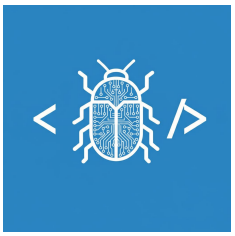


BugBusters

Specifica Tecnica$_G$

Versione 1.0.0

Stato	Approvato
Redattori$_G$	Alberto Autiero, Alberto Pignat, Linor Sadè, Luca Slongo
Verificatori$_G$	Alberto Autiero, Alberto Pignat, Leonardo Salviato, Marco Favero
Distribuzione	BugBusters, Eggon, Prof. Tullio Vardanega, Prof. Riccardo Cardin

Descrizione

Documento dove viene spiegata in dettaglio la struttura architetturale, le tecnologie utilizzate e le API$_G$ esposte del prodotto$_G$ software.

Registro delle Modifiche

Versione	Data	Descrizione	Redatto	Verificato	Approvato
1.0.0	03/04/2026	Approvazione documento	-	-	Linor Sadè
0.10.0	03/04/2026	Verifica documento	-	Alberto Pignat	-
0.9.1	03/04/2026	Aggiunti ai termini presenti nel Glossario$_G$ la G	Alberto Autiero	-	-
0.9.0	01/04/2026	Verifica$_G$ del documento	-	Alberto Autiero	-
0.8.1	01/04/2026	Aggiunta parte di modulo$_G$ AI Assistant$_G$	Alberto Pignat	-	-
0.8.0	30/03/2026	Verifica$_G$ del documento	-	Leonardo Salviato	-
0.7.1	31/03/2026	Prima stesura parte front-end$_G$	Linor Sadè	-	-
0.7.0	30/03/2026	Verifica$_G$ del documento	-	Alberto Pignat	-
0.6.1	31/03/2026	Continuazione paragrafo Pattern e aggiunta diagrammi	Luca Slongo	-	-
0.6.0	30/03/2026	Verifica$_G$ del documento	-	Alberto Autiero	-
0.5.1	31/03/2026	Continuazione paragrafo Pattern e aggiunta diagrammi	Luca Slongo	-	-
0.5.0	18/03/2026	Verifica	-	Marco Favero	-
0.4.1	27/03/2026	Prima stesura paragrafo Database e continuazione paragrafo Design Pattern	Alberto Pignat	-	-
0.4.0	18/03/2026	Verifica	-	Leonardo Salviato	-
0.3.1	25/03/2026	Stesura paragrafo Tecnologie e Linguaggi	Luca Slongo	-	-
0.3.0	18/03/2026	Verifica	-	Alberto Pignat	-
0.2.1	18/03/2026	Prima stesura design pattern	Linor Sadè	-	-
0.2.0	16/03/2026	Verifica	-	Alberto Autiero	-
0.1.1	16/03/2026	Prima stesura della tabella dei requisiti, paragrafo requisiti e paragrafo architettura	Alberto Pignat	-	-
0.1.0	24/02/2026	Verifica	-	Alberto Pignat	-
0.0.1	24/02/2026	Prima stesura del documento	Alberto Autiero	-	-

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

Il documento ha come scopo la descrizione dettagliata dell'architettura del sistema, delle tecnologie utilizzate e delle API$_G$ esposte. Viene inoltre fornita una panoramica dei requisiti funzionali$_G$ soddisfatti dal prodotto$_G$ software.

1.1 Glossario

Il glossario$_G$ raccoglie e definisce i termini, gli acronimi e le abbreviazioni impiegati nel documento e nel progetto$_G$ Nexum. L'obiettivo è fornire definizioni univoche per ridurre ambiguità, garantire coerenza terminologica tra i membri del team e facilitare l'onboarding di nuovi partecipanti.
Per i termini tecnici e specifici utilizzati in questo documento, si fa riferimento al glossario$_G$ disponibile al seguente link. Per maggiore usabilità e facilitá di consultazione, il glossario$_G$ è accessibile anche aprendo dal nostro sito web i vari documenti con il viewer pdf da noi sviluppato.

1.2 Riferimenti normativi

- **Capitolato$_G$ d'appalto C5: Nexum - Piattaforma di consulenza e documentazione previdenziale**
  https://www.math.unipd.it/~tullio/IS-1/2025/Progetto/C5.pdf

1.3 Riferimenti informativi

- **Glossario$_G$ (versione 2.0.0) :**
  https://bugbustersunipd.github.io/DocumentazioneSWE/PB/GLOSSARIO/Glossario.pdf

1.3.1 Linguaggi, Framework e Librerie

- **Documentazione ufficiale di Ruby**
  https://www.ruby-lang.org/en/documentation/
- **Documentazione ufficiale di Ruby on Rails**
  https://rubyonrails.org/docs
- **Documentazione ufficiale di Angular 21**
  https://angular.dev/overview
- **Documentazione ufficiale di PrimeNG**
  https://primeng.org/installation
- **Documentazione ufficiale di Bootstrap**
  https://getbootstrap.com/docs/5.3/getting-started/introduction/

2 Tecnologie

Il presente capitolo illustra lo stack tecnologico adottato per la progettazione e la realizzazione del prodotto$_G$ software. Le scelte implementative sono state guidate dalla necessità di sviluppare un'architettura solida, garantendo al contempo elevati standard di scalabilità$_G$, manutenibilità$_G$ e sicurezza$_G$ del sistema. Nelle sottosezioni seguenti verranno passati in rassegna i linguaggi di programmazione, i framework, le librerie e gli strumenti di supporto operativi impiegati dal team, evidenziando il ruolo specifico di ciascun elemento all'interno del ciclo di vita del progetto$_G$.

2.1 Linguaggi

Questa sezione descrive i linguaggi di programmazione, di markup e di interrogazione fondamentali utilizzati per la stesura del codice sorgente. La selezione comprende sia gli standard consolidati per la strutturazione e stilizzazione dell'interfaccia$_G$ utente (Frontend$_G$), sia i linguaggi orientati agli oggetti scelti per implementare la logica applicativa (Backend$_G$), fino ad arrivare agli strumenti dichiarativi per la gestione delle basi di dati. La Tabella 3 riassume i linguaggi adottati, specificandone la versione di riferimento e il relativo ambito di utilizzo.

Tabella 3: Tabella dei Linguaggi

Nome	Descrizione	Versione
Typescript$_G$	Linguaggio di programmazione basato su JavaScript che aggiunge la tipizzazione statica	6.0
HTML	Linguaggio di markup usato per creare pagine web	5
CSS	Linguaggio usato per definire lo stile delle pagine web	3
Ruby$_G$	Linguaggio di programmazione orientato agli oggetti	3.3.1
SQL	Linguaggio per la gestione e manipolazione di database relazionali	SQL:2023

2.2 Framework

In questa sezione vengono presentati i framework adottati per costituire l'impalcatura architetturale del prodotto$_G$ software. La scelta si è orientata verso soluzioni moderne e ampiamente supportate dalla community, capaci di garantire una netta separazione tra la logica di presentazione e le regole di business. Nello specifico, la Tabella 4 illustra le infrastrutture cardine impiegate per lo sviluppo della Single Page Application lato client e per la realizzazione delle API$_G$ lato server.

Tabella 4: Tabella dei Framework

Nome	Descrizione	Versione
Angular$_G$	Framework open source per la creazione e sviluppo di single page applications	21
Ruby on Rails$_G$	Framework open source server side per applicazioni web	8.1
Bootstrap	Framework CSS per lo sviluppo di interfacce responsive e mobile-first	5.3.8
Cypress	Framework per il testing end-to-end di applicazioni web	15.13.0
Vitest	Framework per il testing unitario e di integrazione in ambiente JavaScript	4.0.8

2.3 Librerie

Oltre alle tecnologie di base, lo sviluppo del sistema ha richiesto l'integrazione di pacchetti e librerie di terze parti. Queste componenti si sono rivelate essenziali per estendere le funzionalità$_G$

native dei framework, ottimizzare i tempi di sviluppo e risolvere specifiche problematiche di dominio in modo standardizzato, affidabile e sicuro. La Tabella 5 elenca le principali dipendenze esterne introdotte all'interno degli ambienti Node.js e Ruby$_G$.

