



SWEetCode

Specifica tecnica

Componenti del gruppo

Bresolin G.

Campese M.

Ciriolo I.

Dugo A.

Feltrin E.

Michelon R.

Orlandi G.



Registro delle versioni

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
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.
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.

Continua nella pagina successiva



Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
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.
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.

Continua nella pagina successiva



Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
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

1	Introduzione	13
1.1	Obiettivo del documento	13
1.2	Glossario	13
1.3	Riferimenti	13
1.3.1	Riferimenti normativi	13
1.3.2	Riferimenti informativi	13
2	Tecnologie utilizzate	16
2.1	Flask	16
2.1.1	Python	17
2.2	Next.js	17
2.2.1	Typescript	18
2.3	Docker	19
2.4	Langchain	19
2.4.1	Pinecone	20
2.4.2	ChromaDB	21
2.4.3	OpenAI	21
2.4.4	HuggingFace	22
2.5	AWS S3	23
2.6	Postgres	23
3	Architettura di sistema	25
3.1	Modello architetturale	25
3.2	Descrizione delle componenti	25
3.2.1	Frontend	25
3.2.2	Backend	25
3.3	Assemblaggio delle componenti	26
3.4	Struttura del sistema	26
3.4.1	Frontend	26
3.4.2	Backend	26
4	Architettura delle componenti	27
4.1	Frontend	27
4.2	Backend	27
4.2.1	AskChatbot	27
4.2.1.1	Descrizione	27
4.2.1.2	Esiti possibili	27
4.2.1.3	Tracciamento dei requisiti	27
4.2.1.4	Lista sottocomponenti	27
4.2.2	ConcealDocuments	28
4.2.2.1	Descrizione	28
4.2.2.2	Esiti possibili	29
4.2.2.3	Tracciamento dei requisiti	29
4.2.2.4	Lista sottocomponenti	29
4.2.3	DeleteChats	30
4.2.3.1	Descrizione	30
4.2.3.2	Esiti possibili	30



4.2.3.3	Tracciamento dei requisiti	30
4.2.3.4	Lista sottocomponenti	30
4.2.4	DeleteDocuments	31
4.2.4.1	Descrizione	31
4.2.4.2	Esiti possibili	31
4.2.4.3	Tracciamento dei requisiti	31
4.2.4.4	Lista sottocomponenti	31
4.2.5	EmbedDocuments	32
4.2.5.1	Descrizione	32
4.2.5.2	Esiti possibili	32
4.2.5.3	Tracciamento dei requisiti	32
4.2.5.4	Lista sottocomponenti	33
4.2.6	EnableDocuments	34
4.2.6.1	Descrizione	34
4.2.6.2	Esiti possibili	34
4.2.6.3	Tracciamento dei requisiti	34
4.2.6.4	Lista sottocomponenti	34
4.2.7	GetChatMessages	35
4.2.7.1	Descrizione	35
4.2.7.2	Esiti possibili	35
4.2.7.3	Tracciamento dei requisiti	35
4.2.7.4	Lista sottocomponenti	36
4.2.8	GetChats	36
4.2.8.1	Descrizione	36
4.2.8.2	Esiti possibili	36
4.2.8.3	Tracciamento dei requisiti	37
4.2.8.4	Lista sottocomponenti	37
4.2.9	GetDocuments	37
4.2.9.1	Descrizione	37
4.2.9.2	Esiti possibili	38
4.2.9.3	Tracciamento dei requisiti	38
4.2.9.4	Lista sottocomponenti	38
4.2.10	RenameChat	39
4.2.10.1	Descrizione	39
4.2.10.2	Esiti possibili	39
4.2.10.3	Tracciamento dei requisiti	39
4.2.10.4	Lista sottocomponenti	40
4.2.11	UploadDocuments	40
4.2.11.1	Descrizione	40
4.2.11.2	Esiti possibili	40
4.2.11.3	Tracciamento dei requisiti	40
4.2.11.4	Lista sottocomponenti	41
4.2.12	ViewDocumentContent	42
4.2.12.1	Descrizione	42
4.2.12.2	Esiti possibili	42
4.2.12.3	Tracciamento dei requisiti	42
4.2.12.4	Lista sottocomponenti	43
4.3	Database	43
4.4	Libreria per la persistenza	43



5	Progettazione di dettaglio	44
5.1	AskChatbot	44
5.1.1	Diagramma delle classi	44
5.1.2	Lista delle sottocomponenti	44
5.1.2.1	AskChatbotController	44
5.1.2.2	ChatId	44
5.1.2.3	Message	44
5.1.2.4	MessageSender (Enumeration)	44
5.1.2.5	AskChatbotUseCase	44
5.1.2.6	AskChatbotService	45
5.1.2.7	AskChatbotPort	45
5.1.2.8	AskChatbotLangchain	45
5.1.2.9	ChatbotLangchain	46
5.1.2.10	LangchainLLM	46
5.1.2.11	OpenAILLM	46
5.1.2.12	HuggingFaceLLM	46
5.1.2.13	LangchainVectorStore	46
5.1.2.14	PineconeVectorStore	46
5.1.2.15	ChromaDBVectorStore	47
5.1.2.16	LangchainEmbeddingModel	47
5.1.2.17	OpenAIEmbeddingModel	47
5.1.2.18	HuggingFaceEmbeddingModel	47
5.1.2.19	ChatHistoryManager	47
5.1.2.20	PostgresORM	48
5.1.2.21	PostgresChat	48
5.1.2.22	PostgresMessage	48
5.2	ConcealDocuments	49
5.2.1	Diagramma delle classi	49
5.2.2	Lista delle sottocomponenti	49
5.2.2.1	ConcealDocumentsController	49
5.2.2.2	ConcealDocumentsUseCase	49
5.2.2.3	ConcealDocumentsService	49
5.2.2.4	ConcealDocumentsPort	49
5.2.2.5	ConcealDocumentsVectorStore	50
5.2.2.6	VectorStoreManager	50
5.2.2.7	VectorStorePineconeManager	50
5.2.2.8	VectorStoreChromaDBManager	51
5.2.2.9	VectorStoreDocumentOperationResponse	52
5.3	DeleteChats	52
5.3.1	Diagramma delle classi	52
5.3.2	Lista delle sottocomponenti	52
5.3.2.1	ChatId	52
5.3.2.2	DeleteChatsController	52
5.3.2.3	DeleteChatsUseCase	53
5.3.2.4	DeleteChatsService	53
5.3.2.5	DeleteChatsPort	53
5.3.2.6	DeleteChatsPostgres	53
5.3.2.7	ChatOperationResponse	53
5.3.2.8	PostgresORM	54



5.4	DeleteDocuments	54
5.4.1	Diagramma delle classi	54
5.4.2	Lista delle sottocomponenti	54
5.4.2.1	DeleteDocumentsController	54
5.4.2.2	DeleteDocumentsUseCase	54
5.4.2.3	DeleteDocumentsService	54
5.4.2.4	DeleteDocuments	54
5.4.2.5	DeleteDocumentsEmbeddings	55
5.4.2.6	DeleteDocumentsPort	55
5.4.2.7	DeleteEmbeddingsPort	55
5.4.2.8	DocumentOperationResponse	55
5.4.2.9	DeleteDocumentsAWSS3	55
5.4.2.10	AWSS3Manager	56
5.4.2.11	AWSDocumentOperationResponse	56
5.4.2.12	DeleteEmbeddingsVectorStore	56
5.4.2.13	VectorStoreManager	56
5.4.2.14	VectorStorePineconeManager	57
5.4.2.15	VectorStoreChromaDBManager	57
5.4.2.16	VectorStoreDocumentOperationResponse	57
5.5	EmbedDocuments	57
5.5.1	Diagramma delle classi	57
5.5.2	Lista delle sottocomponenti	57
5.5.2.1	EmbedDocumentsController	57
5.5.2.2	EmbedDocumentsUseCase	57
5.5.2.3	EmbedDocumentsService	57
5.5.2.4	GetDocuments	58
5.5.2.5	EmbeddingsUploader	58
5.5.2.6	GetDocumentsPort	58
5.5.2.7	EmbeddingsUploaderPort	58
5.5.2.8	GetDocumentsAWSS3	59
5.5.2.9	AWSS3Manager	59
5.5.2.10	AWSDocument	59
5.5.2.11	EmbeddingsUploaderPort	59
5.5.2.12	DocumentOperationResponse	59
5.5.2.13	LangchainDocument	59
5.5.2.14	DocumentToText	59
5.5.2.15	TextExtractor	60
5.5.2.16	PDFTextExtractor	60
5.5.2.17	DOCXTextExtractor	60
5.5.2.18	Chunkerizer	60
5.5.2.19	EmbeddingsCreator	60
5.5.2.20	LangchainEmbeddingModel	60
5.5.2.21	OpenAIEmbeddingModel	60
5.5.2.22	HuggingFaceEmbeddingModel	61
5.5.2.23	EmbeddingsUploaderVectorStore	61
5.5.2.24	VectorStoreManager	61
5.5.2.25	VectorStorePineconeManager	61
5.5.2.26	VectorStoreChromaDBManager	61
5.5.2.27	VectorStoreDocumentOperationResponse	61



