



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
v3.0.0(0)	2024 – 04 – 12	-	-	Approvazione versione finale PB.
v2.27.7(7)	2024 – 04 – 02	Bresolin G.	Dugo A.	Correzione versione e diagramma componente GetChatMessages.
v2.27.7(6)	2024 – 03 – 30	Campese M.	Bresolin G.	Aggiornamento diagrammi delle classi.
v2.27.7(5)	2024 – 03 – 30	Bresolin G.	Campese M.	Aggiornamento viewDocumentContent in getDocumentContent.
v2.27.7(3)	2024 – 03 – 29	Campese M.	Michelon R.	Aggiornamento descrizione generale delle componenti e migliorie diagrammi.
v2.27.7(2)	2024 – 03 – 29	Dugo A.	Campese M.	Aggiornamento sezione Database e diagrammi ER.
v2.27.6(2)	2024 – 03 – 29	Campese M.	Orlandi G.	Aggiunta diagrammi delle classi aggiornati.
v2.27.5(1)	2024 – 03 – 28	Orlandi G.	Dugo A.	Aggiunta tabella riepilogativa tracciamento dei requisiti.
v2.27.2(1)	2024 – 03 – 26	Michelon R.	Campese M.	Aggiornamento progettazione di dettaglio frontend.
v2.26.0(1)	2024 – 03 – 25	Bresolin G.	Ciriolo I.	Aggiornamento tracciamento dei requisiti del frontend.
v2.25.2(1)	2024 – 03 – 22	Campese M.	Feltrin E.	Aggiornamento tracciamento dei requisiti.

Continua nella pagina successiva



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

Continua nella pagina successiva



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

Continua nella pagina successiva



Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
v2.0.0(16)	2024 – 03 – 02	Michelon R.	Dugo A.	Aggiunta pro- gettazione logica DeleteDocuments.
v2.0.0(15)	2024 – 03 – 02	Bresolin G.	Feltrin E.	Aggiunta pro- gettazione logica DeleteChats.
v2.0.0(14)	2024 – 03 – 02	Dugo A.	Orlandi G.	Aggiunta pro- gettazione logica ConcealDocuments.
v2.0.0(13)	2024 – 03 – 02	Michelon R.	Campese M.	Aggiunta pro- gettazione logica AskChatbot.
v2.0.0(8)	2024 – 02 – 28	Feltrin E.	Michelon R.	Aggiunta versioni tecnologie utilizzate.
v2.0.0(7)	2024 – 02 – 28	Dugo A. Bresolin G.	Michelon R.	Aggiunta pro- gettazione logica UploadDocuments.
v2.0.0(6)	2024 – 02 – 28	Feltrin E. Michelon R.	Bresolin G.	Prima stesura archi- tettura di sistema e tecnologie utilizzate.



Indice

1	Introduzione	19
1.1	Obiettivo del documento	19
1.2	Glossario	19
1.3	Riferimenti	19
1.3.1	Riferimenti normativi	19
1.3.2	Riferimenti informativi	19
2	Tecnologie utilizzate	23
2.1	Flask	23
2.1.1	Python	24
2.2	Next.js	24
2.2.1	Typescript	25
2.3	Docker	26
2.4	Langchain	26
2.4.1	Pinecone	27
2.4.2	ChromaDB	28
2.4.3	OpenAI	28
2.4.4	HuggingFace	29
2.5	AWS S3	30
2.6	Postgres	30
3	Architettura di sistema	32
3.1	Modello architetturale	32
3.2	Descrizione delle componenti	32
3.2.1	Frontend	32
3.2.2	Backend	32
3.2.3	Database	33
3.3	Assemblaggio delle componenti	33
3.3.1	Docker	33
3.4	Struttura del sistema	33
3.4.1	Frontend	33
3.4.2	Backend	33
4	Architettura delle componenti	35
4.1	Frontend	35
4.1.1	Chatbot	35
4.1.1.1	Descrizione	35
4.1.1.2	Lista di sottocomponenti	35
4.1.1.3	Tracciamento dei requisiti	35
4.1.2	Dashboard	36
4.1.2.1	Descrizione	36
4.1.2.2	Lista di sottocomponenti	36
4.1.2.3	Tracciamento dei requisiti	36
4.1.3	Documents	36
4.1.3.1	Descrizione	36
4.1.3.2	Lista di sottocomponenti	36
4.1.3.3	Tracciamento dei requisiti	37



4.1.4	DocumentView	37
4.1.4.1	Descrizione	37
4.1.4.2	Lista di sottocomponenti	38
4.1.5	SettingsAppearance	38
4.1.5.1	Descrizione	38
4.1.5.2	Lista di sottocomponenti	38
4.1.5.3	Tracciamento dei requisiti	38
4.1.6	SettingsConfiguration	38
4.1.6.1	Descrizione	38
4.1.6.2	Lista di sottocomponenti	38
4.1.6.3	Tracciamento dei requisiti	38
4.1.7	SetUp (Primo avvio)	39
4.1.7.1	Descrizione	39
4.1.7.2	Lista di sottocomponenti	39
4.1.7.3	Tracciamento dei requisiti	40
4.1.8	SideBar	40
4.1.8.1	Descrizione	40
4.1.8.2	Lista di sottocomponenti	40
4.1.8.3	Tracciamento dei requisiti	40
4.2	Backend	41
4.2.1	AskChatbot	41
4.2.1.1	Descrizione	41
4.2.1.2	Esiti possibili	41
4.2.1.3	Lista sottocomponenti	41
4.2.1.4	Tracciamento dei requisiti	42
4.2.2	ChangeConfiguration	42
4.2.2.1	Descrizione	42
4.2.2.2	Esiti possibili	43
4.2.2.3	Lista sottocomponenti	43
4.2.2.4	Tracciamento dei requisiti	43
4.2.3	ConcealDocuments	44
4.2.3.1	Descrizione	44
4.2.3.2	Esiti possibili	44
4.2.3.3	Lista sottocomponenti	44
4.2.3.4	Tracciamento dei requisiti	45
4.2.4	ConfigurationManager	45
4.2.4.1	Descrizione	45
4.2.4.2	Esiti possibili ConfigurationManager	45
4.2.4.3	Tracciamento dei requisiti	46
4.2.4.4	Lista sottocomponenti	46
4.2.5	DeleteChats	46
4.2.5.1	Descrizione	46
4.2.5.2	Esiti possibili	47
4.2.5.3	Lista sottocomponenti	47
4.2.5.4	Tracciamento dei requisiti	47
4.2.6	DeleteDocuments	48
4.2.6.1	Descrizione	48
4.2.6.2	Esiti possibili	48
4.2.6.3	Lista sottocomponenti	48



4.2.6.4	Tracciamento dei requisiti	49
4.2.7	EmbedDocuments	50
4.2.7.1	Descrizione	50
4.2.7.2	Esiti possibili	50
4.2.7.3	Lista sottocomponenti	51
4.2.7.4	Tracciamento dei requisiti	52
4.2.8	EnableDocuments	53
4.2.8.1	Descrizione	53
4.2.8.2	Esiti possibili	53
4.2.8.3	Lista sottocomponenti	53
4.2.8.4	Tracciamento dei requisiti	54
4.2.9	GetChatMessages	54
4.2.9.1	Descrizione	54
4.2.9.2	Esiti possibili	54
4.2.9.3	Lista sottocomponenti	55
4.2.9.4	Tracciamento dei requisiti	55
4.2.10	GetChats	56
4.2.10.1	Descrizione	56
4.2.10.2	Esiti possibili	56
4.2.10.3	Lista sottocomponenti	56
4.2.10.4	Tracciamento dei requisiti	57
4.2.11	GetConfiguration	57
4.2.11.1	Descrizione	57
4.2.11.2	Esiti possibili	57
4.2.11.3	Lista sottocomponenti	57
4.2.11.4	Tracciamento dei requisiti	58
4.2.12	GetConfigurationOptions	59
4.2.12.1	Descrizione	59
4.2.12.2	Esiti possibili	59
4.2.12.3	Lista sottocomponenti	59
4.2.12.4	Tracciamento dei requisiti	60
4.2.13	GetDocumentContent	60
4.2.13.1	Descrizione	60
4.2.13.2	Esiti possibili	61
4.2.13.3	Lista sottocomponenti	61
4.2.13.4	Tracciamento dei requisiti	62
4.2.14	GetDocuments	62
4.2.14.1	Descrizione	62
4.2.14.2	Esiti possibili	62
4.2.14.3	Lista sottocomponenti	63
4.2.14.4	Tracciamento dei requisiti	64
4.2.15	RenameChat	64
4.2.15.1	Descrizione	64
4.2.15.2	Esiti possibili	64
4.2.15.3	Lista sottocomponenti	64
4.2.15.4	Tracciamento dei requisiti	65
4.2.16	SetConfiguration	65
4.2.16.1	Descrizione	65
4.2.16.2	Esiti possibili	66



4.2.16.3	Lista sottocomponenti	66
4.2.16.4	Tracciamento dei requisiti	67
4.2.17	UploadDocuments	67
4.2.17.1	Descrizione	67
4.2.17.2	Esiti possibili	67
4.2.17.3	Lista sottocomponenti	68
4.2.17.4	Tracciamento dei requisiti	69
4.3	Database	70
4.3.1	Configurazione del sistema	70
4.3.1.1	Diagramma ER	70
4.3.1.2	Definizione	70
4.3.2	Archiviazione chat	72
4.3.2.1	Diagramma ER	72
4.3.2.2	Definizione	72
4.4	Riepilogo tracciamento dei requisiti	73
5	Progettazione di dettaglio frontend	79
5.1	Chatbot	79
5.1.1	Lista delle sottocomponenti	79
5.1.1.1	ChatContent	79
5.1.1.2	ChatForm	79
5.1.1.3	ChatHeader	79
5.1.1.4	ChatListSideBar	79
5.1.1.5	MessageCard	79
5.2	Dashboard	80
5.2.1	Lista delle sottocomponenti	80
5.2.1.1	DashboardHeader	80
5.2.1.2	DocumentPreview	80
5.2.1.3	LatestChatContent	80
5.2.1.4	MessageCard	80
5.2.1.5	RecentChatsList	80
5.2.1.6	RecentlyViewedTab	81
5.2.1.7	RecentlyUploadedTab	81
5.3	Documents	81
5.3.1	Lista delle sottocomponenti	81
5.3.1.1	DocumentsList	81
5.3.1.2	DocumentsListRow	81
5.3.1.3	StagingArea	82
5.3.1.4	StagingAreaRow	82
5.4	DocumentView	82
5.4.1	Lista delle sottocomponenti	82
5.4.1.1	DocumentViewer	82
5.5	SettingsAppearance	82
5.5.1	Lista delle sottocomponenti	82
5.5.1.1	ThemeSelection	82
5.6	SettingsConfiguration	83
5.6.1	Lista delle sottocomponenti	83
5.6.1.1	ChangeLLMConfiguration	83
5.6.1.2	CurrentConfigurationCarousel	83



5.6.1.3	DocumentStoreCard	83
5.6.1.4	EmbeddingModelCard	83
5.6.1.5	LLMCard	84
5.6.1.6	VectorStoreCard	84
5.7	SetUp(Primo avvio)	84
5.7.1	Lista delle sottocomponenti	84
5.7.1.1	ConfirmConfiguration	84
5.7.1.2	DocumentStoreCard	84
5.7.1.3	DocumentStoreInit	84
5.7.1.4	EmbeddingModelCard	85
5.7.1.5	EmbeddingModelInit	85
5.7.1.6	LLMCard	85
5.7.1.7	LLMModelInit	85
5.7.1.8	VectorStoreCard	85
5.7.1.9	VectorStoreInit	85
5.8	SideBar	85
5.8.1	Lista delle sottocomponenti	85
5.8.1.1	NavigationMenu	85
5.8.1.2	SettingShortcut	85
6	Progettazione di dettaglio backend	86
6.1	AskChatbot	86
6.1.1	Diagramma delle classi	86
6.1.2	Lista delle sottocomponenti	87
6.1.2.1	AskChatbotController	87
6.1.2.2	AskChatbotLangchain	87
6.1.2.3	AskChatbotPort	87
6.1.2.4	AskChatbotService	87
6.1.2.5	AskChatbotUseCase	88
6.1.2.6	ChatHistoryManager	88
6.1.2.7	ChatId	88
6.1.2.8	ChatOperationResponse	88
6.1.2.9	Message	88
6.1.2.10	MessageResponse	88
6.1.2.11	MessageSender (Enumeration)	89
6.1.2.12	PersistChatPort	89
6.1.2.13	PostgresChat	89
6.1.2.14	PostgresChatOperationResponse	89
6.1.2.15	PostgresMessage	89
6.1.2.16	PostgresMessageSenderType(Enumeration)	90
6.1.2.17	PostgresPersistChat	90
6.1.2.18	PostgresChatORM	90
6.2	ChangeConfiguration	91
6.2.1	Diagramma delle classi	91
6.2.2	Lista delle sottocomponenti	91
6.2.2.1	ChangeConfigurationController	91
6.2.2.2	ChangeConfigurationPort	92
6.2.2.3	ChangeConfigurationPostgres	92
6.2.2.4	ChangeConfigurationService	92



6.2.2.5	ChangeConfigurationUseCase	92
6.2.2.6	ConfigurationOperationResponse	92
6.2.2.7	PostgresConfigurationOperationResponse	93
6.2.2.8	PostgresConfigurationORM	93
6.3	ConcealDocuments	94
6.3.1	Diagramma delle classi	94
6.3.2	Lista delle sottocomponenti	94
6.3.2.1	ConcealDocumentsController	94
6.3.2.2	ConcealDocumentsPort	95
6.3.2.3	ConcealDocumentsService	95
6.3.2.4	ConcealDocumentsUseCase	95
6.3.2.5	ConcealDocumentsVectorStore	95
6.3.2.6	DocumentId	96
6.3.2.7	DocumentOperationResponse	96
6.3.2.8	VectorStoreChromaDBManager	96
6.3.2.9	VectorStoreDocumentOperationResponse	97
6.3.2.10	VectorStoreManager	97
6.3.2.11	VectorStorePineconeManager	98
6.4	ConfigurationManager	99
6.4.1	Diagramma delle classi	99
6.4.2	Lista delle sottocomponenti	99
6.4.2.1	ConfigurationManager	99
6.4.2.2	PostgresConfiguration	100
6.4.2.3	PostgresConfigurationORM	100
6.4.2.4	PostgresDocumentStoreConfiguration	101
6.4.2.5	PostgresDocumentStoreType(Enum)	101
6.4.2.6	PostgresEmbeddingModelConfiguration	101
6.4.2.7	PostgresEmbeddingModelType(Enum)	101
6.4.2.8	PostgresLLMModelConfiguration	102
6.4.2.9	PostgresLLMModelType(Enum)	102
6.4.2.10	PostgresVectorStoreConfiguration	102
6.4.2.11	PostgresVectorStoreType(Enum)	102
6.5	DeleteChats	103
6.5.1	Diagramma delle classi	103
6.5.2	Lista delle sottocomponenti	103
6.5.2.1	ChatId	103
6.5.2.2	ChatOperationResponse	103
6.5.2.3	DeleteChatsController	104
6.5.2.4	DeleteChatsPort	104
6.5.2.5	DeleteChatsPostgres	104
6.5.2.6	DeleteChatsService	104
6.5.2.7	DeleteChatsUseCase	104
6.5.2.8	PostgresChatOperationResponse	105
6.5.2.9	PostgresChatORM	105
6.6	DeleteDocuments	106
6.6.1	Diagramma delle classi	106
6.6.2	Lista delle sottocomponenti	107
6.6.2.1	AWSDocumentOperationResponse	107
6.6.2.2	AWSS3Manager	107



6.6.2.3	DeleteDocuments	107
6.6.2.4	DeleteDocumentsAWSS3	108
6.6.2.5	DeleteDocumentsController	108
6.6.2.6	DeleteDocumentsEmbeddings	108
6.6.2.7	DeleteDocumentsPort	108
6.6.2.8	DeleteDocumentsService	109
6.6.2.9	DeleteDocumentsUseCase	109
6.6.2.10	DeleteEmbeddingsPort	109
6.6.2.11	DeleteEmbeddingsVectorStore	109
6.6.2.12	DocumentId	109
6.6.2.13	DocumentOperationResponse	109
6.6.2.14	VectorStoreChromaDBManager	110
6.6.2.15	VectorStoreDocumentOperationResponse	110
6.6.2.16	VectorStoreManager	110
6.6.2.17	VectorStorePineconeManager	110
6.7	EmbedDocuments	111
6.7.1	Diagramma delle classi	111
6.7.2	Lista delle sottocomponenti	115
6.7.2.1	AWSDocument	115
6.7.2.2	AWSS3Manager	115
6.7.2.3	Chunkerizer	115
6.7.2.4	DocumentId	115
6.7.2.5	DocumentOperationResponse	115
6.7.2.6	DocumentStatus	115
6.7.2.7	DOCXTextExtractor	115
6.7.2.8	EmbeddingsCreator	116
6.7.2.9	EmbeddingsUploader	116
6.7.2.10	EmbeddingsUploaderFacadeLangchain	116
6.7.2.11	EmbeddingsUploaderPort	117
6.7.2.12	EmbeddingsUploaderVectorStore	117
6.7.2.13	EmbedDocumentsController	117
6.7.2.14	EmbedDocumentsService	117
6.7.2.15	EmbedDocumentsUseCase	118
6.7.2.16	GetDocumentsContent	118
6.7.2.17	GetDocumentsContentAWSS3	118
6.7.2.18	GetDocumentsContentPort	118
6.7.2.19	GetDocumentsStatus	119
6.7.2.20	GetDocumentsStatusPort	119
6.7.2.21	GetDocumentsStatusVectorStore	119
6.7.2.22	HuggingFaceEmbeddingModel	119
6.7.2.23	LangchainDocument	120
6.7.2.24	LangchainEmbeddingModel	120
6.7.2.25	OpenAIEmbeddingModel	120
6.7.2.26	PDFTextExtractor	120
6.7.2.27	Status (Enumeration)	120
6.7.2.28	TextExtractor	121
6.7.2.29	VectorStoreChromaDBManager	121
6.7.2.30	VectorStoreDocumentOperationResponse	121
6.7.2.31	VectorStoreDocumentStatusResponse	121



6.7.2.32	VectorStoreManager	121
6.7.2.33	VectorStorePineconeManager	121
6.8	EnableDocuments	122
6.8.1	Diagramma delle classi	122
6.8.2	Lista delle sottocomponenti	122
6.8.2.1	DocumentId	122
6.8.2.2	DocumentOperationResponse	122
6.8.2.3	EnableDocumentsController	122
6.8.2.4	EnableDocumentsPort	123
6.8.2.5	EnableDocumentsService	123
6.8.2.6	EnableDocumentsUseCase	123
6.8.2.7	EnableDocumentsVectorStore	123
6.8.2.8	VectorStoreChromaDBManager	124
6.8.2.9	VectorStoreDocumentOperationResponse	124
6.8.2.10	VectorStoreManager	124
6.8.2.11	VectorStorePineconeManager	124
6.9	GetChatMessages	125
6.9.1	Diagramma delle classi	125
6.9.2	Lista delle sottocomponenti	125
6.9.2.1	Chat	125
6.9.2.2	ChatId	125
6.9.2.3	GetChatMessagesController	126
6.9.2.4	GetChatMessagesPort	126
6.9.2.5	GetChatMessagesPostgres	126
6.9.2.6	GetChatMessagesService	126
6.9.2.7	GetChatMessagesUseCase	126
6.9.2.8	Message	126
6.9.2.9	MessageSender	127
6.9.2.10	PostgresChat	127
6.9.2.11	PostgresChatORM	127
6.9.2.12	PostgresMessage	127
6.10	GetChats	127
6.10.1	Diagramma delle classi	127
6.10.2	Lista delle sottocomponenti	127
6.10.2.1	ChatFilter	127
6.10.2.2	ChatId	128
6.10.2.3	ChatPreview	128
6.10.2.4	GetChatsController	128
6.10.2.5	GetChatsPort	128
6.10.2.6	GetChatsPostgres	128
6.10.2.7	GetChatsService	128
6.10.2.8	GetChatsUseCase	129
6.10.2.9	Message	129
6.10.2.10	MessageSender	129
6.10.2.11	PostgresChatORM	129
6.10.2.12	PostgresChatPreview	129
6.10.2.13	PostgresMessage	129
6.10.2.14	PostgresMessageSenderType	129
6.11	GetConfiguration	130



6.11.1	Diagramma delle classi	130
6.11.2	Lista delle sottocomponenti	130
6.11.2.1	Configuration	130
6.11.2.2	DocumentStoreConfiguration	130
6.11.2.3	DocumentStoreType (Enumeration)	131
6.11.2.4	EmbeddingModelConfiguration	131
6.11.2.5	EmbeddingModelType (Enumeration)	131
6.11.2.6	GetConfigurationController	131
6.11.2.7	GetConfigurationPort	131
6.11.2.8	GetConfigurationPostgres	131
6.11.2.9	GetConfigurationService	132
6.11.2.10	GetConfigurationUseCase	132
6.11.2.11	LLMModelConfiguration	132
6.11.2.12	LLMModelType (Enumeration)	132
6.11.2.13	PostgresConfiguration	132
6.11.2.14	PostgresConfigurationORM	133
6.11.2.15	PostgresDocumentStoreConfiguration	133
6.11.2.16	PostgresDocumentStoreType	133
6.11.2.17	PostgresEmbeddingModelConfiguration	133
6.11.2.18	PostgresEmbeddingModelType	133
6.11.2.19	PostgresLLMModelConfiguration	133
6.11.2.20	PostgresLLMModelType	133
6.11.2.21	PostgresVectorStoreConfiguration	133
6.11.2.22	PostgresVectorStoreType	133
6.11.2.23	VectorStoreConfiguration	133
6.11.2.24	VectorStoreType (Enumeration)	133
6.12	GetConfigurationOptions	134
6.12.1	Diagramma delle classi	134
6.12.2	Lista delle sottocomponenti	134
6.12.2.1	ConfigurationOptions	134
6.12.2.2	DocumentStoreConfiguration	135
6.12.2.3	DocumentStoreType	135
6.12.2.4	EmbeddingModelConfiguration	135
6.12.2.5	EmbeddingModelType	135
6.12.2.6	GetConfigurationOptions	135
6.12.2.7	GetConfigurationOptionsPort	135
6.12.2.8	GetConfigurationOptionsPostgres	135
6.12.2.9	GetConfigurationOptionsService	136
6.12.2.10	GetConfigurationOptionsUseCase	136
6.12.2.11	LLMModelConfiguration	136
6.12.2.12	LLMModelType	136
6.12.2.13	PostgresConfigurationORM	136
6.12.2.14	PostgresDocumentStoreConfiguration	136
6.12.2.15	PostgresDocumentStoreType	136
6.12.2.16	PostgresEmbeddingModelConfiguration	136
6.12.2.17	PostgresEmbeddingModelType	136
6.12.2.18	PostgresLLMModelConfiguration	136
6.12.2.19	PostgresLLMModelType	136
6.12.2.20	PostgresVectorStoreConfiguration	137



6.12.2.21	PostgresVectorStoreType	137
6.12.2.22	VectorStoreConfiguration	137
6.12.2.23	VectorStoreType	137
6.13	GetDocumentContent	138
6.13.1	Diagramma delle classi	138
6.13.2	Lista delle sottocomponenti	139
6.13.2.1	AWSDocument	139
6.13.2.2	AWSS3Manager	139
6.13.2.3	Document	139
6.13.2.4	DocumentContent	139
6.13.2.5	DocumentId	139
6.13.2.6	DocumentMetadata	139
6.13.2.7	DocumentStatus	139
6.13.2.8	DocumentType (Enumeration)	139
6.13.2.9	GetDocumentContentController	139
6.13.2.10	GetDocumentsContentFacadeService	139
6.13.2.11	GetDocumentsContent	140
6.13.2.12	GetDocumentsContentAWSS3	140
6.13.2.13	GetDocumentsContentPort	140
6.13.2.14	GetDocumentsStatus	140
6.13.2.15	GetDocumentsStatusPort	140
6.13.2.16	GetDocumentsStatusVectorStore	140
6.13.2.17	GetDocumentsContentUseCase	140
6.13.2.18	PlainDocument	140
6.13.2.19	Status (Enumeration)	140
6.13.2.20	VectorStoreChromaDBManager	140
6.13.2.21	VectorStoreDocumentStatusResponse	141
6.13.2.22	VectorStoreManager	141
6.13.2.23	VectorStorePineconeManager	141
6.14	GetDocuments	142
6.14.1	Diagramma delle classi	142
6.14.2	Lista delle sottocomponenti	143
6.14.2.1	AWSDocumentMetadata	143
6.14.2.2	AWSS3Manager	143
6.14.2.3	DocumentFilter	143
6.14.2.4	DocumentId	143
6.14.2.5	DocumentMetadata	143
6.14.2.6	DocumentStatus	143
6.14.2.7	DocumentType (Enumeration)	143
6.14.2.8	ElaborationException	144
6.14.2.9	GetDocumentsController	144
6.14.2.10	GetDocumentsFacadeService	144
6.14.2.11	GetDocumentsListAWSS3	144
6.14.2.12	GetDocumentsMetadata	145
6.14.2.13	GetDocumentsMetadataPort	145
6.14.2.14	GetDocumentsStatus	145
6.14.2.15	GetDocumentsStatusPort	145
6.14.2.16	GetDocumentsStatusVectorStore	145
6.14.2.17	GetDocumentsUseCase	145