Tabella 5: Tabella delle Librerie

Nome	Descrizione	Versione
rails	Framework web full-stack per Ruby$_G$.	8.1.2.1
pg	Adattatore PostgreSQL$_G$ per Active Record.	1.1
puma	Server web concorrente, veloce e multi-thread per Ruby$_G$.	5.0.0
tzinfo-data	Dati sui fusi orari necessari per i sistemi Windows e JRuby.	1.2026.1
solid_cache	Backend$_G$ basato su database per Rails.cache.	1.0.10
solid_queue	Backend$_G$ basato su database per Active Job.	1.4.0
solid_cable	Backend$_G$ basato su database per Action Cable.	3.0.12
bootsnap	Ottimizza e riduce i tempi di avvio dell'applicazione tramite caching.	1.23.0
kamal	Strumento per il deploy di applicazioni web ovunque tramite container Docker$_G$.	2.11.0
thruster	Proxy HTTP per accelerare Puma con caching degli asset e compressione.	0.1.20
image_processing	Fornisce metodi helper per la gestione e la trasformazione delle immagini.	1.2.0
rack-cors	Middleware per la gestione delle richieste CORS (Cross-Origin Resource Sharing).	3.0.0
combine_pdf	Libreria per unire, formattare o manipolare file PDF.	1.0.31
csv	Fornisce un'interfaccia$_G$ standard per leggere e scrivere file CSV.	3.3.5
aws-sdk-textract	Client AWS SDK ufficiale per il servizio OCR$_G$ Amazon Textract.	1.90.0
aws-sdk-bedrockruntime	Client AWS$_G$ SDK ufficiale per interagire con i modelli LLM$_G$ di Amazon Bedrock$_G$.	1.74.0
simplecov	Strumento per l'analisi della copertura del codice durante i test$_G$.	0.22.0
minitest	Suite completa e leggera per il testing (TDD, BDD, benchmarking).	6.0.3
minitest-mock	Estensione per Minitest dedicata al mocking di oggetti nei test$_G$.	5.27.0
mocha	Libreria ampiamente utilizzata per il mocking e lo stubbing nei test$_G$.	3.1.0
debug	Strumento standard per il debugging delle applicazioni Ruby$_G$.	1.11.1
bundler-audit	Scanner per rilevare vulnerabilità di sicurezza$_G$ note nelle gemme installate.	0.9.3
brakeman	Scanner di analisi statica per rilevare vulnerabilità di sicurezza$_G$ in app Rails.	8.0.4
rubocop-rails-omakase	Configurazione standard ufficiale (stile Omakase) per RuboCop in Rails.	1.1.0

Continua nella prossima pagina...

Tabella 5 – continua dalla pagina precedente

Nome	Descrizione	Versione
RxJs	Libreria per la programmazione reattiva.	7.8.0
PrimeNG	Componenti UI per Angular.	21.1.1
ChartJS	Libreria per la visualizzazione di grafici dinamici e interattivi.	4.5.1
Quill	Editor di testo WYSIWYG per Angular.	2.0.3

2.4 Strumenti di Sviluppo

L'efficienza$_G$, la solidità e la riproducibilità dell'intero ciclo di vita del software sono state assicurate attraverso l'impiego di una serie di strumenti operativi e di orchestrazione avanzati. Questa categoria, dettagliata nella Tabella 6, comprende gli ambienti di sviluppo integrati (IDE), i sistemi di containerizzazione volti a uniformare gli ambienti tra i vari membri del team, e le piattaforme per la modellazione e l'amministrazione della persistenza dei dati.

Tabella 6: Tabella degli Strumenti di Sviluppo

Nome	Descrizione	Versione
Visual Studio Code	IDE open source per lo sviluppo software sviluppato da Microsoft	1.114
Docker$_G$	Piattaforma che permette di creare, distribuire e eseguire applicazioni in container	29.3.1
PostgreSQL$_G$	Database relazionale open source	18.3
pgAdmin	Strumento grafico per amministrare e gestire database PostgreSQL$_G$	9.13

3 API

Il sistema espone un set di API$_G$ RESTful che consentono l'interazione tra il frontend$_G$ e il backend$_G$, nonché l'integrazione con eventuali servizi esterni. Le API$_G$ sono progettate per essere intuitive, sicure e performanti, seguendo le best practice$_G$ del settore per la progettazione di interfacce di programmazione. Di seguito viene fornita una panoramica delle principali API$_G$ esposte dal sistema, suddivise per funzionalità$_G$ e endpoint.

3.1 Document Management API

Tabella 7: Document Management API

Endpoint	Metodo	Descrizione
/documents/split	POST	Upload PDF, avvia splitting + estrazione
/documents/process_file	POST	Upload CSV/immagine
/documents/uploads	GET	Lista documenti caricati
/documents/uploads/:id/extracted	GET	Pagine estratte di un documento

Continua nella prossima pagina...

Tabella 7 – continua dalla pagina precedente

Endpoint	Metodo	Descrizione
/documents/uploads/ :id/file	GET	Download file originale
/documents/extracted/ :id	GET	Dettaglio documento estratto
/documents/extracted/ :id/pdf	GET	Download range pagine come PDF
/documents/extracted/ :id/metadata	PATCH	Aggiorna metadati estratti
/documents/extracted/ :id/validate	PATCH	Valida documento
/documents/extracted/ :id/reassign_range	PATCH	Riassegna range pagine estratto
/documents/uploads/ :id/retry	POST	Riprova processing run fallito
/documents/extracted/ :id/retry	POST	Riprova estrazione fallita
/documents/uploads/ :id	DELETE	Elimina documento caricato

3.2 Lookups API

Tabella 8: Lookups API

Endpoint	Metodo	Descrizione
/lookups/companies	GET	Nomi aziende
/lookups/users	GET	Nomi dipendenti

3.3 Sendings API

Tabella 9: Sendings API

Endpoint	Metodo	Descrizione
/sendings	GET	Lista invii effettuati
/sendings	POST	Crea un invio (documento → destinatario)

3.4 Templates API

Tabella 10: Templates API

Endpoint	Metodo	Descrizione
/templates	GET	Lista template$_G$disponibili
/templates/ :id	GET	Dettaglio template$_G$
/templates	POST	Crea nuovo template$_G$

3.5 Tones & Styles API

Tabella 11: Tones & Styles API

Endpoint	Metodo	Descrizione
/tones	GET	Lista toni disponibili
/tones	POST	Crea tono
/tones/:id	DELETE	Elimina tono
/styles	GET	Lista stili disponibili
/styles	POST	Crea stile
/styles/:id	DELETE	Elimina stile

3.6 Posts API

Tabella 12: Posts API

Endpoint	Metodo	Descrizione
/posts	GET	Lista post generati
/posts	POST	Crea post
/posts/:id	DELETE	Elimina post

3.7 Generated Data API

Tabella 13: Generated Data API

Endpoint	Metodo	Descrizione
/generated_data	POST	Crea contenuto AI$_G$ (asincrono)
/generated_data/:id/regenerate	POST	Rigenera contenuto AI$_G$
/generated_data/:id	GET	Stato e contenuto generazione
/generated_data/:id/rating	PATCH	Valuta contenuto generato
/generated_data/:id	DELETE	Elimina generazione

3.8 Analytics API

Tabella 14: Analytics API

Endpoint	Metodo	Descrizione
/ai_copilot_data_analyst	GET	Dati analisi CoPilot
/ai_generator_data_analyst	GET	Dati analisi Generator

4 Architettura del Sistema

Il sistema è stato documentato seguendo un approccio top-down basato sul modello C4, partendo dal contesto generale fino ad arrivare al dettaglio dei componenti interni e all'infrastruttura di

rilascio.
Dato che il sistema comprende una grande quantità di componenti è il modo migliore per evidenziare le interazioni tra di essi e la loro collocazione all'interno dell'architettura complessiva, senza perdersi nei dettagli fin da subito. Nonostante ciò verranno comunque forniti diagrammi UML delle classi e delle descrizioni dettagliate dei componenti, in modo da avere un quadro completo e chiaro del sistema.

4.1 Contesto di Sistema (C4 Level 1)

Il diagramma di contesto illustra le interazioni di altissimo livello tra l'utente finale e i sistemi esterni. La Piattaforma MVP$_G$ si posiziona al centro, fornendo all'Operatore funzionalità$_G$ di caricamento documenti, validazione$_G$ estrazioni e generazione contenuti tramite browser (via HTTPS e WebSocket). Per l'elaborazione dei dati, il sistema si integra con l'ecosistema cloud di Amazon Web Services, delegando l'estrazione ottica dei caratteri (OCR$_G$) ad AWS$_G$ Textract e l'inferenza di intelligenza artificiale ad AWS$_G$ Bedrock$_G$.

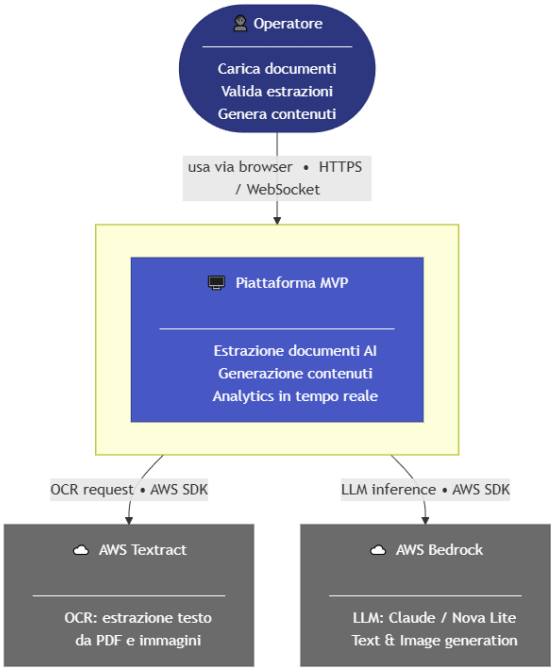


Figura 1: System Context Diagram

4.2 Architettura Applicativa (C4 Level 2 - Container)

Scendendo al livello applicativo, il sistema si divide nei suoi container logici principali. Il frontend$_G$ è costituito da una Single Page Application (SPA) sviluppata in Angular$_G$ 21. Il backend$_G$ è un'API$_G$ sviluppata in Ruby on Rails$_G$ 8, servita tramite Puma. Per garantire fluidità all'utente, le operazioni pesanti (come l'OCR$_G$ e l'invocazione degli LLM$_G$) sono gestite in modo asincrono tramite Solid Queue Workers. Il livello di persistenza è unificato su PostgreSQL$_G$ 16 (Solid Stack), che funge non solo da database primario, ma anche da gestore per le code, cache e backend$_G$ per le connessioni WebSocket (ActionCable).

