List<string>): List<DocumentOperationResponse>`
    Metodo che si occupa di trasformare la lista di stringhe in lista di DocumentId e inoltrare la richiesta di eliminazione dei documenti a DeleteDocumentsUseCase.

5.6.2.6 DeleteDocumentsEmbeddings

- **Attributi**:
  - `outPort: DeleteEmbeddingsPort.`
- **Metodi**:
  - `deleteDocumentsEmbeddings(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo per eliminare gli embeddings di una lista di documenti, a partire dai loro id, tramite outPort dal vector store.

5.6.2.7 DeleteDocumentsPort

- **Metodi**:
  - `deleteDocuments(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo astratto per eliminare una lista di documenti dal sistema di embeddings a partire dai loro id.

5.6.2.8 DeleteDocumentsService

- **Implementa**: DocumentsUseCase;
- **Attributi**:
  - `documentsDeleter: DeleteDocuments;`
  - `deleteDocumentsEmbeddings: DeleteDocumentsEmbeddings.`
- **Metodi**:
  - `deleteDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di DeleteDocumentsUseCase per eliminare una lista di documenti e i loro embeddings a partire dai loro id.

5.6.2.9 DeleteDocumentsUseCase

- **Metodi**:
  - `deleteDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo astratto per eliminare una lista di documenti a partire dai loro id.

5.6.2.10 DeleteEmbeddingsPort

- **Metodi**:
  - `deleteDocumentsEmbeddings(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo astratto per eliminare gli embeddings di una lista di documenti dal vector store a partire dai loro id.

5.6.2.11 DeleteEmbeddingsVectorStore

- **Implementa**: DeleteEmbeddingsPort;
- **Attributi**:
  - vectorStoreManager: VectorStoreManager.
- **Metodi**:
  - `deleteDocumentsEmbeddings(documentIds:List<DocumentId>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di DeleteEmbeddingsPort per eliminare gli embeddings di una lista di documenti dal vector store a partire dai loro id.

5.6.2.12 DocumentId

Vedi (§5.3.2.6) .

5.6.2.13 DocumentOperationResponse

Vedi (§5.3.2.7) .

5.6.2.14 VectorStoreChromaDBManager

Vedi (§5.3.2.8) .

5.6.2.15 VectorStoreDocumentOperationResponse

Vedi (§5.3.2.9) .

5.6.2.16 VectorStoreManager

Vedi (§5.3.2.10) .

5.6.2.17 VectorStorePineconeManager

Vedi (§5.3.2.11) .

5.7 EmbedDocuments

5.7.1 Diagramma delle classi

5.7.2 Lista delle sottocomponenti

5.7.2.1 AWSDocument

- **Attributi**:
  - `content: bytes;`
  - `id: string;`
  - `size: float;`
  - `type: string;`
  - `uploadTime: timestamp.`
- **Metodi**:
  - `toPlainDocument(): PlainDocument`
    Metodo che ottiene ritorna un PlainDocument.

5.7.2.2 AWSS3Manager

Vedi (§5.6.2.2) .

5.7.2.3 Chunkerizer

- **Metodi**:
  - `extractText(document:Document): List<langchain.Document>`
    Metodo per suddividere in chunks una lista di langchain.Document, ritornandoli attraverso una lista di langchain.Document.
  - `getTextExtractorFrom(documentType:string): TextExtractor`
    Metodo che ritorna il tipo di TextExtractor da usare.

5.7.2.4 DocumentId

Vedi (§5.3.2.6) .

5.7.2.5 DocumentOperationResponse

Vedi (§5.3.2.7) .

5.7.2.6 DocumentStatus

- **Attributi**:
  - `status: Status.`

5.7.2.7 DOCXTextExtractor

- **Implementa**: TextExtractor;
- **Metodi**:
  - `extractText(document:DocumentContent): List<langchain.Document>`
    Implementazione del metodo astratto di TextExtractor per estrarre il testo da un DOCX sotto forma di langchain.Document a partire da un DocumentContent.

5.7.2.8 EmbeddingsCreator

- **Attributi**:
  - `langchainEmbeddingModel: LangchainEmbeddingModel.`
- **Metodi**:
  - `embedDocument(documents:List<langchain.Document>): List<List<float>>`
    Metodo per generare una lista di embeddings a partire da una lista di langchain.Document: per ogni chunk di langchain.Document presente nella lista viene generata una lista di float che rappresenta gli embeddings corrispondenti.

5.7.2.9 EmbeddingsUploader

- **Attributi**:
  - `outPort: EmbeddingsUploaderPort.`
- **Metodi**:
  - `uploadEmbeddings(documents:List<Document>): List<DocumentOperationResponse>`
    Metodo per effettuare la generazione e il caricamento degli embeddings di una lista di Document tramite outPort in un vector store, ritornando infine una lista di DocumentOperationResponse.

5.7.2.10 EmbeddingsUploaderFacadeLangchain

- **Implementa**: EmbeddingsUploaderPort;
- **Attributi**:
  - `chunkerizer: Chunkerizer;`
  - `embeddingsCreator: EmbeddingsCreator;`
  - `embeddingsUploaderVectorStore: EmbeddingsUploaderVectorStore.`
- **Metodi**:
  - `uploadEmbeddings(documents:List<Document>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di EmbeddingsUploaderPort per la generazione e il caricamento di embeddings di una lista di Document nel vector

store tramite l'utilizzo di un pattern facade, ritornando infine una lista di DocumentOperationResponse.
Le operazioni all'interno della facade vengono svolte nel seguente ordine:

1. Estrazione del testo dai documenti tramite documentToText;
2. Suddivisione in chunks del testo estratto dai documenti tramite chunkerizer;
3. Generazione degli embeddings a partire dai chunks tramite embeddingsCreator;
4. Caricamento degli embeddings creati nel vector store tramite embeddingsUploaderVectorStore.

5.7.2.11 EmbeddingsUploaderPort

- **Metodi**:
  - `uploadEmbeddings(documents:List<Document>): List<DocumentOperationResponse>`
    Metodo astratto per effettuare la generazione e il caricamento degli embeddings di una lista di Document in un vector store.