6.14.2.18	LightDocument	145
6.14.2.19	Status (Enumeration)	145
6.14.2.20	VectorStoreChromaDBManager	145
6.14.2.21	VectorStoreDocumentStatusResponse	146
6.14.2.22	VectorStoreManager	146
6.14.2.23	VectorStorePineconeManager	146
6.15	RenameChat	146
6.15.1	Diagramma delle classi	146
6.15.2	Lista delle sottocomponenti	146
6.15.2.1	ChatId	146
6.15.2.2	ChatOperationResponse	147
6.15.2.3	RenameChatController	147
6.15.2.4	RenameChatPort	147
6.15.2.5	RenameChatPostgres	147
6.15.2.6	RenameChatService	147
6.15.2.7	RenameChatUseCase	147
6.15.2.8	PostgresChatOperationResponse	148
6.15.2.9	PostgresChatORM	148
6.16	SetConfiguration	148
6.16.1	Diagramma delle classi	148
6.16.2	Lista delle sottocomponenti	148
6.16.2.1	ConfigurationOperationResponse	148
6.16.2.2	PostgresConfigurationOperationResponse	148
6.16.2.3	PostgresConfigurationORM	149
6.16.2.4	SetConfigurationController	149
6.16.2.5	SetConfigurationPort	149
6.16.2.6	SetConfigurationPostgres	149
6.16.2.7	SetConfigurationService	150
6.16.2.8	SetConfigurationUseCase	150
6.17	UploadDocuments	151
6.17.1	Diagramma delle classi	151
6.17.2	Lista delle sottocomponenti	154
6.17.2.1	AWSDocument	154
6.17.2.2	AWSDocumentOperationResponse	154
6.17.2.3	AWSS3Manager	154
6.17.2.4	Chunkerizer	154
6.17.2.5	Document	154
6.17.2.6	DocumentContent	154
6.17.2.7	DocumentId	154
6.17.2.8	DocumentMetadata	154
6.17.2.9	DocumentOperationResponse	154
6.17.2.10	DocumentStatus	154
6.17.2.11	DocumentType (Enumeration)	154
6.17.2.12	DocumentsUploader	155
6.17.2.13	DocumentUploaderAWSS3	155
6.17.2.14	DocumentsUploaderPort	155
6.17.2.15	DOCXTextExtractor	155
6.17.2.16	EmbeddingsCreator	155
6.17.2.17	EmbeddingsUploader	155



6.17.2.18 EmbeddingsUploaderFacadeLangchain	156
6.17.2.19 EmbeddingsUploaderPort	156
6.17.2.20 EmbeddingsUploaderVectorStore	156
6.17.2.21 HuggingFaceEmbeddingModel	156
6.17.2.22 LangchainDocument	156
6.17.2.23 LangchainEmbeddingModel	156
6.17.2.24 NewDocument	156
6.17.2.25 OpenAIEmbeddingModel	156
6.17.2.26 PDFTextExtractor	156
6.17.2.27 PlainDocument	156
6.17.2.28 Status (Enumeration)	157
6.17.2.29 TextExtractor	157
6.17.2.30 UploadDocumentsController	157
6.17.2.31 UploadDocumentsService	157
6.17.2.32 UploadDocumentsUseCase	157
6.17.2.33 VectorStoreChromaDBManager	157
6.17.2.34 VectorStoreDocumentOperationResponse	158
6.17.2.35 VectorStoreManager	158
6.17.2.36 VectorStorePineconeManager	158



Elenco delle figure

1	Diagramma ER: Configurations	70
2	Diagramma ER: Chats	72
3	Diagramma delle classi della componente AskChatbot	86
4	Diagramma delle classi della componente ChangeConfiguration	91
5	Diagramma delle classi della componente ConcealDocuments	94
6	Diagramma delle classi della componente ConfigurationManager	99
7	Diagramma delle classi della componente DeleteChats	103
8	Diagramma delle classi della componente DeleteDocuments	106
9	Diagramma delle classi componente EmbedDocuments (continua nei prossimi diagrammi)	111
10	Diagramma delle classi componente EmbedDocuments (continua nel prossimo diagramma)	112
11	Diagramma delle classi componente EmbedDocuments (continua nel prossimo diagramma)	113
12	Diagramma delle classi componente EmbedDocuments	114
13	Diagramma delle classi della componente EnableDocuments	122
14	Diagramma delle classi della componente GetChatMessages	125
15	Diagramma delle classi della componente GetChats	127
16	Diagramma delle classi della componente GetConfiguration	130
17	Diagramma delle classi della componente GetConfigurationOptions	134
18	Diagramma delle classi componente GetDocumentContent	138
19	Diagramma delle classi componente GetDocuments	142
20	Diagramma delle classi componente RenameChat	146
21	Diagramma delle classi componente SetConfiguration	148
22	Diagramma delle classi componente UploadDocuments (continua nei prossimi diagrammi)	151
23	Diagramma delle classi componente UploadDocuments (continua nel prossimo diagramma)	152
24	Diagramma delle classi componente UploadDocuments	153



Elenco delle tabelle

1	Tracciamento dei requisiti nella componente Chatbot	35
2	Tracciamento dei requisiti nella componente Documents	37
3	Tracciamento dei requisiti nella componente SettingsConfiguration	38
4	Tracciamento dei requisiti nella componente SetUp	40
5	Esiti possibili AskChatbot	41
6	Tracciamento dei requisiti nella componente AskChatbot	42
7	Esiti possibili ChangeConfiguration	43
8	Tracciamento dei requisiti nell componente ChangeConfiguration	44
9	Esiti possibili ConcealDocuments	44
10	Tracciamento dei requisiti nella componente ConcealDocuments	45
11	Esiti possibili	46
12	Esiti possibili DeleteChats	47
13	Tracciamento dei requisiti nella componente DeleteChats	48
14	Esiti possibili DeleteDocuments	48
15	Tracciamento dei requisiti nella componente DeleteDocuments	49
16	Esiti possibili EmbedDocuments	50
17	Tracciamento dei requisiti nella componente EmbedDocuments	53
18	Esiti possibili EnableDocuments	53
19	Tracciamento dei requisiti nella componente EnableDocuments	54
20	Esiti possibili GetChatMessages	55
21	Tracciamento dei requisiti nella componente GetChatMessages	55
22	Esiti possibili GetChats	56
23	Esiti possibili GetConfiguration	57
24	Tracciamento dei requisiti nella componente GetConfiguration	59
25	Esiti possibili GetConfigurationOptions	59
26	Esiti possibili GetDocumentContent	61
27	Tracciamento dei requisiti nella componente GetDocumentContent	62
28	Esiti possibili GetDocuments	62
29	Tracciamento dei requisiti nella componente GetDocuments	64
30	Esiti possibili RenameChat	64
31	Tracciamento dei requisiti nella componente RenameChat	65
32	Esiti possibili SetConfiguration	66
33	Tracciamento dei requisiti nella componente SetConfiguration	67
34	Esiti possibili UploadDocuments	67
35	Tracciamento dei requisiti nella componente UploadDocuments	69
36	Tabella riepilogativa del tracciamento dei requisiti	73



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_{lg}Management_{lg}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 *lg* 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 v3.0.0(0))*;
- *Regolamento del progetto didattico*:
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf>
(Ultimo accesso: 2024-03-29);
- *Standard ISO/IEC 9126*:
https://it.wikipedia.org/wiki/ISO/IEC_9126
(Ultimo accesso: 2024-03-29).

1.3.2 Riferimenti informativi

- *(Analisi dei requisiti v3.0.0(0))*;
- *Capitolato_{lg}C1: Knowledge Management AI*
 - <https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C1.pdf>
(Ultimo accesso: 2024-03-29);
 - <https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C1p.pdf>
(Ultimo accesso: 2024-03-29).
- *Dispense su Dependency Injection*:
<https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Architetturali%20-%20Dependency%20Injection.pdf>
(Ultimo accesso: 2024-03-29);
- *Dispense su OOP*:
<https://www.math.unipd.it/~rcardin/swea/2023/Object-Oriented%20Programming%20Principles%20Revised.pdf>
(Ultimo accesso: 2024-03-29);



- *Dispense su Diagrammi delle classi:*
<https://www.math.unipd.it/~rcardin/swea/2023/Diagrammi%20delle%20Classi.pdf>
(Ultimo accesso: 2024-03-29);
- *Dispense su Pattern architetturali:*
<https://www.math.unipd.it/~rcardin/swea/2022/Software%20Architecture%20Patterns.pdf>
(Ultimo accesso: 2024-03-29);
- *Dispense su Pattern creazionali:*
<https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Creazionali.pdf>
(Ultimo accesso: 2024-03-29);
- *Dispense su Pattern strutturali:*
<https://www.math.unipd.it/~rcardin/swea/2022/Design%20Pattern%20Strutturali.pdf>
(Ultimo accesso: 2024-03-29);
- *Dispense su Principi SOLID:*
https://www.math.unipd.it/~rcardin/swea/2021/SOLID%20Principles%20of%20Object-Oriented%20Design_4x4.pdf
(Ultimo accesso: 2024-03-29);
- *Dispense sulla Progettazione software (argomento T6):*
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T6.pdf>
(Ultimo accesso: 2024-03-29);
- *Dispense sulla Qualità del software (argomento T7):*
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T7.pdf>
(Ultimo accesso: 2024-03-29);
- *Repository su Architettura esagonale:*
<https://github.com/rcardin/hexagonal>
(Ultimo accesso: 2024-03-29); <https://github.com/rcardin/hexagonal-java/>
(Ultimo accesso: 2024-03-29);
- *Repository Ingegneria del software_{ig} professor Cardin:*
<https://github.com/rcardin/swe-imdb>
(Ultimo accesso: 2024-03-29);
- Riferimenti a scelte tecnologiche:
 - *AWS_{ig}S3:*
<https://aws.amazon.com/it/s3/>
(Ultimo accesso: 2024-03-29);
 - *ChromaDB:*
<https://www.trychroma.com/>
(Ultimo accesso: 2024-03-29);
 - *Docker:*
<https://www.docker.com/>
(Ultimo accesso: 2024-03-29);
 - *Flask_{ig}:*
<https://flask.palletsprojects.com/en/3.0.x/>
(Ultimo accesso: 2024-03-29);



- *HuggingFace*_{lgl}:
<https://huggingface.co/>
(Ultimo accesso: 2024-03-29);
- *Langchain*_{lgl}:
https://python.langchain.com/docs/get_started/introduction
(Ultimo accesso: 2024-03-29);
- *Next.js*_{lgl}:
<https://nextjs.org/>
(Ultimo accesso: 2024-03-29);
- *OpenAI*_{lgl}:
<https://openai.com/>
(Ultimo accesso: 2024-03-29));
- *Pinecone*_{lgl}:
<https://www.pinecone.io/>
(Ultimo accesso: 2024-03-29));
- *Postgres*_{lgl}:
<https://www.postgresql.org/>
(Ultimo accesso: 2024-03-29));
- *Python*_{lgl}:
<https://www.python.org/>
(Ultimo accesso: 2024-03-29));
- *Typescript*_{lgl}:
<https://www.typescriptlang.org/>
(Ultimo accesso: 2024-03-29));
- *(Glossario v3.0.0(0))*;
- *(Piano di progetto v3.0.0(0))*;
- *(Piano di qualifica v3.0.0(0))*;
- *Verballi interni*:
 - 2024-02-19;
 - 2024-02-26;
 - 2024-03-07;
 - 2024-03-12;
 - 2024-03-19;
 - 2024-03-25;
 - 2024-04-01;
 - 2024-04-12.
- *Verballi esterni*:
 - 2023-02-28;
 - 2023-03-13;



- 2023-03-26.



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*_{lgj} per applicazioni web. È stato scelto per la sua leggerezza e la sua flessibilità, rispetto ad altri framework come *Django*_{lgj} 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*_{lgj} 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*_{lgj} 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_{lgl}*. È progettato per essere altamente leggibile.

Rispetto ad altri linguaggi come *Java_{lgl}* o *C++_{lgl}*, 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_{lgl}*;
- **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 3.12-slim

2.2 Next.js

Next.js è un framework per applicazioni web basato su *React_{lgl}*. È stato scelto per la sua *efficienza_{lgl}* e per le sue funzionalità di rendering lato server. Rispetto ad altri framework come *Angular_{lgl}*, 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_{gl}* 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_{IGI} di sviluppo.

Versione scelta: Typescript 5.3.3.

2.3 Docker

Docker_{IGI} 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_{IGI}:** 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_{IGI}* o altri servizi web. È stato scelto per la sua comodità e flessibilità. Rispetto ad altri framework come *TensorFlow_{IGI}* o *PyTorch_{IGI}*,



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_{lgl}* o *Annoy_{lgl}*, 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_{IGI} 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_{IGI} 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_{IGI}* (LLM) che per gli *embeddings_{IGI}* 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_{gl} 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_{igl} (Simple Storage Service) è un servizio di storage di oggetti offerto da Amazon Web Services_{igl}. È stato scelto per la sua scalabilità, affidabilità, e sicurezza. Rispetto ad altre soluzioni di storage come Google Cloud Storage_{igl} o Azure Blob Storage_{igl}, 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_{igl}, 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_{igl}, 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_{igl} come MySQL_{igl} o SQLite_{igl}, 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 ($MVCC_{igl}$), punti di controllo, logging di scrittura anticipata (WAL_{igl}) 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_{igl} tradizionale che come una soluzione $NoSQL_{igl}$ 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_{IGI}* 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 tre componenti: *frontend_{IGI}* e *backend_{IGI}* e database.

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 archiviazione documenti, il *vector store_{igl}*, il database delle chat e con i modelli di intelligenza artificiale necessari per il corretto funzionamento dell'applicazione.

3.2.3 Database

Il database è responsabile della memorizzazione dei dati relativi alle chat, tra cui messaggi e relativi documenti rilevanti, e alla configurazione attuale dell'utente.

3.3 Assemblaggio delle componenti

3.3.1 Docker

Le componenti sono assemblate tramite Docker Compose per facilitare l'esecuzione e la gestione di più container docker.

In particolare sono stati prodotti i seguenti container Docker:

- **database**: espone l'istanza del database nella porta 5432, permettendo al backend di accedere ai dati relativi alle chat e alla configurazione dell'utente;
- **flaskapp**: espone la componente backend nella porta 4000, dando al frontend la possibilità di chiamare i servizi offerti;
- **nextapp**: 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. In particolare, all'interno della cartella app sono presenti le seguenti cartelle, che nel framework sopracitato corrispondono alle route visitabili dall'utente:

- **chatbot**: contiene i componenti relativi alla gestione e alla visualizzazione delle chat;
- **configuration**: contiene i componenti relativi alla configurazione da eseguire durante il primo avvio dell'applicazione;
- **documents**: contiene i componenti relativi alla gestione e alla visualizzazione dei documenti;
- **settings**: contiene i componenti relativi alla gestione delle impostazioni dell'applicazione, cioè il suo aspetto e la configurazione scelta.

La root del frontend corrisponde alla pagina **dashboard**, che contiene i componenti per il riepilogo e la navigazione tra le varie sezioni dell'applicazione. I componenti grafici di base, condivisi tra le varie interfacce, sono contenuti nella cartella **components**, localizzata allo stesso livello di app.

3.4.2 Backend

La struttura organizzativa del backend segue la seguente struttura:

```
backend
├── adapter
│   └── _in
```



```
|
| |
| | | web-- controllers
| | |
| | | out -- implementazioni di Outbound Port
| | |
| | | application
| | | |
| | | | port
| | | | |
| | | | | _in -- Inbound Ports (Use Cases)
| | | | |
| | | | | out -- Outbound Ports
| | | | |
| | | | | service -- implementazioni di Inbound Port
| | | | |
| | | | | blueprints
| | | | |
| | | | | domain -- classi di business
```

Questa struttura riflette il modello architetturale scelto, facilitando il passaggio da progettazione a *codifica*_[g].



4 Architettura delle componenti

4.1 Frontend

4.1.1 Chatbot

4.1.1.1 Descrizione

4.1.1.2 Lista di sottocomponenti

- **ChatContent**: componente che si occupa di mostrare i dati della chat selezionata e di interagire con il *chatbot_{lgI}*;
- **ChatForm**: elemento che permette all'utente di inserire i messaggi e le richieste da fare al Chatbot;
- **ChatHeader**: componente che permette di visualizzare il titolo e le interazioni possibili con la chat corrente;
- **ChatListSideBar**: componente che permette di navigare tra le chat salvate nel sistema e di visualizzarne le preview;
- **MessageCard**: componente che rappresenta un messaggio in chat e tutte le sue informazioni.

4.1.1.3 Tracciamento dei requisiti

Tabella 1: Tracciamento dei requisiti nella componente Chatbot

Requisito_{lgI}	Soddisfacimento del requisito
RF.O.34	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1.1	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1.2	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1.3	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1.4	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.35	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.36.1	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.37.1	ChatHeader: vedi (§5.1.1.3) .
RF.O.38.1	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.38.2	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.40.1	ChatHeader: vedi (§5.1.1.3) .
RF.O.41.1	ChatHeader: vedi (§5.1.1.3) .
RF.O.43	ChatContent: vedi (§5.1.1.1) .
RF.O.43.1	ChatContent: vedi (§5.1.1.1) .
RF.O.43.1.1	MessageCard: vedi (§5.1.1.5) .
RF.O.43.1.2	MessageCard: vedi (§5.1.1.5) .
RF.D.44	MessageCard: vedi (§5.1.1.5) .
RF.D.48	MessageCard: vedi (§5.1.1.5) .
RF.O.51	ChatForm: vedi (§5.1.1.2) .

Continua nella pagina successiva



Tabella 1: Tracciamento dei requisiti nella componente Chatbot (cont)

Requisito	Soddisfacimento del requisito
RF.O.51.1	ChatForm: vedi (§5.1.1.2) .
RF.O.51.2	ChatForm: vedi (§5.1.1.2) .
RF.O.51.2.1	ChatForm: vedi (§5.1.1.2) .
RF.O.51.2.2	ChatForm: vedi (§5.1.1.2) .
RF.O.53	MessageCard: vedi (§5.1.1.5) .

4.1.2 Dashboard

4.1.2.1 Descrizione

Pagina che raccoglie alcune funzionalità importanti dell'applicazione e i documenti e chat con cui l'utente ha interagito più recentemente.

4.1.2.2 Lista di sottocomponenti

- **DashboardHeader**: elemento che permette di accedere velocemente a delle funzioni principali dell'applicazione;
- **DocumentPreview**: componente pop-up che permette di vedere una preview delle informazioni del documento in archivio a cui si riferisce;
- **LatestChatContent**: componente che mostra il contenuto della chat con l'interazione più recente da parte dell'utente;
- **MessageCard**: vedi (§4.1.1.2) ;
- **RecentChatsList**: elenco di chat ordinate cronologicamente con le quali l'utente ha interagito più recentemente;
- **RecentlyViewedTab**: elenco di documenti che l'utente ha visualizzato più recentemente;
- **RecentlyUploadedTab**: elenco di documenti che l'utente ha caricato più recentemente.

4.1.2.3 Tracciamento dei requisiti

-

4.1.3 Documents

4.1.3.1 Descrizione

4.1.3.2 Lista di sottocomponenti

- **DocumentsList**: componente che permette di navigare tra i documenti presenti nel sistema e di interagire con essi;
- **DocumentsListRow**: componente che permette di visualizzare le informazioni del documento, di eliminarlo o di modificarne lo status;
- **StagingArea**: component componente che permette di caricare i documenti e successivamente di fare l'upload degli stessi;



- **StagingAreaRow**: componente che permette di interagire con i documenti caricati nell'area di staging.

4.1.3.3 Tracciamento dei requisiti

Tabella 2: Tracciamento dei requisiti nella componente Documents

Requisito	Soddisfacimento del requisito
RF.O.15	DocumentsList: vedi (§5.3.1.1) .
RF.O.15.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.2	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.3	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.4	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.16	StagingArea: vedi (§5.3.1.3) .
RF.O.16.1	StagingArea: vedi (§5.3.1.3) .
RF.O.16.2	StagingArea: vedi (§5.3.1.3) .
RF.O.17	StagingArea: vedi (§5.3.1.3) .
RF.O.17.1	StagingArea: vedi (§5.3.1.3) .
RF.O.17.2	StagingArea: vedi (§5.3.1.3) .
RF.O.17.3	StagingArea: vedi (§5.3.1.3) .
RF.O.18	StagingArea: vedi (§5.3.1.3) .
RF.O.18.1	StagingArea: vedi (§5.3.1.3) .
RF.O.19	StagingAreaRow: vedi (§5.3.1.4) .
RF.O.20	StagingAreaRow: vedi (§5.3.1.4) .
RF.O.21.1	StagingArea: vedi (§5.3.1.3) .
RF.O.21.2	StagingArea: vedi (§5.3.1.3) .
RF.O.21.3	StagingArea: vedi (§5.3.1.3) .
RF.O.21.4	StagingArea: vedi (§5.3.1.3) .
RF.O.21.5	StagingArea: vedi (§5.3.1.3) .
RF.O.21.6	StagingArea: vedi (§5.3.1.3) .
RF.O.22.1.1	StagingArea: vedi (§5.3.1.3) .
RF.O.23.1.1	StagingArea: vedi (§5.3.1.3) .
RF.O.26.1.1	DocumentsList: vedi (§5.3.1.1) .
RF.O.27.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.28.1.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.30.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.32.1	DocumentsListRow: vedi (§5.3.1.2) .

4.1.4 DocumentView

4.1.4.1 Descrizione

Componente che permette visualizzare l'intero contenuto del documento salvato nel sistema di archiviazione e altre sue informazioni rilevanti.



4.1.4.2 Lista di sottocomponenti

- **DocumentViewer**: componente che permette di visionare per intero il documento selezionato.

4.1.5 SettingsAppearance

4.1.5.1 Descrizione

Pagina della finestra Settings che permette all'utente di modificare il tema dell'applicazione.

4.1.5.2 Lista di sottocomponenti

- **ThemeSelection**: componente che permette di scegliere il tema dell'applicazione.

4.1.5.3 Tracciamento dei requisiti

-

4.1.6 SettingsConfiguration

4.1.6.1 Descrizione

Pagina della finestra Settings che permette di visualizzare e modificare alcuni elementi della configurazione del sistema.

4.1.6.2 Lista di sottocomponenti

- **ChangeLLMConfiguration**: sezione che consente di cambiare LLM Model configurato;
- **CurrentConfigurationCarousel**: componente che permette di conoscere la configurazione corrente di vector store, sistema di archiviazione documenti, embedding model e LLM;
- **DocumentStoreCard**: elemento che contiene informazioni rilevanti di un sistema di archiviazione dei documenti;
- **EmbeddingModelCard**: elemento che contiene informazioni rilevanti di un modello di embeddings;
- **LLMCard**: elemento che contiene informazioni rilevanti di un modello LLM;
- **VectorStoreCard**: elemento che contiene informazioni rilevanti di un vector store.

4.1.6.3 Tracciamento dei requisiti

Tabella 3: Tracciamento dei requisiti nella componente SettingsConfiguration

Requisito	Soddisfacimento del requisito
RF.O.2	ChangeLLMConfiguration: vedi (§5.6.1.1) .
RF.O.2.1	ChangeLLMConfiguration: vedi (§5.6.1.1) .

Continua nella pagina successiva



Tabella 3: Tracciamento dei requisiti nella componente SettingsConfiguration (cont)

Requisito	Soddisfacimento del requisito
RF.O.6.1	ChangeLLMConfiguration: vedi (§5.6.1.1) .
RF.O.10	LLMCard: vedi (§5.6.1.5) .
RF.O.10.1	LLMCard: vedi (§5.6.1.5) .
RF.O.10.2	LLMCard: vedi (§5.6.1.5) .
RF.O.10.3	LLMCard: vedi (§5.6.1.5) .
RF.O.10.4	LLMCard: vedi (§5.6.1.5) .
RF.O.10.5	LLMCard: vedi (§5.6.1.5) .
RF.O.11	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.1	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.2	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.3	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.4	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.5	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.12	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.1	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.2	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.3	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.4	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.5	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.13	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.1	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.2	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.3	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.4	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.5	DocumentStoreCard: vedi (§5.6.1.3) .

4.1.7 SetUp (Primo avvio)

4.1.7.1 Descrizione

Pagina che permette di inizializzare le configurazioni di vector store, sistema di archiviazione dei documenti, modello di embeddings e LLM al primo avvio dell'applicazione.