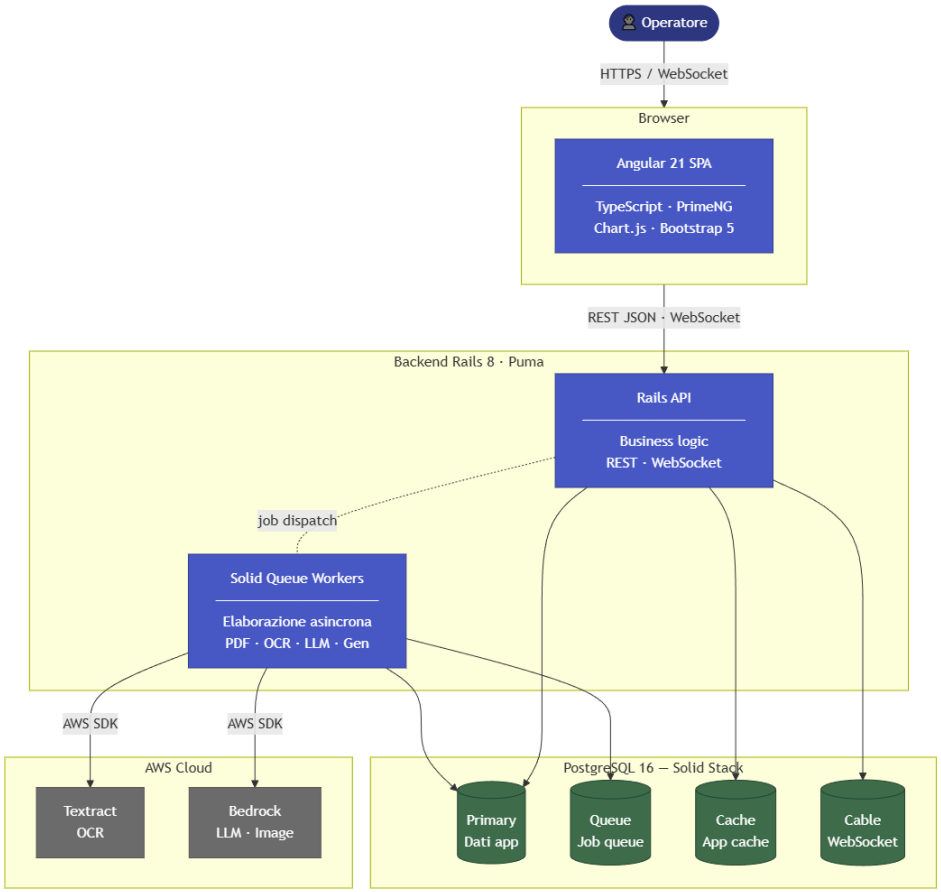


Figura 2: Container Diagram

4.3 Architettura Logica Interna

L'architettura prevede una netta separazione tra il frontend, dedicato all'interfaccia utente, e il backend$_G$, sviluppato in Ruby on Rails$_G$.

4.3.1 Frontend: Interfaccia Utente

Sviluppato in Angular$_G$ (Single Page Application), è il responsabile esclusivo dell'interfaccia$_G$ utente e dell'interazione con l'utente finale.

- Gestisce gli input dell'utente (es. caricamento documenti, richieste di generazione testo).
- Comunica con il backend$_G$ tramite chiamate API$_G$ RESTful per inviare richieste e ricevere risposte.
- Si occupa del routing lato client e della renderizzazione reattiva dei dati.

4.3.2 Backend - Architettura Layered

Il sistema backend è stato progettato seguendo un pattern architetturale di tipo Layered (a livelli). Questa separazione garantisce un alto grado di disaccoppiamento, facilitando la manutenzione, i test$_G$ e la scalabilità$_G$ del sistema. Quest'ultimo sfrutta le convenzioni del framework per implementare in modo naturale i vari livelli interni, senza forzare l'introduzione di pattern complessi e contro il linguaggio stesso.

Il sistema si struttura quindi nei seguenti livelli logici:

4.3.2.1 Backend: Livello Applicativo (Application Layer)

Costituisce il punto di ingresso nel backend$_G$ per le richieste provenienti dal frontend ed è implementato dai **Controller** di Ruby on Rails$_G$.

- Riceve e instrada le richieste HTTP in ingresso.
- Valida i parametri e i payload della richiesta.
- Delega l'effettiva esecuzione delle operazioni ai livelli sottostanti e formatta le risposte (es. in JSON) da restituire al client.

4.3.2.2 Backend: Livello di Business Logic (Business Layer)

Rappresenta il cuore elaborativo del sistema, in cui risiedono le logiche e le regole di dominio.

- Implementato tramite **Service**, **Job** asincroni e classi di orchestrazione.
- Gestisce la logica complessa, come l'integrazione con i modelli di intelligenza artificiale (AWS$_G$ Bedrock$_G$), i servizi di estrazione (AWS$_G$ Textract) e le pipeline di elaborazione dei documenti.
- Coordina il flusso dei dati tra il livello applicativo e il livello di persistenza.

4.3.2.3 Backend: Livello di Persistenza (Persistence Layer)

Gestisce in modo esclusivo la comunicazione con il database PostgreSQL$_G$ e il salvataggio dello stato del sistema.

- Implementato tramite i **Model** di Rails, sfruttando il modulo$_G$ ORM ActiveRecord.
- Rappresenta le entità del dominio (es. Utente, Documento, Sessione) astraendo la complessità del database relazionale.
- Traduce le operazioni sui dati (CRUD) in query SQL sicure e ottimizzate, garantendo l'integrità dei dati e prevenendo vulnerabilità come le SQL Injection.

4.3.2.4 Esempio architettura Layered: Modulo AIGenerator

Il modulo$_G$ di generazione dei contenuti AI$_G$ (AIGenerator) segue rigorosamente i principi dell'architettura Layered. Il controller `AIGeneratorController` si occupa di ricevere le richieste di generazione, validare i parametri e delegare l'esecuzione a un job asincrono (`AIGeneratorJob`), che rappresenta il livello di business logic. Il job coordina i servizi di generazione testuale (AWS$_G$ Bedrock$_G$) e visiva, gestendo la logica di orchestrazione e le eventuali eccezioni. Infine, i risultati vengono salvati tramite i Data Manager e Model, che rappresentano il livello di persistenza, astraggono le operazioni sul database e garantiscono la consistenza dei dati. Come si può vedere ogni livello interagisce solo con quello immediatamente sottostante, mantenendo una chiara separazione delle responsabilità e facilitando la manutenzione e l'estensione futura del sistema. Inoltre, nonostante ci siano lavori asincroni, viene usata una funzionalità nativa di Rails$_G$ *ActiveService::Notifications* per comunicare con il livello application senza dover conoscere e introdurre dipendenze non consone all'architettura, mantenendo dipendenze solo verso lo strato sottostante.

Questo esempio è rappresentativo di come l'architettura Layered sia stata applicata in modo coerente in tutto il backend$_G$ e completamente analogo agli altri moduli.

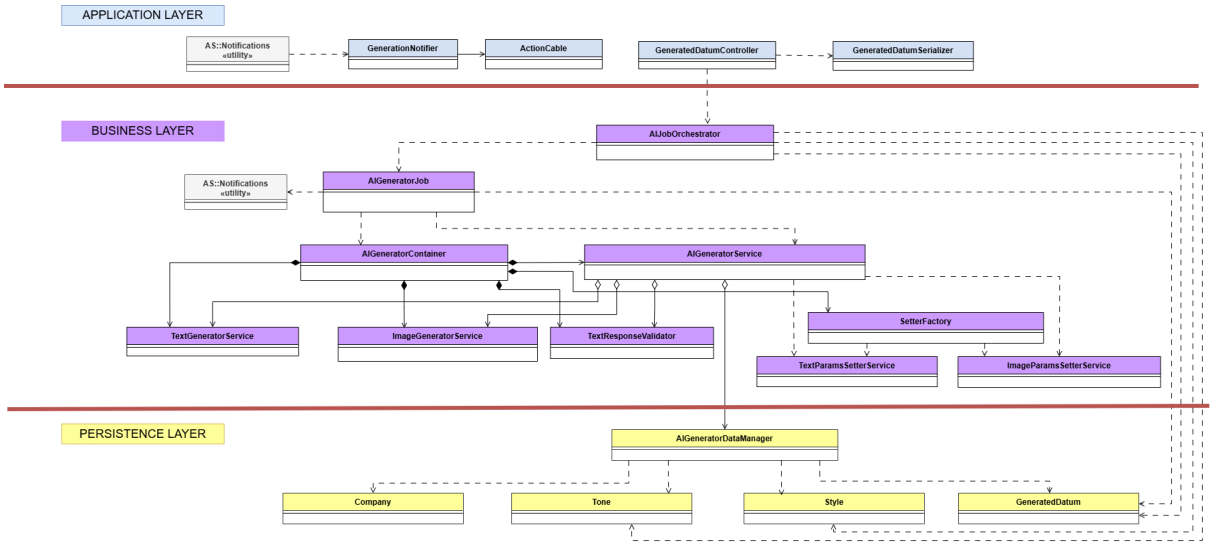


Figura 3: Architettura Layered – Esempio Modulo AIGenerator

4.4 Dettaglio dei Componenti (C4 Level 3)

I principi dell'architettura Layered appena descritti trovano applicazione pratica nell'organizzazione interna dei macro-moduli del backend$_G$ Rails. I vari riquadri indicano i componenti principali (i nomi delle classi di ruby on Rails), evidenziando le interazioni tra di essi. Di seguito si illustra il dettaglio (Livello 3) dei tre domini principali del sistema:

4.4.1 Modulo$_G$ OCR$_G$ ed Estrazione Dati:

Questo modulo$_G$ gestisce il caricamento dei file, lo splitting asincrono dei PDF e l'orchestrazione con AWS$_G$ Textract e Bedrock$_G$. Si nota la netta separazione tra il livello HTTP (Controllers), i Commands per l'accodamento dei job, gli Orchestratori e i Domain Services (Adapter per le API$_G$ esterne e logica di validazione$_G$).

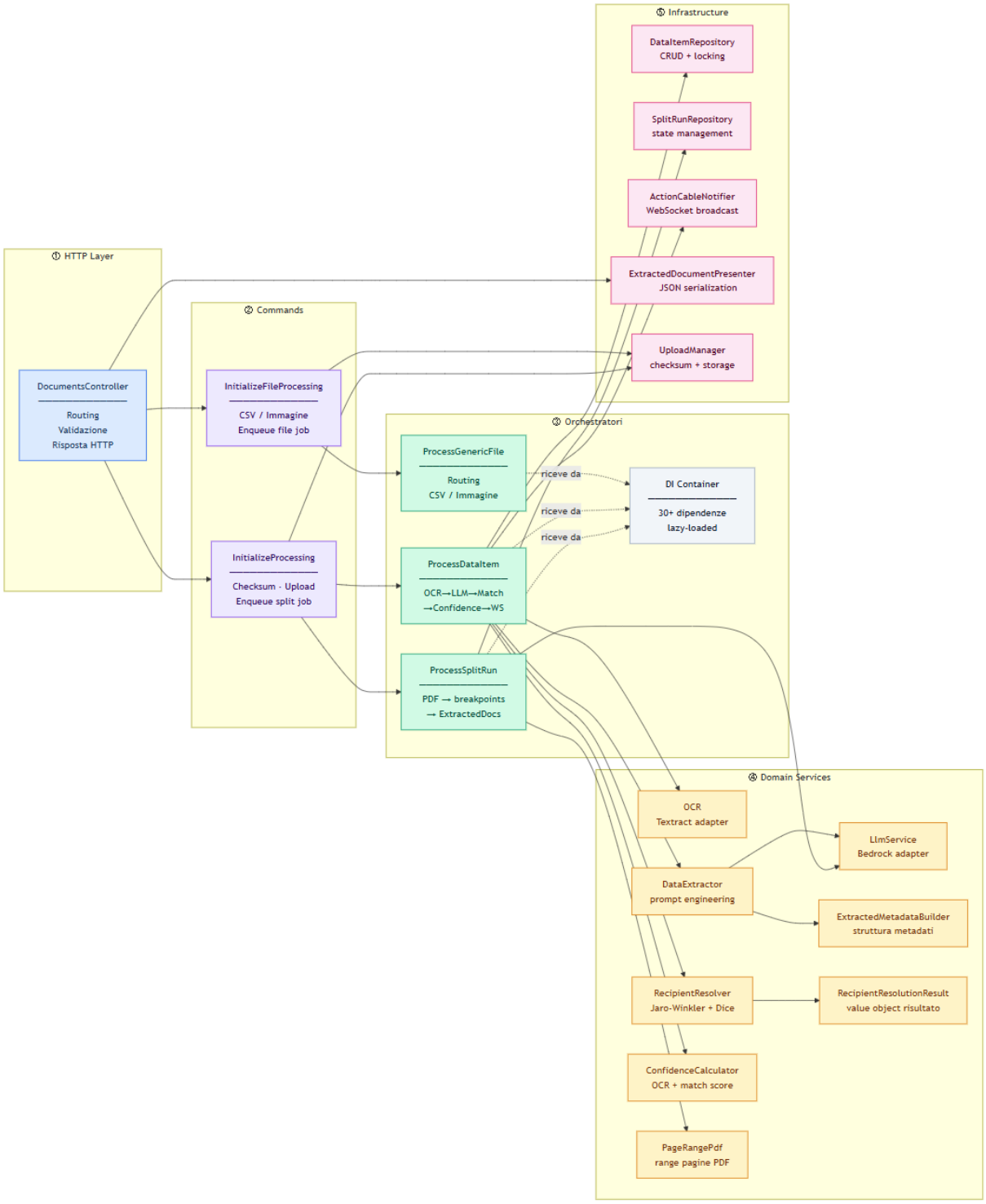


Figura 4: Component Diagram – Document Processing

4.4.2 Modulo$_G$ Generazione Contenuti:

Gestisce la pipeline di generazione multimodale (testo e immagini). Il controller delega immediatamente l'esecuzione a un job asincrono (AIGeneratorJob), il quale coordina i servizi di generazione testuale (Bedrock$_G$) e visiva, salvando infine i record generati tramite i Data Manager.

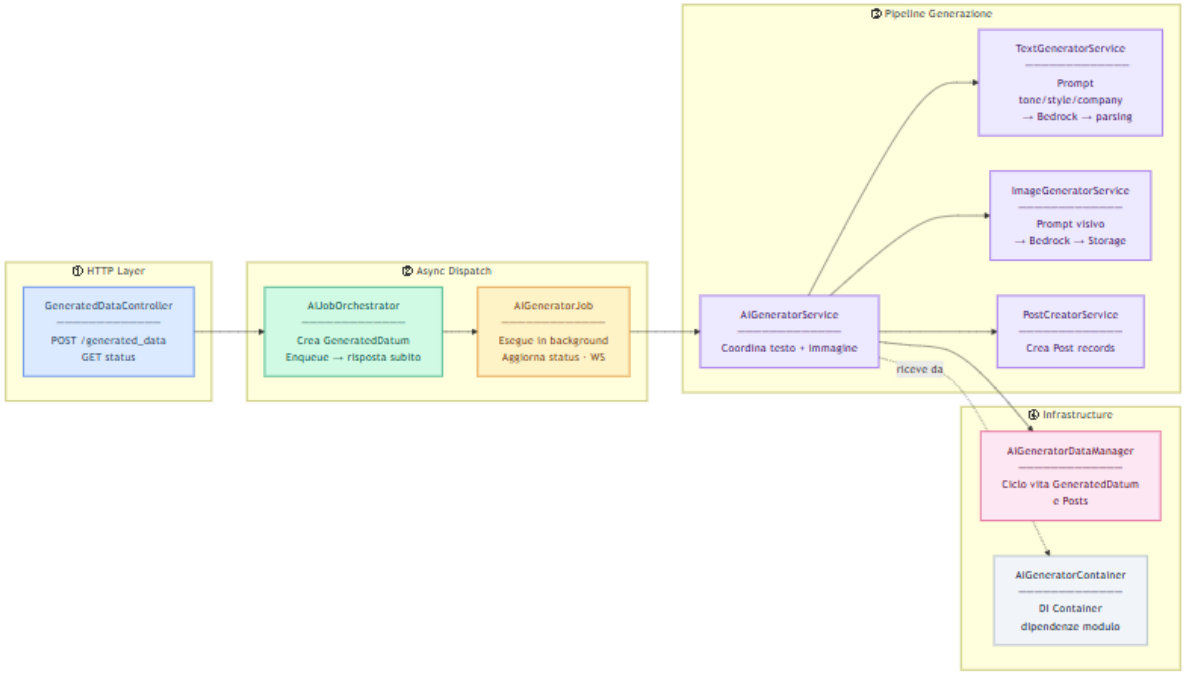


Figura 5: Component Diagram – AI Generation

4.4.3 Modulo$_G$ Analisi e Reportistica (Data Analyst):

Questo componente si occupa dell'aggregazione dei dati per le dashboard$_G$. L'architettura demanda l'esecuzione delle query aggregate (es. calcolo della confidence media o dei tempi di elaborazione) a specifici *Data Managers*.