5.7.2.12 EmbeddingsUploaderVectorStore

- **Attributi**:
  - `vectorStoreManager: VectorStoreManager.`
- **Metodi**:
  - `uploadEmbeddings(documents:List<LangchainDocument>): List<VectorStoreDocumentOperationResponse>`
    Metodo per effettuare l'upload di una lista di LangchainDocument nel vector store tramite vectorStoreManager, ritornando una lista di VectorStoreDocumentOperationResponse.

5.7.2.13 EmbedDocumentsController

- **Attributi**:
  - `useCase: EmbedDocumentsUseCase.`
- **Metodi**:
  - `embedDocuments(documentsIds:List<DocumentIds>): List<DocumentOperationResponse>`
    Metodo che si occupa di trasformare le stringhe di id in DocumentId e inoltrare la generazione degli embeddings dei documenti a EmbedDocumentsUseCase.

5.7.2.14 EmbedDocumentsService

- **Implementa**: EmbedDocumentsUseCase;
- **Attributi**:

- `embeddingsUploader: EmbeddingsUploader;`
- `getDocumentsContent: GetDocumentsContent;`
- `getDocumentsStatus: GetDocumentsStatus.`

- **Metodi**:
  - `embedDocuments(documentsIds:List<DocumentIds>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di EmbedDocumentsUseCase, per il recupero dei documenti e il loro contenuto attraverso getDocuments, e la generazione degli embeddings della lista dei documenti recuperati tramite embeddingsUploader. Ritorna una lista di DocumentOperationResponse, indicando per ogni documento l'esito della generazione degli embeddings.

5.7.2.15 EmbedDocumentsUseCase

- **Metodi**:
  - `embedDocuments(documentsIds:List<DocumentIds>): List<DocumentOperationResponse>`
    Metodo astratto per generare gli embeddings di una lista di documenti a partire dal loro DocumentId. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.7.2.16 GetDocumentsContent

- **Attributi**:
  - `outPort: GetDocumentsContentPort.`
- **Metodi**:
  - `getDocumentsContent(documentsIds:List<DocumentId>): List<PlainDocument>`
    Metodo per recuperare il contenuto di una lista di Document tramite outPort dal sistema di archiviazione documenti a partire da una lista di id.

5.7.2.17 GetDocumentsContentAWSS3

- **Implementa**: DocumentsContentPort;
- **Attributi**:
  - `awsS3Manager: AWSS3Manager.`
- **Metodi**:
  - `getDocumentsContent(documentsIds:List<DocumentId>): List<PlainDocument>`
    Implementazione del metodo astratto di GetDocumentsContentPort per recuperare il contenuto di una lista di documenti dal sistema di archiviazione a partire dal loro id;
  - `toPlainDocumentFrom(document:AWSDocument): PlainDocument`
    Metodo che trasforma un AWSDocument in un Document della business logic.

5.7.2.18 GetDocumentsContentPort

- **Metodi**:
  - `getDocumentsContent(documentsIds:List<DocumentId>): List<PlainDocument>`
    Metodo astratto per recuperare il contenuto di una lista di documenti dal sistema di archiviazione a partire dal loro id.

5.7.2.19 GetDocumentsStatus

- **Attributi**:
  - `outPort: GetDocumentsStatusPort.`
- **Metodi**:
  - `getDocumentsStatus(documentsIds:List<DocumentId>) List<DocumentStatus>`
    Metodo per recuperare gli status di una lista di documenti tramite outPort dal vector store a partire dai loro id.

5.7.2.20 GetDocumentsStatusPort

- **Metodi**:
  - `getDocumentsStatus(documentsIds:List<DocumentId>): List<DocumentStatus>`
    Metodo astratto per recuperare gli status di una lista di documenti a partire dal loro id.

5.7.2.21 GetDocumentsStatusVectorStore

- **Implementa**: GetDocumentsStatusPort;
- **Attributi**:
  - `vectorStoreManager: VectorStoreManager.`
- **Metodi**:
  - `getDocumentsStatus(documentsIds:List<DocumentId>): List<DocumentStatus>`
    Implementazione del metodo astratto di GetDocumentsStatusVectorStore per recuperare gli status di una lista di documenti a partire dal loro id.

5.7.2.22 HuggingFaceEmbeddingModel

- **Implementa**: LangchainEmbeddingModel;
- **Attributi**:
  - `embeddingsDimension: int;`
  - `model: langchain.Embedding.`
- **Metodi**:

- `embedDocument(documentChunks:List<string>): List<List<float>>`
  Implementazione del metodo astratto di LangchainEmbeddingModel per generare gli embeddings di un insieme di chunks di stringhe tramite un modello di embeddings di HuggingFace: per ogni chunk in lista viene generata una lista di float che rappresenta gli embeddings;
- `getEmbeddingFunction(): langchain.Embedding`
  Implementazione del metodo astratto per ottenere il campo `model`.

5.7.2.23 LangchainDocument

- **Attributi**:
  - `documentId: string;`
  - `embeddings: List<List<float>>;`
  - `text: List<langchain.Document>.`

5.7.2.24 LangchainEmbeddingModel

- **Metodi**:
  - `embedDocument(documentChunks: List<string>): List<List<float>>`
    Metodo astratto per generare gli embeddings di un insieme di chunks sotto forma di lista di stringhe: per ogni chunk in lista viene generata una lista di float che rappresenta gli embeddings;
  - `getEmbeddingFunction(): langchain.Embedding`
    Metodo astratto per ottenere il campo `model`.

5.7.2.25 OpenAIEmbeddingModel

- **Implementa**: LangchainEmbeddingModel;
- **Attributi**:
  - `embeddingsDimension: int;`
  - `model: langchain.Embedding.`
- **Metodi**:
  - `embedDocument(documentChunks:List<string>): List<List<float>>`
    Implementazione del metodo astratto di LangchainEmbeddingModel per generare gli embeddings di un insieme di chunks di stringhe tramite un modello di embeddings OpenAI: per ogni chunk in lista viene generata una lista di float che rappresenta gli embeddings;
  - `getEmbeddingFunction(): langchain.Embedding`
    Implementazione del metodo astratto per ottenere il campo `model`.