4.1.7.2 Lista di sottocomponenti

- : per confermare la configurazione scelta;
- **DocumentStoreCard**: vedi (§4.1.6.2) ;
- **DocumentStoreInit**: componente che permette di scegliere e inizializzare la configurazione del sistema di archiviazione dei documenti;
- **EmbeddingModelCard**: vedi (§4.1.6.2) ;
- **EmbeddingModelInit**: componente che permette di scegliere e inizializzare la configurazione del modello di embedding;
- **LLMCard**: vedi (§4.1.6.2) ;



- **LLMModelInit**: componente che permette di scegliere e inizializzare la configurazione del modello LLM;
- **VectorStoreCard**: vedi (§4.1.6.2) ;
- **VectorStoreInit**: componente che permette di scegliere e inizializzare la configurazione del vector store.

4.1.7.3 Tracciamento dei requisiti

Tabella 4: Tracciamento dei requisiti nella componente SetUp

Requisito	Soddisfacimento del requisito
RF.O.1.1.1	LLMModelInit: vedi (§5.7.1.7) .
RF.O.1.2.1	VectorStoreInit: vedi (§5.7.1.9) .
RF.O.1.3.1	EmbeddingModelInit: vedi (§5.7.1.5) .
RF.O.1.4.1	DocumentStoreInit: vedi (§5.7.1.3) .
RF.O.3	VectorStoreInit: vedi (§5.7.1.9) .
RF.O.3.1	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.4	EmbeddingModelInit: vedi (§5.7.1.5) .
RF.O.4.1	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.5	DocumentStoreInit: vedi (§5.7.1.3) .
RF.O.5.1	DocumentStoreCard: vedi (§5.6.1.3) .

4.1.8 SideBar

4.1.8.1 Descrizione

Componente fissa dell'interfaccia utente che permette di navigare tramite tabs le seguenti pagine dell'applicazione:

- Chatbot;
- Dashboard;
- Documents;
- Settings.

4.1.8.2 Lista di sottocomponenti

- **NavigationMenu**: componente che permette di raggiungere altre pagine dell'applicazione;
- **SettingShortcut**: componente che permette di raggiungere la pagina Settings per modificare le impostazioni di sistema.

4.1.8.3 Tracciamento dei requisiti

-



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

4.2.1.2 Esiti possibili

Tabella 5: Esiti possibili AskChatbot

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

4.2.1.3 Lista sottocomponenti

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



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

4.2.1.4 Tracciamento dei requisiti

Tabella 6: Tracciamento dei requisiti nella componente AskChatbot

Requisito	Soddisfacimento del requisito
RF.O.35	PostgresChatORM: vedi (§6.1.2.18) .
RF.O.36	PostgresChatORM: vedi (§6.1.2.18) .
RF.O.52	AskChatbotService: vedi (§6.1.2.4) .
RF.D.56	AskChatbotService: vedi (§6.1.2.4) .
RF.OP.57	AskChatbotService: vedi (§6.1.2.4) .
RF.O.59	AskChatbotService: vedi (§6.1.2.4) .
RF.O.60	AskChatbotService: vedi (§6.1.2.4) .

4.2.2 ChangeConfiguration

4.2.2.1 Descrizione

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

- **Route API_{lg}** : `/changeConfiguration`;
- **Metodo**: POST;
- **Lista parametri HTTP**:



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

4.2.2.2 Esiti possibili

Tabella 7: Esiti possibili `ChangeConfiguration`

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

4.2.2.3 Lista sottocomponenti

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

4.2.2.4 Tracciamento dei requisiti



Tabella 8: Tracciamento dei requisiti nell componente ChangeConfiguration

Requisito	Soddisfacimento del requisito
RF.O.6	ChangeConfigurationService: vedi (§6.2.2.4).

4.2.3 ConcealDocuments

4.2.3.1 Descrizione

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

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

4.2.3.2 Esiti possibili

Tabella 9: Esiti possibili ConcealDocuments

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

4.2.3.3 Lista sottocomponenti

- **ConcealDocumentsController:** classe controller che si occupa del passaggio di una lista di `DocumentId` allo use case `ConcealDocumentsUseCase` a partire da una lista di stringhe che rappresentano l'id dei documenti da occultare;
- **ConcealDocumentsPort:** interfaccia che rappresenta la porta per effettuare l'occultamento dei documenti nel vector store;
- **ConcealDocumentsService:** classe service che implementa lo use case `ConcealDocumentsUseCase`;



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

4.2.3.4 Tracciamento dei requisiti

Tabella 10: Tracciamento dei requisiti nella componente ConcealDocuments

Requisito	Soddisfacimento del requisito
RF.O.29	ConcealDocumentsService: vedi (§6.3.2.3) .
RF.O.30	ConcealDocumentsService: vedi (§6.3.2.3) .

4.2.4 ConfigurationManager

4.2.4.1 Descrizione

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

È costituita da:

- **Route API:**
- **Metodo:**
- **Lista parametri HTTP:**

4.2.4.2 Esiti possibili ConfigurationManager



Tabella 11: Esiti possibili

Codice	Descrizione	Risposta
-	-	-

4.2.4.3 Tracciamento dei requisiti

•

4.2.4.4 Lista sottocomponenti

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

4.2.5 DeleteChats

4.2.5.1 Descrizione

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

- **Route API**: `/deleteChats`;
- **Metodo**: `POST`;
- **Lista parametri HTTP**:



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

4.2.5.2 Esiti possibili

Tabella 12: Esiti possibili DeleteChats

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

4.2.5.3 Lista sottocomponenti

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

4.2.5.4 Tracciamento dei requisiti



Tabella 13: Tracciamento dei requisiti nella componente DeleteChats

Requisito	Soddisfacimento del requisito
RF.O.37	DeleteChatsService: vedi (§6.5.2.6) .
RF.O.38	DeleteChatsService: vedi (§6.5.2.6) .

4.2.6 DeleteDocuments

4.2.6.1 Descrizione

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

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

4.2.6.2 Esiti possibili

Tabella 14: Esiti possibili DeleteDocuments

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

4.2.6.3 Lista sottocomponenti

- **AWSDocumentOperationResponse:** classe che contiene l'esito di un'operazione effettuata su un AWS S3 riguardo un documento;
- **AWSS3Manager:** classe che si occupa di effettuare le operazioni sul bucket di Amazon S3 configurato, cioè il sistema di archiviazione documenti dell'applicazione;



- **DeleteDocuments**: classe che si occupa di inoltrare la richiesta di eliminare una lista di documenti nella porta esterna DeleteDocumentsPort diretta verso il sistema di archiviazione;
- **DeleteDocumentsAWS3**: classe adapter che implementa la porta DeleteDocumentsPort, adattando la chiamata di DeleteDocuments a AWS3Manager;
- **DeleteDocumentsController**: classe controller che si occupa del passaggio di una lista di DocumentId allo use case DeleteDocumentsUseCase a partire da una lista di stringhe che rappresentano gli id dei documenti da eliminare;
- **DeleteDocumentsEmbeddings**: classe che si occupa di eliminare gli embeddings di una lista di documenti nella porta esterna DeleteEmbeddingsPort diretta verso il vector store;
- **DeleteDocumentsPort**: interfaccia che rappresenta la porta in uscita per eliminare una lista di documenti dal sistema di archiviazione;
- **DeleteDocumentsService**: classe service che implementa lo use case DeleteDocumentsUseCase;
- **DeleteDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per eliminare i documenti;
- **DeleteEmbeddingsPort**: interfaccia che rappresenta la porta in uscita per eliminare gli embeddings di una lista di documenti dal vector store;
- **DeleteEmbeddingsVectorStore**: classe adapter che implementa la porta DeleteEmbeddingsPort, adattando la chiamata di DeleteDocumentsEmbeddings a VectorStoreManager;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.6.4 Tracciamento dei requisiti

Tabella 15: Tracciamento dei requisiti nella componente DeleteDocuments

Requisito	Soddisfacimento del requisito
RF.O.25	DeleteDocumentsService: vedi (§6.6.2.8) .
RF.O.26	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.26.1	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.26.2	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .

Continua nella pagina successiva



Tabella 15: Tracciamento dei requisiti nella componente DeleteDocuments (cont)

Requisito	Soddisfacimento del requisito
RF.O.26.3	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .
RF.O.27	DeleteDocumentsService: vedi (§6.6.2.8) .
RF.O.28	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.28.1	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.28.2	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .
RF.O.28.3	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .

4.2.7 EmbedDocuments

4.2.7.1 Descrizione

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

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

4.2.7.2 Esiti possibili

Tabella 16: Esiti possibili EmbedDocuments

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



4.2.7.3 Lista sottocomponenti

- **AWSDocument**: classe che rappresenta i documenti gestibili da AWSS3Manager;
- **AWSS3Manager**: vedi (§4.2.6.3) ;
- **Chunkerizer**: classe che crea *chunks_{igi}* a partire da testo;
- **Document**: classe di business che rappresenta i documenti completi;
- **DocumentContent**: classe di business che rappresenta il contenuto di un documento;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentMetadata**: vedi (§4.2.14.3) ;
- **DocumentOperationResponse**: vedi (§4.2.3.3) ;
- **DocumentStatus**: classe di business che rappresenta lo status di un documento;
- **DOCXTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti docx;
- **DocumentType**: enumeration che rappresenta i valori che può assumere il tipo di un documento;
- **EmbeddingsCreator**: classe che dialoga con il modello di embeddings per creare gli embeddings a partire dai chunk forniti in input;
- **EmbeddingsUploader**: classe che si occupa di effettuare la chiamata dell'upload degli embeddings nella porta esterna diretta verso il vector store;
- **EmbeddingsUploaderFacadeLangchain**: classe adapter che implementa la porta EmbeddingsUploaderPort, adattando la chiamata di EmbeddingsUploader alla sequenza di operazioni necessarie per il calcolo degli embeddings e il loro upload nel vector store;
- **EmbeddingsUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload degli embeddings verso i vector store;
- **EmbeddingsUploaderVectorStore**: classe che offre un metodo per eseguire l'upload degli embeddings nel vector store;
- **EmbedDocumentsController**: classe controller che si occupa del passaggio di una lista di DocumentId allo use case EmbedDocumentsUseCase a partire da una lista di stringhe che rappresentano l'id dei documenti di cui generare gli embeddings;
- **EmbedDocumentsService**: classe service che implementa lo use case EmbedDocumentsUseCase;
- **EmbedDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per generare gli embeddings dei documenti;
- **GetDocumentsContent**: classe che si occupa di inoltrare la richiesta di recuperare i documenti nella porta esterna diretta verso il sistema di archiviazione;
- **GetDocumentsContentAWSS3**: classe adapter che implementa la porta GetDocumentsContentPort, adattando la chiamata di GetDocumentsContent a AWSS3Manager;



- **GetDocumentsContentPort**: interfaccia che rappresenta la porta in uscita per recuperare il contenuto dei documenti dal sistema di archiviazione;
- **GetDocumentsStatus**: classe che si occupa di inoltrare la richiesta di recuperare gli status di una lista di documenti nella porta esterna GetDocumentsStatusPort diretta verso il vector store;
- **GetDocumentsStatusPort**: interfaccia che rappresenta la porta in uscita per recuperare gli status di una lista di documenti dal vector store;
- **GetDocumentsStatusVectorStore**: classe adapter che implementa la porta GetDocumentsStatusPort, adattando la chiamata di GetDocumentsStatus a VectorStoreManager;
- **HuggingFaceEmbeddingModel**: classe che implementa LangchainEmbeddingModel offrendo la possibilità di creare embeddings attraverso un embedding model di HuggingFace;
- **LangchainDocument**: classe che rappresenta un documento e i suoi embeddings;
- **LangchainEmbeddingModel**: classe astratta che permette di interagire con un oggetto langchain.Embeddings;
- **OpenAIEmbeddingModel**: classe che implementa la classe astratta LangchainEmbeddingModel per interagire con un modello di generazione di embeddings di OpenAI;
- **PDFTextExtractor**: classe che implementa l'interfaccia TextExtractor, offrendo un metodo per l'estrazione di testo da documenti PDF_{igi} ;
- **PlainDocument**: classe di business che rappresenta i documenti, compresi i metadati e il contenuto;
- **Status**: enumeration che rappresenta i valori che può assumere lo status di un documento;
- **TextExtractor**: interfaccia che espone il metodo astratto di estrazione di testo da un documento;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentStatusResponse**: classe che rappresenta la risposta generata da un vector store, contenente lo status del documento richiesto;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.7.4 Tracciamento dei requisiti



Tabella 17: Tracciamento dei requisiti nella componente EmbedDocuments

Requisito	Soddisfacimento del requisito
RF.O.58	EmbedDocumentsService: vedi (§6.7.2.14) .

4.2.8 EnableDocuments

4.2.8.1 Descrizione

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

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

4.2.8.2 Esiti possibili

Tabella 18: Esiti possibili EnableDocuments

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

4.2.8.3 Lista sottocomponenti

- **DocumentId:** vedi (§4.2.3.3) ;
- **DocumentOperationResponse:** vedi (§4.2.3.3) ;
- **EnableDocumentsController:** classe controller che si occupa del passaggio di una lista di `DocumentId` allo use case `EnableDocumentsUseCase` a partire da una lista di stringhe che rappresentano l'id dei documenti da riabilitare;



- **EnableDocumentsPort**: interfaccia che rappresenta la porta in uscita per effettuare la riabilitazione dei documenti nel vector store;
- **EnableDocumentsService**: classe service che implementa lo use case EnableDocumentsUseCase;
- **EnableDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la riabilitazione dei documenti;
- **EnableDocumentsVectorStore**: classe adapter che implementa la porta EnableDocumentsPort, adattando la chiamata di EnableDocumentsService a VectorStoreManager;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentOperationResponse**: vedi (§4.2.3.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .

4.2.8.4 Tracciamento dei requisiti

Tabella 19: Tracciamento dei requisiti nella componente EnableDocuments

Requisito	Soddisfacimento del requisito
RF.O.31	EnableDocumentsService: vedi (§6.8.2.5) .
RF.O.32	EnableDocumentsService: vedi (§6.8.2.5) .

4.2.9 GetChatMessages

4.2.9.1 Descrizione

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

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

4.2.9.2 Esiti possibili



Tabella 20: Esiti possibili GetChatMessages

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

4.2.9.3 Lista sottocomponenti

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

4.2.9.4 Tracciamento dei requisiti

Tabella 21: Tracciamento dei requisiti nella componente GetChatMessages

Requisito	Soddisfacimento del requisito
RF.O.42	GetChatMessagesService: vedi (§6.9.2.6) .



4.2.10 GetChats

4.2.10.1 Descrizione

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

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

4.2.10.2 Esiti possibili

Tabella 22: Esiti possibili GetChats

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

4.2.10.3 Lista sottocomponenti

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



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

4.2.10.4 Tracciamento dei requisiti

-

4.2.11 GetConfiguration

4.2.11.1 Descrizione

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

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

4.2.11.2 Esiti possibili

Tabella 23: Esiti possibili GetConfiguration

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

4.2.11.3 Lista sottocomponenti

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



- **EmbeddingModelConfiguration**: classe che rappresenta la configurazione di un modello di embedding;
- **EmbeddingModelType**: enumeration che rappresenta i valori che può assumere il campo `name` in `EmbeddingModelConfiguration`;
- **GetConfigurationController**: classe che si occupa di ritornare la configurazione di vector store, document store, embedding model e LLM;
- **GetConfigurationPort**: interfaccia che rappresenta la porta in uscita per ottenere la configurazione del sistema;
- **GetConfigurationPostgres**: classe che implementa la porta `GetConfigurationPort`, adattando la chiamata di `GetConfigurationService` a `PostgresConfigurationORM`;
- **GetConfigurationService**: classe service che implementa lo use case `GetConfigurationUseCase`;
- **GetConfigurationUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare il recupero della configurazione;
- **LLMModelConfiguration**: classe che rappresenta la configurazione di un modello LLM;
- **LLMModelType**: enumeration che rappresenta il tipo di valori del campo `name` di `LLMModelConfiguration`;
- **PostgresConfiguration**: vedi (§4.2.4.4) ;
- **PostgresConfigurationORM**: vedi (§4.2.2.3) ;
- **PostgresDocumentStoreConfiguration**: vedi (§4.2.4.4) ;
- **PostgresDocumentStoreType**: vedi (§4.2.4.4) ;
- **PostgresEmbeddingModelConfiguration**: vedi (§4.2.4.4) ;
- **PostgresEmbeddingModelType**: vedi (§4.2.4.4) ;
- **PostgresLLMModelConfiguration**: vedi (§4.2.4.4) ;
- **PostgresLLMModelType**: vedi (§4.2.4.4) ;
- **PostgresVectorStoreConfiguration**: vedi (§4.2.4.4) ;
- **PostgresVectorStoreType**: vedi (§4.2.4.4) ;
- **VectorStoreConfiguration**: classe che rappresenta la configurazione di un vector store;
- **VectorStoreType**: enumeration che rappresenta i valori che può assumere il campo `name` in `VectorStoreConfiguration`.

4.2.11.4 Tracciamento dei requisiti



Tabella 24: Tracciamento dei requisiti nella componente GetConfiguration

Requisito	Soddisfacimento del requisito
RF.O.14	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.1	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.2	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.3	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.4	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.54.	LLMModelType: vedi (§6.11.2.12) .

4.2.12 GetConfigurationOptions

4.2.12.1 Descrizione

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

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

4.2.12.2 Esiti possibili

Tabella 25: Esiti possibili GetConfigurationOptions

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

4.2.12.3 Lista sottocomponenti

- **ConfigurationOptions:** classe che contiene liste delle possibili configurazioni per vector store, document store, embedding model e LLM;
- **DocumentStoreConfiguration:** vedi (§4.2.11.3) ;
- **DocumentStoreType:** vedi (§4.2.11.3) ;
- **EmbeddingModelConfiguration:** vedi (§4.2.11.3) ;



- **EmbeddingModelType**: vedi (§4.2.11.3) ;
- **GetConfigurationOptions**: classe che si occupa di ottenere le possibili configurazioni di vector store, document store, embedding model e LLM;
- **GetConfigurationOptionsPort**: interfaccia che rappresenta la porta in uscita per ottenere le possibili configurazione di vector store, document store, embedding model e LLM;
- **GetConfigurationOptionsPostgres**: classe che implementa la porta GetConfigurationOptionsPort, adattando la chiamata di GetConfigurationOptionsService a PostgresConfigurationORM;
- **GetConfigurationOptionsService**: classe service che implementa lo use case GetConfigurationOptionsUseCase;
- **GetConfigurationOptionsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare il recupero dell'opzioni di configurazione per vector store, document store, embedding model e LLM;
- **LLMModelConfiguration**: vedi (§4.2.11.3) ;
- **LLMModelType**: vedi (§4.2.11.3) ;
- **PostgresConfigurationORM**: vedi (§4.2.2.3) ;
- **PostgresDocumentStoreConfiguration**: vedi (§4.2.4.4) ;
- **PostgresDocumentStoreType**: vedi (§4.2.4.4) ;
- **PostgresEmbeddingModelConfiguration**: vedi (§4.2.4.4) ;
- **PostgresEmbeddingModelType**: vedi (§4.2.4.4) ;
- **PostgresLLMModelConfiguration**: vedi (§4.2.4.4) ;
- **PostgresLLMModelType**: vedi (§4.2.4.4) ;
- **PostgresVectorStoreConfiguration**: vedi (§4.2.4.4) ;
- **PostgresVectorStoreType**: vedi (§4.2.4.4) ;
- **VectorStoreConfiguration**: vedi (§4.2.11.3) ;
- **VectorStoreType**: vedi (§4.2.11.3) .

4.2.12.4 Tracciamento dei requisiti

-

4.2.13 GetDocumentContent

4.2.13.1 Descrizione

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

- **Route API**: /getDocumentContent;
- **Metodo**: GET;
- **Lista parametri HTTP**:



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

4.2.13.2 Esiti possibili

Tabella 26: Esiti possibili `GetDocumentContent`

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

4.2.13.3 Lista sottocomponenti

- **AWSDocument:** vedi (§4.2.7.3) ;
- **AWSS3Manager:** vedi (§4.2.6.3) ;
- **Document:** vedi (§4.2.7.3) ;
- **DocumentContent:** vedi (§4.2.7.3) ;
- **DocumentId:** vedi (§4.2.3.3) ;
- **DocumentMetadata:** vedi (§4.2.14.3) ;
- **DocumentStatus:** vedi (§4.2.7.3) ;
- **DocumentType:** vedi (§4.2.7.3) ;
- **GetDocumentContentController:** classe controller che si occupa del passaggio di un `DocumentId` allo use case `GetDocumentsContentUseCase` a partire da una stringa che rappresenta l'id del documento da recuperare;
- **GetDocumentsContentFacadeService:** classe service che implementa lo use case `GetDocumentsContentUseCase`;
- **GetDocumentsContent:** vedi (§4.2.7.3) ;
- **GetDocumentsContentAWSS3:** vedi (§4.2.7.3) ;
- **GetDocumentsContentPort:** vedi (§4.2.7.3) ;
- **GetDocumentsStatus:** vedi (§4.2.7.3) ;
- **GetDocumentsStatusPort:** (§4.2.7.3) ;
- **GetDocumentsStatusVectorStore:** vedi (§4.2.7.3) ;



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

4.2.13.4 Tracciamento dei requisiti

Tabella 27: Tracciamento dei requisiti nella componente GetDocumentContent

Requisito	Soddisfacimento del requisito
RF.O.24	GetDocumentsContentFacadeService: vedi (§6.13.2.10) .

4.2.14 GetDocuments

4.2.14.1 Descrizione

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

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

4.2.14.2 Esiti possibili

Tabella 28: Esiti possibili GetDocuments

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



4.2.14.3 Lista sottocomponenti

- **AWSDocumentMetadata**: classe che rappresenta i metadati di un documento ricavato da AWS;
- **AWSS3Manager**: vedi (§4.2.6.3) ;
- **DocumentFilter**: classe di business che rappresenta il filtro applicabile alla ricerca di documenti;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentMetadata**: classe di business che rappresenta i metadati di un documento;
- **DocumentStatus**: vedi (§4.2.7.3) ;
- **DocumentType**: vedi (§4.2.7.3) ;
- **ElaborationException**: classe che rappresenta il messaggio di eccezione;
- **GetDocumentsController**: classe controller che si occupa del passaggio di un oggetto DocumentFilter allo use case GetDocumentsUseCase a partire da una stringa che rappresenta il filtro della ricerca;
- **GetDocumentsFacadeService**: classe service che implementa lo use case GetDocumentsUseCase;
- **GetDocumentsListAWSS3**: classe adapter che implementa la porta GetDocumentsMetadataPort, adattando la chiamata di GetDocumentsMetadata a AWSS3Manager;
- **GetDocumentsMetadata**: classe che si occupa di ricavare i metadati di una lista di documenti nella porta esterna GetDocumentsMetadataPort diretta verso il sistema di archiviazione documenti a partire da un filtro di ricerca;
- **GetDocumentsMetadataPort**: interfaccia che rappresenta la porta in uscita per ricavare i metadati di una lista di documenti dal sistema di archiviazione a partire da un filtro di ricerca;
- **GetDocumentsStatus**: vedi (§4.2.7.3) ;
- **GetDocumentsStatusPort**: vedi (§4.2.7.3) ;
- **GetDocumentsStatusVectorStore**: (§4.2.7.3) ;
- **GetDocumentsUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per ricercare dei documenti;
- **LightDocument**: classe di business che fornisce una rappresentazione leggera dei documenti, escludendo il contenuto;
- **Status**: vedi (§4.2.7.3) ;
- **VectorStoreChromaDBManager**: vedi (§4.2.3.3) ;
- **VectorStoreDocumentStatusResponse**: (§4.2.7.3) ;
- **VectorStoreManager**: vedi (§4.2.3.3) ;
- **VectorStorePineconeManager**: vedi (§4.2.3.3) .



4.2.14.4 Tracciamento dei requisiti

Tabella 29: Tracciamento dei requisiti nella componente GetDocuments

Requisito	Soddisfacimento del requisito
RF.O.33	GetDocumentsFacadeService: vedi (§6.14.2.10) .

4.2.15 RenameChat

4.2.15.1 Descrizione

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

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

4.2.15.2 Esiti possibili

Tabella 30: Esiti possibili RenameChat

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

4.2.15.3 Lista sottocomponenti

- **ChatId:** vedi (§4.2.1.3) ;



- **ChatOperationResponse**: vedi (§4.2.1.3) ;
- **RenameChatController**: classe controller che si occupa di effettuare la rinomina di una chat;
- **RenameChatPort**: interfaccia che rappresenta la porta in uscita per effettuare la rinomina di una chat verso il database per la storicizzazione delle chat;
- **RenameChatPostgres**: classe che implementa la porta RenameChatPort, adattando la chiamata di RenameChatService a PostgresChatORM;
- **RenameChatService**: classe service che implementa lo use case RenameChatUseCase;
- **RenameChatUseCase**: interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la rinomina di una chat;
- **PostgresChatOperationResponse**: vedi (§4.2.1.3) ;
- **PostgresChatORM**: vedi (§4.2.1.3) .