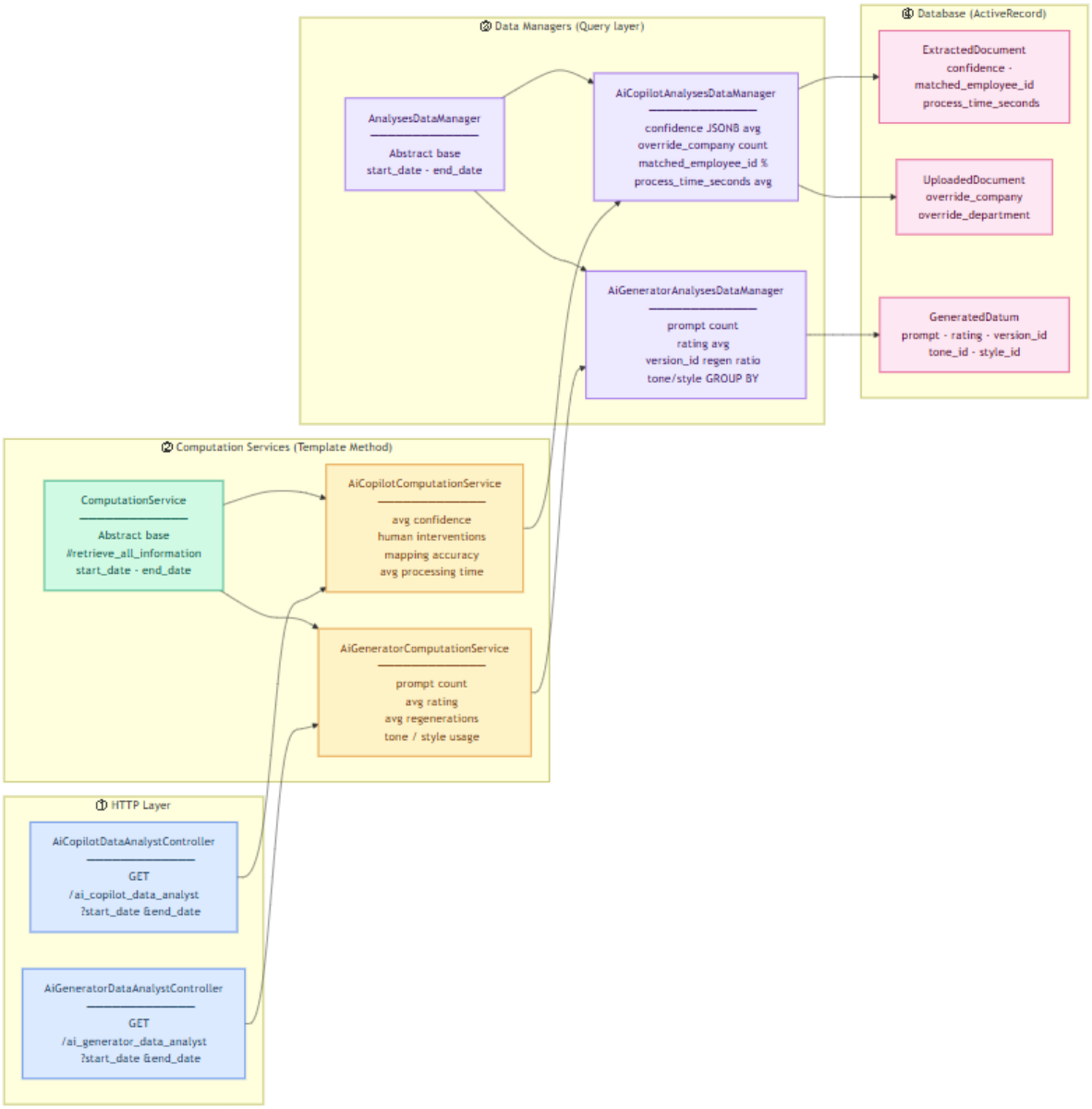


Figura 6: Component Diagram – Data Analysis

4.5 Architettura di Deployment

Dal punto di vista del rilascio, il sistema è composto da un **monolite backend** (Rails) e una **SPA frontend** (Angular), distribuiti come container separati ma orchestrati insieme tramite Docker$_G$ Compose su un singolo nodo ospite.

La scelta del monolite per il backend è motivata dalla **Solid Suite** di Rails 8, una suite di gemme che offre nativamente la capacità di centralizzare code di job (`solid_queue`), cache (`solid_cache`) e WebSocket (`solid_cable`) su PostgreSQL$_G$, eliminando la necessità di infrastrutture esterne come Redis$_G$ o Sidekiq. Nonostante l'uso di alcuni servizi esterni (AWS$_G$ Textract e Bedrock$_G$), l'architettura rimane un monolite logico, con tutti i componenti interni che condividono lo stesso database e processo di esecuzione, in quanto sono delle semplici chiamate, non richiedono un'architettura a microservizi e non introducono complessità aggiuntiva significativa.

4.6 Containerizzazione con Docker$_G$

Per avere un progetto affidabile e riproducibile, l'intero stack è containerizzato con Docker$_G$:

- **Docker Host (Server Node)**: L'infrastruttura hardware o virtuale che ospita il motore Docker$_G$. Al suo interno una Docker$_G$ Network privata permette ai container di comunicare in isolamento senza esporre porte non necessarie all'esterno.
- **Container Frontend$_G$ (SPA)**: Esegue l'applicativo Angular$_G$ 21 come Single Page Application. Serve i file statici al browser; tutta la logica di rendering avviene lato client. Non contiene logica di business.
- **Container Backend$_G$ (Monolite Rails)**: Basato su Ruby$_G$, esegue il server Puma che gestisce in un unico processo: le richieste HTTP REST$_G$, i WebSocket tramite Action Cable (`solid_cable`), la cache (`solid_cache`) e i background job tramite SolidQueue.
- **Container Database (PostgreSQL$_G$ 16)**: Esegue il database relazionale. In sviluppo è presente un singolo database (`backend_development`) che contiene dati applicativi, code di job, cache e stato dei WebSocket. I dati sono persistiti su un Docker$_G$ Volume esterno al container, garantendone la sopravvivenza a riavvii e aggiornamenti.

5 Descrizione architettura (Backend$_G$)

5.1 Modulo$_G$ AI Assistant

Il modulo$_G$ AI Assistant$_G$ è la componente del sistema dedicata alla generazione di contenuti testuali e visivi tramite modelli di intelligenza artificiale. Il suo obiettivo è fornire agli utenti uno strumento potente e intuitivo per creare post personalizzati, arricchendo l'esperienza utente con funzionalità$_G$ avanzate di generazione automatica di contenuti. Il modulo$_G$ si integra con i servizi di AI$_G$ (AWS$_G$ Bedrock$_G$) per estrarre il massimo valore dalle capacità di generazione testuale e visiva, offrendo un'interfaccia$_G$ semplice per la creazione di contenuti complessi e personalizzati, supportando al contempo la gestione e la valutazione dei risultati generati. Nei diagrammi seguenti sono rappresentati i componenti del modulo$_G$, divisi per controller, evidenziando i componenti principali e le loro interazioni.

Il controller più importante del modulo$_G$ è *AIGeneratorController*, che espone gli endpoint per la creazione e gestione dei contenuti generati. Il controller delega la logica di business a un job asincrono *(AIGeneratorJob)*, che si occupa di orchestrare le chiamate ai servizi di generazione testuale e visiva, gestire la persistenza dei risultati e aggiornare lo stato delle generazioni in corso.