5.7.2.26 PDFTextExtractor

- **Implementa**: TextExtractor;
- **Metodi**:

- `extractText(document:DocumentContent): List<langchain.Document>` Implementazione del metodo astratto di TextExtractor per estrarre il testo da un PDF sotto forma di langchain.Document a partire da un DocumentContent.

5.7.2.27 Status (Enumeration)

- **Valori**:
  - `Concealed;`
  - `Enabled;`
  - `Inconsistent;`
  - `Not Embedded.`

5.7.2.28 TextExtractor

- **Metodi**:
  - `extractText(document:DocumentContent): List<langchain.Document>` Metodo astratto per estrarre il testo sotto forma di langchain.Document a partire da un DocumentContent.

5.7.2.29 VectorStoreChromaDBManager

Vedi (§5.3.2.8) .

5.7.2.30 VectorStoreDocumentOperationResponse

Vedi (§5.3.2.9) .

5.7.2.31 VectorStoreDocumentStatusResponse

- **Attributi**:
  - `documentStatus: string;`
  - `id: string.`

5.7.2.32 VectorStoreManager

Vedi (§5.3.2.10) .

5.7.2.33 VectorStorePineconeManager

Vedi (§5.3.2.11) .

5.8 EnableDocuments

5.8.1 Diagramma delle classi

5.8.2 Lista delle sottocomponenti

5.8.2.1 DocumentId

Vedi (§5.3.2.6) .

5.8.2.2 DocumentOperationResponse

Vedi (§5.3.2.7) .

5.8.2.3 EnableDocumentsController

- **Attributi**:
  - `useCase: EnableDocumentsUseCase .`
- **Metodi**:
  - `enableDocuments(documentsIds:List<string>): List<DocumentOperationResponse>`
    Metodo che si occupa di trasformare le stringhe di id in DocumentId e inoltrare la riabilitazione dei documenti a EnableDocumentsUseCase;

5.8.2.4 EnableDocumentsPort

- **Metodi**:
  - `enableDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo astratto per riabilitare una lista di documenti a partire dal loro DocumentId, dialogando con il vector store. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.8.2.5 EnableDocumentsService

- **Implementa**: EnableDocumentsUseCase;
- **Attributi**:
  - `outPort: EnableDocumentsPort .`
- **Metodi**:
  - `enableDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di EnableDocumentsUseCase per riabilitare una lista di documenti a partire dal loro DocumentId. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.8.2.6 EnableDocumentsUseCase

- **Metodi**:
  - `enableDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`
    Metodo astratto per riabilitare una lista di documenti a partire dal loro DocumentId. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.8.2.7 EnableDocumentsVectorStore

- **Implementa**: EnableDocumentsPort;
- **Attributi**:
  - `vectorStoreManager: VectorStoreManager.`
- **Metodi**:
  - `enableDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di EnableDocumentsPort per riabilitare una lista di documenti a partire dal loro DocumentId, dialogando con il vector store. Ritorna una lista di DocumentOperationResponse, che indica l'esito dell'operazione per ogni documento.

5.8.2.8 VectorStoreChromaDBManager

Vedi (§5.3.2.8) .

5.8.2.9 VectorStoreDocumentOperationResponse

Vedi (§5.3.2.9) .

5.8.2.10 VectorStoreManager

Vedi (§5.3.2.10) .

5.8.2.11 VectorStorePineconeManager

Vedi (§5.3.2.11) .

5.9 GetChatMessages

5.9.1 Diagramma delle classi

5.9.2 Lista delle sottocomponenti

5.9.2.1 Chat

- **Attributi**:
  - `id: ChatId;`
  - `messages: List<Message>;`
  - `title: string.`

5.9.2.2 ChatId

Vedi (§5.1.2.7) .

5.9.2.3 GetChatMessagesController

- **Attributi**:
  - `useCase: GetChatMessagesUseCase.`

- **Metodi**:
  - `getChatMessages(chatId:int): Chat`
    Metodo che ritorna una Chat tramite useCase a partire da un intero rappresentante l'id della chat.

5.9.2.4 GetChatMessagesPort

- **Metodi**:
  - `getChatMessages(chatId:ChatId): Chat`
    Metodo astratto per ottenere una Chat a partire da un ChatId.

5.9.2.5 GetChatMessagesPostgres

- **Implementa**: GetChatMessagesPort;
- **Attributi**:
  - `postgresChatORM: PostgresChatORM.`
- **Metodi**:
  - `getChatMessages(chatId:ChatId): Chat`
    Implementazione del metodo astratto di GetChatMessagesPort per ottenere una Chat a partire da un ChatId.

5.9.2.6 GetChatMessagesService

- **Implementa**: GetChatMessagesUseCase;
- **Attributi**:
  - `outPort: GetChatMessagesPort.`
- **Metodi**:
  - `getChatMessages(chatId:ChatId): Chat`
    Implementazione del metodo astratto di GetChatMessagesUseCase per ottenere una Chat a partire da un ChatId.

5.9.2.7 GetChatMessagesUseCase

- **Metodi**:
  - `getChatMessages(chatId:ChatId): Chat`
    Metodo astratto per ottenere una Chat a partire da un ChatId.

5.9.2.8 Message

Vedi (§5.1.2.9) .

5.9.2.9 MessageSender

Vedi (§5.1.2.11) .

5.9.2.10 PostgresChat

Vedi (§5.1.2.13) .

5.9.2.11 PostgresChatORM

Vedi (§5.1.2.18) .

5.9.2.12 PostgresMessage

Vedi (§5.1.2.15) .

5.10 GetChats

5.10.1 Diagramma delle classi

5.10.2 Lista delle sottocomponenti

5.10.2.1 ChatFilter

- **Attributi**:
  - `searchFilter: string.`

5.10.2.2 ChatId

Vedi (§5.1.2.7) .

5.10.2.3 ChatPreview

- **Attributi**:
  - `id: ChatId;`
  - `lastMessage: Message;`
  - `title: string.`

5.10.2.4 GetChatsController

- **Attributi**:
  - `useCase: GetChatsUseCase.`
- **Metodi**:
  - `getChats(searchFilter: string): List<ChatPreview>`
    Metodo che ritorna una lista di ChatPreview tramite useCase, eventualmente filtrando la ricerca tramite searchFilter.