5.6	EnableDocuments	61
5.6.1	Diagramma delle classi	61
5.6.2	Lista delle sottocomponenti	61
5.6.2.1	EnableDocumentsController	61
5.6.2.2	EnableDocumentsUseCase	62
5.6.2.3	EnableDocumentsService	62
5.6.2.4	EnableDocumentsPort	62
5.6.2.5	EnableDocumentsVectorStore	62
5.6.2.6	VectorStoreManager	62
5.6.2.7	VectorStorePineconeManager	62
5.6.2.8	VectorStoreChromaDBManager	63
5.6.2.9	VectorStoreDocumentOperationResponse	63
5.7	GetChatMessages	63
5.7.1	Diagramma delle classi	63
5.7.2	Lista delle sottocomponenti	63
5.7.2.1	GetChatMessagesController	63
5.7.2.2	ChatId	63
5.7.2.3	GetChatMessagesUseCase	63
5.7.2.4	GetChatMessagesService	63
5.7.2.5	Chat	63
5.7.2.6	ChatInfo	64
5.7.2.7	ChatId	64
5.7.2.8	Message	64
5.7.2.9	MessageSender	64
5.7.2.10	GetChatMessagesPort	64
5.7.2.11	GetChatMessagesPostgres	64
5.7.2.12	PostgresChat	64
5.7.2.13	PostgresMessage	64
5.7.2.14	PostgresORM	65
5.8	GetChats	65
5.8.1	Diagramma delle classi	65
5.8.2	Lista delle sottocomponenti	65
5.8.2.1	ChatFilter	65
5.8.2.2	GetChatsController	65
5.8.2.3	ChatFilter	65
5.8.2.4	GetChatsUseCase	65
5.8.2.5	GetChatsService	65
5.8.2.6	ChatPreview	65
5.8.2.7	ChatInfo	66
5.8.2.8	ChatId	66
5.8.2.9	GetChatsPort	66
5.8.2.10	GetChatsPostgres	66
5.8.2.11	PostgresChatPreview	66
5.8.2.12	PostgresORM	66
5.9	GetDocuments	66
5.9.1	Diagramma delle classi	66
5.9.2	Lista delle sottocomponenti	66
5.9.2.1	DocumentMetadata	66
5.9.2.2	DocumentType (Enumeration)	67



5.9.2.3	DocumentStatus	67
5.9.2.4	Status (Enumeration)	67
5.9.2.5	DocumentId	67
5.9.2.6	LightDocument	67
5.9.2.7	DocumentFilter	67
5.9.2.8	GetDocumentsController	67
5.9.2.9	GetDocumentsUseCase	68
5.9.2.10	GetDocumentsFacadeService	68
5.9.2.11	GetDocumentsStatus	68
5.9.2.12	GetDocumentsMetadata	68
5.9.2.13	GetDocumentsStatusPort	69
5.9.2.14	GetDocumentsMetadataPort	69
5.9.2.15	GetDocumentsStatusVectorStore	69
5.9.2.16	GetDocumentsListAWS3	69
5.9.2.17	AWS3Manager	70
5.9.2.18	AWSDocumentMetadata	70
5.9.2.19	VectorStoreManager	70
5.9.2.20	VectorStorePineconeManager	70
5.9.2.21	VectorStoreChromaDBManager	70
5.9.2.22	VectorStoreDocumentOperationResponse	70
5.10	RenameChat	70
5.10.1	Diagramma delle classi	70
5.10.2	Lista delle sottocomponenti	70
5.10.2.1	ChatId	70
5.10.2.2	RenameChatController	70
5.10.2.3	RenameChatUseCase	71
5.10.2.4	RenameChatService	71
5.10.2.5	RenameChatPort	71
5.10.2.6	RenameChatPostgres	71
5.10.2.7	PostgresORM	71
5.11	UploadDocuments	71
5.11.1	Diagramma delle classi	71
5.11.2	Lista delle sottocomponenti	71
5.11.2.1	NewDocument	71
5.11.2.2	Document	72
5.11.2.3	PlainDocument	72
5.11.2.4	DocumentMetadata	72
5.11.2.5	DocumentType (Enumeration)	72
5.11.2.6	DocumentStatus	72
5.11.2.7	Status (Enumeration)	72
5.11.2.8	DocumentId	72
5.11.2.9	DocumentContent	72
5.11.2.10	UploadDocumentsController	72
5.11.2.11	UploadDocumentsUseCase	73
5.11.2.12	UploadDocumentsService	73
5.11.2.13	DocumentUploader	73
5.11.2.14	EmbeddingsUploader	73
5.11.2.15	DocumentUploaderPort	74
5.11.2.16	EmbeddingsUploaderPort	74



5.11.2.17 DocumentUploaderAWSS3	74
5.11.2.18 AWSDocument	74
5.11.2.19 AWSDocumentOperationResponse	74
5.11.2.20 AWSS3Manager	74
5.11.2.21 EmbeddingsUploaderFacadeLangchain	74
5.11.2.22 DocumentOperationResponse	75
5.11.2.23 LangchainDocument	75
5.11.2.24 DocumentToText	75
5.11.2.25 TextExtractor	75
5.11.2.26 PDFTextExtractor	75
5.11.2.27 DOCXTextExtractor	75
5.11.2.28 Chunkerizer	75
5.11.2.29 EmbeddingsCreator	75
5.11.2.30 LangchainEmbeddingModel	75
5.11.2.31 OpenAIEmbeddingModel	76
5.11.2.32 HuggingFaceEmbeddingModel	76
5.11.2.33 EmbeddingsUploaderVectorStore	76
5.11.2.34 VectorStoreDocumentStatusResponse	76
5.11.2.35 VectorStoreManager	76
5.11.2.36 VectorStorePineconeManager	76
5.11.2.37 VectorStoreChromaDBManager	76
5.11.2.38 VectorStoreDocumentOperationResponse	76



Elenco delle figure



Elenco delle tabelle

1	Esiti possibili AskChatbot	27
2	Esiti possibili ConcealDocuments	29
3	Esiti possibili DeleteChats	30
4	Esiti possibili DeleteDocuments	31
5	Esiti possibili EmbedDocuments	32
6	Esiti possibili EnableDocuments	34
7	Esiti possibili GetChatMessages	35
8	Esiti possibili GetChats	36
9	Esiti possibili GetDocuments	38
10	Esiti possibili RenameChat	39
11	Esiti possibili UploadDocuments	40
12	Esiti possibili ViewDocumentContent	42



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%20Programming%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. Inoltre le richieste dell'utente al backend e mostra i risultati ottenuti.

3.2.2 Backend

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



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

3.3 Assemblaggio delle componenti

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

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

3.4 Struttura del sistema

3.4.1 Frontend

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

3.4.2 Backend

La struttura organizzativa del backend segue la seguente struttura:

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

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



4 Architettura delle componenti

4.1 Frontend

4.2 Backend

4.2.1 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.

4.2.1.2 Esiti possibili

Tabella 1: Esiti possibili AskChatbot

Codice	Descrizione	Risposta
200	Risposta ottenuta con successo.	-
500	Risposta del chatbot fallita.	Errore di generazione della risposta del chatbot.