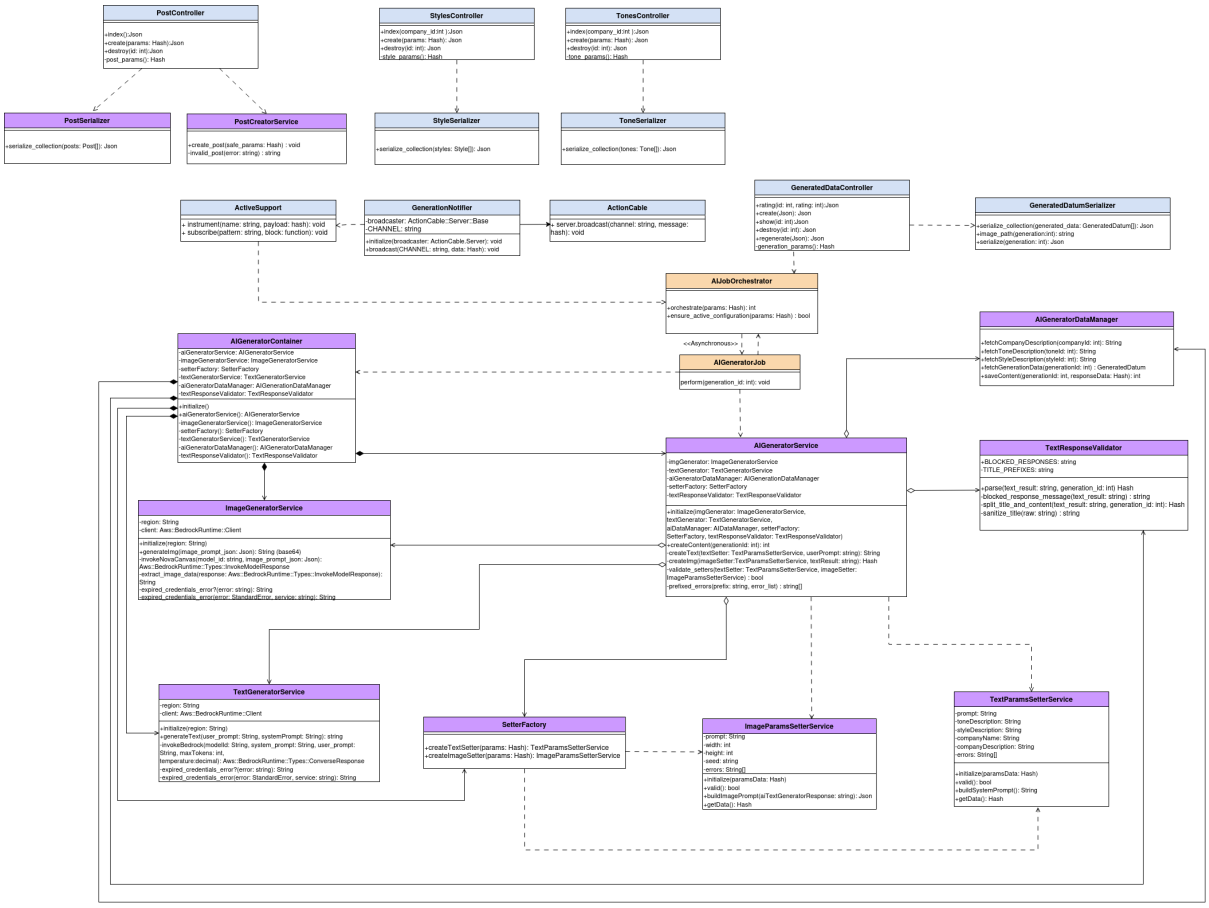


Figura 7: Diagramma delle classi – Modulo$_G$ AI Assistant$_G$

5.1.1 Diagrammi delle classi del modulo$_G$ AI Assistant$_G$

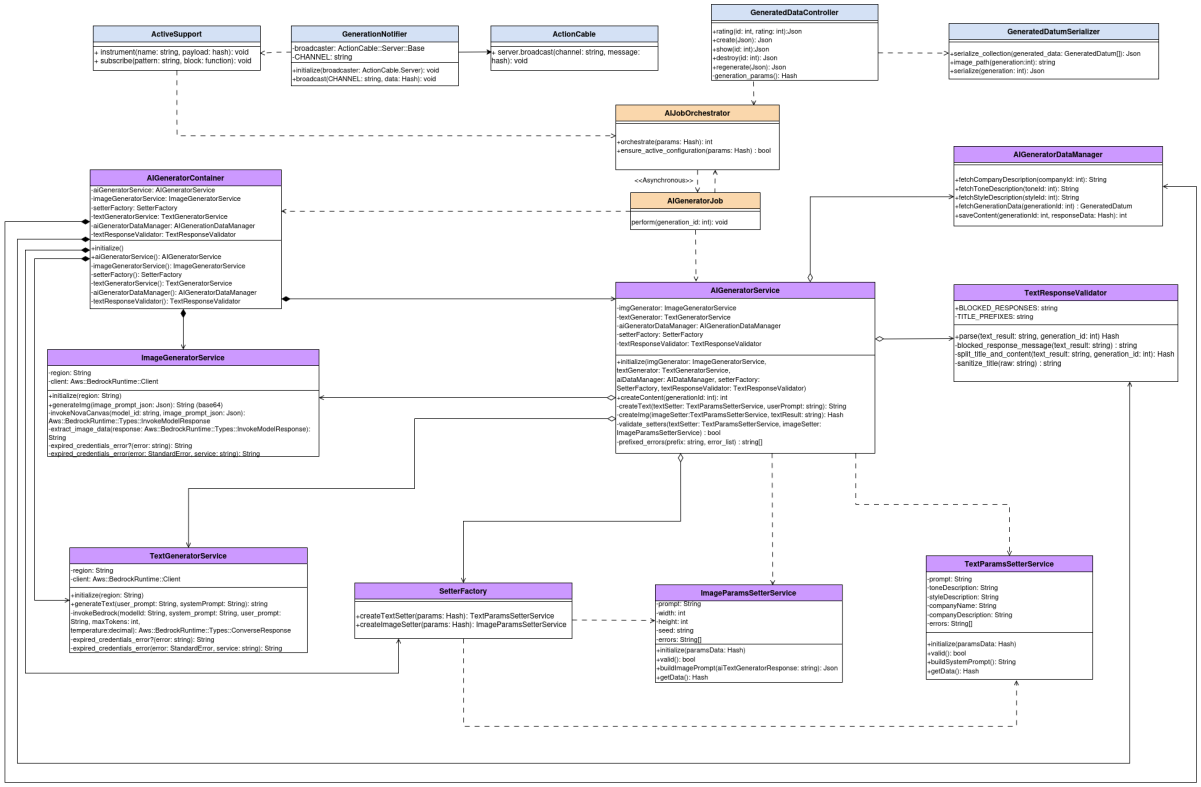


Figura 8: Diagramma classi parziale – AIGeneratorController

Il diagramma mostra il controller di ingresso del modulo$_G$: riceve le richieste HTTP, valida i parametri e delega l'esecuzione al job asincrono. Tramite l'utilizzo di un container di dipendenza, il controller inietta i servizi necessari per la generazione, mantenendo un alto grado di disaccoppiamento e testabilità. I vari services gestiscono la logica di interazione con i modelli di intelligenza artificiale, la persistenza dei dati e l'aggiornamento dello stato delle generazioni.

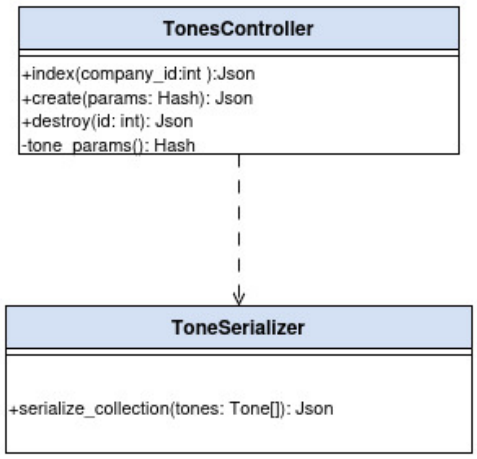


Figura 9: Diagramma classi parziale – TonesController

Il diagramma mostra il controller dedicato alla gestione dei toni di generazione. Espone endpoint per la creazione, eliminazione e recupero dei toni disponibili, delegando la logica di business al

servizio specifico che si occupa della persistenza e della validazione$_G$ dei dati: *ToneSerializer*.

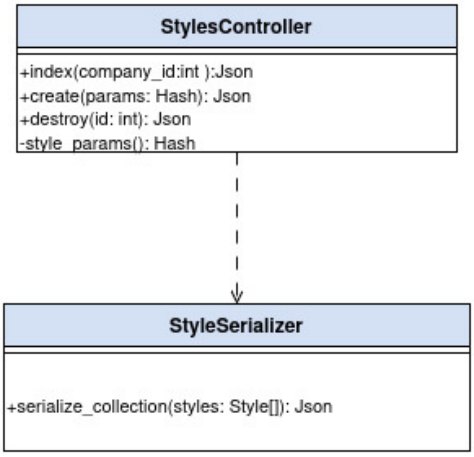


Figura 10: Diagramma classi parziale – StylesController

Il diagramma mostra il controller dedicato alla gestione dei stili di generazione. Il funzionamento è analogo a quello del controller dei toni, delegando la logica di business al servizio specifico *StyleSerializer*.

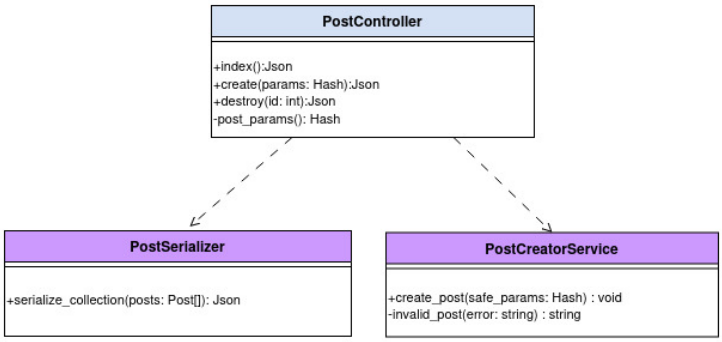


Figura 11: Diagramma classi parziale – PostsController

Il diagramma mostra il controller dedicato alla pubblicazione di post. Anche qui il funzionamento di base è analogo a quello del controller dei toni, fatto utilizzando *PostsSerializer*, ma utilizza anche *PostCreatorService* per gestire la logica di creazione dei post, con la sua logica specifica di validazione$_G$.