5.10.2.5 GetChatsPort

- **Metodi**:
  - `getChats(chatFilter:ChatFilter): List<ChatPreview>`
    Metodo astratto per ottenere una lista di ChatPreview, filtrando eventualmente la ricerca tramite chatFilter.

5.10.2.6 GetChatsPostgres

- **Implementa**: GetChatsPort;
- **Attributi**:
  - `postgresChatORM: PostgresChatORM.`
- **Metodi**:
  - `getChats(chatFilter:ChatFilter): List<ChatPreview>`
    Implementazione del metodo astratto di GetChatPort per ottenere una lista di ChatPreview tramite postgresChatORM, filtrando eventualmente la ricerca tramite chatFilter.

5.10.2.7 GetChatsService

- **Implementa**: GetChatsUseCase;
- **Attributi**:
  - `outPort: GetChatsPort.`
- **Metodi**:
  - `getChats(chatFilter:ChatFilter): List<ChatPreview>`
    Implementazione del metodo astratto di GetChatsUseCase per ottenere una lista di ChatPreview tramite outPort, filtrando eventualmente la ricerca tramite chatFilter.

5.10.2.8 GetChatsUseCase

- **Metodi**:
  - `getChats(chatFilter:ChatFilter): List<ChatPreview>`
    Metodo astratto per ottenere una lista di ChatPreview, filtrando eventualmente la ricerca tramite chatFilter.

5.10.2.9 Message

Vedi (§5.1.2.9) .

5.10.2.10 MessageSender

Vedi (§5.1.2.11) .

5.10.2.11 PostgresChatORM

Vedi (§5.1.2.18) .

5.10.2.12 PostgresChatPreview

- **Attributi**:
  - `id: int;`
  - `title: string;`
  - `lastMessage: PostgresMessage.`

- **Metodi**:
  -
  - `toChatPreviewFrom()`: `ChatPreview` Metodo per ottenere una ChatPreview a partire dal PostgresChatPreview.

5.10.2.13 PostgresMessage

Vedi (§5.1.2.15) .

5.10.2.14 PostgresMessageSenderType

Vedi (§5.1.2.16) .

5.11 GetConfiguration

5.11.1 Diagramma delle classi

5.11.2 Lista delle sottocomponenti

5.11.2.1 Configuration

- **Attributi**:
  - `embeddingModel: EmbeddingModelConfiguration;`
  - `documentStore: DocumentStoreConfiguration;`
  - `LLMModel: LLMModelConfiguration;`
  - `vectorStore: VectorStoreConfiguration.`

5.11.2.2 DocumentStoreConfiguration

- **Attributi**:
  - `costIndicatore: string;`
  - `description: string;`
  - `name: DocumentStoreType;`
  - `organization: string;`
  - `type: string.`

5.11.2.3 DocumentStoreType (Enumeration)

- **Valori**:
  - `AWS.`

5.11.2.4 EmbeddingModelConfiguration

- **Attributi**:
  - `costIndicator: string;`
  - `description: string;`

- `name: EmbeddingModelType;`
- `organization: string;`
- `type: string.`

5.11.2.5 EmbeddingModelType (Enumeration)

- **Valori**:
  - `HUGGINGFACE;`
  - `OPENAI.`

5.11.2.6 GetConfigurationController

- **Attributi**:
  - `useCase: GetConfigurationUseCase.`
- **Metodi**:
  - `getConfiguration(): Configuration`
    Metodo che si occupa di ritornare la configurazione di vector store, document store, embedding e LLM model tramite oggetto Configuration.

5.11.2.7 GetConfigurationPort

- **Metodi**:
  - `getConfiguration(): Configuration`
    Metodo atratto che ritorna la configurazione di vector store, document store, embedding e LLM model tramite oggetto Configuration.

5.11.2.8 GetConfigurationPostgres

- **Implementa**: GetConfigurationPort;
- **Attributi**:
  - `postgresConfigurationORM: PostgresConfigurationORM.`
- **Metodi**:
  - `getConfiguration(): Configuration`
    Implementazione del metodo astratto di GetConfigurationPort per ottenere la configurazione di vector store, document store, embedding e LLM model ritornando un oggetto Configuration.

5.11.2.9 GetConfigurationService

- **Implementa**: GetConfigurationUseCase;
- **Attributi**:
  - `outPort: GetConfigurationPort.`
- **Metodi**:

- `getConfiguration(): Configuration`
  Implementazione del metodo astratto di GetConfigurationUseCase per la configurazione di vector store, document store, embedding e LLM model ritornando un oggetto Configuration.

5.11.2.10 GetConfigurationUseCase

- **Metodi**:
  - `getConfiguration(): Configuration`
    Metodo atratto che ritorna la configurazione di vector store, document store, embedding e LLM model tramite oggetto Configuration.

5.11.2.11 LLMModelConfiguration

- **Attributi**:
  - `costIndicator: string;`
  - `description: string;`
  - `name: LLMModelType;`
  - `organization: string;`
  - `type: string.`

5.11.2.12 LLMModelType (Enumeration)

- **Valori**:
  - `HUGGINGFACE;`
  - `OPENAI.`

5.11.2.13 PostgresConfiguration

Vedi (§5.5.2.2) .

5.11.2.14 PostgresConfigurationORM

Vedi (§5.2.2.8) .

5.11.2.15 PostgresDocumentStoreConfiguration

Vedi (§5.5.2.4) .

5.11.2.16 PostgresDocumentStoreType

Vedi (§5.5.2.5) .

5.11.2.17 PostgresEmbeddingModelConfiguration

Vedi (§5.5.2.6) .

5.11.2.18 PostgresEmbeddingModelType

Vedi (§5.5.2.7) .

5.11.2.19 PostgresLLMModelConfiguration

Vedi (§5.5.2.8) .

5.11.2.20 PostgresLLMModelType

Vedi (§5.5.2.9) .

5.11.2.21 PostgresVectorStoreConfiguration

Vedi (§5.5.2.10) .