4.2.1.3 Tracciamento dei requisiti

- RF.D.40;
- RF.O.47.1;
- RF.O.48.

4.2.1.4 Lista sottocomponenti

- **AskChatbotController:** classe controller che si occupa del richiedere una risposta al chatbot ad un messaggio appartenente ad una chat allo use case AskChatbotUseCase;
- **Message:** classe di business che rappresenta un messaggio di una chat;
- **MessageSender:** classe di business che rappresenta il mittente di un messaggio;
- **AskChatbotUseCase:** interfaccia use case che rappresenta la porta della business logic in entrata per richiedere una risposta al chatbot ad un messaggio appartenente ad una chat;
- **AskChatbotService:** classe service che implementa lo use case AskChatbotUseCase;



- **ChatId**: classe di business che rappresenta l'id di una chat utilizzato nella ricerca;
- **AskChatbotPort**: interfaccia che rappresenta la porta in uscita per effettuare la richiesta di una risposta al chatbot ad un messaggio appartenente ad una chat, con storicizzazione all'interno del database delle chat;
- **AskChatbotLangchain**: classe che implementa la porta AskChatbotPort, adattando la chiamata di AskChatbotService a classi offerte dal framework Langchain;
- **ChatHistoryManager**: classe per interagire con le chat come oggetti langchain.Memory;
- **ChatbotLangchain**: classe che permette di interagire con un oggetto langchain.Chain;
- **LangchainLLM**: classe astratta che permette di interagire con un oggetto langchain.BaseLanguageModel;
- **OpenAILLM**: classe che implementa la classe astratta LangchainLLM per interagire con un modello LLM di OpenAI;
- **HuggingFaceLLM**: classe che implementa la classe astratta LangchainLLM per interagire con un modello LLM di HuggingFace;
- **LangchainVectorStore**: classe astratta che permette di interagire con un oggetto langchain.VectorStore;
- **PineconeVectorStore**: classe che implementa la classe astratta Langchain-VectorStore per interagire con un vector store Pinecone;
- **ChromaDBVectorStore**: classe che implementa la classe astratta Langchain-VectorStore per interagire con un vector store ChromaDB;
- **LangchainEmbeddingModel**: classe astratta che permette di interagire con un oggetto langchain.Embeddings;
- **OpenAIEmbeddingModel**: classe che implementa la classe astratta LangchainEmbeddingModel per interagire con un modello di generazione di embeddings di OpenAI;
- **HuggingFaceEmbeddingModel**: classe che implementa la classe astratta LangchainEmbeddingModel per interagire con un modello di generazione di embeddings di HuggingFace;
- **PostgresORM**: classe che si occupa di effettuare le operazioni su Postgres configurato, cioè il sistema di storicizzazione delle chat dell'applicazione;
- **PostgresChat**: classe di persistence che rappresenta una chat;
- **PostgresMessage**: classe di persistence che rappresenta un messaggio.

4.2.2 ConcealDocuments

4.2.2.1 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.

4.2.2.2 Esiti possibili

Tabella 2: Esiti possibili ConcealDocuments

Codice	Descrizione	Risposta
200	Occultamento avvenuto con successo.	-
500	Occultamento fallito.	Errore nell'occultamento dei documenti.

4.2.2.3 Tracciamento dei requisiti

- RF.O.25;
- RF.O.26.

4.2.2.4 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;
- **ConcealDocumentsUseCase:** interfaccia use case che rappresenta la porta della business logic in entrata per effettuare l'occultamento dei documenti;
- **ConcealDocumentsService:** classe service che implementa lo use case ConcealDocumentsUseCase;
- **ConcealDocumentsPort:** interfaccia che rappresenta la porta in uscita per effettuare l'occultamento dei documenti nel vector store;
- **ConcealDocumentsVectorStore:** classe adapter che implementa la porta ConcealDocumentsPort, adattando la chiamata di ConcealDocumentsService a VectorStoreManager;
- **VectorStoreManager:** interfaccia che rende disponibile metodi per dialogare con i vector store;
- **VectorStorePineconeManager:** classe che implementa VectorStoreManager, offrendo la possibilità di dialogare con il vector store Pinecone;
- **VectorStoreChromaDBManager:** classe che implementa VectorStoreManager, offrendo la possibilità di dialogare con il vector store Chroma;
- **VectorStoreDocumentOperationResponse:** classe che contiene l'esito di un'operazione effettuata su un vector store riguardo un documento.



4.2.3 DeleteChats

4.2.3.1 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.

4.2.3.2 Esiti possibili

Tabella 3: Esiti possibili DeleteChats

Codice	Descrizione	Risposta
200	Eliminazione avvenuta con successo.	-
500	Eliminazione fallita.	Errore nella eliminazione della chat.

4.2.3.3 Tracciamento dei requisiti

- RF.O.33
- RF.O.34
- RF.O.34.1

4.2.3.4 Lista sottocomponenti

- **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;
- **DeleteChatsUseCase:** interfaccia use case che rappresenta la porta della business logic in entrata per effettuare l'eliminazione di una lista di chat;
- **DeleteChatsService:** classe service che implementa lo use case DeleteChatsUseCase;
- **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 PostgresORM;
- **PostgresORM:** vedi (§4.2.1.4) .



4.2.4 DeleteDocuments

4.2.4.1 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.

4.2.4.2 Esiti possibili

Tabella 4: Esiti possibili DeleteDocuments

Codice	Descrizione	Risposta
200	Eliminazione avvenuta con successo.	-
500	Eliminazione fallita.	Errore nell'eliminazione dei documenti.

4.2.4.3 Tracciamento dei requisiti

- TODO.

4.2.4.4 Lista sottocomponenti

- **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;
- **DeleteDocumentsUseCase:** interfaccia use case che rappresenta la porta della business logic in entrata per eliminare i documenti;
- **DeleteDocumentsService:** classe service che implementa lo use case DeleteDocumentsUseCase;
- **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;
- **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;
- **DeleteEmbeddingsPort:** interfaccia che rappresenta la porta in uscita per eliminare gli embeddings di una lista di documenti dal vector store;



- **DocumentOperationResponse**: vedi (§4.2.5.4) ;
- **DeleteDocumentsAWSS3**: classe adapter che implementa la porta DeleteDocumentsPort, adattando la chiamata di DeleteDocuments a AWSS3Manager;
- **AWSS3Manager**: classe che si occupa di effettuare le operazioni sul bucket di Amazon S3 configurato, cioè il sistema di archiviazione documenti dell'applicazione;
- **AWSDocumentOperationResponse**: classe che contiene l'esito di un'operazione effettuata su un AWS S3 riguardo un documento;
- **DeleteEmbeddingsVectorStore**: classe adapter che implementa la porta DeleteEmbeddingsPort, adattando la chiamata di DeleteDocumentsEmbeddings a VectorStoreManager;
- **VectorStoreManager**: vedi (§4.2.2.4) ;
- **VectorStorePineconeManager**: vedi (§4.2.2.4) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.2.4) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.2.4) .

4.2.5 EmbedDocuments

4.2.5.1 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.

4.2.5.2 Esiti possibili

Tabella 5: Esiti possibili EmbedDocuments

Codice	Descrizione	Risposta
200	Generazione embeddings avvenuta con successo.	-
500	Generazione embeddings fallita.	Errore nella generazione degli embeddings.

4.2.5.3 Tracciamento dei requisiti

- TODO.