4.2.15.4 Tracciamento dei requisiti

Tabella 31: Tracciamento dei requisiti nella componente RenameChat

Requisito	Soddisfacimento del requisito
RF.O.39	RenameChatService: vedi (§6.15.2.6) .
RF.O.40	RenameChatService: vedi (§6.15.2.6) .
RF.O.41	RenameChatService: vedi (§6.15.2.6) .

4.2.16 SetConfiguration

4.2.16.1 Descrizione

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

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



2. Il parametro `documentStore` non deve essere vuoto o nullo;
3. Il parametro `LLMModel` non deve essere vuoto o nullo;
4. Il parametro `vectorStore` non deve essere vuoto o nullo.

4.2.16.2 Esiti possibili

Tabella 32: Esiti possibili SetConfiguration

Codice	Descrizione	Risposta
200	Configurazione impostata con successo.	-
400	Uno o più dei parametri (<code>embeddingModel</code> , <code>documentStore</code> , <code>LLMModel</code> , <code>vectorStore</code>) è nullo.	Parametri insufficienti.
400	Il parametro <code>LLMModel</code> è vuoto.	Modello LLM ' <code>LLMModel</code> ' non valido.
400	Il parametro <code>embeddingModel</code> è vuoto.	Modello di embedding ' <code>embeddingModel</code> ' non valido.
400	Il parametro <code>documentStore</code> è vuoto.	Document Store ' <code>documentStore</code> ' non valido.
400	Il parametro <code>vectorStore</code> è vuoto.	Vector Store ' <code>vectorStore</code> ' non valido.
500	Configurazione fallita.	Errore nella inizializzazione della configurazione.

4.2.16.3 Lista sottocomponenti

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



- **SetConfigurationUseCase:** interfaccia use case che rappresenta la porta della business logic in entrata per effettuare la inizializzazione della configurazione di sistema.

4.2.16.4 Tracciamento dei requisiti

Tabella 33: Tracciamento dei requisiti nella componente SetConfiguration

Requisito	Soddisfacimento del requisito
RF.O.1	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.1	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.2	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.3	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.4	SetConfigurationService: vedi (§6.16.2.7) .

4.2.17 UploadDocuments

4.2.17.1 Descrizione

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

- **Route API:** /uploadDocuments;
- **Metodo:** POST;
- **Lista parametri HTTP:**
 - `documents` : una lista di documenti.
- **Implementazione generale:** La componente viene implementata dal blueprint `uploadDocuments` che esegue i seguenti controlli:
 1. Ogni documento deve avere un titolo;
 2. Ogni documento deve essere di tipo *PDF* o *DOC*;
 3. Devono esserci dei documenti nell'area di staging.

4.2.17.2 Esiti possibili

Tabella 34: Esiti possibili UploadDocuments

Codice	Descrizione	Risposta
200	Upload avvenuto con successo.	-

Continua nella pagina successiva



Tabella 34: Esiti possibili UploadDocuments (cont)

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

4.2.17.3 Lista sottocomponenti

- **AWSDocument**: vedi (§4.2.7.3) ;
- **AWSDocumentOperationResponse**: vedi (§4.2.6.3) ;
- **AWSS3Manager**: vedi (§4.2.6.3) ;
- **Chunkerizer**: vedi (§4.2.7.3) ;
- **Document**: vedi (§4.2.7.3) ;
- **DocumentContent**: vedi (§4.2.7.3) ;
- **DocumentId**: vedi (§4.2.3.3) ;
- **DocumentMetadata**: vedi (§4.2.14.3) ;
- **DocumentOperationResponse**: vedi (§4.2.3.3) ;
- **DocumentStatus**: vedi (§4.2.7.3) ;
- **DocumentType**: vedi (§4.2.7.3) ;
- **DocumentUploader**: classe che si occupa di effettuare la chiamata dell'upload dei documenti nella porta esterna diretta verso il sistema di archiviazione documenti;
- **DocumentUploaderAWSS3**: classe adapter che implementa la porta DocumentUploaderPort, adattando la chiamata di DocumentUploader a AWSS3Manager;
- **DocumentUploaderPort**: interfaccia che rappresenta la porta in uscita per effettuare l'upload dei documenti verso il sistema di archiviazione documenti;
- **DOCXTextExtractor**: vedi (§4.2.7.3) ;
- **EmbeddingsCreator**: vedi (§4.2.7.3) ;
- **EmbeddingsUploader**: vedi (§4.2.7.3) ;
- **EmbeddingsUploaderFacadeLangchain**: vedi (§4.2.7.3) ;
- **EmbeddingsUploaderPort**: vedi (§4.2.7.3) ;
- **EmbeddingsUploaderVectorStore**: vedi (§4.2.7.3) ;
- **HuggingFaceEmbeddingModel**: vedi (§4.2.7.3) ;



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

4.2.17.4 Tracciamento dei requisiti

Tabella 35: Tracciamento dei requisiti nella componente UploadDocuments

Requisito	Soddisfacimento del requisito
RF.O.21	UploadDocumentsService: vedi (§6.17.2.31) .
RF.O.22	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.22.1	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.22.2	EmbeddingsUploader: vedi (§6.7.2.9) .
RF.O.22.3	EmbeddingsUploader: vedi (§6.7.2.9) .
RF.O.23	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.23.1	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.23.2	EmbeddingsUploader: vedi (§6.7.2.9) .
RF.O.23.3	EmbeddingsUploader: vedi (§6.7.2.9) .



4.3 Database

Il Database è composto dalle seguenti tabelle riguardanti la configurazione del sistema e la archiviazione delle chat:

4.3.1 Configurazione del sistema

4.3.1.1 Diagramma ER

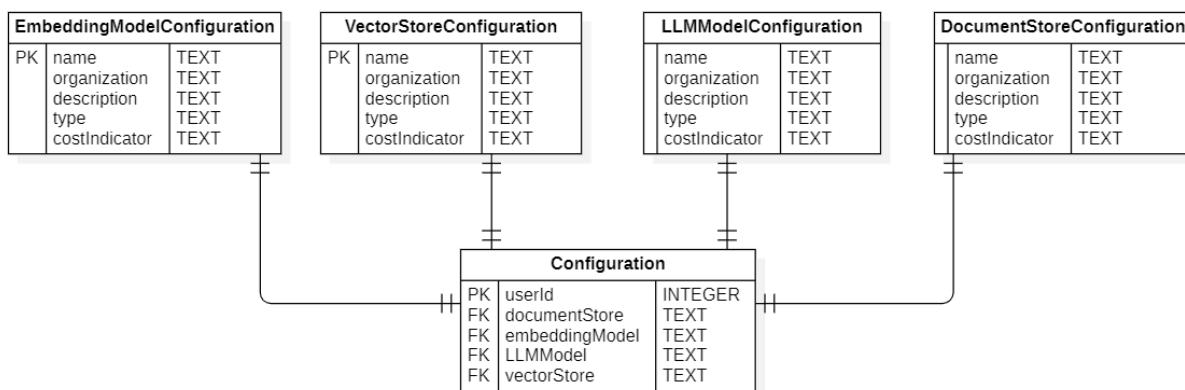


Figura 1: Diagramma ER: Configurations

4.3.1.2 Definizione

- **configuration** che rappresenta la scelta di configurazione di vector store, document store, modello LLM e di embedding:
Questa tabella contiene:
 - `documentStore`: fa riferimento alla colonna `name` nella tabella `documentStoreConfiguration` e indica la configurazione del document store scelto;
 - `embeddingModel`: fa riferimento alla colonna `name` nella tabella `embeddingModelConfiguration` e indica la configurazione del modello di embedding scelto;
 - `LLMModel`: fa riferimento alla colonna `name` nella tabella `LLMModelConfiguration` e indica la configurazione del modello LLM scelto;
 - `userId`: questa colonna funge da chiave primaria e identifica univocamente la configurazione;
 - `vectorStore`: fa riferimento alla colonna `name` nella tabella `vectorStoreConfiguration` e indica la configurazione del vector store scelto.
- **documentStoreConfiguration** che rappresenta la configurazione di un sistema di archiviazione dei documenti e contiene:
 - `costIndicator`: indica il costo associato all'uso del document store;



- `description` : descrizione della organizzazione proprietaria del document store e del document store;
 - `name` : il nome univoco del document store;
 - `organization` : rappresenta l'associazione proprietaria del document store;
 - `type` : indica il tipo di document store (locale o meno).
- **embeddingModelConfiguration** che rappresenta la configurazione di un modello di embeddings e contiene:
 - `costIndicator` : indicatore del costo associato all'uso del modello di embeddings;
 - `description` : descrizione della organizzazione proprietaria del modello di embeddings e del modello di embeddings stesso;
 - `name` : il nome univoco del modello di embeddings;
 - `organization` : rappresenta l'associazione proprietaria del modello di embeddings;
 - `type` : indica il tipo di modello di embeddings (locale o meno).
- **LLMModelConfiguration** che rappresenta la configurazione di un modello LLM e contiene:
 - `costIndicator` : indicatore del costo associato all'uso del modello LLM;
 - `description` : descrizione della organizzazione proprietaria del modello LLM e del modello di LLM stesso;
 - `name` : il nome univoco del modello di LLM;
 - `organization` : rappresenta l'associazione proprietaria del modello LLM;
 - `type` : indica il tipo di modello LLM (locale o meno).
- **vectorStoreConfiguration** che rappresenta la configurazione di un vector store e contiene:
 - `costIndicator` : indicatore del costo associato all'uso del vector store;
 - `description` : descrizione della organizzazione proprietaria del vector store e del vector store;
 - `name` : il nome univoco del vector store;
 - `organization` : rappresenta l'associazione proprietaria del vector store;
 - `type` : indica il tipo di vector store (locale o meno).



4.3.2 Archiviazione chat

4.3.2.1 Diagramma ER

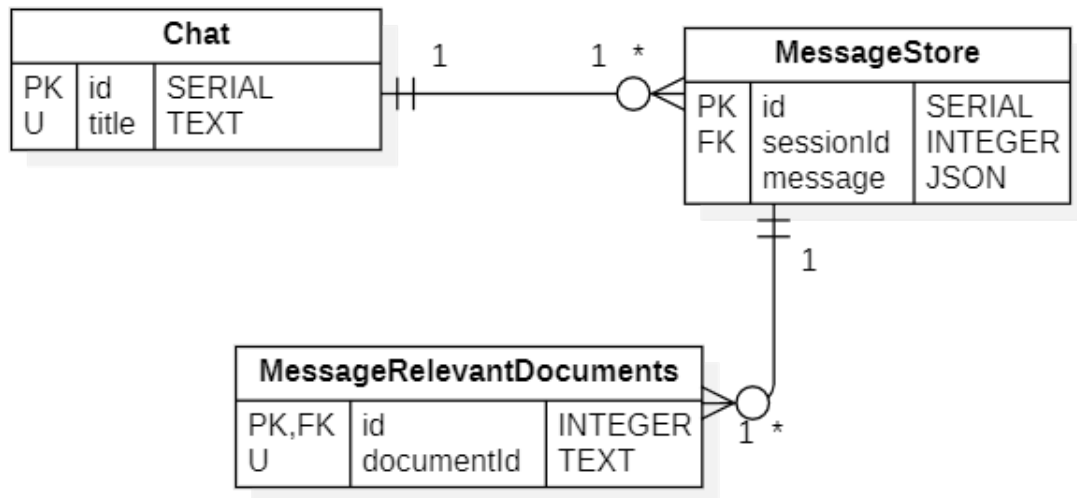


Figura 2: Diagramma ER: Chats

4.3.2.2 Definizione

- **chat** tabella che rappresenta una chat del Chatbot e contiene:
 - **id**: chiave primaria univoca che identifica la chat;
 - **title**: campo che sta per il titolo della chat, unico e non nullo.
- **messageRelevantDocuments** tabella che rappresenta i documenti rilevanti usati per un messaggio del Chatbot:
 - **documentId**: campo che identifica un documento rilevante;
 - **id**: chiave primaria univoca che fa riferimento alla colonna **id** nella tabella **messageStore**.
- **messageStore** tabella che rappresenta un messaggio di una chat del Chatbot e contiene:
 - **id**: chiave primaria univoca che identifica il messaggio;
 - **message**: contenuto del messaggio, memorizzato come JSON;
 - **sessionId**: fa riferimento alla colonna **id** nella tabella **chat** e indica la chat di appartenenza del messaggio. Permette inoltre la cancellazione a cascata.



4.4 Riepilogo tracciamento dei requisiti

Tabella 36: Tabella riepilogativa del tracciamento dei requisiti

Requisito	Soddisfacimento del requisito
RF.O.1	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.1	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.1.1	LLMModelInit: vedi (§5.7.1.7) .
RF.O.1.2	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.3	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.3.1	EmbeddingModelInit: vedi (§5.7.1.5) .
RF.O.1.4	SetConfigurationService: vedi (§6.16.2.7) .
RF.O.1.4.1	DocumentStoreInit: vedi (§5.7.1.3) .
RF.O.2	ChangeLLMConfiguration: vedi (§5.6.1.1) .
RF.O.2.1	ChangeLLMConfiguration: vedi (§5.6.1.1) .
RF.O.3	VectorStoreInit: vedi (§5.7.1.9) .
RF.O.3.1	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.4	EmbeddingModelInit: vedi (§5.7.1.5) .
RF.O.4.1	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.5	DocumentStoreInit: vedi (§5.7.1.3) .
RF.O.5.1	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.6	ChangeConfigurationService: vedi (§6.2.2.4) .
RF.O.6.1	ChangeLLMConfiguration: vedi (§5.6.1.1) .
RF.OP.7	NS
RF.OP.7.1	NS
RF.OP.8	NS
RF.OP.8.1	NS
RF.OP.9	NS
RF.OP.9.1	NS
RF.O.10	LLMCard: vedi (§5.6.1.5) .
RF.O.10.1	LLMCard: vedi (§5.6.1.5) .

Continua nella pagina successiva



Tabella 36: Tabella riepilogativa del tracciamento dei requisiti (cont)

Requisito	Soddisfacimento del requisito
RF.O.10.2	LLMCard: vedi (§5.6.1.5) .
RF.O.10.3	LLMCard: vedi (§5.6.1.5) .
RF.O.10.4	LLMCard: vedi (§5.6.1.5) .
RF.O.10.5	LLMCard: vedi (§5.6.1.5) .
RF.O.11	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.1	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.2	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.3	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.4	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.11.5	VectorStoreCard: vedi (§5.6.1.6) .
RF.O.12	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.1	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.2	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.3	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.4	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.12.5	EmbeddingModelCard: vedi (§5.6.1.4) .
RF.O.13	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.1	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.2	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.3	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.4	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.13.5	DocumentStoreCard: vedi (§5.6.1.3) .
RF.O.14	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.1	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.2	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.3	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.14.4	GetConfigurationService: vedi (§6.11.2.9) .
RF.O.15	DocumentsList: vedi (§5.3.1.1) .

Continua nella pagina successiva



Tabella 36: Tabella riepilogativa del tracciamento dei requisiti (cont)

Requisito	Soddisfacimento del requisito
RF.O.15.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.2	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.3	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.15.1.4	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.16	StagingArea: vedi (§5.3.1.3) .
RF.O.16.1	StagingArea: vedi (§5.3.1.3) .
RF.O.16.2	StagingArea: vedi (§5.3.1.3) .
RF.O.17	StagingArea: vedi (§5.3.1.3) .
RF.O.17.1	StagingArea: vedi (§5.3.1.3) .
RF.O.17.2	StagingArea: vedi (§5.3.1.3) .
RF.O.17.3	StagingArea: vedi (§5.3.1.3) .
RF.O.18	StagingArea: vedi (§5.3.1.3) .
RF.O.18.1	StagingArea: vedi (§5.3.1.3) .
RF.O.19	StagingAreaRow: vedi (§5.3.1.4) .
RF.O.20	StagingAreaRow: vedi (§5.3.1.4) .
RF.O.21	UploadDocumentsService: vedi (§6.17.2.31) .
RF.O.21.1	StagingArea: vedi (§5.3.1.3) .
RF.O.21.2	StagingArea: vedi (§5.3.1.3) .
RF.O.21.3	StagingArea: vedi (§5.3.1.3) .
RF.O.21.4	StagingArea: vedi (§5.3.1.3) .
RF.O.21.5	StagingArea: vedi (§5.3.1.3) .
RF.O.21.6	StagingArea: vedi (§5.3.1.3) .
RF.O.22	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.22.1	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.22.1.1	StagingArea: vedi (§5.3.1.3) .
RF.O.22.2	EmbeddingsUploader: vedi (§6.7.2.9) .
RF.O.22.3	EmbeddingsUploader: vedi (§6.7.2.9) .

Continua nella pagina successiva



Tabella 36: Tabella riepilogativa del tracciamento dei requisiti (cont)

Requisito	Soddisfacimento del requisito
RF.O.23	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.23.1	DocumentsUploader: vedi (§6.17.2.12) .
RF.O.23.1.1	StagingArea: vedi (§5.3.1.3) .
RF.O.23.2	EmbeddingsUploader: vedi (§6.7.2.9) .
RF.O.23.3	EmbeddingsUploader: vedi (§6.7.2.9) .
RF.O.24	GetDocumentsContentFacadeService: vedi (§6.13.2.10) .
RF.O.25	DeleteDocumentsService: vedi (§6.6.2.8) .
RF.O.26	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.26.1	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.26.1.1	DocumentsList: vedi (§5.3.1.1) .
RF.O.26.2	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .
RF.O.26.3	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .
RF.O.27	DeleteDocumentsService: vedi (§6.6.2.8) .
RF.O.27.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.28	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.28.1	DeleteDocuments: vedi (§6.6.2.3) .
RF.O.28.1.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.28.2	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .
RF.O.28.3	DeleteDocumentsEmbeddings: vedi (§6.6.2.6) .
RF.O.29	ConcealDocumentsService: vedi (§6.3.2.3) .
RF.O.30	ConcealDocumentsService: vedi (§6.3.2.3) .
RF.O.30.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.31	EnableDocumentsService: vedi (§6.8.2.5) .
RF.O.32	EnableDocumentsService: vedi (§6.8.2.5) .
RF.O.32.1	DocumentsListRow: vedi (§5.3.1.2) .
RF.O.33	GetDocumentsFacadeService: vedi (§6.14.2.10) .
RF.O.34	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1	ChatListSideBar: vedi (§5.1.1.4) .

Continua nella pagina successiva



Tabella 36: Tabella riepilogativa del tracciamento dei requisiti (cont)

Requisito	Soddisfacimento del requisito
RF.O.34.1.1	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1.2	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1.3	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.34.1.4	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.35	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.36	PostgresChatORM: vedi (§6.1.2.18) .
RF.O.36.1	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.37	DeleteChatsService: vedi (§6.5.2.6) .
RF.O.37.1	ChatHeader: vedi (§5.1.1.3) .
RF.O.38	DeleteChatsService: vedi (§6.5.2.6) .
RF.O.38.1	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.38.2	ChatListSideBar: vedi (§5.1.1.4) .
RF.O.39	RenameChatService: vedi (§6.15.2.6) .
RF.O.40	RenameChatService: vedi (§6.15.2.6) .
RF.O.40.1	ChatHeader: vedi (§5.1.1.3) .
RF.O.41	RenameChatService: vedi (§6.15.2.6) .
RF.O.41.1	ChatHeader: vedi (§5.1.1.3) .
RF.O.42	GetChatMessagesService: vedi (§6.9.2.6) .
RF.O.43	ChatContent: vedi (§5.1.1.1) .
RF.O.43.1	ChatContent: vedi (§5.1.1.1) .
RF.O.43.1.1	MessageCard: vedi (§5.1.1.5) .
RF.O.43.1.2	MessageCard: vedi (§5.1.1.5) .
RF.D.44	MessageCard: vedi (§5.1.1.5) .
RF.D.45	NS
RF.D.46	NS
RF.D.46.1	NS
RF.D.46.2	NS
RF.D.48	MessageCard: vedi (§5.1.1.5) .

Continua nella pagina successiva



Tabella 36: Tabella riepilogativa del tracciamento dei requisiti (cont)

Requisito	Soddisfacimento del requisito
RF.OP.49	NS
RF.OP.50	NS
RF.O.51	ChatForm: vedi (§5.1.1.2) .
RF.O.51.1	ChatForm: vedi (§5.1.1.2) .
RF.O.51.2	ChatForm: vedi (§5.1.1.2) .
RF.O.51.2.1	ChatForm: vedi (§5.1.1.2) .
RF.O.51.2.2	ChatForm: vedi (§5.1.1.2) .
RF.O.52	AskChatbotService: vedi (§6.1.2.4) .
RF.O.53	MessageCard: vedi (§5.1.1.5) .
RF.O.54.	LLMModelType: vedi (§6.1.2.12) .
RF.OP.55	NS
RF.D.56	AskChatbotService: vedi (§6.1.2.4) .
RF.OP.57	NS
RF.O.58	EmbedDocumentsService: vedi (§6.7.2.14) .
RF.O.59	AskChatbotService: vedi (§6.1.2.4) .
RF.O.60	AskChatbotService: vedi (§6.1.2.4) .



5 Progettazione di dettaglio frontend

5.1 Chatbot

5.1.1 Lista delle sottocomponenti

5.1.1.1 ChatContent

- **Elementi determinanti:**
 - Messages: i messaggi rappresentati sotto forma di MessageCard ordinati cronologicamente.

5.1.1.2 ChatForm

- **Elementi determinanti:**
 - Send Button: per l'invio della richiesta contenuta nella texting Area;
 - Pause-Play Button: per mettere in pausa o in play la registrazione del messaggio vocale;
 - Record Button: per iniziare la registrazione del messaggio vocale;
 - Texting Area: per l'inserimento di messaggi testuali;
 - Vocal Message Button: per l'inserimento di messaggi vocali.

5.1.1.3 ChatHeader

- **Elementi determinanti:**
 - Delete Chat Button: per eliminare la chat corrente;
 - Rename Chat Button: per rinominare la chat corrente;
 - Title: campo per la visualizzazione e la modifica del titolo.

5.1.1.4 ChatListSideBar

- **Elementi determinanti:**
 - Abort Delete Button: per annullare l'eliminazione delle chat selezionate;
 - Chat Search Bar: campo di inserimento per permettere la ricerca delle chat;
 - Chat List: sezione che contiene i record delle chat salvate;
 - Confirm Delete Button: per confermare l'eliminazione delle chat selezionate;
 - New Chat Button: per creare una nuova chat;
 - Select Chat: componente per permettere di selezionare più chat da eliminare.

5.1.1.5 MessageCard

- **Elementi determinanti:**
 - Copy Button: elemento che permette di copiare il testo del messaggio;



- Message Content: elemento che contiene il testo del messaggio e i collegamenti ai documenti pertinenti nel caso dei messaggi del chatbot;
- Referenced Docs Shortcut: pulsante presente nei messaggi del chatbot che permette di visualizzare i documenti pertinenti alla risposta;
- Timestamp: campo contenente data e ora di invio del messaggio.

5.2 Dashboard

5.2.1 Lista delle sottocomponenti

5.2.1.1 DashboardHeader

- **Elementi determinanti:**
 - New Chat Shortcut: elemento che permette di passare velocemente alla finestra Chatbot con una nuova chat aperta;
 - Upload Document Shortcut: elemento che permette di passare velocemente alla finestra Documents per caricare un nuovo documento.

5.2.1.2 DocumentPreview

- **Elementi determinanti:**
 - Close Preview Button: elemento che permette di chiudere la preview pop-up del documento selezionato;
 - Document Info: scheda che permette di vedere lo status del documento, la dimensione, il formato, e la data di caricamento;
 - Document Title: campo che contiene il nome del documento;
 - View Document Button: elemento che permette di vedere per intero il documento selezionato.

5.2.1.3 LatestChatContent

- **Elementi determinanti:**
 - Expand Chat Button: elemento che permette di aprire la finestra Chatbot con la chat più recente aperta;
 - Recent Chat Messages: i messaggi della chat più recente rappresentati sotto forma di MessageCard ordinati cronologicamente.

5.2.1.4 MessageCard

Vedi (§5.1.1.5).

5.2.1.5 RecentChatsList

- **Elementi determinanti:**
 - Recent Chat Row: componente che rappresenta una chat con la quale l'utente ha recentemente interagito e ne mostra una preview.



5.2.1.6 RecentlyViewedTab

- **Elementi determinanti:**
 - Recently Uploaded Document Record: elemento che rappresenta un documento recentemente visualizzato e ne mostra il titolo, formato, dimensioni e status.

5.2.1.7 RecentlyUploadedTab

- **Elementi determinanti:**
 - Recently Uploaded Document Record: elemento che rappresenta un documento recentemente caricato nel sistema e ne mostra il titolo, formato, dimensioni e status.