5.11.2.22 PostgresVectorStoreType

Vedi (§5.5.2.11) .

5.11.2.23 VectorStoreConfiguration

- **Attributi**:
  - `costIndicator: string;`
  - `description: string;`
  - `name: VectorStoreType;`
  - `organization: string;`
  - `type: string.`

5.11.2.24 VectorStoreType (Enumeration)

- **Valori**:
  - `CHROMA DB;`
  - `PINECONE.`

5.12 GetConfigurationOptions

5.12.1 Diagramma delle classi

5.12.2 Lista delle sottocomponenti

5.12.2.1 ConfigurationOptions

- **Attributi**:
  - `documentStoreOptions: List<DocumentStoreConfiguration>;`
  - `embeddingModelOptions: List<EmbeddingModelConfiguration>;`
  - `LLMModelOptions: List<LLMModelConfiguration>;`
  - `vectorStoreOptions: List<VectorStoreConfiguration>.`

5.12.2.2 DocumentStoreConfiguration

Vedi (§5.11.2.2) .

5.12.2.3 DocumentStoreType

Vedi (§5.11.2.3) .

5.12.2.4 EmbeddingModelConfiguration

Vedi (§5.11.2.4) .

5.12.2.5 EmbeddingModelType

Vedi (§5.11.2.5) .

5.12.2.6 GetConfigurationOptions

- **Attributi**:
  - `useCase: GetConfigurationOptionsUseCase .`
- **Metodi**:
  - `getConfigurationOptions(): ConfigurationOptions`
    Metodo che si occupa di ritornare le opzioni disponibili per le configurazioni di vector store, document store, embedding model e LLM model.

5.12.2.7 GetConfigurationOptionsPort

- **Metodi**:
  - `getConfigurationOptions(): ConfigurationOptions`
    Metodo atratto che si occupa di ritornare le opzioni disponibili per le configurazioni di vector store, document store, embedding model e LLM model.

5.12.2.8 GetConfigurationOptionsPostgres

- **Implementa**: GetConfigurationOptionsPort;
- **Attributi**:
  - `postgresConfigurationORM: PostgresConfigurationORM .`
- **Metodi**:
  - `getConfigurationOptions(): ConfigurationOptions`
    Implementazione del metodo astratto di GetConfigurationOptionsPort che ritorna le opzioni disponibili per le configurazioni di vector store, document store, embedding model e LLM model.

5.12.2.9 GetConfigurationOptionsService

- **Implementa**: GetConfigurationOptionsUseCase;
- **Attributi**:
  - `outPort: GetConfigurationOptionsPort .`
- **Metodi**:

- `getConfigurationOptions(): ConfigurationOptions`
  Implementazione del metodo astratto di GetConfigurationOptionsUseCase che ritorna le opzioni disponibili per le configurazioni di vector store, document store, embedding model e LLM model.

5.12.2.10 GetConfigurationOptionsUseCase

- **Metodi**:
  - `getConfigurationOptions(): ConfigurationOptions`
    Metodo astratto per ritornare le opzioni disponibili per le configurazioni di vector store, document store, embedding model e LLM model.

5.12.2.11 LLMModelConfiguration

Vedi (§5.11.2.11)

5.12.2.12 LLMModelType

Vedi (§5.11.2.12) .

5.12.2.13 PostgresConfigurationORM

Vedi (§5.2.2.8) .

5.12.2.14 PostgresDocumentStoreConfiguration

Vedi (§5.5.2.4) .

5.12.2.15 PostgresDocumentStoreType

Vedi (§5.5.2.5) .

5.12.2.16 PostgresEmbeddingModelConfiguration

Vedi (§5.5.2.6) .

5.12.2.17 PostgresEmbeddingModelType

Vedi (§5.5.2.7) .

5.12.2.18 PostgresLLMModelConfiguration

Vedi (§5.5.2.8) .

5.12.2.19 PostgresLLMModelType

Vedi (§5.5.2.9) .

5.12.2.20 PostgresVectorStoreConfiguration

Vedi (§5.5.2.10) .

5.12.2.21 PostgresVectorStoreType

Vedi (§5.5.2.11) .

5.12.2.22 VectorStoreConfiguration

Vedi (§5.11.2.23) .

5.12.2.23 VectorStoreType

Vedi (§5.11.2.24) .

5.13 GetDocuments

5.13.1 Diagramma delle classi

5.13.2 Lista delle sottocomponenti

5.13.2.1 AWSDocumentMetadata

- **Attributi**:
  - `id: string;`
  - `size: float;`
  - `uploadTime: datetime.`
- **Metodi**:
  - `toDocumentMetadataFrom(): DocumentMetadata`
    Metodo che trasforma un AWSDocumentMetadata della persistence in un DocumentMetadata della business logic.

5.13.2.2 AWSS3Manager

Vedi (§5.6.2.2) .

5.13.2.3 DocumentFilter

- **Attributi**:
  - `searchFilter: string.`

5.13.2.4 DocumentId

Vedi (§5.3.2.6) .

5.13.2.5 DocumentMetadata

- **Attributi**:
  - `id: DocumentId;`
  - `size: float;`
  - `type: DocumentType;`
  - `uploadTime: datetime.`

5.13.2.6 DocumentStatus

Vedi (§5.7.2.6) .

5.13.2.7 DocumentType (Enumeration)

- **Valori**:
  - `DOCX`;
  - `PDF`.

5.13.2.8 ElaborationException

- **Attributi**:
  - `message: string`.

5.13.2.9 GetDocumentsController

- **Attributi**:
  - `useCase: GetDocumentsUseCase`.
- **Metodi**:
  - `getDocuments(searchFilter:string = ""): List<LightDocument>`
    Metodo che si occupa di inoltrare la richiesta di ricerca di documenti a GetDocumentsUseCase, trasformando la stringa filter in oggetto di business SearchFilter.