5.1.2 Diagramma di sequenza di generazione contenuti

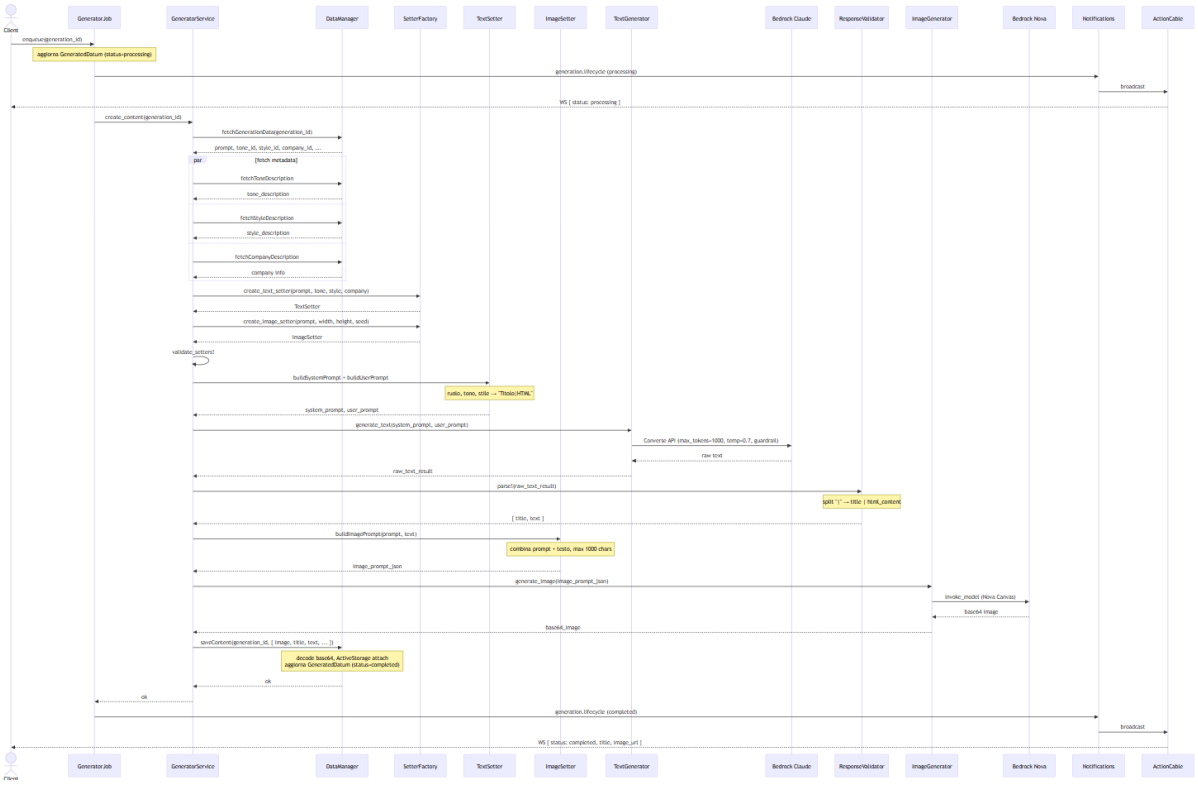
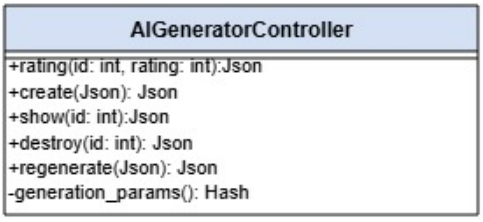


Figura 12: Diagramma di sequenza – Flusso di generazione contenuti

Il diagramma descrive la sequenza delle operazioni che avvengono quando un utente richiede la generazione di un contenuto tramite l'endpoint dedicato. Il controller riceve la richiesta, valida i parametri e delega l'esecuzione a un job asincrono. Il job, una volta avviato, coordina le chiamate ai servizi di generazione testuale e visiva, gestisce la persistenza dei risultati e aggiorna lo stato delle generazioni in corso.

5.1.2.1 AIGeneratorController



Classe di tipo Controller che espone gli endpoint per la generazione dei contenuti nel modulo$_G$ AI Assistant$_G$. Gestisce le operazioni CRUD sulle generazioni, espone un endpoint per rigenerare i contenuti e un endpoint per valutare i risultati generati.

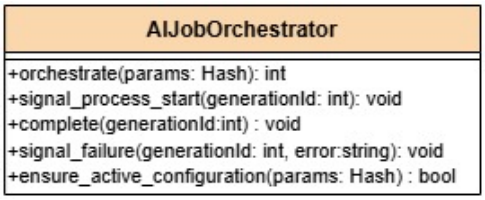
Attributi

- Nessuno

Metodi

- create(params: Hash): Json
- regenerate(id: int): Json
- show(id: int): Json
- rating(id: int, rating: int): Json
- destroy(id: int): Json
- generation_params(): Hash

5.1.2.2 AIJobOrchestrator



Classe orchestratrice che gestisce la logica di esecuzione dei job di generazione dei contenuti. Si occupa di coordinare le chiamate ai servizi di generazione testuale e visiva, gestire la persistenza dei risultati e aggiornare lo stato delle generazioni in corso.

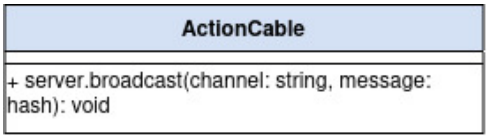
Attributi

- Nessuno

Metodi

- orchestrate(params: Hash): int
- signal_process_started(generationId: int): void
- signal_failure(generation_id: int, error: string): void
- complete(generationId: int): void
- ensure_active_configuration(params: Hash): bool

5.1.2.3 ActionCable



Classe nativa di rails per la gestione delle connessioni WebSocket. Nel contesto del modulo$_G$ AI Assistant$_G$, viene utilizzata per inviare aggiornamenti in tempo reale al frontend$_G$ riguardo lo stato delle generazioni in corso.

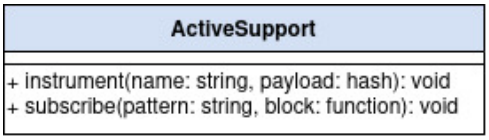
Attributi

- Nessuno

Metodi

- server.broadcast(channel: string, message: Json): void

5.1.2.4 ActiveSupport



Classe nativa di rails per la gestione delle connessioni WebSocket. Nel contesto del modulo$_G$ AI Assistant$_G$, viene utilizzata insieme ad altri componenti per gestire le comunicazioni in tempo reale. Recupera le notifiche inviate da AiGeneratorJob e le distribuisce ai client sottoscritti, facilitando l'aggiornamento dinamico dell'interfaccia$_G$ utente.

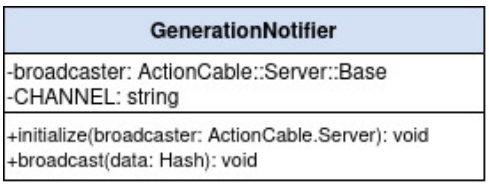
Attributi

- Nessuno

Metodi

- instrument(name: string, payload: hash): void
- subscribe(pattern: string, block: function): void

5.1.2.5 GenerationNotifier



Classe di servizio dedicata alla gestione delle notifiche in tempo reale riguardo lo stato delle generazioni in corso. Utilizza ActionCable e ActiveSupport per inviare aggiornamenti al frontend$_G$ e permettere un'esperienza utente più fluida e interattiva durante il processo di generazione dei contenuti.

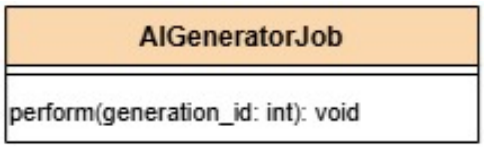
Attributi

- CHANNEL: string
- broadcaster: ActionCable::Server

Metodi

- initialize(broadcaster: ActionCable::Server): void
- subscribe(data: Hash): void

5.1.2.6 AIGeneratorJob



Classe di tipo Job asincrono che si occupa di eseguire la logica di generazione dei contenuti in background. Viene invocata dal *AIJobOrchestrator* quando viene richiesta la creazione o rigenerazione di un contenuto, e si occupa di orchestrare le chiamate ai servizi di generazione testuale e visiva.

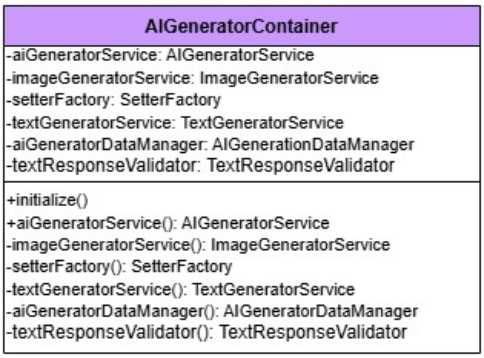
Attributi

- Nessuno

Metodi

- perform(generationId: int): void

5.1.2.7 AIGeneratorContainer



Container che si occupa di gestire le dipendenze necessarie per l'esecuzione dei job di generazione dei contenuti. Viene utilizzato per iniettare i servizi necessari all'esecuzione dei job all'interno di *AIGeneratorService*, mantenendo un alto grado di disaccoppiamento e testabilità, ritornando alla chiamata del job i servizi necessari per l'esecuzione della logica di generazione.