5.3 Documents

5.3.1 Lista delle sottocomponenti

5.3.1.1 DocumentsList

- **Elementi determinanti:**
 - Documents: i documenti rappresentati sotto forma di DocumentsListRow;
 - Delete Selected Button: per eliminare i DocumentsListRow selezionati;
 - Nav Row Pages Buttons: per navigare tra le pagine di DocumentsListRow;
 - Order By Button: per ordinare secondo i diversi campi informazione dei DocumentsListRow;
 - Rows Per Page: per scegliere il numero di DocumentsListRow per pagina;
 - Status Filter: per visualizzare i documenti in base allo status selezionato;
 - Title Search Bar: per ricercare i documenti per titolo;
 - Type Filter: per visualizzare i documenti in base al formato selezionato;

5.3.1.2 DocumentsListRow

- **Elementi determinanti:**
 - Document Info: parte del record che contiene informazioni significative del documento quali la dimensione, lo status, il formato e la data di upload;
 - Document Title: parte del record che contiene il titolo del documento;
 - Conceal Document Button: per occultare gli embeddings di un documento abilitato;
 - Delete Document Button: per eliminare il documento selezionato;
 - Embed Document Button: per creare gli embeddings di un documento *Not Embedded*;
 - Enable Document Button: per abilitare un documento occultato;
 - Select Row Button: per selezionare il record del documento;



- View Document Button: per visualizzare il documento selezionato.

5.3.1.3 StagingArea

- **Elementi determinanti:**
 - Cancel Upload Button: per annullare l'upload dei documenti caricati in area di staging;
 - Drag&Drop Area: area dove è possibile trascinare e rilasciare i file nell'area di staging;
 - Staged Documents: i documenti caricati nell'area di staging, rappresentati sotto forma di StagingAreaRow;
 - Upload Button: per confermare l'upload dei documenti caricati in area di staging.

5.3.1.4 StagingAreaRow

- **Elementi determinanti:**
 - Cancel Staging Button: per eliminare il documento caricato in area di staging;
 - Staged Document Info: parte del record che contiene informazioni significative del documento caricato in area di staging, quali il titolo, la dimensione e il formato.

5.4 DocumentView

5.4.1 Lista delle sottocomponenti

5.4.1.1 DocumentViewer

- **Elementi determinanti:**
 - Document Viewer Box: componente che permette di visualizzare l'intero documento *PDF* o *DOC* selezionato.

5.5 SettingsAppearance

5.5.1 Lista delle sottocomponenti

5.5.1.1 ThemeSelection

- **Elementi determinanti:**
 - System Theme Choice: per selezionare il tema del sistema sul quale gira l'applicazione;
 - Theme Preview: elemento che mostra una anteprima del tema;
 - Update Preferences Button: elemento che permette di applicare il tema scelto.



5.6 SettingsConfiguration

5.6.1 Lista delle sottocomponenti

5.6.1.1 ChangeLLMConfiguration

- **Elementi determinanti:**
 - Change LLM Carousel: contiene una LLMCard selezionabile per ogni opzione disponibile di modello LLM;
 - Confirm Configuration Button: elemento per confermare la scelta di LLM selezionata nel Change LLM Carousel.

5.6.1.2 CurrentConfigurationCarousel

- **Elementi determinanti:**
 - Chosen Document Store: una DocumentStoreCard che indica il sistema di archiviazione della configurazione corrente;
 - Chosen Embedding Model: una EmbeddingModelCard che indica il modello di embeddings della configurazione corrente;
 - Chosen LLM Model: una LLMCard che indica il modello LLM della configurazione corrente;
 - Chosen Vector Store: una VectorStoreCard che indica il vector store della configurazione corrente.

5.6.1.3 DocumentStoreCard

- **Elementi determinanti:**
 - Document Store Cost Indicator: campo che fornisce informazioni sul costo del document store;
 - Document Store Description: campo che descrive il document store e la organizzazione proprietaria;
 - Document Store Name: campo che rappresenta il nome del document store;
 - Document Store Organization: campo che rappresenta il nome dell'organizzazione proprietaria di quel document store;
 - Document Store Type: campo che indica il tipo di document store.

5.6.1.4 EmbeddingModelCard

- **Elementi determinanti:**
 - Embedding Model Cost Indicator: campo che fornisce informazioni sul costo del modello di embedding;
 - Embedding Model Description: campo che descrive il modello di embedding e la organizzazione proprietaria;
 - Embedding ModelName: campo che rappresenta il nome del modello di embedding;



- Embedding Model Organization: campo che rappresenta il nome dell'organizzazione proprietaria di quel modello di embedding;
- Embedding Model Type: campo che indica il tipo di modello di embedding.

5.6.1.5 LLMCard

- **Elementi determinanti:**
 - LLM Cost Indicator: campo che fornisce informazioni sul costo del modello LLM;
 - LLM Description: campo che descrive il modello LLM e la organizzazione proprietaria;
 - LLM Name: campo che rappresenta il nome del modello LLM;
 - LLM Organization: campo che rappresenta il nome dell'organizzazione proprietaria di quel modello LLM;
 - LLM Type: campo che indica il tipo di modello LLM.

5.6.1.6 VectorStoreCard

- **Elementi determinanti:**
 - Vector Store Cost Indicator: campo che fornisce informazioni sul costo del vector store;
 - Vector Store Description: campo che descrive il vector store e la organizzazione proprietaria;
 - Vector Store Name: campo che rappresenta il nome del vector store;
 - Vector Store Organization: campo che rappresenta il nome dell'organizzazione proprietaria di quel vector store;
 - Vector Store Type: campo che indica il tipo di vector store.

5.7 SetUp(Primo avvio)

5.7.1 Lista delle sottocomponenti

5.7.1.1 ConfirmConfiguration

- **Elementi determinanti:**
 - Confirm Configuration Choices Button: elemento che permette di confermare le scelte di vector store, document store, modello di embeddings e LLM al primo avvio dell'applicazione.

5.7.1.2 DocumentStoreCard

Vedi (§5.6.1.3) .

5.7.1.3 DocumentStoreInit

- **Elementi determinanti:**



- Document Store Choices: componente che contiene una DocumentStoreCard selezionabile per ogni sistema di archiviazione dei documenti disponibile.

5.7.1.4 EmbeddingModelCard

Vedi (§5.6.1.4).

5.7.1.5 EmbeddingModelInit

- **Elementi determinanti:**
 - Embedding Model Choices: componente che contiene una EmbeddingModelCard selezionabile per ogni modello di embeddings disponibile.

5.7.1.6 LLMCard

Vedi (§5.6.1.5).

5.7.1.7 LLMModelInit

- **Elementi determinanti:**
 - LLM Model Choices: componente che contiene una LLMCard selezionabile per ogni modello LLM disponibile.

5.7.1.8 VectorStoreCard

Vedi (§5.6.1.6).

5.7.1.9 VectorStoreInit

- **Elementi determinanti:**
 - Vector Store Choices: componente che contiene una VectorStoreCard selezionabile per ogni vector store disponibile.

5.8 SideBar

5.8.1 Lista delle sottocomponenti

5.8.1.1 NavigationMenu

- **Elementi determinanti:**
 - Chatbot Nav Button: per raggiungere la pagina Chatbot;
 - Dashboard Nav Button: per raggiungere la Dashboard dell'applicazione;
 - Documents Nav Button: per raggiungere la pagine Documents.

5.8.1.2 SettingShortcut

- **Elementi determinanti:**
 - Settings Nav Button: per raggiungere le impostazioni dell'applicazione.



6.1.2 Lista delle sottocomponenti

6.1.2.1 AskChatbotController

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

6.1.2.2 AskChatbotLangchain

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

6.1.2.3 AskChatbotPort

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

6.1.2.4 AskChatbotService

- **Implementa:** `AskChatbotUseCase;`
- **Attributi:**
 - `askChatbotoutPort: AskChatbotPort;`
 - `persistChatOutPort: PersistChatPort.`
- **Metodi:**
 - `askChatbot(message:Message, chatId:ChatId): MessageResponse`
Implementazione del metodo astratto di `AskChatbotUseCase` per ottenere tramite `outPort` una risposta dal chatbot ad un messaggio appartenente ad



una chat identificata da un ChatId. Nel caso in cui il chatId sia vuoto e venga inoltre generata correttamente una risposta, viene creata una nuova chat.

6.1.2.5 AskChatbotUseCase

- **Metodi:**

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

6.1.2.6 ChatHistoryManager

- **Metodi:**

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

6.1.2.7 ChatId

- **Attributi:**

- `id: int.`

6.1.2.8 ChatOperationResponse

- **Attributi:**

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

6.1.2.9 Message

- **Attributi:**

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

6.1.2.10 MessageResponse

- **Attributi:**

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



6.1.2.11 MessageSender (Enumeration)

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

6.1.2.12 PersistChatPort

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

6.1.2.13 PostgresChat

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

6.1.2.14 PostgresChatOperationResponse

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

6.1.2.15 PostgresMessage

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



- `toMessage(): Message`
Metodo che permette di passare da `PostgresMessage` a un oggetto `Message`.

6.1.2.16 PostgresMessageSenderType(Enumeration)

- **Valori:**

- `AI` ;
- `human` .

6.1.2.17 PostgresPersistChat

- **Implementa:** `PersistChatPort`;

- **Attributi:**

- `postgresChatORM: PostgresChatORM` .

- **Metodi:**

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

6.1.2.18 PostgresChatORM

- **Metodi:**

- `createChat(): PostgresChatOperationResponse`
Metodo per creare una nuova `PostgresChat` sul database `Postgres` utilizzato per la storicizzazione della chat;
- `deleteChats(chatIds:List<int>): List<PostgresChatOperationResponse>`
Metodo per eliminare chat sul database `Postgres`, utilizzato per la storicizzazione delle chat. Ritorna l'esito dell'operazione con una lista di `PostgresChatOperationResponse`;
- `getChatMessages(chatId:int): PostgresChat`
Metodo per ottenere una `PostgresChat` dal database `Postgres` utilizzato per la storicizzazione delle chat a partire da un intero rappresentante l'id della chat;
- `getChats(chatFilter:string): List<PostgresChatPreview>`
Metodo per ottenere una lista di `PostgresChatPreview` dal database `Postgres` utilizzato per la storicizzazione delle chat, eventualmente filtrando la ricerca attraverso `chatFilter`;
- `persistChat(messages:List<PostgresMessage>, chatId:int): PostgresChatOperationResponse`
Metodo che prende una lista di `PostgresMessages` e la rende persistente uti-



lizzando il metodo `saveMessages`. Restituisce un `PostgresChatOperationResponse`, che rappresenta l'esito della operazione.

- `renameChat(chatId:int, title:string): PostgresChatOperationResponse`
Metodo per rinominare con `title` una chat identificata tramite `chatId` sul database `Postgres` utilizzato per la storicizzazione delle chat. Ritorna l'esito dell'operazione con un `PostgresChatOperationResponse`;
- `saveMessages(messages:List<PostgresMessage>, chatId:int): PostgresChatOperationResponse`
Metodo per salvare una lista di `PostgresMessages`. Restituisce l'esito dell'operazione come `PostgresChatOperationResponse`.

6.2 ChangeConfiguration

6.2.1 Diagramma delle classi

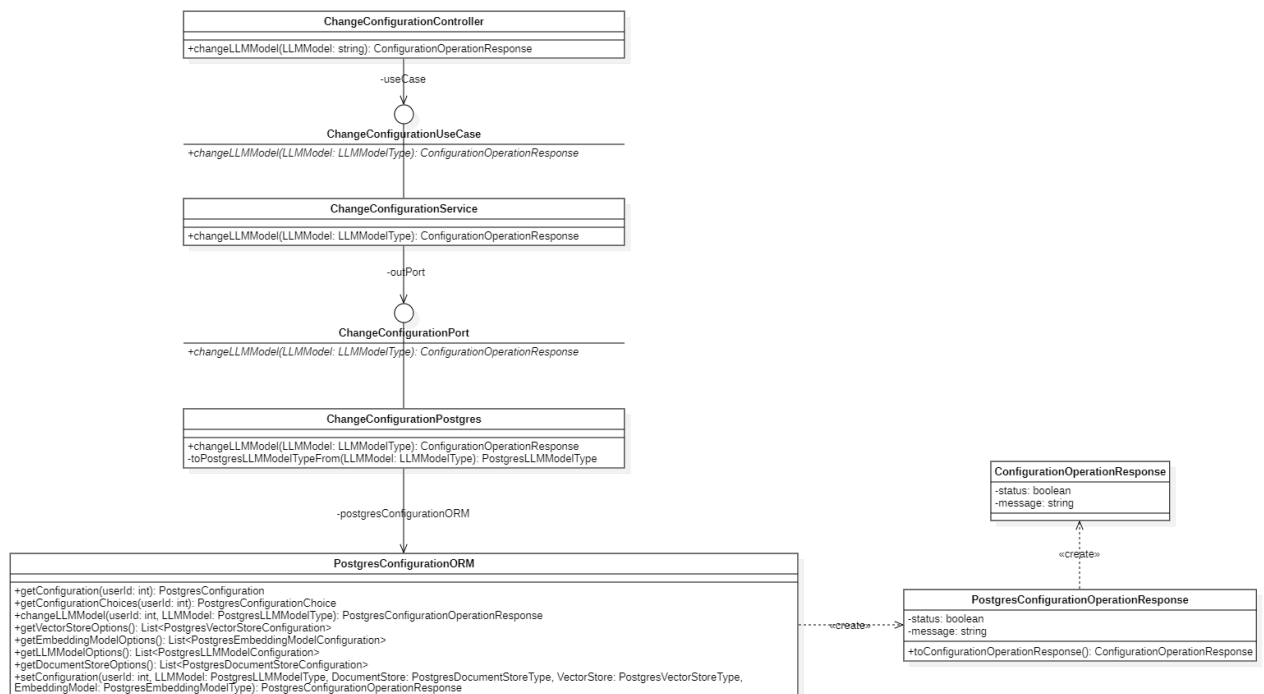


Figura 4: Diagramma delle classi della componente `ChangeConfiguration`

6.2.2 Lista delle sottocomponenti

6.2.2.1 ChangeConfigurationController

- **Attributi:**
 - `useCase: ChangeConfigurationUseCase`.
- **Metodi:**
 - `changeLLMModel(LLMModel:string): ConfigurationOperationResponse`
Metodo che si occupa di effettuare il cambio di configurazione del `LLMModel`



identificato da una stringa e ritorna l'esito della operazione come `ConfigurationOperationResponse`.

6.2.2.2 ChangeConfigurationPort

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

6.2.2.3 ChangeConfigurationPostgres

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

6.2.2.4 ChangeConfigurationService

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

6.2.2.5 ChangeConfigurationUseCase

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

6.2.2.6 ConfigurationOperationResponse

- **Attributi:**
 - `message: string`;



- status: boolean.

6.2.2.7 PostgresConfigurationOperationResponse

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

6.2.2.8 PostgresConfigurationORM

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



Metodo per impostare le configurazioni di vector store, modello di embedding, LLM e metodo di storicizzazione dei documenti.

6.3 ConcealDocuments

6.3.1 Diagramma delle classi

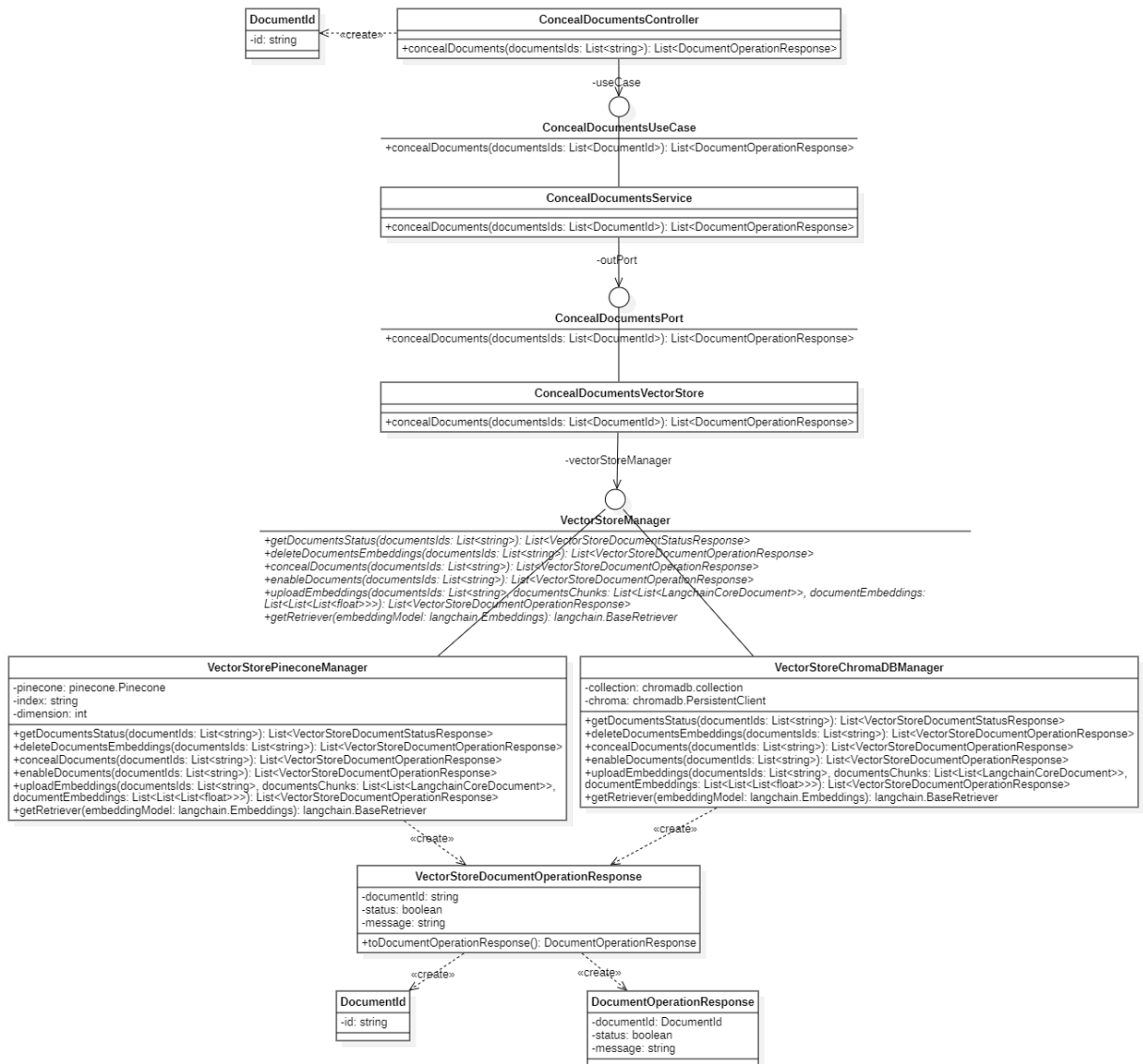


Figura 5: Diagramma delle classi della componente ConcealDocuments

6.3.2 Lista delle sottocomponenti

6.3.2.1 ConcealDocumentsController

- **Attributi:**
 - useCase: ConcealDocumentsUseCase.



- **Metodi:**

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

6.3.2.2 ConcealDocumentsPort

- **Metodi:**

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

6.3.2.3 ConcealDocumentsService

- **Implementa:** ConcealDocumentsUseCase;

- **Attributi:**

- `outPort: ConcealDocumentsPort.`

- **Metodi:**

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

6.3.2.4 ConcealDocumentsUseCase

- **Metodi:**

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

6.3.2.5 ConcealDocumentsVectorStore

- **Implementa:** ConcealDocumentsPort;

- **Attributi:**

- `vectorStoreManager: VectorStoreManager.`

- **Metodi:**

- `concealDocuments(documentIds:List<DocumentId>): List<DocumentOperationResponse>`



Implementazione del metodo astratto di `ConcealDocumentsPort` per occultare una lista di documenti a partire dal loro `DocumentId`, dialogando con il vector store. Ritorna una lista di `DocumentOperationResponse`, che indica l'esito dell'operazione per ogni documento.

6.3.2.6 DocumentId

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

6.3.2.7 DocumentOperationResponse

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

6.3.2.8 VectorStoreChromaDBManager

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



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

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

6.3.2.9 VectorStoreDocumentOperationResponse

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

6.3.2.10 VectorStoreManager

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



tadati a partire da una lista di stringhe che rappresentano i documentIds dei documenti, ritornandoli in una lista di VectorStoreDocumentStatusResponse;

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

6.3.2.11 VectorStorePineconeManager

- **Attributi:**

- `dimension: int;`
- `index: string;`
- `pinecone: pinecone.Pinecone.`

- **Implementa:** VectorStoreManager;

- **Metodi:**

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



List<List<List<float>>>>):List<VectorStoreDocumentOperationResponse>
Implementazione del metodo astratto di VectorStoreManager per effettuare in Pinecone l'upload degli embeddings di un documento. Ritorna un VectorStoreDocumentOperationResponse;

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

6.4 ConfigurationManager

6.4.1 Diagramma delle classi

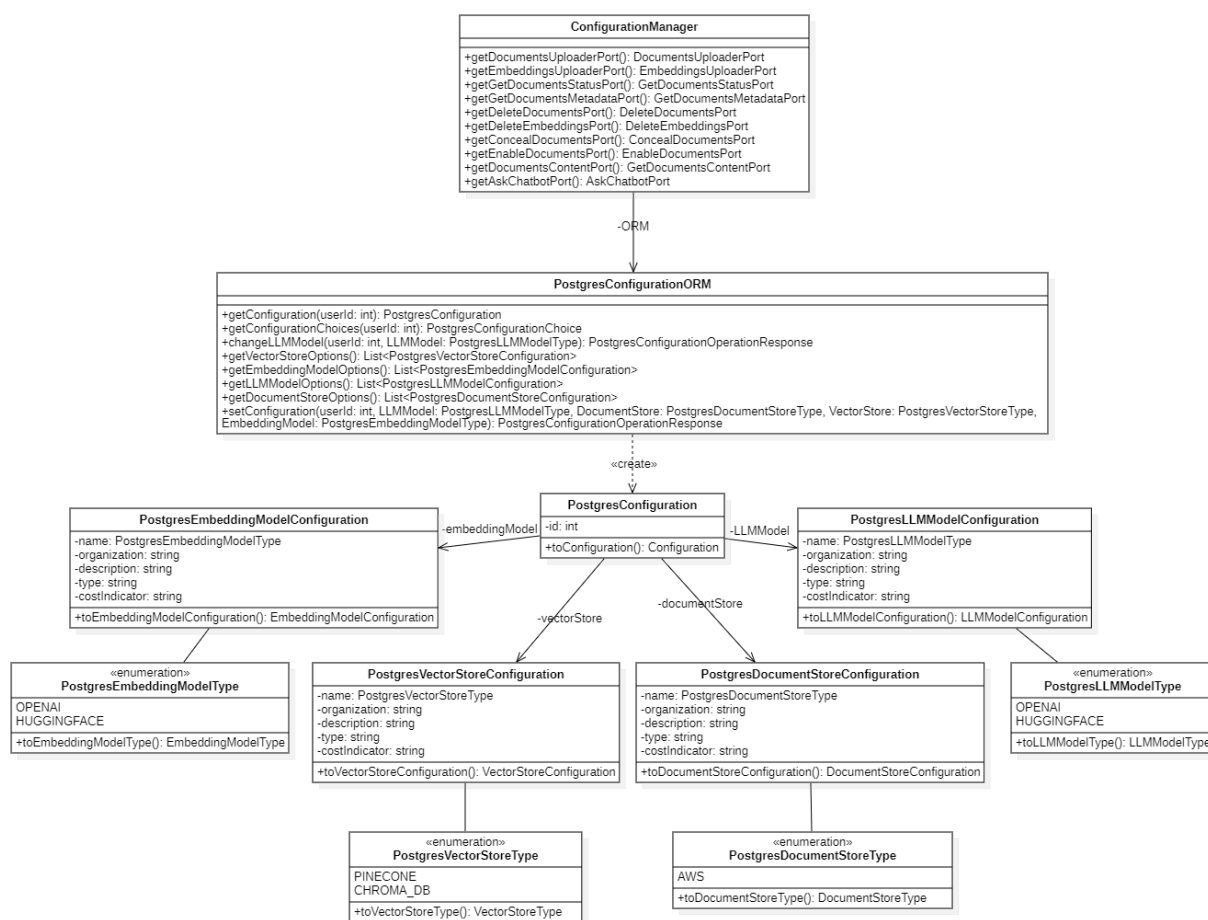


Figura 6: Diagramma delle classi della componente ConfigurationManager

6.4.2 Lista delle sottocomponenti

6.4.2.1 ConfigurationManager

- **Attributi:**
 - ORM: `PostgresConfigurationORM`.
- **Metodi:**



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

6.4.2.2 PostgresConfiguration

- **Attributi:**
 - `documentStore: PostgresDocumentStoreConfiguration;`
 - `embeddingModel: PostgresEmbeddingModelConfiguration;`
 - `id: int;`
 - `LLMModel: PostgresLLMModelConfiguration;`
 - `vectorStore: PostgresVectorStoreConfiguration.`
- **Metodi:**
 - `toConfiguration(): Configuration`
Metodo che ricava un oggetto Configuration dal PostgresConfiguration

6.4.2.3 PostgresConfigurationORM

Vedi (§6.2.2.8);



6.4.2.4 PostgresDocumentStoreConfiguration

- **Attributi:**

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

- **Metodi:**

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

6.4.2.5 PostgresDocumentStoreType(Enum)

- **Valori:**

- `AWS .`

- **Metodi:**

- `toDocumentStoreType(): DocumentStoreType`
Metodo che permette di ottenere un `DocumentStoreType` a partire dal `PostgresDocumentStoreType(Enum)`.