4.2.5.4 Lista sottocomponenti

- **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;
- **EmbedDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per generare gli embeddings dei documenti;
- **EmbedDocumentsService**: classe service che implementa lo use case EmbedDocumentsUseCase;
- **GetDocuments**: classe che si occupa di inoltrare la richiesta di recuperare i documenti nella porta esterna diretta verso il sistema di archiviazione;
- **EmbeddingsUploader**: classe che si occupa di effettuare la chiamata dell'upload degli embeddings nella porta esterna diretta verso il vector store;
- **GetDocumentsPort**: interfaccia che rappresenta la porta in uscita per recuperare i documenti dal sistema di archiviazione;
- **EmbeddingsUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload degli embeddings verso i vector store;
- **GetDocumentsAWSS3**: classe adapter che implementa la porta GetDocumentsPort, adattando la chiamata di GetDocuments a AWSS3Manager;
- **AWSS3Manager**: vedi (§4.2.4.4) ;
- **AWSDocument**: classe che rappresenta i documenti gestibili da AWSS3Manager;
- **DocumentOperationResponse**: classe che contiene l'esito di un'operazione effettuata su un documento;
- **LangchainDocument**: classe che rappresenta un documento e i suoi embeddings;
- **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;
- **DocumentToText**: classe che si occupa di convertire documenti;
- **TextExtractor**: interfaccia che espone il metodo astratto di estrazione di testo da un documento;
- **PDFTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti PDF;
- **DOCXTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti docx;
- **Chunkerizer**: classe che crea chunks a partire da testo;
- **EmbeddingsCreator**: classe che dialoga con il modello di embeddings per creare gli embeddings a partire dai chunk forniti in input;
- **LangchainEmbeddingModel**: vedi (§4.2.1.4) ;



- **OpenAIEmbeddingModel**: vedi (§4.2.1.4) ;
- **HuggingFaceEmbeddingModel**: classe che implementa LangchainEmbeddingModel offrendo la possibilità di creare embeddings attraverso un embedding model di HuggingFace;
- **EmbeddingsUploaderVectorStore**: classe che offre un metodo per eseguire l'upload degli embeddings nel vector store;
- **VectorStoreManager**: vedi (§4.2.2.4) ;
- **VectorStorePineconeManager**: vedi (§4.2.2.4) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.2.4) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.2.4) .

4.2.6 EnableDocuments

4.2.6.1 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.

4.2.6.2 Esiti possibili

Tabella 6: Esiti possibili EnableDocuments

Codice	Descrizione	Risposta
200	Riabilitazione avvenuta con successo.	-
500	Riabilitazione fallita.	Errore nella riabilitazione dei documenti.

4.2.6.3 Tracciamento dei requisiti

- RF.O.25;
- RF.O.26.

4.2.6.4 Lista sottocomponenti

- **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;



- **EnableDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la riabilitazione dei documenti;
- **EnableDocumentsService**: classe service che implementa lo use case EnableDocumentsUseCase;
- **EnableDocumentsPort**: interfaccia che rappresenta la porta in uscita per effettuare la riabilitazione dei documenti nel vector store;
- **EnableDocumentsVectorStore**: classe adapter che implementa la porta EnableDocumentsPort, adattando la chiamata di EnableDocumentsService a VectorStoreManager;
- **VectorStoreManager**: vedi (§4.2.2.4) ;
- **VectorStorePineconeManager**: vedi (§4.2.2.4) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.2.4) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.2.4) .

4.2.7 GetChatMessages

4.2.7.1 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**: POST;
- **Lista parametri HTTP**:
 - "chatId": id della chat da recuperare.

4.2.7.2 Esiti possibili

Tabella 7: Esiti possibili GetChatMessages

Codice	Descrizione	Risposta
200	Chat recuperata con successo.	-
500	Recupero della Chat fallito.	Errore nella recupero della chat.

4.2.7.3 Tracciamento dei requisiti

- RF.O.39;
- RF.O.39.1;
- RF.O.39.1.1;
- RF.O.39.1.2.



4.2.7.4 Lista sottocomponenti

- **GetChatMessagesController**: classe controller che si occupa del richiedere una chat allo use case GetChatMessagesUseCase;
- **ChatId**: vedi (§4.2.1.4) ;
- **GetChatMessagesUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per richiedere una chat;
- **GetChatMessagesService**: classe service che implementa lo use case GetChatMessagesUseCase;
- **Chat**: classe di business che rappresenta una chat completa;
- **ChatInfo**: classe di business che rappresenta le informazioni di una chat;
- **Message**: vedi (§4.2.1.4) ;
- **MessageSender**: vedi (§4.2.1.4) ;
- **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 PostgresORM;
- **PostgresORM**: vedi (§4.2.1.4) ;
- **PostgresChat**: vedi (§4.2.1.4) ;
- **PostgresMessage**: vedi (§4.2.1.4) ;

4.2.8 GetChats

4.2.8.1 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**: POST;
- **Lista parametri HTTP**:
 - "filter": filtro da applicare per la ricerca tra le chat.

4.2.8.2 Esiti possibili

Tabella 8: Esiti possibili GetChats

Codice	Descrizione	Risposta
200	Ricerca avvenuta con successo.	-
500	Ricerca fallita.	Errore nella ricerca della chat.



4.2.8.3 Tracciamento dei requisiti

- RF.O.30;
- RF.O.30.1;
- RF.O.30.1.1;
- RF.O.30.1.2;
- RF.O.30.1.3;
- RF.O.30.1.4;
- RF.O.38.

4.2.8.4 Lista sottocomponenti

- **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;
- **ChatFilter**: classe di business che rappresenta il filtro utilizzato nella ricerca;
- **GetChatsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per richiedere le chat, eventualmente filtrate;
- **GetChatsService**: classe service che implementa lo use case GetChatsUseCase;
- **ChatPreview**: classe di business che rappresenta la preview di una chat;
- **ChatInfo**: vedi (§4.2.7.4) ;
- **ChatId**: vedi (§4.2.1.4) ;
- **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 PostgresORM;
- **PostgresORM**: vedi (§4.2.1.4) ;
- **PostgresChatPreview**: classe di persistence che rappresenta la preview di una chat.

4.2.9 GetDocuments

4.2.9.1 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**:



- "searchFilter": stringa che rappresenta il filtro opzionale da applicare alla ricerca dei documenti.

4.2.9.2 Esiti possibili

Tabella 9: Esiti possibili GetDocuments

Codice	Descrizione	Risposta
200	Ricerca avvenuta con successo.	-
500	Ricerca fallita.	Errore nella ricerca dei documenti.

4.2.9.3 Tracciamento dei requisiti

- TODO.

4.2.9.4 Lista sottocomponenti

- **DocumentMetadata**: classe di business che rappresenta i metadati di un documento;
- **DocumentType**: enumeration che rappresenta i valori che può assumere il tipo di un documento;
- **DocumentId**: classe di business che rappresenta l'id di un documento;
- **DocumentStatus**: classe di business che rappresenta lo status di un documento;
- **Status**: enumeration che rappresenta i valori che può assumere lo status di un documento;
- **LightDocument**: classe di business che fornisce una rappresentazione leggera dei documenti, escludendo il contenuto;
- **DocumentFilter**: classe di business che rappresenta il filtro applicabile alla ricerca di documenti;
- **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;
- **GetDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per ricercare dei documenti;
- **GetDocumentsFacadeService**: classe service che implementa lo use case GetDocumentsUseCase.
- **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;
- **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;



- **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;
- **GetDocumentsStatusVectorStore:** classe adapter che implementa la porta GetDocumentsStatusPort, adattando la chiamata di GetDocumentsStatus a VectorStoreManager;
- **GetDocumentsListAWSS3:** classe adapter che implementa la porta GetDocumentsMetadataPort, adattando la chiamata di GetDocumentsMetadata a AWSS3Manager;
- **AWSS3Manager:** vedi (§4.2.4.4) ;
- **AWSDocumentMetadata:** classe che rappresenta i metadati di un documento ricavato da AWS;
- **VectorStoreManager:** vedi (§4.2.2.4) ;
- **VectorStorePineconeManager:** vedi (§4.2.2.4) ;
- **VectorStoreChromaDBManager:** vedi (§4.2.2.4) .

4.2.10 RenameChat

4.2.10.1 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.

4.2.10.2 Esiti possibili

Tabella 10: Esiti possibili RenameChat

Codice	Descrizione	Risposta
500	Rinomina fallita.	Errore nella rinomina della chat.
500	Rinomina fallita.	Errore nella rinomina della chat: titolo vuoto.
500	Rinomina fallita.	Errore nella rinomina della chat: titolo già utilizzato per un'altra chat.