5.13.2.10 GetDocumentsFacadeService

- **Implementa**: GetDocumentsUseCase;
- **Attributi**:
  - `getDocumentMetadatas: GetDocumentMetadatas`;
  - `getDocumentsStatus: GetDocumentsStatus`.
- **Metodi**:
  - `getDocuments(documentFilter:DocumentFilter): List<LightDocument>`
    Implementazione del metodo astratto di GetDocumentsUseCase per ricercare documenti a partire da un filtro di ricerca, tramite l'utilizzo di un pattern facade.
    Le operazioni all'interno della facade vengono svolte nel seguente ordine:
    1. Recupero dei documenti che soddisfano la ricerca dal sistema di archiviazione;
    2. Recupero degli status dei documenti ritornati dal passo precedente;
    3. Creazione di oggetti LightDocument a partire dai risultati dei passi precedenti.

5.13.2.11 GetDocumentsListAWSS3

- **Implementa**: GetDocumentsMetadataPort;
- **Attributi**:
  - `awsS3Manager: AWSS3Manager`.

- **Metodi**:
  - `getDocumentsMetadata(documentFilter:DocumentFilter): List<DocumentMetadata>`
    Implementazione del metodo astratto di GetDocumentsMetadataPort per ricercare documenti tramite outPort nel sistema di archiviazione a partire da un filtro.

5.13.2.12 GetDocumentsMetadata

- **Attributi**:
  - `outPort: GetDocumentsMetadataPort .`
- **Metodi**:
  - `getDocumentsMetadata(documentFilter:DocumentFilter): List<DocumentMetadata>`
    Metodo per ricercare documenti tramite outPort nel sistema di archiviazione a partire da un filtro.

5.13.2.13 GetDocumentsMetadataPort

- **Metodi**:
  - `getDocumentsMetadata(documentFilter:DocumentFilter): List<DocumentMetadata>`
    Metodo astratto per ricercare documenti nel sistema di archiviazione a partire da un filtro di ricerca.

5.13.2.14 GetDocumentsStatus

Vedi (§5.7.2.19) .

5.13.2.15 GetDocumentsStatusPort

Vedi (§5.7.2.20) .

5.13.2.16 GetDocumentsStatusVectorStore

Vedi (§5.7.2.21) .

5.13.2.17 GetDocumentsUseCase

- **Metodi**:
  - `getDocuments(documentFilter:DocumentFilter): List<LightDocument>`
    Metodo astratto per ricercare documenti a partire da un filtro di ricerca.

5.13.2.18 LightDocument

- **Attributi**:
  - `metadata: DocumentMetadata;`
  - `status: DocumentStatus .`

5.13.2.19 Status (Enumeration)

Vedi (§5.7.2.27) .

5.13.2.20 VectorStoreChromaDBManager

Vedi (§5.3.2.8) .

5.13.2.21 VectorStoreDocumentStatusResponse

Vedi (§5.7.2.31) .

5.13.2.22 VectorStoreManager

Vedi (§5.3.2.10) .

5.13.2.23 VectorStorePineconeManager

Vedi (§5.3.2.11) .

5.14 RenameChat

5.14.1 Diagramma delle classi

5.14.2 Lista delle sottocomponenti

5.14.2.1 ChatId

Vedi (§5.1.2.7) .

5.14.2.2 ChatOperationResponse

Vedi (§5.1.2.8) .

5.14.2.3 RenameChatController

- **Attributi**:
  - `useCase: RenameChatUseCase .`
- **Metodi**:
  - `renameChat(chatId:int, title:string): ChatOperationResponse`
    Metodo che rinomina tramite useCase una chat identificate da un intero, ritornando un ChatOperationResponse.

5.14.2.4 RenameChatPort

- **Metodi**:
  - `renameChat(chatId:ChatId, title:string): ChatOperationResponse`
    Metodo astratto per rinominare con title una chat identificata da un ChatId, ritornando un ChatOperationResponse rappresentante l'esito dell'operazione.

5.14.2.5 RenameChatPostgres

- **Implementa**: RenameChatPort;
- **Attributi**:
  - `outPort: PostgresChatORM.`
- **Metodi**:
  - `renameChat(chatId:ChatId, title:string): ChatOperationResponse` Implementazione del metodo astratto di RenameChatPort per rinominare con title una chat identificata da un ChatId, ritornando un ChatOperationResponse rappresentante l'esito dell'operazione.

5.14.2.6 RenameChatService

- **Implementa**: RenameChatUseCase;
- **Attributi**:
  - `outPort: RenameChatPort.`
- **Metodi**:
  - `renameChat(chatId:ChatId, title:string): ChatOperationResponse` Implementazione del metodo astratto di RenameChatUseCase per rinominare con title tramite outPort una chat identificata da un ChatId, ritornando un ChatOperationResponse rappresentante l'esito dell'operazione.

5.14.2.7 RenameChatUseCase

- **Metodi**:
  - `renameChat(chatId:ChatId, title:string): ChatOperationResponse` Metodo astratto per che rinominare con title una chat identificata da un ChatId, ritornando un ChatOperationResponse rappresentante l'esito dell'operazione.

5.14.2.8 PostgresChatOperationResponse

Vedi (§5.1.2.14) .

5.14.2.9 PostgresChatORM

Vedi (§5.1.2.18) .

5.15 SetConfiguration

5.15.1 Diagramma delle classi

5.15.2 Lista delle sottocomponenti

5.15.2.1 ConfigurationOperationResponse

Vedi (§5.2.2.6) .

5.15.2.2 PostgresConfigurationOperationResponse

Vedi (§5.2.2.7) .

5.15.2.3 PostgresConfigurationORM

Vedi (§5.2.2.8) .

5.15.2.4 SetConfigurationController

- **Attributi**: `useCase: SetConfigurationUseCase;`
- **Metodi**:
  - `setConfiguration(LLModel:string, DocumentStore:string, VectorStore:string EmbeddingModel:string): ConfigurationOperationResponse`
    Metodo che si occupa di inizializzare la configurazione del sistema al primo avvio, passando stringhe che rappresentano rispettivamente LLModel, document store, vector store ed embedding model. Ritorna un ConfigurationOperationResponse rappresentante l'esito dell'operazione.