6.4.2.6 PostgresEmbeddingModelConfiguration

- **Attributi:**

- `costIndicator: string;`
- `description: string;`
- `name: PostgresEmbeddingModelType;`
- `organization: string;`
- `type: string.`

- **Metodi:**

- `toEmbeddingModelConfiguration(): EmbeddingModelConfiguration`
Metodo che permette di ottenere un `EmbeddingModelConfiguration` a partire dal `PostgresEmbeddingModelConfiguration`.

6.4.2.7 PostgresEmbeddingModelType(Enum)

- **Valori:**

- `HUGGINGFACE ;`
- `OPENAI .`

- **Metodi:**



- `toEmbeddingModelType(): EmbeddingModelType`
Metodo che permette di ottenere un `EmbeddingModelType` a partire dal `PostgresEmbeddingModelType(Enum)`.

6.4.2.8 PostgresLLMModelConfiguration

- **Attributi:**

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

- **Metodi:**

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

6.4.2.9 PostgresLLMModelType(Enum)

- **Valori:**

- `HUGGINGFACE;`
- `OPENAI.`

- **Metodi:**

- `toLLMModelType(): LLMModelType`
Metodo che permette di ottenere un `LLMModelType` a partire dal `PostgresLLMModelType(Enum)`.

6.4.2.10 PostgresVectorStoreConfiguration

- **Attributi:**

- `costIndicator: string;`
- `description: string;`
- `name: PostgresVectorStoreType;`
- `organization: string;`
- `type: string.`

- **Metodi:**

- `toVectorStoreConfiguration(): VectorStoreConfiguration`
Metodo che permette di ottenere un `VectorStoreConfiguration` a partire dal `PostgresVectorStoreConfiguration`.

6.4.2.11 PostgresVectorStoreType(Enum)

- **Valori:**



- CHROMA DB ;
- PINECONE .
- **Metodi:**
 - `toVectorStoreType(): VectorStoreType`
Metodo che permette di ottenere un `VectorStoreType` a partire dal `PostgresVectorStoreType(Enum)`.

6.5 DeleteChats

6.5.1 Diagramma delle classi

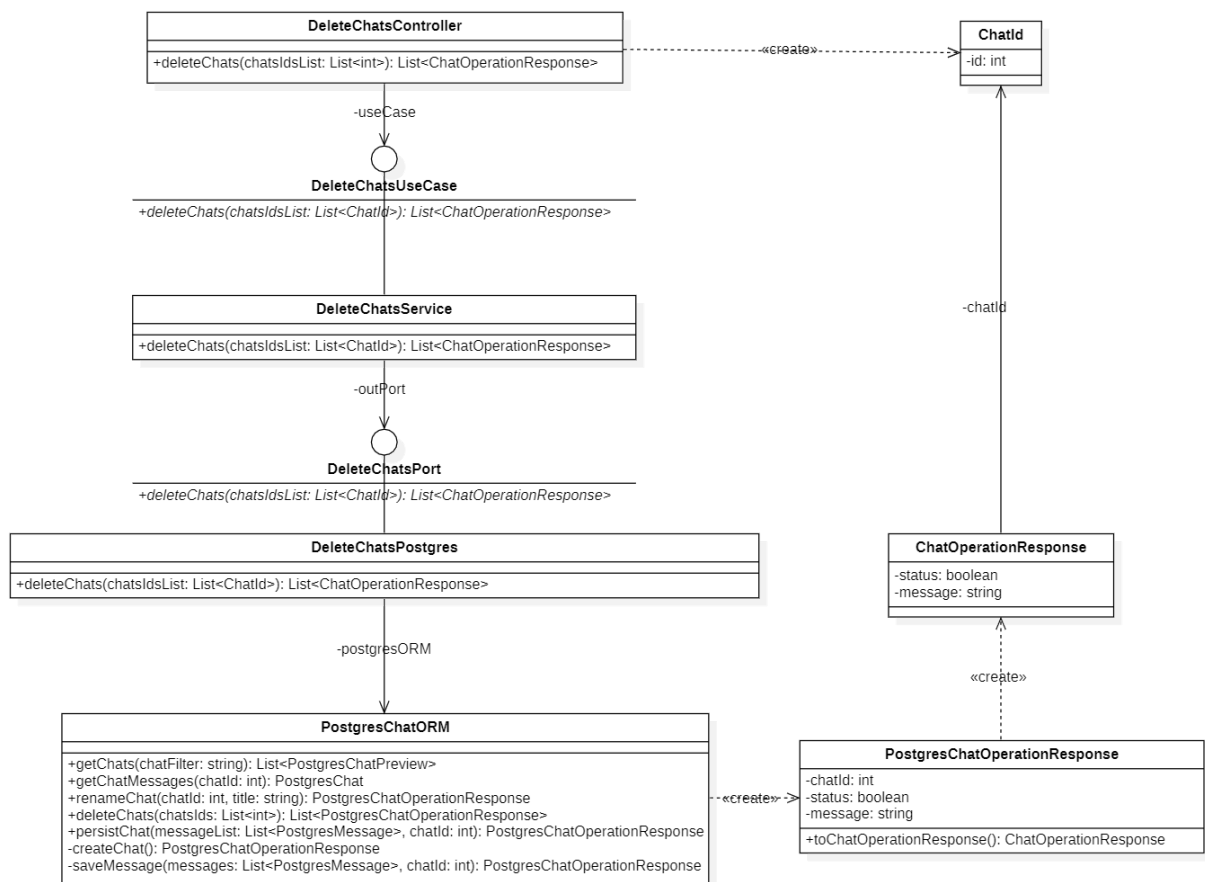


Figura 7: Diagramma delle classi della componente DeleteChats

6.5.2 Lista delle sottocomponenti

6.5.2.1 ChatId

Vedi (§6.1.2.7) .

6.5.2.2 ChatOperationResponse

Vedi (§6.1.2.8) ;



6.5.2.3 DeleteChatsController

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

6.5.2.4 DeleteChatsPort

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

6.5.2.5 DeleteChatsPostgres

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

6.5.2.6 DeleteChatsService

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

6.5.2.7 DeleteChatsUseCase

- **Metodi:**



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

6.5.2.8 PostgresChatOperationResponse

Vedi (§6.1.2.14) ;

6.5.2.9 PostgresChatORM

Vedi (§6.1.2.18) .





6.6.2 Lista delle sottocomponenti

6.6.2.1 AWSDocumentOperationResponse

- **Attributi:**

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

- **Metodi:**

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

6.6.2.2 AWSS3Manager

- **Attributi:**

- `awsBucketName: string;`
- `boto3/g/ boto3.session.Session.`

- **Metodi:**

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

6.6.2.3 DeleteDocuments

- **Attributi:**

- `outPort: DeleteDocumentsPort.`

- **Metodi:**



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

6.6.2.4 DeleteDocumentsAWS3

- **Implementa:** `DeleteDocumentsPort`;
- **Attributi:**
 - `awsS3Manager`: `AWSS3Manager`.
- **Metodi:**
 - `deleteDocuments(documentIds:List<DocumentId>):`
`List<DocumentOperationResponse>`
Implementazione del metodo astratto di `DeleteDocumentsPort` per eliminare una lista di documenti da AWS S3 a partire dai loro id.

6.6.2.5 DeleteDocumentsController

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

6.6.2.6 DeleteDocumentsEmbeddings

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

6.6.2.7 DeleteDocumentsPort

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



6.6.2.8 DeleteDocumentsService

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

6.6.2.9 DeleteDocumentsUseCase

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

6.6.2.10 DeleteEmbeddingsPort

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

6.6.2.11 DeleteEmbeddingsVectorStore

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

6.6.2.12 DocumentId

Vedi (§6.3.2.6).

6.6.2.13 DocumentOperationResponse

Vedi (§6.3.2.7).

**6.6.2.14 VectorStoreChromaDBManager**

Vedi (§6.3.2.8) .

6.6.2.15 VectorStoreDocumentOperationResponse

Vedi (§6.3.2.9) .

6.6.2.16 VectorStoreManager

Vedi (§6.3.2.10) .

6.6.2.17 VectorStorePineconeManager

Vedi (§6.3.2.11) .



6.7 EmbedDocuments

6.7.1 Diagramma delle classi

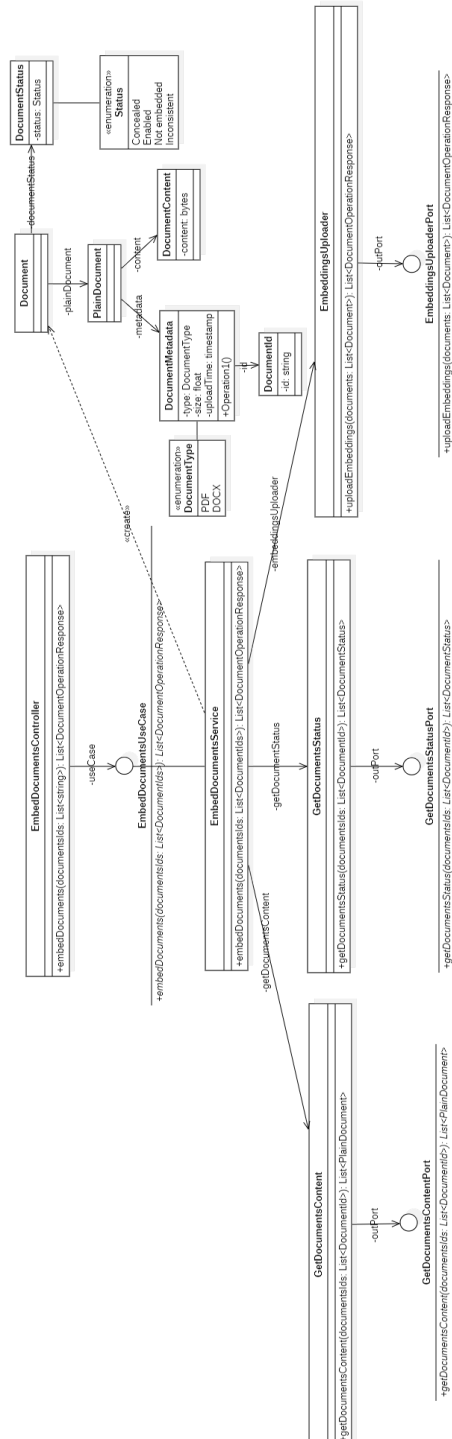


Figura 9: Diagramma delle classi componente EmbedDocuments (continua nei prossimi diagrammi)

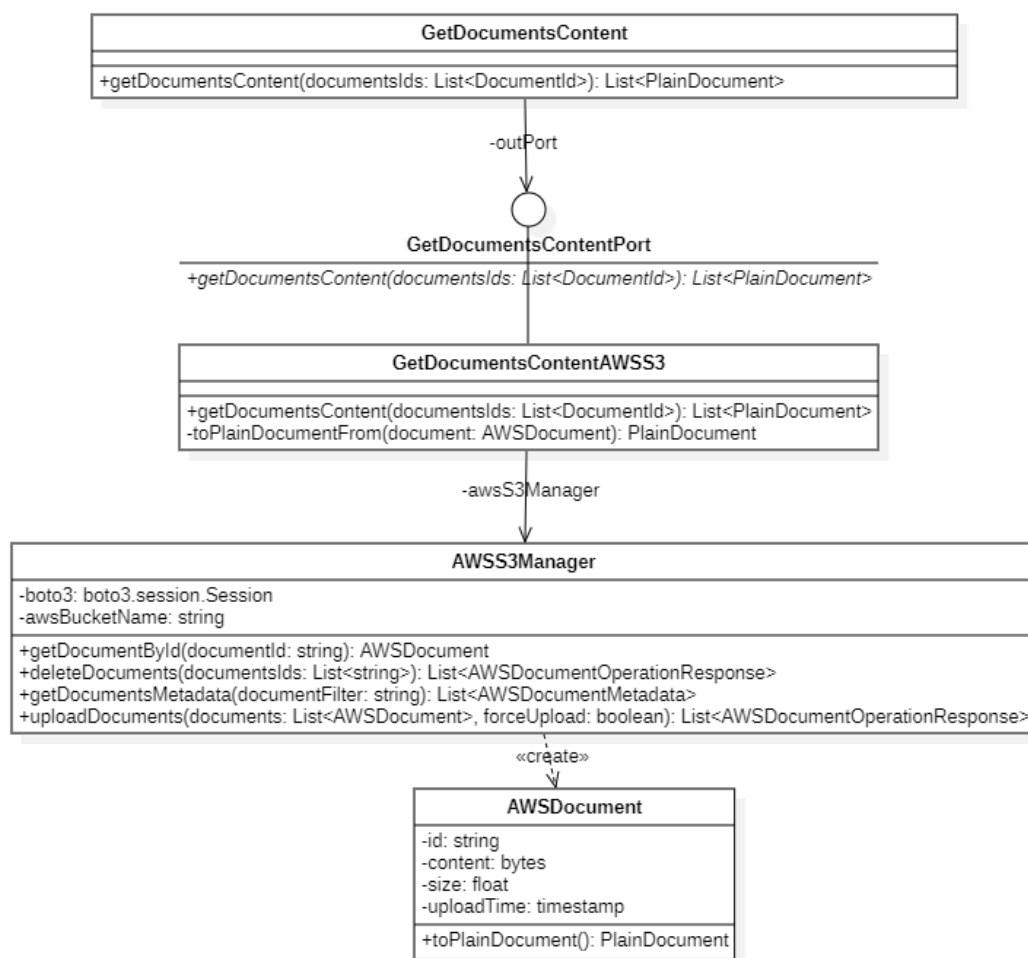


Figura 10: Diagramma delle classi componente EmbedDocuments (continua nel prossimo diagramma)

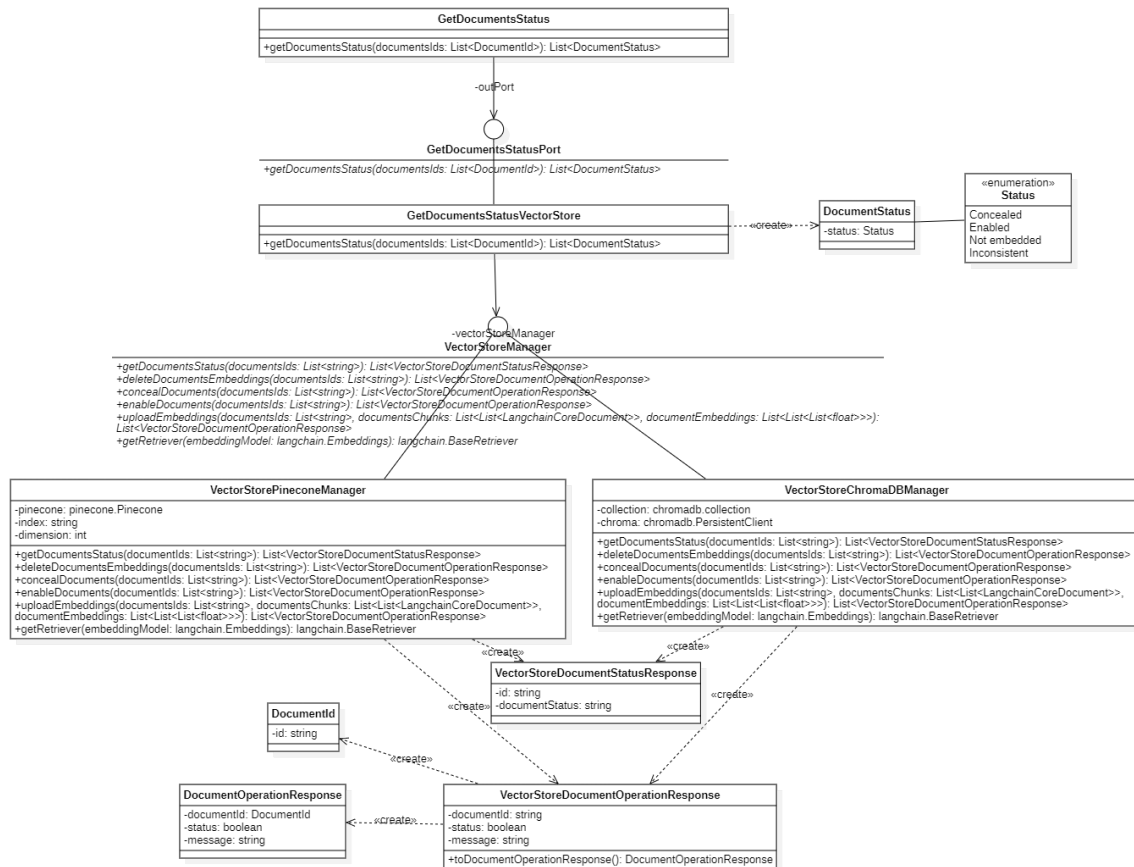
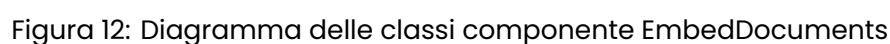


Figura 11: Diagramma delle classi componente EmbedDocuments (continua nel prossimo diagramma)





6.7.2 Lista delle sottocomponenti

6.7.2.1 AWSDocument

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

6.7.2.2 AWSS3Manager

Vedi (§6.6.2.2).

6.7.2.3 Chunkerizer

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

6.7.2.4 DocumentId

Vedi (§6.3.2.6).

6.7.2.5 DocumentOperationResponse

Vedi (§6.3.2.7).

6.7.2.6 DocumentStatus

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

6.7.2.7 DOCXTextExtractor

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



6.7.2.8 EmbeddingsCreator

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

6.7.2.9 EmbeddingsUploader

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

6.7.2.10 EmbeddingsUploaderFacadeLangchain

- **Implementa:** `EmbeddingsUploaderPort;`
- **Attributi:**
 - `chunkerizer: Chunkerizer;`
 - `embeddingsCreator: EmbeddingsCreator;`
 - `embeddingsUploaderVectorStore: EmbeddingsUploaderVectorStore.`
- **Metodi:**
 - `uploadEmbeddings(documents:List<Document>): List<DocumentOperationResponse>`
Implementazione del metodo astratto di `EmbeddingsUploaderPort` per la generazione e il caricamento di embeddings di una lista di `Document` nel vector store tramite l'utilizzo di un pattern facade, ritornando infine una lista di `DocumentOperationResponse`.
Le operazioni all'interno della facade vengono svolte nel seguente ordine:
 1. Estrazione del testo dai documenti tramite `documentToText`;
 2. Suddivisione in chunks del testo estratto dai documenti tramite `chunkerizer`;
 3. Generazione degli embeddings a partire dai chunks tramite `embeddingsCreator`;



4. Caricamento degli embeddings creati nel vector store tramite `embeddingsUploaderVectorStore`.

6.7.2.11 `EmbeddingsUploaderPort`

- **Metodi:**

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

6.7.2.12 `EmbeddingsUploaderVectorStore`

- **Attributi:**

- `vectorStoreManager: VectorStoreManager`.

- **Metodi:**

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

6.7.2.13 `EmbedDocumentsController`

- **Attributi:**

- `useCase: EmbedDocumentsUseCase`.

- **Metodi:**

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

6.7.2.14 `EmbedDocumentsService`

- **Implementa:** `EmbedDocumentsUseCase`;

- **Attributi:**

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

- **Metodi:**

- `embedDocuments(documentsIds:List<DocumentIds>): List<DocumentOperationResponse>`
Implementazione del metodo astratto di `EmbedDocumentsUseCase`, per il recupero dei documenti e il loro contenuto attraverso `getDocuments`, e la



generazione degli embeddings della lista dei documenti recuperati tramite `embeddingsUploader`. Ritorna una lista di `DocumentOperationResponse`, indicando per ogni documento l'esito della generazione degli embeddings.

6.7.2.15 `EmbedDocumentsUseCase`

- **Metodi:**

- `embedDocuments(documentsIds:List<DocumentIds>): List<DocumentOperationResponse>`

Metodo astratto per generare gli embeddings di una lista di documenti a partire dal loro `DocumentId`. Ritorna una lista di `DocumentOperationResponse`, che indica l'esito dell'operazione per ogni documento.

6.7.2.16 `GetDocumentsContent`

- **Attributi:**

- `outPort: GetDocumentsContentPort.`

- **Metodi:**

- `getDocumentsContent(documentsIds:List<DocumentId>): List<PlainDocument>`

Metodo per recuperare il contenuto di una lista di `Document` tramite `outPort` dal sistema di archiviazione documenti a partire da una lista di id.

6.7.2.17 `GetDocumentsContentAWSS3`

- **Implementa:** `DocumentsContentPort`;

- **Attributi:**

- `awsS3Manager: AWSS3Manager.`

- **Metodi:**

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

6.7.2.18 `GetDocumentsContentPort`

- **Metodi:**

- `getDocumentsContent(documentsIds:List<DocumentId>): List<PlainDocument>`

Metodo astratto per recuperare il contenuto di una lista di documenti dal sistema di archiviazione a partire dal loro id.



6.7.2.19 GetDocumentsStatus

- **Attributi:**

- `outPort: GetDocumentsStatusPort.`

- **Metodi:**

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

6.7.2.20 GetDocumentsStatusPort

- **Metodi:**

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

6.7.2.21 GetDocumentsStatusVectorStore

- **Implementa:** `GetDocumentsStatusPort;`

- **Attributi:**

- `vectorStoreManager: VectorStoreManager.`

- **Metodi:**

- `getDocumentsStatus(documentsIds:List<DocumentId>):`
`List<DocumentStatus>`
Implementazione del metodo astratto di `GetDocumentsStatusVectorStore` per recuperare gli status di una lista di documenti a partire dal loro id.

6.7.2.22 HuggingFaceEmbeddingModel

- **Implementa:** `LangchainEmbeddingModel;`

- **Attributi:**

- `embeddingsDimension: int;`
- `model: langchain.Embedding.`

- **Metodi:**

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



6.7.2.23 LangchainDocument

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

6.7.2.24 LangchainEmbeddingModel

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

6.7.2.25 OpenAIEmbeddingModel

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

6.7.2.26 PDFTextExtractor

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

6.7.2.27 Status (Enumeration)

- **Valori:**
 - `Concealed;`



- Enabled;
- Inconsistent;
- Not Embedded.

6.7.2.28 TextExtractor

- **Metodi:**

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

6.7.2.29 VectorStoreChromaDBManager

Vedi (§6.3.2.8).

6.7.2.30 VectorStoreDocumentOperationResponse

Vedi (§6.3.2.9).

6.7.2.31 VectorStoreDocumentStatusResponse

- **Attributi:**

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

6.7.2.32 VectorStoreManager

Vedi (§6.3.2.10).

6.7.2.33 VectorStorePineconeManager

Vedi (§6.3.2.11).



6.8 EnableDocuments

6.8.1 Diagramma delle classi

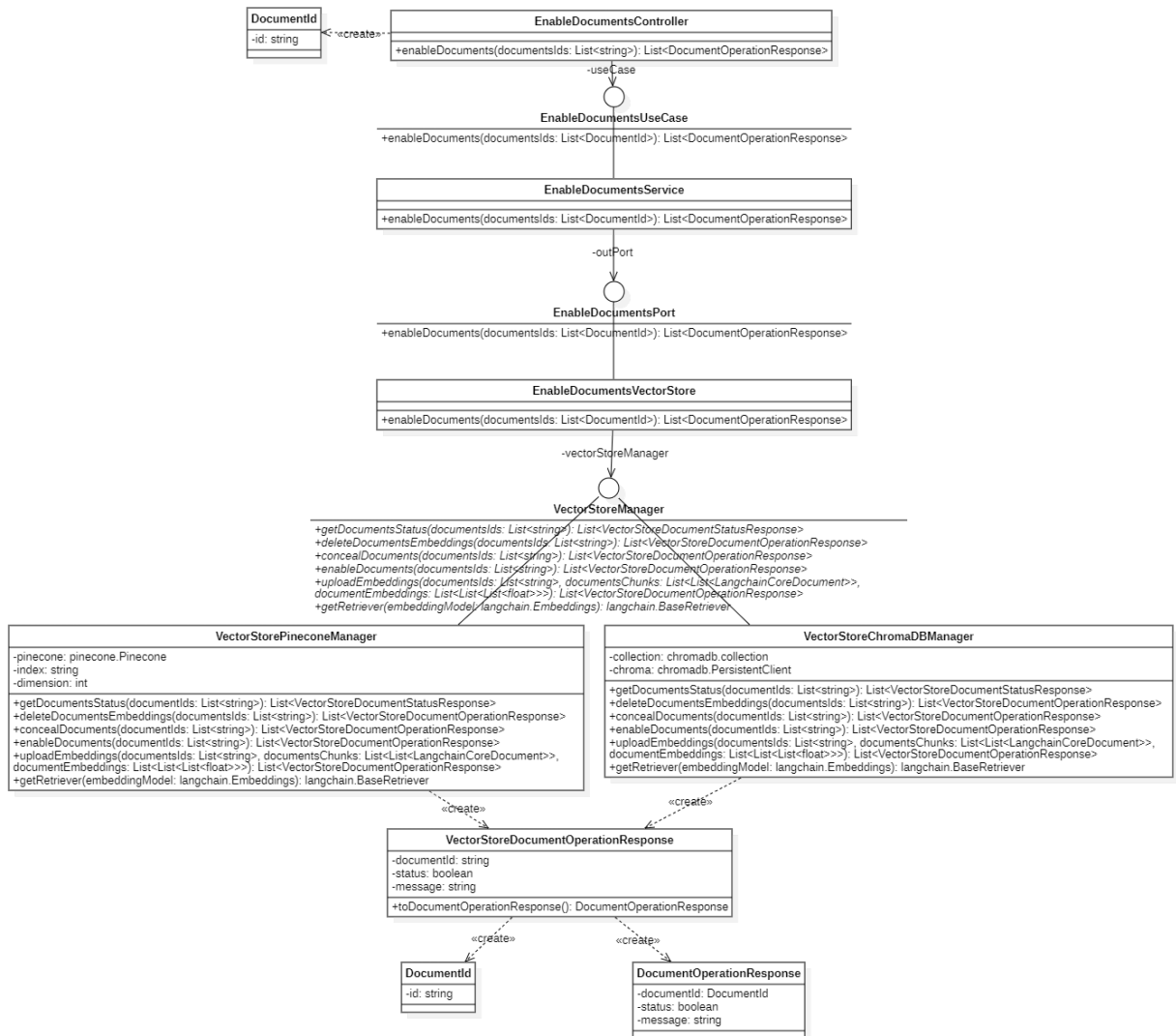


Figura 13: Diagramma delle classi della componente EnableDocuments

6.8.2 Lista delle sottocomponenti

6.8.2.1 DocumentId

Vedi (§6.3.2.6).