4.2.10.3 Tracciamento dei requisiti

- RF.O.35
- RF.O.36



- RF.O.37

4.2.10.4 Lista sottocomponenti

- **RenameChatController**: classe controller che si occupa del passaggio allo use case CreateChatUseCase di un intero rappresentante l'id di una chat e un titolo per eseguire la sua rinomina;
- **RenameChatUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la rinomina di una chat;
- **RenameChatService**: classe service che implementa lo use case RenameChatUseCase;
- **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 PostgresORM;
- **PostgresORM**: vedi (§4.2.1.4) .

4.2.11 UploadDocuments

4.2.11.1 Descrizione

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

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

4.2.11.2 Esiti possibili

Tabella 11: Esiti possibili UploadDocuments

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

4.2.11.3 Tracciamento dei requisiti

- RF.O.21;
- RF.O.21.1;
- RF.O.21.3;



- RF.O.21.4;
- RF.O.21.6.

4.2.11.4 Lista sottocomponenti

- **NewDocument**: classe di presentation che contiene le informazioni relative ai documenti appena inseriti dall'utente, presenti nella richiesta HTTP;
- **DocumentMetadata**: vedi (§4.2.9.4) ;
- **DocumentType**: vedi (§4.2.9.4) ;
- **DocumentId**: vedi (§4.2.9.4) ;
- **DocumentStatus**: vedi (§4.2.9.4) ;
- **Status**: vedi (§4.2.9.4) ;
- **DocumentContent**: classe di business che rappresenta il contenuto di un documento;
- **Document**: classe di business che rappresenta i documenti completi;
- **PlainDocument**: classe di business che rappresenta i documenti, compresi i metadati e il contenuto;
- **UploadDocumentsController**: classe controller che si occupa del passaggio di Documents allo use case UploadDocumentsUseCase a partire da NewDocuments;
- **UploadDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare l'upload dei documenti;
- **UploadDocumentsService**: classe service che implementa lo use case UploadDocumentsUseCase;
- **DocumentUploader**: classe che si occupa di effettuare la chiamata dell'upload dei documenti nella porta esterna diretta verso il sistema di archiviazione documenti;
- **EmbeddingsUploader**: vedi (§4.2.5.4) ;
- **DocumentUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload dei documenti verso il sistema di archiviazione documenti;
- **EmbeddingsUploaderPort**: vedi (§4.2.5.4) ;
- **DocumentUploaderAWSS3**: classe adapter che implementa la porta DocumentUploaderPort, adattando la chiamata di DocumentUploader a AWSS3Manager;
- **AWSDocument**: vedi (§4.2.5.4) ;
- **AWSDocumentOperationResponse**: vedi (§4.2.4.4) ;
- **DocumentOperationResponse**: vedi (§4.2.5.4) ;
- **EmbeddingsUploaderFacadeLangchain**: vedi (§4.2.5.4) ;
- **LangchainDocument**: vedi (§4.2.5.4) ;
- **AWSS3Manager**: vedi (§4.2.4.4) ;



- **DocumentToText**: vedi (§4.2.5.4) ;
- **TextExtractor**: vedi (§4.2.5.4) ;
- **PDFTextExtractor**: vedi (§4.2.5.4) ;
- **DOCXTextExtractor**: vedi (§4.2.5.4) ;
- **Chunkerizer**: vedi (§4.2.5.4) ;
- **EmbeddingsCreator**: vedi (§4.2.5.4) ;
- **LangchainEmbeddingModel**: vedi (§4.2.1.4) ;
- **OpenAIEmbeddingModel**: vedi (§4.2.1.4) ;
- **HuggingFaceEmbeddingModel**: vedi (§4.2.1.4) ;
- **EmbeddingsUploaderVectorStore**: vedi (§4.2.5.4) ;
- **VectorStoreDocumentStatusResponse**: classe che rappresenta la risposta generata da un vector store, contenente lo status del documento richiesto;
- **VectorStoreManager**: vedi (§4.2.2.4) ;
- **VectorStorePineconeManager**: vedi (§4.2.2.4) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.2.4) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.2.4) .

4.2.12 ViewDocumentContent

4.2.12.1 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.

4.2.12.2 Esiti possibili

Tabella 12: Esiti possibili ViewDocumentContent

Codice	Descrizione	Risposta
200	Documento recuperato con successo.	-
500	Recupero documento fallito.	Errore nel recupero del documento.

4.2.12.3 Tracciamento dei requisiti

- TODO.



4.2.12.4 Lista sottocomponenti

- **DocumentMetadata:** vedi (§4.2.9.4) ;
- **DocumentType:** vedi (§4.2.9.4) ;
- **DocumentId:** vedi (§4.2.9.4) ;
- **DocumentStatus:** vedi (§4.2.9.4) ;
- **Status:** vedi (§4.2.9.4) ;
- **DocumentContent:** vedi (§4.2.11.4) ;
- **Document:** vedi (§4.2.11.4) ;
- **PlainDocument:** vedi (§4.2.11.4) ;
- **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;
- **GetDocumentUseCase:** interfaccia use case che rappresenta la porta della business logic in entrata per recuperare un documento;
- **GetDocumentFacadeService:** classe service che implementa lo use case GetDocumentUseCase;
- **GetDocuments:** vedi (§4.2.5.4) ;
- **GetDocumentsStatus:** vedi (§4.2.9.4) ;
- **GetDocumentsPort:** vedi (§4.2.5.4) ;
- **GetDocumentsStatusPort:** interfaccia che rappresenta la porta in uscita per recuperare gli status di una lista di documenti dal vector store;
- **GetDocumentsAWS3:** vedi (§4.2.5.4) ;
- **AWS3Manager:** vedi (§4.2.4.4) ;
- **AWSDocument:** vedi (§4.2.5.4) ;
- **GetDocumentsStatusVectorStore:** vedi (§4.2.9.4) ;
- **VectorStoreManager:** vedi (§4.2.2.4) ;
- **VectorStorePineconeManager:** vedi (§4.2.2.4) ;
- **VectorStoreChromaDBManager:** vedi (§4.2.2.4) ;
- **VectorStoreDocumentOperationResponse:** vedi (§4.2.2.4) .

4.3 Database

4.4 Libreria per la persistenza



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): Message`
Metodo che ritorna un Message 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;
- `toMessageFrom(message: string): Message`
Metodo che trasforma una stringa un Message;
- `toChatIdFrom(chatIdInt:int): ChatId`
Metodo che trasforma un intero in un ChatId.

5.1.2.2 ChatId

- **Attributi:**

- `id: int.`

5.1.2.3 Message

- **Attributi:**

- `content: string;`
- `timestamp: timestamp;`
- `relevantDocument: List<string>;`
- `sender: MessageSender.`

5.1.2.4 MessageSender (Enumeration)

- **Valori:**

- `Chatbot;`
- `User;`

5.1.2.5 AskChatbotUseCase

- **Metodi:**



- `askChatbot(message:Message, chatId:ChatId): Message`
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 AskChatbotService

- **Attributi:**
 - `outPort: GetChatMessagesPort.`
- **Metodi:**
 - `askChatbot(message:Message, chatId:ChatId): Message`
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.

5.1.2.7 AskChatbotPort

- **Metodi:**
 - `askChatbot(message: Message, chatId: ChatId): Message`
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.

5.1.2.8 AskChatbotLangchain

- **Attributi:**
 - `chatbot: ChatbotLangchain;`
 - `chatHistoryManager: ChatHistoryManager;`
 - `postgresORM: PostgresORM.`
- **Metodi:**
 - `askChatbot(message:Message, chatId:ChatId): Message`
Implementazione del metodo astratto di `AskChatbotPort` 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 tramite `postgresORM`;
 - `toMessageFrom(message: BaseMessage): Message`
Metodo per ottenere un `Message` a partire da un `BaseMessage`;
 - `toBaseMessageFrom(message: Message): BaseMessage`
Metodo per ottenere un `BaseMessage` a partire da un `Message`;
 - `createChat(): ChatId`
Metodo per creare una chat tramite `postgresORM`, ritornando il suo `ChatId`;



- `createMemory(chatId:ChatId): langchain.Memory`
Metodo per ottenere una `langchain.Memory` a partire da un `ChatId` tramite `chatHistoryManager`.