5.15.2.5 SetConfigurationPort

- **Metodi**:
  - `setConfiguration(LLModel:LLMModelType, DocumentStore:DocumentStoreType, VectorStore:VectorStoreType, EmbeddingModel:EmbeddingModelType): ConfigurationOperationResponse`
    Metodo astratto per inizializzare la configurazione del sistema al primo avvio passando LLMModelType, DocumentStoreType, VectorStoreType ed EmbeddingModelType. Ritorna un ConfigurationOperationResponse.

5.15.2.6 SetConfigurationPostgres

- **Implementa**: SetConfigurationPort;
- **Attributi**:
  - `postgresConfigurationORM: PostgresConfigurationORM.`
- **Metodi**:
  - `setConfiguration(LLModel:LLMModelType, DocumentStore:DocumentStoreType, VectorStore:VectorStoreType, EmbeddingModel:EmbeddingModelType): ConfigurationOperationResponse`
    Implementazione del metodo astratto di SetConfigurationPort inizializzare la configurazione del sistema al primo avvio passando LLMModelType, DocumentStoreType, VectorStoreType ed EmbeddingModelType. Ritorna l'esito dell'operazione con un ConfigurationOperationResponse;
  - `toPostgresDocumentStoreTypeFrom(DocumentStore:DocumentStoreType): PostgresDocument`
    Metodo per convertire un DocumentStoreType in PostgresDocumentStoreType;
  - `toPostgresEmbeddingModelFrom(EmbeddingModel: EmbeddingModelType): PostgresEmbeddi`
    Metodo per convertire un EmbeddingModelType in PostgresEmbeddingModelType;

- `toPostgresLLMModelTypeFrom(LLMModel:LLMModelType): PostgresLLMModelType`
  Metodo per convertire un LLLMModelType in PostgresLLMModelType;
- `toPostgresVectorStoreTypeFrom(VectorStore:VectorStoreType): PostgresVectorStoreTyp`
  Metodo per convertire un VectorStoreType in PostgresVectorStoreType.

5.15.2.7 SetConfigurationService

- **Implementa**: SetConfigurationUseCase;
- **Attributi**:
  - `outPort: SetConfigurationPort.`
- **Metodi**:
  - `setConfiguration(LLModel:LLMModelType, DocumentStore:DocumentStoreType, VectorStore:VectorStoreType, EmbeddingModel:EmbeddingModelType): ConfigurationOperationResponse`
    Implementazione del metodo astratto di SetConfigurationUseCase per inizializzare la configurazione del sistema al primo avvio tramite outPort. Ritorna l'esito dell'operazione con un ConfigurationOperationResponse.

5.15.2.8 SetConfigurationUseCase

- **Metodi**:
  - `setConfiguration(LLModel:LLMModelType, DocumentStore:DocumentStoreType, VectorStore:VectorStoreType, EmbeddingModel:EmbeddingModelType): ConfigurationOperationResponse`
    Metodo astratto per inizializzare la configurazione del sistema al primo avvio, ritornando l'esito come ConfigurationOperationResponse.

5.16 UploadDocuments

5.16.1 Diagramma delle classi

5.16.2 Lista delle sottocomponenti

5.16.2.1 AWSDocument

Vedi (§5.7.2.1) .

5.16.2.2 AWSDocumentOperationResponse

Vedi (§5.6.2.1) .

5.16.2.3 AWSS3Manager

Vedi (§5.6.2.2) .

5.16.2.4 Chunkerizer

Vedi (§5.7.2.3) .

5.16.2.5 Document

- **Attributi**:
  - `plainDocument: PlainDocument;`
  - `documentStatus: DocumentStatus.`

5.16.2.6 DocumentContent

- **Attributi**:
  - `content: bytes.`

5.16.2.7 DocumentId

Vedi (§5.3.2.6) .

5.16.2.8 DocumentMetadata

Vedi (§5.13.2.5) .

5.16.2.9 DocumentOperationResponse

Vedi (§5.3.2.7) .

5.16.2.10 DocumentStatus

Vedi (§5.7.2.6) .

5.16.2.11 DocumentType (Enumeration)

Vedi (§5.13.2.7) .

5.16.2.12 DocumentsUploader

- **Attributi**:
  - `outPort: DocumentsUploaderPort.`
- **Metodi**:
  - `uploadDocuments(documents:List<Document>, forceUpload:boolean): List<DocumentOperationResponse>`
    Metodo per effettuare il caricamento di una lista di Document tramite outPort sul sistema di archiviazione, con flag per forzare il caricamento di documenti già presenti nel sistema attraverso sostituzione, ritornando infine una lista di DocumentOperationResponse.

5.16.2.13 DocumentUploaderAWSS3

- **Implementa**: DocumentsUploaderPort;
- **Attributi**:
  - `awsS3Manager: AWSS3Manager.`
- **Metodi**:

- `uploadDocuments(documents:List<Document>, forceUpload:boolean): List<DocumentOperationResponse>`
  Implementazione del metodo astratto di DocumentsUploaderPort per il caricamento di una lista di Document nel sistema di archiviazione tramite awsS3Manager, con flag per forzare il caricamento di documenti già presenti nel sistema attraverso sostituzione, ritornando infine una lista di DocumentOperationResponse;
- `toAWSDocumentFrom(document:Document): AWSDocument`
  Metodo che trasforma un Document in un AWSDocument.

5.16.2.14 DocumentsUploaderPort

- **Metodi**:
  - `uploadDocuments(documents:List<Document>, forceUpload:boolean): List<DocumentOperationResponse>`
    Metodo astratto per effettuare il caricamento di una lista di Document in un sistema di archiviazione documenti, con flag per forzare il caricamento di documenti già presenti nel sistema attraverso sostituzione.

5.16.2.15 DOCXTextExtractor

Vedi (§5.7.2.7) .

5.16.2.16 EmbeddingsCreator

Vedi (§5.7.2.8) .

5.16.2.17 EmbeddingsUploader

Vedi (§5.7.2.9) .

5.16.2.18 EmbeddingsUploaderFacadeLangchain

Vedi (§5.7.2.10) .

5.16.2.19 EmbeddingsUploaderPort

Vedi (§5.7.2.11) .

5.16.2.20 EmbeddingsUploaderVectorStore

Vedi (§5.7.2.12) .