6.8.2.2 DocumentOperationResponse

Vedi (§6.3.2.7).

6.8.2.3 EnableDocumentsController

- **Attributi:**



- `useCase: EnableDocumentsUseCase.`

- **Metodi:**

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

6.8.2.4 EnableDocumentsPort

- **Metodi:**

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

6.8.2.5 EnableDocumentsService

- **Implementa:** `EnableDocumentsUseCase`;

- **Attributi:**

- `outPort: EnableDocumentsPort.`

- **Metodi:**

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

6.8.2.6 EnableDocumentsUseCase

- **Metodi:**

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

6.8.2.7 EnableDocumentsVectorStore

- **Implementa:** `EnableDocumentsPort`;

- **Attributi:**

- `vectorStoreManager: VectorStoreManager.`

- **Metodi:**

- `enableDocuments(documentsIds:List<DocumentId>): List<DocumentOperationResponse>`



Implementazione del metodo astratto di `EnableDocumentsPort` per riabilitare una lista di documenti a partire dal loro `DocumentId`, dialogando con il vector store. Ritorna una lista di `DocumentOperationResponse`, che indica l'esito dell'operazione per ogni documento.

6.8.2.8 VectorStoreChromaDBManager

Vedi (§6.3.2.8) .

6.8.2.9 VectorStoreDocumentOperationResponse

Vedi (§6.3.2.9) .

6.8.2.10 VectorStoreManager

Vedi (§6.3.2.10) .

6.8.2.11 VectorStorePineconeManager

Vedi (§6.3.2.11) .



6.9 GetChatMessages

6.9.1 Diagramma delle classi

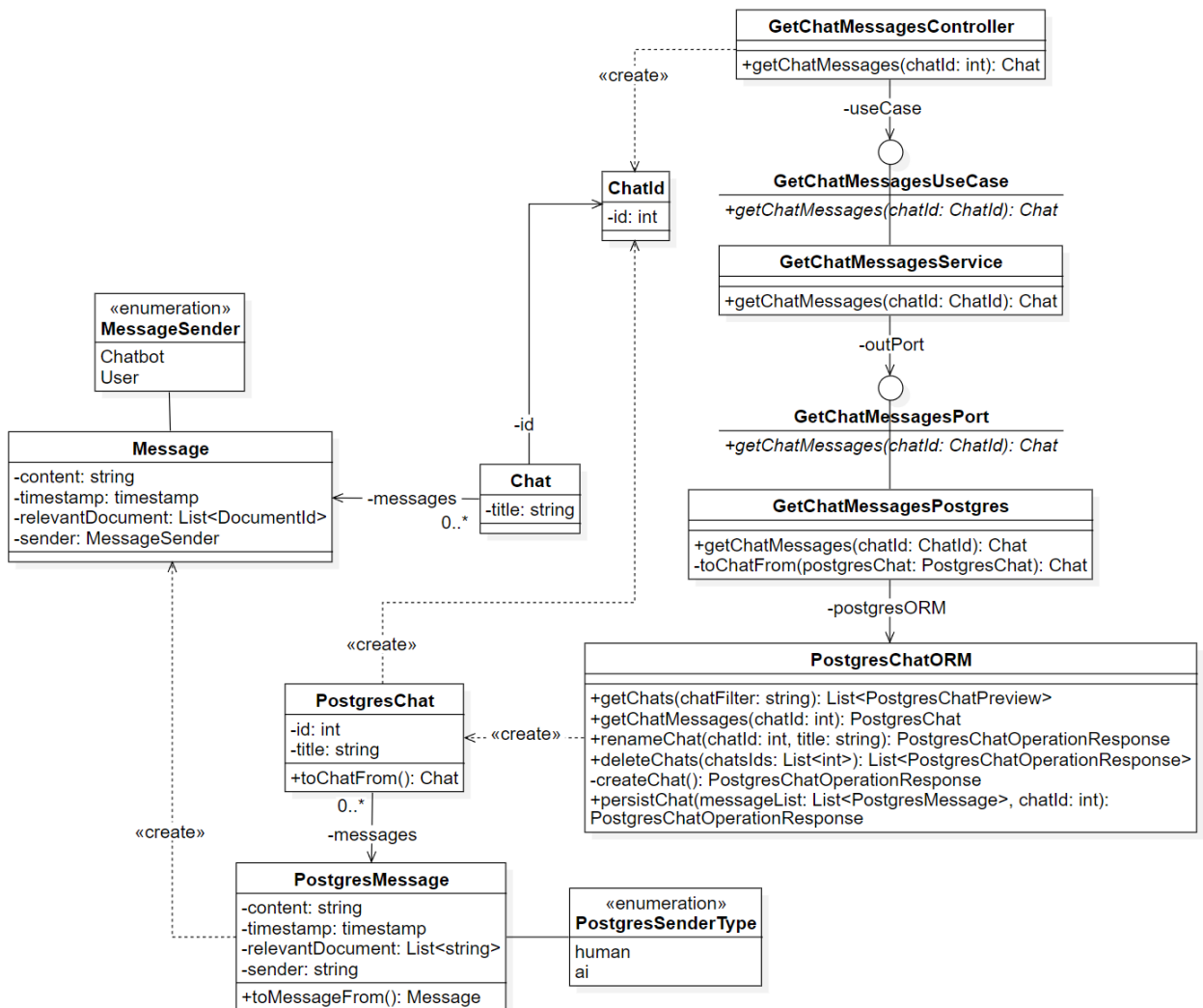


Figura 14: Diagramma delle classi della componente GetChatMessages

6.9.2 Lista delle sottocomponenti

6.9.2.1 Chat

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

6.9.2.2 ChatId

Vedi (§6.1.2.7).



6.9.2.3 GetChatMessagesController

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

6.9.2.4 GetChatMessagesPort

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

6.9.2.5 GetChatMessagesPostgres

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

6.9.2.6 GetChatMessagesService

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

6.9.2.7 GetChatMessagesUseCase

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

6.9.2.8 Message

Vedi (§6.1.2.9).



6.9.2.9 MessageSender

Vedi (§6.1.2.11) .

6.9.2.10 PostgresChat

Vedi (§6.1.2.13) .

6.9.2.11 PostgresChatORM

Vedi (§6.1.2.18) .

6.9.2.12 PostgresMessage

Vedi (§6.1.2.15) .

6.10 GetChats

6.10.1 Diagramma delle classi

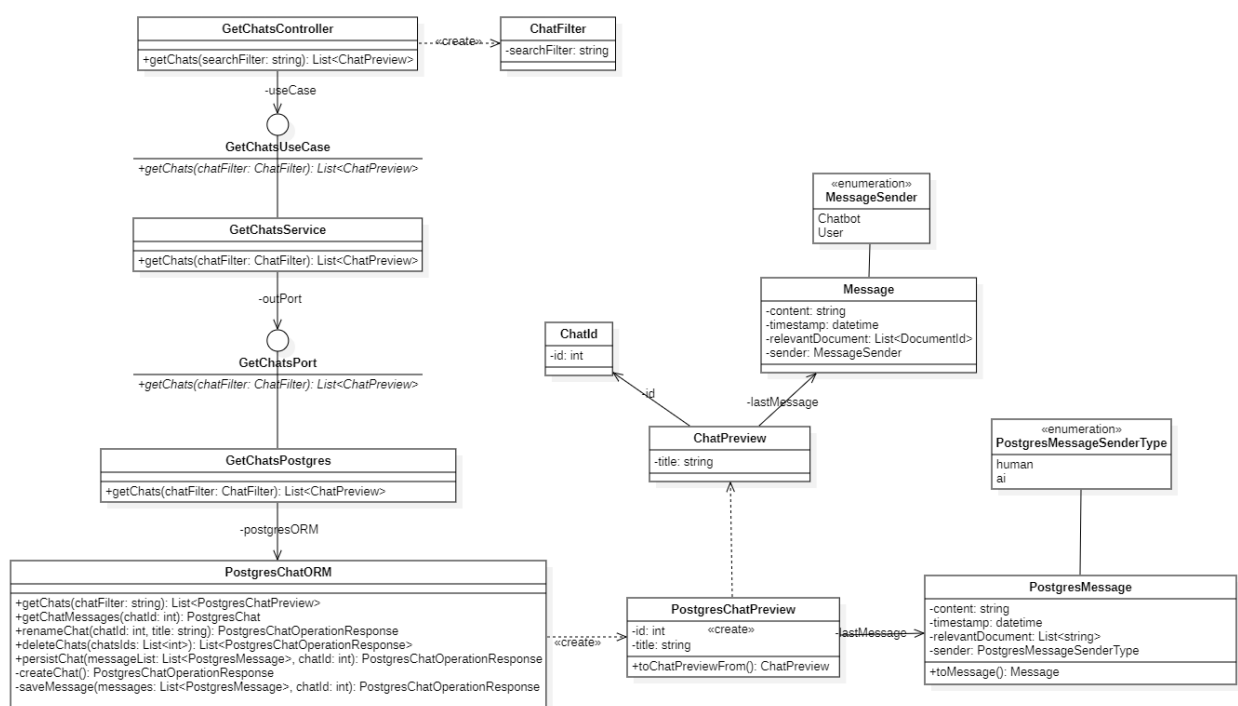


Figura 15: Diagramma delle classi della componente GetChats

6.10.2 Lista delle sottocomponenti

6.10.2.1 ChatFilter

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



6.10.2.2 ChatId

Vedi (§6.1.2.7).

6.10.2.3 ChatPreview

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

6.10.2.4 GetChatsController

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

6.10.2.5 GetChatsPort

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

6.10.2.6 GetChatsPostgres

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

6.10.2.7 GetChatsService

- **Implementa:** GetChatsUseCase;
- **Attributi:**
 - `outPort: GetChatsPort.`
- **Metodi:**



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

6.10.2.8 GetChatsUseCase

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

6.10.2.9 Message

Vedi (§6.1.2.9).

6.10.2.10 MessageSender

Vedi (§6.1.2.11).

6.10.2.11 PostgresChatORM

Vedi (§6.1.2.18).

6.10.2.12 PostgresChatPreview

- **Attributi:**
 - `id: int;`
 - `title: string;`
 - `lastMessage: PostgresMessage.`
- **Metodi:**
 - `toChatPreviewFrom(): ChatPreview` Metodo per ottenere una `ChatPreview` a partire dal `PostgresChatPreview`.

6.10.2.13 PostgresMessage

Vedi (§6.1.2.15).

6.10.2.14 PostgresMessageSenderType

Vedi (§6.1.2.16).



6.11 GetConfiguration

6.11.1 Diagramma delle classi

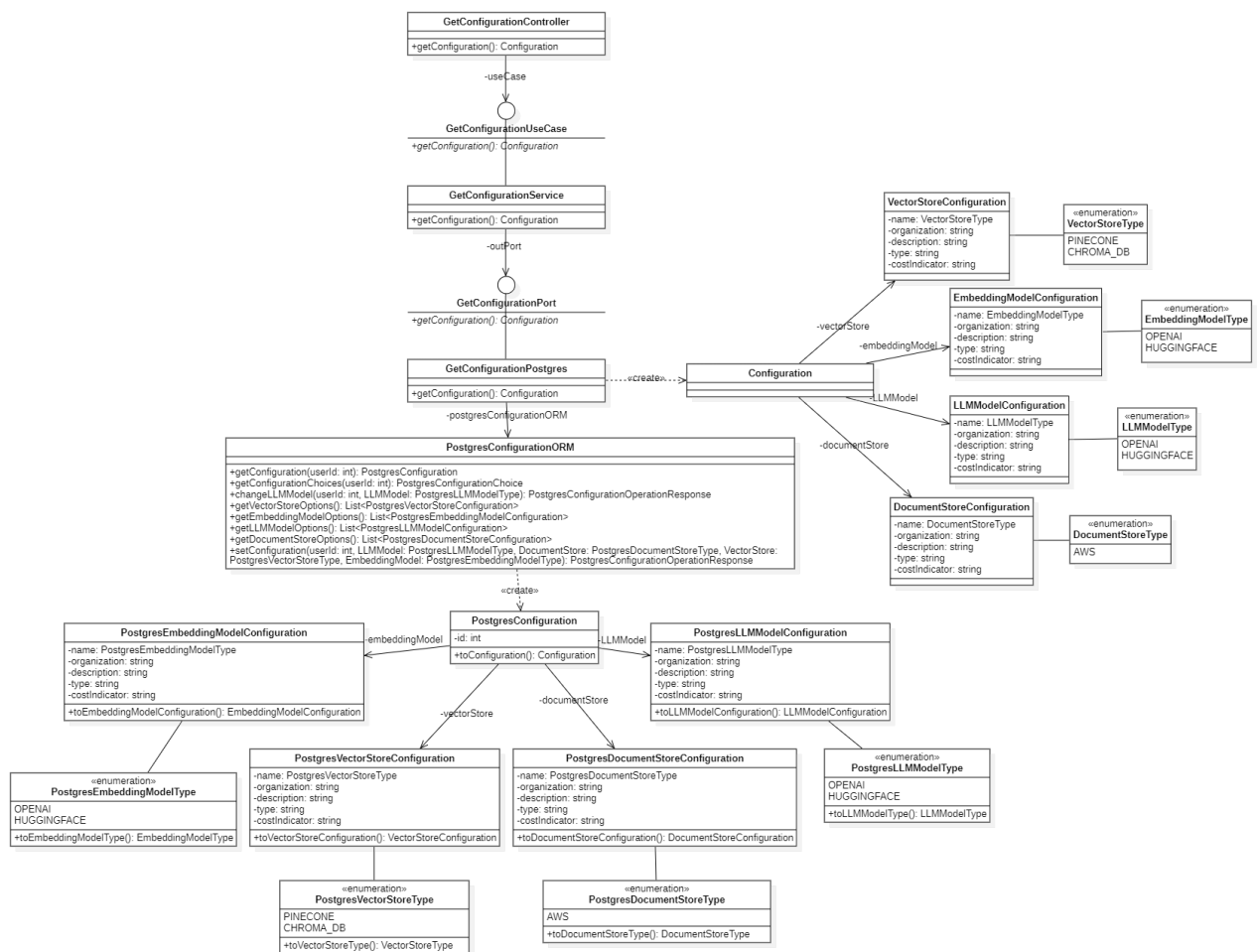


Figura 16: Diagramma delle classi della componente GetConfiguration

6.11.2 Lista delle sottocomponenti

6.11.2.1 Configuration

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

6.11.2.2 DocumentStoreConfiguration

- **Attributi:**
 - costIndicatore: string;



- `description: string;`
- `name: DocumentStoreType;`
- `organization: string;`
- `type: string.`

6.11.2.3 DocumentStoreType (Enumeration)

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

6.11.2.4 EmbeddingModelConfiguration

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

6.11.2.5 EmbeddingModelType (Enumeration)

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

6.11.2.6 GetConfigurationController

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

6.11.2.7 GetConfigurationPort

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

6.11.2.8 GetConfigurationPostgres

- **Implementa:** `GetConfigurationPort;`
- **Attributi:**



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

6.11.2.9 GetConfigurationService

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

6.11.2.10 GetConfigurationUseCase

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

6.11.2.11 LLMModelConfiguration

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

6.11.2.12 LLMModelType (Enumeration)

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

6.11.2.13 PostgresConfiguration

Vedi (§6.4.2.2).

**6.11.2.14 PostgresConfigurationORM**

Vedi (§6.2.2.8) .

6.11.2.15 PostgresDocumentStoreConfiguration

Vedi (§6.4.2.4) .

6.11.2.16 PostgresDocumentStoreType

Vedi (§6.4.2.5) .

6.11.2.17 PostgresEmbeddingModelConfiguration

Vedi (§6.4.2.6) .

6.11.2.18 PostgresEmbeddingModelType

Vedi (§6.4.2.7) .

6.11.2.19 PostgresLLMModelConfiguration

Vedi (§6.4.2.8) .

6.11.2.20 PostgresLLMModelType

Vedi (§6.4.2.9) .

6.11.2.21 PostgresVectorStoreConfiguration

Vedi (§6.4.2.10) .

6.11.2.22 PostgresVectorStoreType

Vedi (§6.4.2.11) .

6.11.2.23 VectorStoreConfiguration

- **Attributi:**

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

6.11.2.24 VectorStoreType (Enumeration)

- **Valori:**

- `CHROMA DB;`
- `PINECONE.`



6.12.1 Diagramma delle classi



6.12.2.1 ConfigurationOptions

- 134 di 158



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

6.12.2.2 DocumentStoreConfiguration

Vedi (§6.11.2.2).

6.12.2.3 DocumentStoreType

Vedi (§6.11.2.3).

6.12.2.4 EmbeddingModelConfiguration

Vedi (§6.11.2.4).

6.12.2.5 EmbeddingModelType

Vedi (§6.11.2.5).

6.12.2.6 GetConfigurationOptions

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

6.12.2.7 GetConfigurationOptionsPort

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

6.12.2.8 GetConfigurationOptionsPostgres

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



6.12.2.9 GetConfigurationOptionsService

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

6.12.2.10 GetConfigurationOptionsUseCase

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

6.12.2.11 LLMModelConfiguration

Vedi (§6.11.2.11)

6.12.2.12 LLMModelType

Vedi (§6.11.2.12).

6.12.2.13 PostgresConfigurationORM

Vedi (§6.2.2.8).

6.12.2.14 PostgresDocumentStoreConfiguration

Vedi (§6.4.2.4).

6.12.2.15 PostgresDocumentStoreType

Vedi (§6.4.2.5).

6.12.2.16 PostgresEmbeddingModelConfiguration

Vedi (§6.4.2.6).

6.12.2.17 PostgresEmbeddingModelType

Vedi (§6.4.2.7).

6.12.2.18 PostgresLLMModelConfiguration

Vedi (§6.4.2.8).

6.12.2.19 PostgresLLMModelType

Vedi (§6.4.2.9).

**6.12.2.20 PostgresVectorStoreConfiguration**

Vedi (§6.4.2.10) .

6.12.2.21 PostgresVectorStoreType

Vedi (§6.4.2.11) .

6.12.2.22 VectorStoreConfiguration

Vedi (§6.11.2.23) .

6.12.2.23 VectorStoreType

Vedi (§6.11.2.24) .





6.13.2 Lista delle sottocomponenti

6.13.2.1 AWSDocument

Vedi (§6.7.2.1) .

6.13.2.2 AWSS3Manager

Vedi (§6.6.2.2) .

6.13.2.3 Document

Vedi (§6.17.2.5) .

6.13.2.4 DocumentContent

Vedi (§6.17.2.6) .

6.13.2.5 DocumentId

Vedi (§6.3.2.6) .

6.13.2.6 DocumentMetadata

Vedi (§6.14.2.5) .

6.13.2.7 DocumentStatus

Vedi (§6.7.2.6) .

6.13.2.8 DocumentType (Enumeration)

Vedi (§6.14.2.7) .

6.13.2.9 GetDocumentContentController

- **Attributi:**

- `useCase: GetDocumentsContentUseCase .`

- **Metodi:**

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

6.13.2.10 GetDocumentsContentFacadeService

- **Implementa:** GetDocumentsContentUseCase;

- **Attributi:**

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

- **Metodi:**



- `getDocumentsContent(documentId: List<DocumentId>): List<Document>`
Implementazione del metodo astratto di `GetDocumentUseCase`. Le operazioni all'interno della facade vengono svolte nel seguente ordine:
 1. Recupero della lista di status dei documenti dal vector store;
 2. Recupero dei documenti e il loro contenuto dal sistema di archiviazione;
 3. Costruzione di una lista di oggetti `Document` a partire dagli output dei passaggi precedenti.

6.13.2.11 GetDocumentsContent

Vedi (§6.7.2.16) .

6.13.2.12 GetDocumentsContentAWSS3

Vedi (§6.7.2.17) .

6.13.2.13 GetDocumentsContentPort

Vedi (§6.7.2.18) .

6.13.2.14 GetDocumentsStatus

Vedi (§6.7.2.19) .

6.13.2.15 GetDocumentsStatusPort

Vedi (§6.7.2.20) .

6.13.2.16 GetDocumentsStatusVectorStore

Vedi (§6.7.2.21) .

6.13.2.17 GetDocumentsContentUseCase

- **Metodi:**
 - `getDocumentsContent(documentId: List<DocumentId>): List<Document>`
Metodo astratto per recuperare tutte le informazioni di una lista di documenti, compresi i loro contenuti, a partire da una lista dei loro id.

6.13.2.18 PlainDocument

Vedi (§6.17.2.27) .

6.13.2.19 Status (Enumeration)

Vedi (§6.7.2.27) .

6.13.2.20 VectorStoreChromaDBManager

Vedi (§6.3.2.8) .

**6.13.2.21 VectorStoreDocumentStatusResponse**

Vedi (§6.7.2.31) .

6.13.2.22 VectorStoreManager

Vedi (§6.3.2.10) .

6.13.2.23 VectorStorePineconeManager

Vedi (§6.3.2.11) .



6.14 GetDocuments

6.14.1 Diagramma delle classi

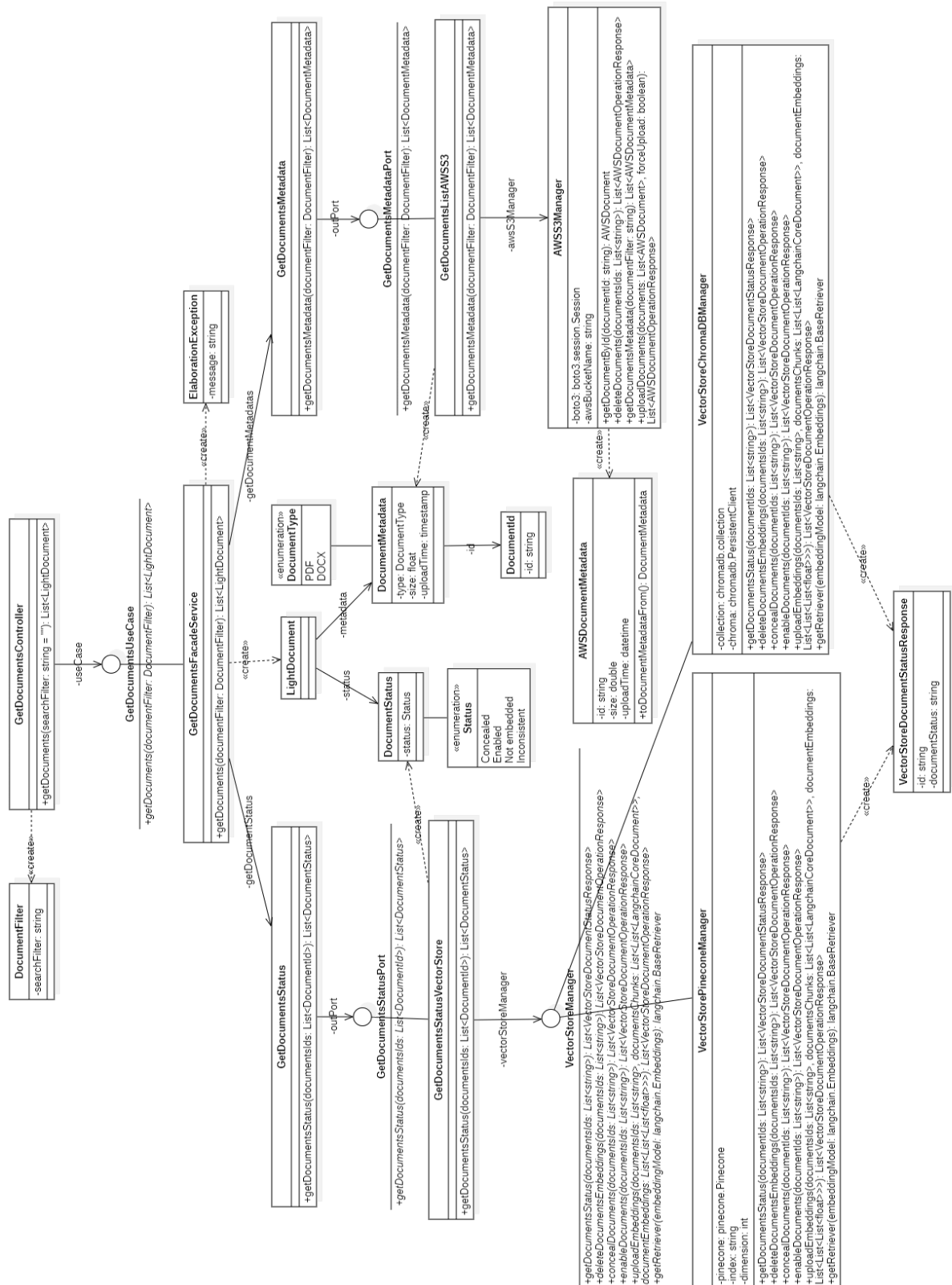


Figura 19: Diagramma delle classi componente GetDocuments



6.14.2 Lista delle sottocomponenti

6.14.2.1 AWSDocumentMetadata

- **Attributi:**

- `id: string;`
- `size: float;`
- `uploadTime: datetime.`

- **Metodi:**

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

6.14.2.2 AWSS3Manager

Vedi (§6.6.2.2).