5.1.2.9 ChatbotLangchain

- **Attributi:**

- `chain: langchain.Chain.`

- **Metodi:**

- `ask(userMessage:BaseMessage, memory:langchain.Memory): langchain.BaseMessage`
Metodo per ottenere una risposta di tipo `langchain.BaseMessage` dal chatbot tramite `chain` ad un `userMessage` di tipo `BaseMessage`, richiedendo inoltre il contesto della chat attraverso la `memory` di tipo `langchain.Memory`.

5.1.2.10 LangchainLLM

- **Attributi:**

- `llm: langchain.BaseLanguageModel.`

5.1.2.11 OpenAILLM

- **Attributi:**

- `llm: langchain.BaseLanguageModel.`

5.1.2.12 HuggingFaceLLM

- **Attributi:**

- `llm: langchain.BaseLanguageModel.`

5.1.2.13 LangchainVectorStore

- **Attributi:**

- `vectorStore: langchain.VectorStore.`

- **Metodi:**

- `getRetriever(): BaseRetriever`
Metodo astratto per ottenere un `langchain.BaseRetriever`.

5.1.2.14 PineconeVectorStore

- **Attributi:**

- `vectorStore: langchain.VectorStore.`

- **Metodi:**

- `getRetriever(): BaseRetriever`
Implementazione del metodo astratto di `LangchainVectorStore` per ottenere un `langchain.BaseRetriever` da un `vector store` `Pinecone`.



5.1.2.15 ChromaDBVectorStore

- **Attributi:**

- `vectorStore: langchain.VectorStore.`

- **Metodi:**

- `getRetriever(): BaseRetriever`
Implementazione del metodo astratto di `LangchainVectorStore` per ottenere un `langchain.BaseRetriever` da un vector store ChromaDB.

5.1.2.16 LangchainEmbeddingModel

- **Attributi:**

- `embeddingModel: Embeddings.`

- **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.

5.1.2.17 OpenAIEmbeddingModel

- **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.

5.1.2.18 HuggingFaceEmbeddingModel

- **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.

5.1.2.19 ChatHistoryManager

- **Metodi:**

- `getChatHistory(chatId: ChatId): langchain.PostgresChatMessageHistory`
Metodo per ottenere una `langchain.PostgresChatMessageHistory` rappresentante il contesto di una chat identificata da un `ChatId`;
- `saveMessage(message: langchain.BaseMessage, memory: langchain.Memory): void`
Metodo per salvare un messaggio di tipo `langchain.BaseMessage` in una memoria di tipo `langchain.Memory` rappresentante il contesto di una chat.



5.1.2.20 PostgresORM

- **Metodi:**

- `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`;
- `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;
- `renameChat(chatId:int, title:string): boolean`
Metodo per rinominare con `title` una chat identificata tramite `chatId` sul database `Postgres` utilizzato per la storicizzazione delle chat. L'esito dell'operazione viene comunicato ritornando un `boolean`;
- `deleteChats(chatId:int): boolean`
Metodo per eliminare una `PostgresChat` sul database `Postgres` utilizzato per la storicizzazione delle chat. L'esito dell'operazione viene comunicato ritornando un `boolean`;
- `createChat(): PostgresChat`
Metodo per creare una nuova `PostgresChat` sul database `Postgres` utilizzato per la storicizzazione della chat. L'istanza della chat creata viene ritornata dal metodo.

5.1.2.21 PostgresChat

- **Attributi:**

- `id: int`
- `title: string;`
- `timestamp: timestamp;`
- `messages: List<PostgresMessage>.`

5.1.2.22 PostgresMessage

- **Attributi:**

- `content: string;`
- `timestamp: timestamp;`
- `relevantDocument: List<string>;`
- `sender: string.`



5.2 ConcealDocuments

5.2.1 Diagramma delle classi

5.2.2 Lista delle sottocomponenti

5.2.2.1 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;

5.2.2.2 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.

5.2.2.3 ConcealDocumentsService

- **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.

5.2.2.4 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.



5.2.2.5 ConcealDocumentsVectorStore

- **Attributi:**

-

- **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.

5.2.2.6 VectorStoreManager

- **Metodi:**

- `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`;
- `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`;
- `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`;
- `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`;
- `uploadEmbeddings(documentEmbeddings: List<Tuple<string; List<float>; Dict<string; Any>>>): List<VectorStoreDocumentOperationResponse>`
Metodo astratto per effettuare l'upload degli embeddings di un documento a partire da una lista di tuple, ciascuna formata da una stringa (`documentId`), una lista di float (embeddings del documento) e un dizionario con coppie `string: Any` (metadata del documento). Ritorna un `VectorStoreDocumentOperationResponse`.

5.2.2.7 VectorStorePineconeManager

- **Metodi:**



- `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`;
- `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`;
- `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`;
- `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`;
- `uploadEmbeddings(documentEmbeddings:`
`List<Tuple<string; List<float>; Dict<string; Any>>>):`
`List<VectorStoreDocumentOperationResponse>`
Implementazione del metodo astratto di `VectorStoreManager` per effettuare in Pinecone l'upload degli embeddings di un documento a partire da una lista di tuple, ciascuna formata da una stringa (`documentId`), una lista di float (embeddings del documento) e un dizionario con coppie `string: Any` (meta-data del documento). Ritorna un `VectorStoreDocumentOperationResponse`.

5.2.2.8 VectorStoreChromaDBManager

- **Metodi:**

- `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`;
- `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;

- `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;
- `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;
- `uploadEmbeddings(documentEmbeddings: List<Tuple<string; List<float>; Dict<string; Any>>>): List<VectorStoreDocumentOperationResponse>`
Implementazione del metodo astratto di VectorStoreManager per effettuare in ChromaDB l'upload degli embeddings di un documento a partire da una lista di tuple, ciascuna formata da una stringa (documentId), una lista di float (embeddings del documento) e un dizionario con coppie string: Any (meta-data del documento). Ritorna un VectorStoreDocumentOperationResponse.

5.2.2.9 VectorStoreDocumentOperationResponse

- **Attributi:**
 - `documentId: string;`
 - `responseStatus: boolean;`
 - `responseMessage: string.`

5.3 DeleteChats

5.3.1 Diagramma delle classi

5.3.2 Lista delle sottocomponenti

5.3.2.1 ChatId

Vedi (§5.1.2.2) .

5.3.2.2 DeleteChatsController

- **Attributi:**
 - `useCase: DeleteChatsUseCase.`
- **Metodi:**
 - `deleteChats(chatIdList:List<int>): List<ChatOperationResponse>`



Metodo che elimina tramite useCase una lista di chat identificate da una lista di interi, ritornando una lista di ChatOperationResponse;

- `toChatIdFrom(chatIdInt: int): ChatId`
Metodo che crea e ritorna un ChatId a partire da un intero.

5.3.2.3 DeleteChatsUseCase

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

5.3.2.4 DeleteChatsService

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

5.3.2.5 DeleteChatsPort

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

5.3.2.6 DeleteChatsPostgres

- **Attributi:**
 - `postgresORM: PostgresORM.`
- **Metodi:**
 - `deleteChats(chatIdList: List<ChatId>): List<ChatOperationResponse>`
Implementazione del metodo astratto di DeleteChatsPort per eliminare una lista di chat identificare da una lista di ChatId, ritornando una lista di ChatOperationResponse.

5.3.2.7 ChatOperationResponse

- **Attributi:**
 - `chatId: ChatId;`



- `response: boolean.`

5.3.2.8 PostgresORM

Vedi (§5.1.2.20).

5.4 DeleteDocuments

5.4.1 Diagramma delle classi

5.4.2 Lista delle sottocomponenti

5.4.2.1 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`.

5.4.2.2 DeleteDocumentsUseCase

- **Metodi:**

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

5.4.2.3 DeleteDocumentsService

- **Attributi:**

- `deleteDocuments: 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.

5.4.2.4 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.

5.4.2.5 DeleteDocumentsEmbeddings

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

5.4.2.6 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.

5.4.2.7 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.

5.4.2.8 DocumentOperationResponse

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

5.4.2.9 DeleteDocumentsAWS3

- **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.