5.16.2.21 HuggingFaceEmbeddingModel

Vedi (§5.7.2.22) .

5.16.2.22 LangchainDocument

Vedi (§5.7.2.23) .

5.16.2.23 LangchainEmbeddingModel

Vedi (§5.7.2.24) .

5.16.2.24 NewDocument

- **Attributi**:
  - `content: bytes;`
  - `documentId: string;`
  - `size: float;`
  - `type: string.`
- **Metodi**:
  - `toDocument(): Document`
    Metodo che trasforma un NewDocument della presentation logic in un Document della business logic.

5.16.2.25 OpenAIEmbeddingModel

Vedi (§5.7.2.25) .

5.16.2.26 PDFTextExtractor

Vedi (§5.7.2.26) .

5.16.2.27 PlainDocument

- **Attributi**:
  - `content: DocumentContent;`
  - `metadata: DocumentMetadata.`

5.16.2.28 Status (Enumeration)

Vedi (§5.7.2.27) .

5.16.2.29 TextExtractor

Vedi (§5.7.2.28) .

5.16.2.30 UploadDocumentsController

- **Attributi**:
  - `useCase: UploadDocumentsUseCase.`
- **Metodi**:
  - `uploadDocuments(newDocuments:List<NewDocument>): List<DocumentOperationResponse>`
    Metodo che effettua l'upload di una lista di NewDocument tramite useCase, ritornando una lista di DocumentOperationResponse.

5.16.2.31 UploadDocumentsService

- **Implementa**: UploadDocumentsUseCase;
- **Attributi**:
  - `documentUploader: DocumentsUploader;`
  - `embeddingsUploader: EmbeddingsUploader.`
- **Metodi**:
  - `uploadDocuments(documents:List<Document>, forceUpload:boolean): List<DocumentOperationResponse>`
    Implementazione del metodo astratto di UploadDocumentsUseCase, per il caricamento di una lista di Documents tramite documentsUploader nel sistema di archiviazione documenti, con flag per forzare il caricamento di documenti già presenti nel sistema attraverso sostituzione, occupandosi inoltre della generazione e caricamento dei relativi embeddings dei documenti tramite embeddingsUploader in un vector store, ritornando una lista di DocumentOperationResponse che tenga conto di entrambe le risposte delle due sotto-operazioni precedentemente descritte ed eseguite.

5.16.2.32 UploadDocumentsUseCase

- **Metodi**:
  - `uploadDocuments(documents:List<Document>, forceUpload:boolean): List<DocumentOperationResponse>`
    Metodo astratto per caricare una lista di Document nel sistema, con possibilità di forzare un caricamento di un documento già presente tramite una flag, ritornando una lista di DocumentOperationResponse.

5.16.2.33 VectorStoreChromaDBManager

Vedi (§5.3.2.8) .

5.16.2.34 VectorStoreDocumentOperationResponse

Vedi (§5.3.2.9) .

5.16.2.35 VectorStoreManager

Vedi (§5.3.2.10) .

5.16.2.36 VectorStorePineconeManager

Vedi (§5.3.2.11) .

5.17 ViewDocumentContent

5.17.1 Diagramma delle classi

5.17.2 Lista delle sottocomponenti

5.17.2.1 AWSDocument

Vedi (§5.7.2.1) .

5.17.2.2 AWSS3Manager

Vedi (§5.6.2.2) .

5.17.2.3 Document

Vedi (§5.16.2.5) .

5.17.2.4 DocumentContent

Vedi (§5.16.2.6) .

5.17.2.5 DocumentId

Vedi (§5.3.2.6) .

5.17.2.6 DocumentMetadata

Vedi (§5.13.2.5) .

5.17.2.7 DocumentStatus

Vedi (§5.7.2.6) .

5.17.2.8 DocumentType (Enumeration)

Vedi (§5.13.2.7) .

5.17.2.9 GetDocumentController

- **Attributi**:
  - `useCase: GetDocumentUseCase .`
- **Metodi**:
  - `getDocument(documentId:String): Document`
    Metodo che si occupa di trasformare la stringa id in DocumentId e inoltrare la richiesta di recupero del documento a GetDocumentsUseCase.

5.17.2.10 GetDocumentFacadeService

- **Implementa**: GetDocumentUseCase;
- **Attributi**:
  - `getDocumentsContent: GetDocumentsContent ;`
  - `getDocumentsStatus: GetDocumentsStatus .`

- **Metodi**:
  - `getDocument(documentId:DocumentId): Document`
    Implementazione del metodo astratto di GetDocumentUseCase. Le operazioni all'interno della facade vengono svolte nel seguente ordine:
    1. Recupero dello status del documento dal vector store;
    2. Recupero del documento e il suo contenuto dal sistema di archiviazione;
    3. Costruzione di un oggetto Document a partire dagli output dei passaggi precedenti.

5.17.2.11 GetDocumentsContent

Vedi (§5.7.2.16) .

5.17.2.12 GetDocumentsContentAWSS3

Vedi (§5.7.2.17) .

5.17.2.13 GetDocumentsContentPort

Vedi (§5.7.2.18) .

5.17.2.14 GetDocumentsStatus

Vedi (§5.7.2.19) .

5.17.2.15 GetDocumentsStatusPort

Vedi (§5.7.2.20) .

5.17.2.16 GetDocumentsStatusVectorStore

Vedi (§5.7.2.21) .

5.17.2.17 GetDocumentUseCase

- **Metodi**:
  - `getDocument(documentId:DocumentId): Document`
    Metodo astratto per recuperare tutte le informazioni di un documento, compreso il suo contenuto, a partire dal suo id.

5.17.2.18 PlainDocument

Vedi (§5.16.2.27) .

5.17.2.19 Status (Enumeration)

Vedi (§5.7.2.27) .

5.17.2.20 VectorStoreChromaDBManager

Vedi (§5.3.2.8) .

5.17.2.21 VectorStoreDocumentStatusResponse

Vedi (§5.7.2.31) .

5.17.2.22 VectorStoreManager

Vedi (§5.3.2.10) .

5.17.2.23 VectorStorePineconeManager

Vedi (§5.3.2.11) .