6.14.2.3 DocumentFilter

- **Attributi:**

- `searchFilter: string.`

6.14.2.4 DocumentId

Vedi (§6.3.2.6).

6.14.2.5 DocumentMetadata

- **Attributi:**

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

6.14.2.6 DocumentStatus

Vedi (§6.7.2.6).

6.14.2.7 DocumentType (Enumeration)

- **Valori:**

- `DOCX;`
- `PDF.`



6.14.2.8 ElaborationException

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

6.14.2.9 GetDocumentsController

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

6.14.2.10 GetDocumentsFacadeService

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

6.14.2.11 GetDocumentsListAWSS3

- **Implementa:** `GetDocumentsMetadataPort;`
- **Attributi:**
 - `awsS3Manager: AWSS3Manager.`
- **Metodi:**
 - `getDocumentsMetadata(documentFilter:DocumentFilter): List<DocumentMetadata>`
Implementazione del metodo astratto di `GetDocumentsMetadataPort` per ricercare documenti tramite `outPort` nel sistema di archiviazione a partire da un filtro.



6.14.2.12 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.

6.14.2.13 GetDocumentsMetadataPort

- **Metodi:**

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

6.14.2.14 GetDocumentsStatus

Vedi (§6.7.2.19) .

6.14.2.15 GetDocumentsStatusPort

Vedi (§6.7.2.20) .

6.14.2.16 GetDocumentsStatusVectorStore

Vedi (§6.7.2.21) .

6.14.2.17 GetDocumentsUseCase

- **Metodi:**

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

6.14.2.18 LightDocument

- **Attributi:**

- metadata: DocumentMetadata ;
 - status: DocumentStatus .

6.14.2.19 Status (Enumeration)

Vedi (§6.7.2.27) .

6.14.2.20 VectorStoreChromaDBManager

Vedi (§6.3.2.8) .



6.14.2.21 VectorStoreDocumentStatusResponse

Vedi (§6.7.2.31) .

6.14.2.22 VectorStoreManager

Vedi (§6.3.2.10) .

6.14.2.23 VectorStorePineconeManager

Vedi (§6.3.2.11) .

6.15 RenameChat

6.15.1 Diagramma delle classi

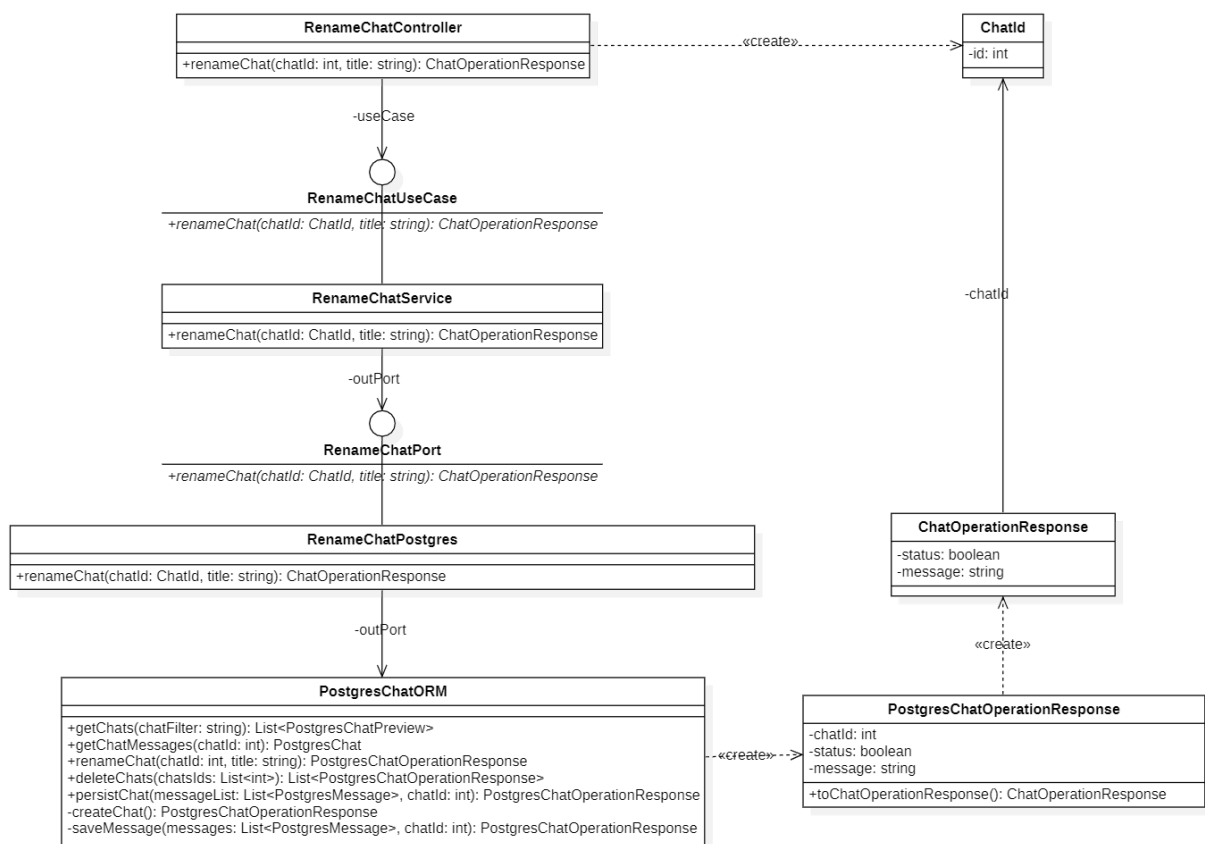


Figura 20: Diagramma delle classi componente RenameChat

6.15.2 Lista delle sottocomponenti

6.15.2.1 ChatId

Vedi (§6.1.2.7) .



6.15.2.2 ChatOperationResponse

Vedi (§6.1.2.8).

6.15.2.3 RenameChatController

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

6.15.2.4 RenameChatPort

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

6.15.2.5 RenameChatPostgres

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

6.15.2.6 RenameChatService

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

6.15.2.7 RenameChatUseCase

- **Metodi:**



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

6.15.2.8 PostgresChatOperationResponse

Vedi (§6.1.2.14) .

6.15.2.9 PostgresChatORM

Vedi (§6.1.2.18) .

6.16 SetConfiguration

6.16.1 Diagramma delle classi

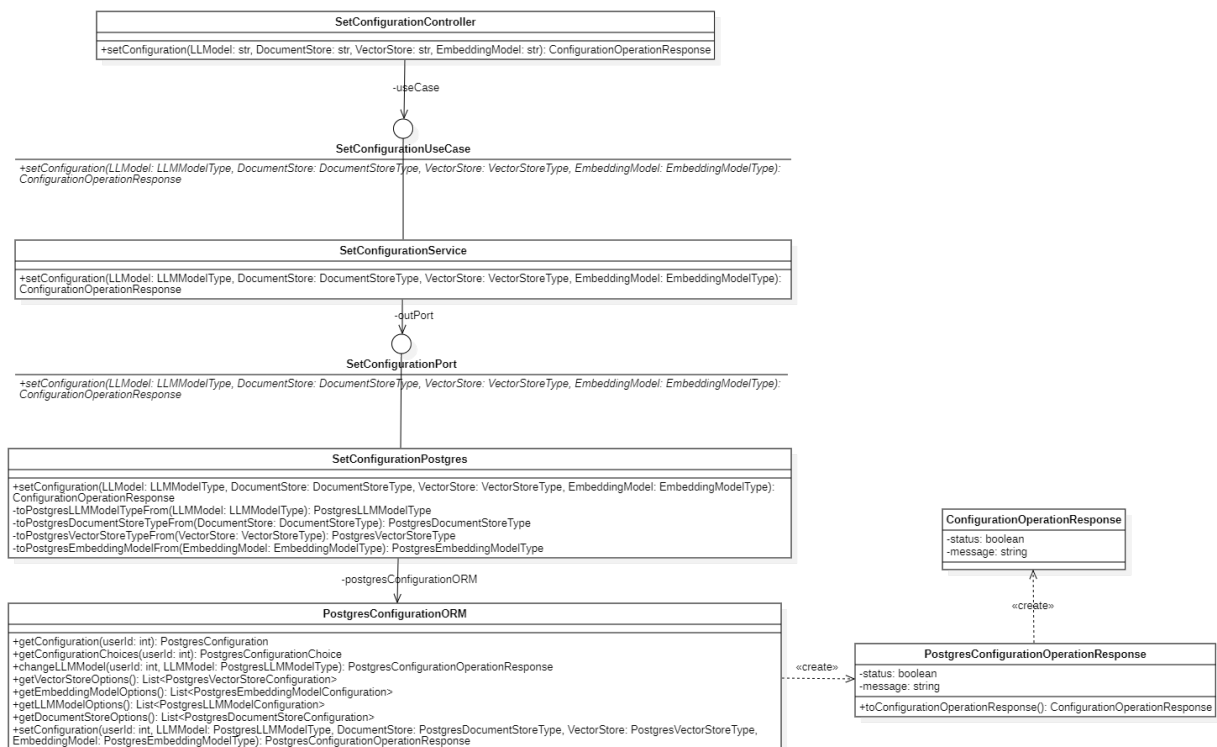


Figura 21: Diagramma delle classi componente SetConfiguration

6.16.2 Lista delle sottocomponenti

6.16.2.1 ConfigurationOperationResponse

Vedi (§6.2.2.6) .

6.16.2.2 PostgresConfigurationOperationResponse

Vedi (§6.2.2.7) .



6.16.2.3 PostgresConfigurationORM

Vedi (§6.2.2.8) .

6.16.2.4 SetConfigurationController

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

6.16.2.5 SetConfigurationPort

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

6.16.2.6 SetConfigurationPostgres

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



- `toPostgresVectorStoreTypeFrom(VectorStore:VectorStoreType): PostgresVectorStoreType`
Metodo per convertire un `VectorStoreType` in `PostgresVectorStoreType`.

6.16.2.7 SetConfigurationService

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

6.16.2.8 SetConfigurationUseCase

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



6.17 UploadDocuments

6.17.1 Diagramma delle classi

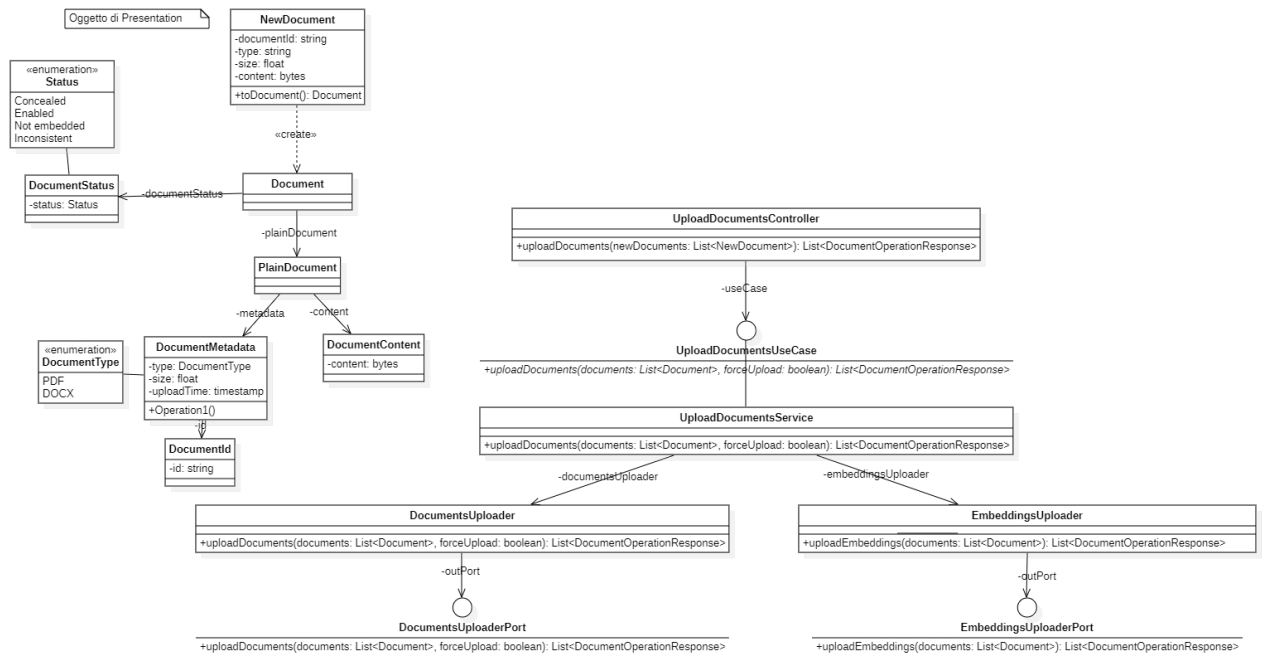


Figura 22: Diagramma delle classi componente UploadDocuments (continua nei prossimi diagrammi)

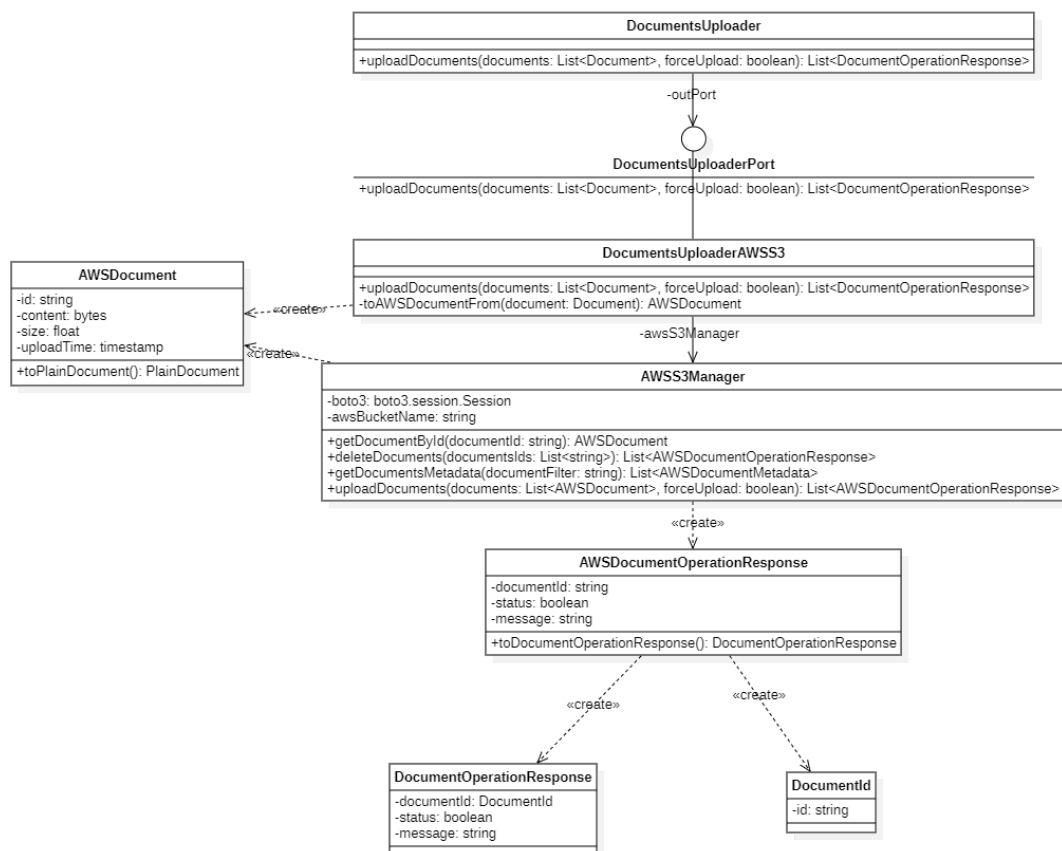


Figura 23: Diagramma delle classi componente UploadDocuments (continua nel prossimo diagramma)

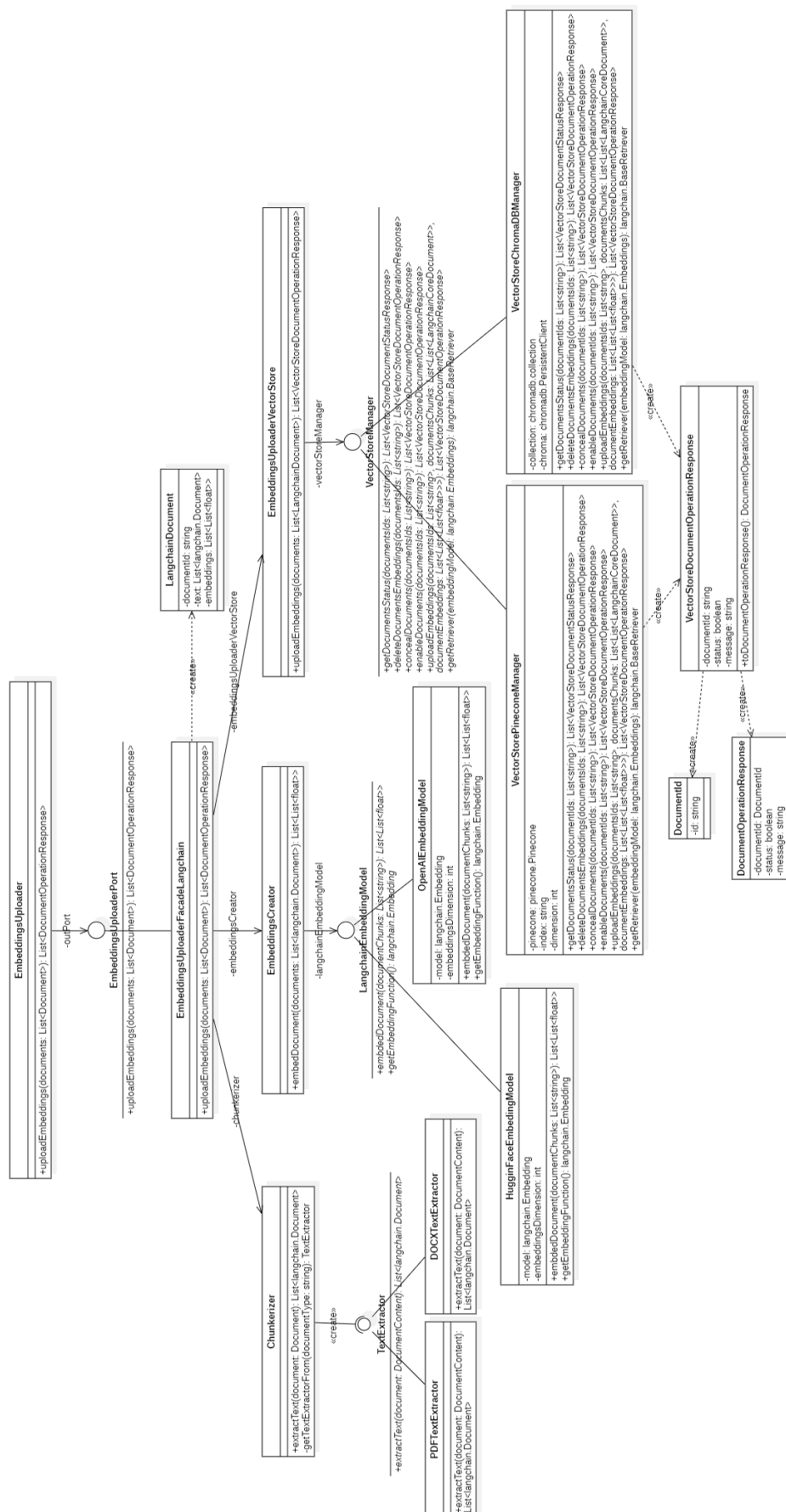


Figura 24: Diagramma delle classi componente UploadDocuments



6.17.2 Lista delle sottocomponenti

6.17.2.1 AWSDocument

Vedi (§6.7.2.1) .

6.17.2.2 AWSDocumentOperationResponse

Vedi (§6.6.2.1) .

6.17.2.3 AWSS3Manager

Vedi (§6.6.2.2) .

6.17.2.4 Chunkerizer

Vedi (§6.7.2.3) .

6.17.2.5 Document

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

6.17.2.6 DocumentContent

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

6.17.2.7 DocumentId

Vedi (§6.3.2.6) .

6.17.2.8 DocumentMetadata

Vedi (§6.14.2.5) .

6.17.2.9 DocumentOperationResponse

Vedi (§6.3.2.7) .

6.17.2.10 DocumentStatus

Vedi (§6.7.2.6) .

6.17.2.11 DocumentType (Enumeration)

Vedi (§6.14.2.7) .



6.17.2.12 DocumentsUploader

- **Attributi:**

- `outPort: DocumentsUploaderPort.`

- **Metodi:**

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

6.17.2.13 DocumentUploaderAWSS3

- **Implementa:** DocumentsUploaderPort;

- **Attributi:**

- `awsS3Manager: AWSS3Manager.`

- **Metodi:**

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

6.17.2.14 DocumentsUploaderPort

- **Metodi:**

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

6.17.2.15 DOCXTextExtractor

Vedi (§6.7.2.7).

6.17.2.16 EmbeddingsCreator

Vedi (§6.7.2.8).

6.17.2.17 EmbeddingsUploader

Vedi (§6.7.2.9).

**6.17.2.18 EmbeddingsUploaderFacadeLangchain**

Vedi (§6.7.2.10) .

6.17.2.19 EmbeddingsUploaderPort

Vedi (§6.7.2.11) .

6.17.2.20 EmbeddingsUploaderVectorStore

Vedi (§6.7.2.12) .

6.17.2.21 HuggingFaceEmbeddingModel

Vedi (§6.7.2.22) .

6.17.2.22 LangchainDocument

Vedi (§6.7.2.23) .

6.17.2.23 LangchainEmbeddingModel

Vedi (§6.7.2.24) .

6.17.2.24 NewDocument

- **Attributi:**

- `content: bytes;`
- `documentId: string;`
- `size: float;`
- `type: string.`

- **Metodi:**

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

6.17.2.25 OpenAIEmbeddingModel

Vedi (§6.7.2.25) .

6.17.2.26 PDFTextExtractor

Vedi (§6.7.2.26) .

6.17.2.27 PlainDocument

- **Attributi:**

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



6.17.2.28 Status (Enumeration)

Vedi (§6.7.2.27) .

6.17.2.29 TextExtractor

Vedi (§6.7.2.28) .

6.17.2.30 UploadDocumentsController

- **Attributi:**

- `useCase: UploadDocumentsUseCase .`

- **Metodi:**

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

6.17.2.31 UploadDocumentsService

- **Implementa:** UploadDocumentsUseCase;

- **Attributi:**

- `documentUploader: DocumentsUploader ;`
- `embeddingsUploader: EmbeddingsUploader .`

- **Metodi:**

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

6.17.2.32 UploadDocumentsUseCase

- **Metodi:**

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

6.17.2.33 VectorStoreChromaDBManager

Vedi (§6.3.2.8) .

**6.17.2.34 VectorStoreDocumentOperationResponse**

Vedi (§6.3.2.9) .

6.17.2.35 VectorStoreManager

Vedi (§6.3.2.10) .

6.17.2.36 VectorStorePineconeManager

Vedi (§6.3.2.11) .