5.4.2.10 AWSS3Manager

- **Attributi:**

- `boto3: boto3.session.Session;`

- **Metodi:**

- `getDocumentById(documentId:string): AWSDocument`
Metodo per ottenere un `AWSDocument` a partire da una stringa corrispondente al `documentId` del documento tramite `boto3`;
- `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`;
- `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`.

5.4.2.11 AWSDocumentOperationResponse

- **Attributi:**

- `documentId: string;`
- `status: boolean;`
- `message: string.`

5.4.2.12 DeleteEmbeddingsVectorStore

- **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.

5.4.2.13 VectorStoreManager

Vedi (§5.2.2.6).



5.4.2.14 VectorStorePineconeManager

Vedi (§5.2.2.7) .

5.4.2.15 VectorStoreChromaDBManager

Vedi (§5.2.2.8) .

5.4.2.16 VectorStoreDocumentOperationResponse

Vedi (§5.2.2.9) .

5.5 EmbedDocuments

5.5.1 Diagramma delle classi

5.5.2 Lista delle sottocomponenti

5.5.2.1 EmbedDocumentsController

- **Attributi:**

- `useCase: EmbedDocumentsUseCase .`

- **Metodi:**

- `embedDocuments(documentIds: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;
- `toDocument(document:NewDocument): Document`
Metodo che trasforma un NewDocument della presentation logic in un Document della business logic.

5.5.2.2 EmbedDocumentsUseCase

- **Metodi:**

- `embedDocuments(documentIds: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.

5.5.2.3 EmbedDocumentsService

- **Attributi:**

- `getDocuments: GetDocuments ;`
- `embeddingsUploader: EmbeddingsUploader .`

- **Metodi:**



- `embedDocuments(documentIds: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.

5.5.2.4 GetDocuments

- **Attributi:**

- `outPort: GetDocumentsPort.`

- **Metodi:**

- `getDocuments(documentIds:List<DocumentId>):`
`List<PlainDocument>`
Metodo per recuperare una lista di `Document` tramite `outPort` dal sistema di archiviazione documenti a partire da una lista di id.

5.5.2.5 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`.

5.5.2.6 GetDocumentsPort

- **Metodi:**

- `getDocuments(documentIds:List<DocumentId>):`
`List<PlainDocument>`
Metodo astratto per recuperare una lista di documenti dal sistema di archiviazione a partire dal loro id.

5.5.2.7 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.



5.5.2.8 GetDocumentsAWSS3

- **Attributi:**

- `awsS3Manager: AWSS3Manager.`

- **Metodi:**

- `getDocuments(documentIds: List<DocumentId>): List<PlainDocument>`
Implementazione del metodo astratto di `GetDocumentsPort` per recuperare 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.

5.5.2.9 AWSS3Manager

Vedi (§5.4.2.10).

5.5.2.10 AWSDocument

- **Attributi:**

- `id: string;`
- `content: bytes;`
- `type: string;`
- `size: float;`
- `uploadTime: timestamp.`

5.5.2.11 EmbeddingsUploaderPort

Vedi (§5.5.2.7).

5.5.2.12 DocumentOperationResponse

Vedi (§5.4.2.8).

5.5.2.13 LangchainDocument

- **Attributi:**

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

5.5.2.14 DocumentToText

- **Metodi:**

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



- `getTextExtractorFrom(documentType:string): TextExtractor`
Metodo per ottenere un `TextExtractor` a partire dal tipo del documento, indicato sotto forma di stringa.

5.5.2.15 TextExtractor

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

5.5.2.16 PDFTextExtractor

- **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`.

5.5.2.17 DOCXTextExtractor

- **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`.

5.5.2.18 Chunkerizer

- **Metodi:**
 - `createChunks(documents:List<langchain.Document>): List<langchain.Document>`
Metodo per suddividere in chunks una lista di `langchain.Document`, ritornandoli attraverso una lista di `langchain.Document`.

5.5.2.19 EmbeddingsCreator

- **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.

5.5.2.20 LangchainEmbeddingModel

Vedi (§5.1.2.16) .

5.5.2.21 OpenAIEmbeddingModel

Vedi (§5.1.2.17) .



5.5.2.22 HuggingFaceEmbeddingModel

Vedi (§5.1.2.18).

5.5.2.23 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.

5.5.2.24 VectorStoreManager

Vedi (§5.2.2.6).

5.5.2.25 VectorStorePineconeManager

Vedi (§5.2.2.7).

5.5.2.26 VectorStoreChromaDBManager

Vedi (§5.2.2.8).

5.5.2.27 VectorStoreDocumentOperationResponse

Vedi (§5.2.2.9).

5.6 EnableDocuments

5.6.1 Diagramma delle classi

5.6.2 Lista delle sottocomponenti

5.6.2.1 EnableDocumentsController

- **Attributi:**

- `useCase: EnableDocumentsUseCase.`

- **Metodi:**

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



5.6.2.2 EnableDocumentsUseCase

- **Metodi:**

- `enableDocuments(documentIds: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.

5.6.2.3 EnableDocumentsService

- **Attributi:**

- `outPort: EnableDocumentsPort.`

- **Metodi:**

- `enableDocuments(documentIds: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.

5.6.2.4 EnableDocumentsPort

- **Metodi:**

- `enableDocuments(documentIds: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.

5.6.2.5 EnableDocumentsVectorStore

- **Attributi:**

- `vectorStoreManager: VectorStoreManager.`

- **Metodi:**

- `enableDocuments(documentIds: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.

5.6.2.6 VectorStoreManager

Vedi (§5.2.2.6).

5.6.2.7 VectorStorePineconeManager

Vedi (§5.2.2.7).



5.6.2.8 VectorStoreChromaDBManager

Vedi (§5.2.2.8) .

5.6.2.9 VectorStoreDocumentOperationResponse

Vedi (§5.2.2.9) .

5.7 GetChatMessages

5.7.1 Diagramma delle classi

5.7.2 Lista delle sottocomponenti

5.7.2.1 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.

5.7.2.2 ChatId

Vedi (§5.1.2.2) .

5.7.2.3 GetChatMessagesUseCase

- **Metodi:**

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

5.7.2.4 GetChatMessagesService

- **Attributi:**

- `outPort: GetChatMessagesPort .`

- **Metodi:**

- `getChatMessages(chatId:ChatId): Chat`
Implementazione del metodo astratto di GetChatMessagesUseCase per ottenere una Chat a partire da un ChatId.

5.7.2.5 Chat

- **Attributi:**

- `info: ChatInfo;`
- `messages: List<Message> .`



5.7.2.6 ChatInfo

- **Attributi:**

- chatId: ChatId
- title: string;
- timestamp: timestamp.

5.7.2.7 ChatId

Vedi (§5.1.2.2) .

5.7.2.8 Message

Vedi (§5.1.2.3) .

5.7.2.9 MessageSender

Vedi (§5.1.2.4) .

5.7.2.10 GetChatMessagesPort

- **Metodi:**

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

5.7.2.11 GetChatMessagesPostgres

- **Attributi:**

- postgresORM: PostgresORM.

- **Metodi:**

- getChatMessages(chatId:ChatId): Chat
Implementazione del metodo astratto di GetChatMessagesPort per ottenere una Chat a partire da un ChatId;
- toChatFrom(in postgresChat:PostgresChat): Chat
Metodo che trasforma un oggetto di persistence PostgresChat in un oggetto di business Chat;
- toMessageFrom(in postgresMessage:PostgresMessage): Message
Metodo che trasforma un oggetto di persistence PostgresMessage in un oggetto di business Message.

5.7.2.12 PostgresChat

Vedi (§5.1.2.21) .

5.7.2.13 PostgresMessage

Vedi (§5.1.2.22) .



5.7.2.14 PostgresORM

Vedi (§5.1.2.20).

5.8 GetChats

5.8.1 Diagramma delle classi

5.8.2 Lista delle sottocomponenti

5.8.2.1 ChatFilter

- **Attributi:**

- `filter: string.`

5.8.2.2 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.

5.8.2.3 ChatFilter

Vedi (§5.8.2.1).

5.8.2.4 GetChatsUseCase

- **Metodi:**

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

5.8.2.5 GetChatsService

- **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.

5.8.2.6 ChatPreview

- **Attributi:**

- `info: ChatInfo;`
- `lastMessageSnippet: string.`



5.8.2.7 ChatInfo

Vedi (§5.7.2.6).

5.8.2.8 ChatId

Vedi (§5.1.2.2).

5.8.2.9 GetChatsPort

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

5.8.2.10 GetChatsPostgres

- **Attributi:**
 - `postgresORM: PostgresORM.`
- **Metodi:**
 - `getChats(chatFilter: ChatFilter): List<ChatPreview>`
Implementazione del metodo astratto di GetChatPort per ottenere una lista di ChatPreview tramite postgresORM, filtrando eventualmente la ricerca tramite chatFilter;
 - `getChatPreviewFrom(chat: PostgresChatPreview): ChatPreview`
Metodo per ottenere una ChatPreview a partire da una PostgresChatPreview.

5.8.2.11 PostgresChatPreview

- **Attributi:**
 - `id: int`
 - `title: string;`
 - `timestamp: timestamp;`
 - `lastMessageSnippet: string.`

5.8.2.12 PostgresORM

Vedi (§5.1.2.20).

5.9 GetDocuments

5.9.1 Diagramma delle classi

5.9.2 Lista delle sottocomponenti

5.9.2.1 DocumentMetadata

- **Attributi:**



- `documentId: DocumentId;`
- `type: DocumentType;`
- `size: float;`
- `uploadTime: timestamp.`

5.9.2.2 DocumentType (Enumeration)

- **Valori:**

- `PDF;`
- `DOCX.`

5.9.2.3 DocumentStatus

- **Attributi:**

- `status: Status.`

5.9.2.4 Status (Enumeration)

- **Valori:**

- `Concealed;`
- `Enabled;`
- `Not Embedded.`

5.9.2.5 DocumentId

- **Attributi:**

- `id: string.`

5.9.2.6 LightDocument

- **Attributi:**

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

5.9.2.7 DocumentFilter

- **Attributi:**

- `searchFilter: string.`

5.9.2.8 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`.

5.9.2.9 `GetDocumentsUseCase`

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

5.9.2.10 `GetDocumentsFacadeService`

- **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.

5.9.2.11 `GetDocumentsStatus`

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

5.9.2.12 `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.

5.9.2.13 GetDocumentsStatusPort

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

5.9.2.14 GetDocumentsMetadataPort

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

5.9.2.15 GetDocumentsStatusVectorStore

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

5.9.2.16 GetDocumentsListAWS3

- **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.
 - `toDocumentMetadataFrom(document:AWSDocumentMetadata): DocumentMetadata`
Metodo che trasforma un AWSDocumentMetadata della persistence in un DocumentMetadata della business logic.



5.9.2.17 AWSS3Manager

Vedi (§5.4.2.10) .

5.9.2.18 AWSDocumentMetadata

- **Attributi:**

- id: string;
- size: float;
- uploadTime: timestamp.

5.9.2.19 VectorStoreManager

Vedi (§5.2.2.6) .

5.9.2.20 VectorStorePineconeManager

Vedi (§5.2.2.7) .

5.9.2.21 VectorStoreChromaDBManager

Vedi (§5.2.2.8) .

5.9.2.22 VectorStoreDocumentOperationResponse

Vedi (§5.2.2.9) .

5.10 RenameChat

5.10.1 Diagramma delle classi

5.10.2 Lista delle sottocomponenti

5.10.2.1 ChatId

Vedi (§5.1.2.2) .

5.10.2.2 RenameChatController

- **Attributi:**

- useCase: DeleteChatsUseCase .

- **Metodi:**

- renameChat(chatId:int, title:string): boolean
Metodo che rinomina tramite useCase una chat identificate da un intero, ritornando un boolean rappresentante l'esito dell'operazione;
- toChatIdFrom(chatIdInt:int): ChatId
Metodo che crea e ritorna un ChatId a partire da un intero.



5.10.2.3 RenameChatUseCase

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

5.10.2.4 RenameChatService

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

5.10.2.5 RenameChatPort

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

5.10.2.6 RenameChatPostgres

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

5.10.2.7 PostgresORM

Vedi (§5.1.2.20) .

5.11 UploadDocuments

5.11.1 Diagramma delle classi

5.11.2 Lista delle sottocomponenti

5.11.2.1 NewDocument

- **Attributi:**
 - `documentId: string;`
 - `type: string;`



- `size: float.`

5.11.2.2 Document

- **Attributi:**

- `plainDocument: PlainDocument;`
- `status: DocumentStatus.`

5.11.2.3 PlainDocument

- **Attributi:**

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

5.11.2.4 DocumentMetadata

Vedi (§5.9.2.1).

5.11.2.5 DocumentType (Enumeration)

Vedi (§5.9.2.2).

5.11.2.6 DocumentStatus

Vedi (§5.9.2.3).

5.11.2.7 Status (Enumeration)

Vedi (§5.9.2.4).

5.11.2.8 DocumentId

Vedi (§5.9.2.5).

5.11.2.9 DocumentContent

- **Attributi:**

- `content: bytes.`

5.11.2.10 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;



- `toDocument(document:NewDocument): Document`
Metodo che trasforma un `NewDocument` della presentation logic in un `Document` della business logic.

5.11.2.11 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`.

5.11.2.12 UploadDocumentsService

- **Attributi:**
 - `documentUploader: DocumentUploader;`
 - `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 `documentUploader` 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.

5.11.2.13 DocumentUploader

- **Attributi:**
 - `outPort: DocumentUploaderPort.`
- **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`.

5.11.2.14 EmbeddingsUploader

Vedi (§5.5.2.5).



5.11.2.15 DocumentUploaderPort

- **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.

5.11.2.16 EmbeddingsUploaderPort

Vedi (§5.5.2.7) .

5.11.2.17 DocumentUploaderAWSS3

- **Attributi:**

- `awsS3Manager: AWSS3Manager .`

- **Metodi:**

- `uploadDocuments(documents:List<Document>, forceUpload:boolean): List<DocumentOperationResponse>`
Implementazione del metodo astratto di DocumentUploaderPort 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.

5.11.2.18 AWSDocument

Vedi (§5.5.2.10) .

5.11.2.19 AWSDocumentOperationResponse

Vedi (§5.4.2.11) .

5.11.2.20 AWSS3Manager

Vedi (§5.4.2.10) .

5.11.2.21 EmbeddingsUploaderFacadeLangchain

- **Attributi:**

- `documentToText: DocumentToText;`
- `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`.

5.11.2.22 DocumentOperationResponse

Vedi (§5.4.2.8) .

5.11.2.23 LangchainDocument

Vedi (§5.5.2.13) .

5.11.2.24 DocumentToText

Vedi (§5.5.2.14) .

5.11.2.25 TextExtractor

Vedi (§5.5.2.15) .

5.11.2.26 PDFTextExtractor

Vedi (§5.5.2.16) .

5.11.2.27 DOCXTextExtractor

Vedi (§5.5.2.17) .

5.11.2.28 Chunkerizer

Vedi (§5.5.2.18) .

5.11.2.29 EmbeddingsCreator

Vedi (§5.5.2.19) .

5.11.2.30 LangchainEmbeddingModel

Vedi (§5.1.2.16) .

**5.11.2.31 OpenAIEmbeddingModel**

Vedi (§5.1.2.17) .

5.11.2.32 HuggingFaceEmbeddingModel

Vedi (§5.1.2.18) .

5.11.2.33 EmbeddingsUploaderVectorStore

Vedi (§5.5.2.23) .

5.11.2.34 VectorStoreDocumentStatusResponse

- **Attributi:**

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

5.11.2.35 VectorStoreManager

Vedi (§5.2.2.6) .

5.11.2.36 VectorStorePineconeManager

Vedi (§5.2.2.7) .

5.11.2.37 VectorStoreChromaDBManager

Vedi (§5.2.2.8) .

5.11.2.38 VectorStoreDocumentOperationResponse

Vedi (§5.2.2.9) .