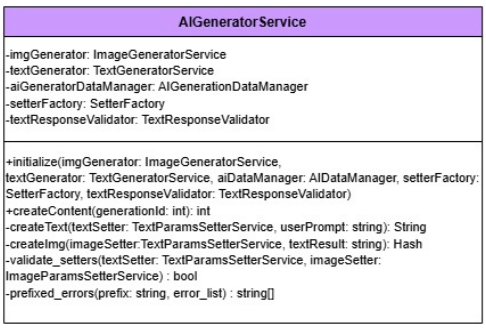
Attributi

- aiGeneratorService
- imageGeneratorService
- setterFactory
- textGeneratorService
- aiGeneratorDataManager
- textResponseValidator

Metodi

- initialize(): void
- aiGeneratorService(): AIGeneratorService
- imageGeneratorService(): ImageGeneratorService
- setterFactory(): SetterFactory
- textGeneratorService(): TextGeneratorService
- aiGeneratorDataManager(): AIGeneratorDataManager
- textResponseValidator(): TextResponseValidator

5.1.2.8 AIGeneratorService



Classe orchestratrice che gestisce la logica di esecuzione dei job di generazione dei contenuti. Si occupa di coordinare le chiamate ai servizi di generazione testuale e visiva, gestire la persistenza dei risultati e aggiornare lo stato delle generazioni in corso.

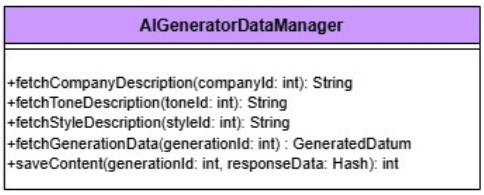
Attributi

- imgGenerator
- textGenerator
- aiGeneratorDataManager
- textResponseValidator
- setterFactory

Metodi

- initialize(imgGenerator: ImageGeneratorService, textGenerator: TextGeneratorService, aiGeneratorDataManager: AIGeneratorDataManager, textResponseValidator: TextResponseValidator, setterFactory: SetterFactory): void
- createContent(generationId: int): int
- createText(textSetter: TextParamsSetterService, userPrompt: string): string
- createImage(imageSetter: ImageParamsSetterService, textResult: string): Hash
- validateSetters(textSetter: TextParamsSetterService, imageSetter: ImageParamsSetterService): bool
- prefixed_errors(prefix: string, errors: string[]): string[]

5.1.2.9 AIGeneratorDataManager



Classe di tipo Data Manager che si occupa della gestione dei dati relativi alle generazioni dei contenuti. Si occupa di salvare i risultati generati, aggiornare lo stato delle generazioni in corso e fornire un'interfaccia$_G$ per recuperare i dati delle generazioni quando necessario.

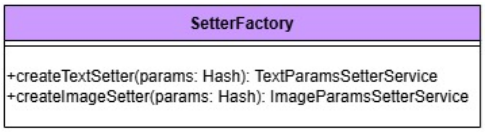
Attributi

- Nessuno

Metodi

- fetchCompanyDescription(companyId: int): string
- fetchToneDescription(toneId: int): string
- fetchStyleDescription(styleId: int): string
- fetchGenerationData(generationId: int): GeneratedDatum
- saveContent(generationId: int, responseData: Hash): int

5.1.2.10 SetterFactory



Classe di tipo Factory che si occupa di creare i setter per i parametri di generazione dei contenuti. Viene utilizzata per creare i setter specifici per i parametri di testo e immagine, in base alle esigenze della generazione richiesta.

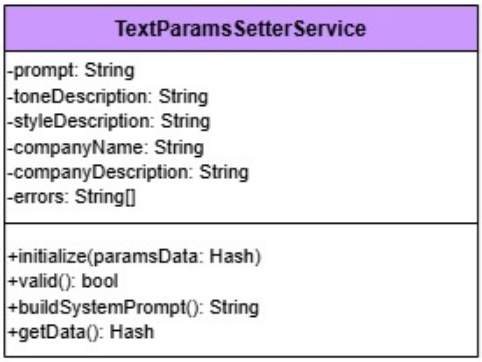
Attributi

- Nessuno

Metodi

- createTextSetter(params: Hash): TextParamsSetterService
- createImageSetter(params: Hash): ImageParamsSetterService

5.1.2.11 TextParamsSetterService



Classe Service che si occupa di impostare i parametri di generazione per i contenuti testuali. Viene utilizzata per impostare i parametri specifici per la generazione di testo,come il tono, lo stile e la descrizione dell'azienda, in base alle esigenze della generazione richiesta.

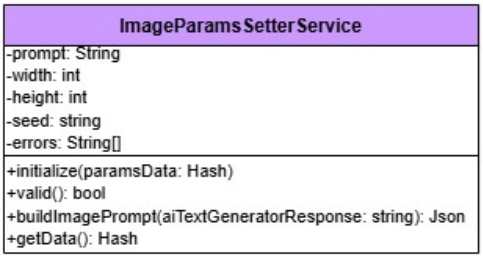
Attributi

- prompt: string
- toneDescription: string
- styleDescription: string
- companyName: string
- companyDescription: string
- errors: string[]

Metodi

- initialize(params: Hash): void
- valid(): bool
- buildSystemPrompt(): string
- getData(): Hash

5.1.2.12 ImageParamsSetterService



Classe Service che si occupa di impostare i parametri di generazione per i contenuti immagine. Viene utilizzata per impostare i parametri specifici per la generazione di immagini, in base alle

esigenze della generazione richiesta, utilizzando anche la risposta della generazione testuale per ulteriori dettagli.

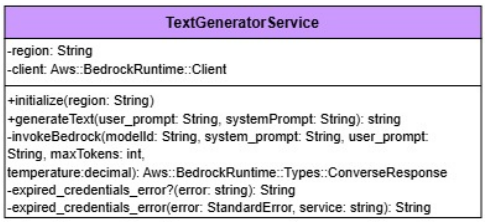
Attributi

- prompt: string
- width: int
- height: int
- seed: string
- errors: string[]

Metodi

- initialize(params: Hash): void
- valid(): bool
- buildImagePrompt(aiTextGeneratorResponse: string): Json
- getData(): Hash

5.1.2.13 TextGeneratorService



Classe service che si occupa di gestire la logica di generazione dei contenuti testuali. Si occupa di interagire con il servizio di generazione testuale (AWS$_G$ Bedrock$_G$) per generare i contenuti richiesti, utilizzando i parametri impostati dal *TextParamsSetterService*.

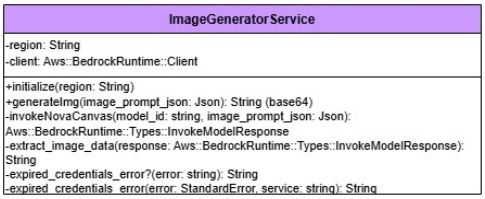
Attributi

- region: string
- client: Aws::BedrockRuntime::Client

Metodi

- initialize(region: string): void
- generateText(user_prompt: string, system_prompt: string): string
- invokeBedrock(modelId: string, user_prompt: string, system_prompt: string, max_tokens: int, temperature: float): AWS::BedrockRuntime::Types::ConverseResponse
- expired_credentials_error?(error: string): string
- expired_credentials_error(error: StandardError, service: Symbol): string

5.1.2.14 ImageGeneratorService



Classe service che si occupa di gestire la logica di generazione dei contenuti immagine. Si occupa di interagire con il servizio di generazione immagini (AWS$_G$ Bedrock$_G$) per generare i contenuti richiesti, utilizzando i parametri impostati dal *ImageParamsSetterService*.

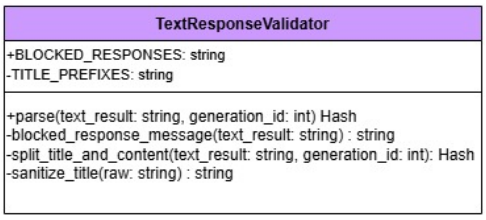
Attributi

- region: string
- client: Aws::BedrockRuntime::Client

Metodi

- initialize(region: string): void
- generateImg(user_prompt: string, system_prompt: string): string
- invokeNovaCanvas(modelId: string, image_prompt_json: Json): AWS::BedrockRuntime::Types::InvokeMode
- extract_image_data(response: AWS::BedrockRuntime::Types::InvokeModelResponse): String
- expired_credentials_error?(error: string): string
- expired_credentials_error(error: StandardError, service: Symbol): string

5.1.2.15 TextResponseValidator



Classe Service che si occupa di validare la risposta della generazione testuale. Si occupa di verificare che la risposta della generazione testuale sia valida e che lo split del titolo dal contenuto sia avvenuto correttamente, restituendo eventuali errori riscontrati durante la validazione$_G$.

Attributi

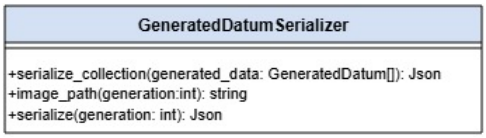
- BLOCKED_RESPONSES: string[]
- TITLE_PREFIXES: string[]

Metodi

- parse(text_result: string, generationId: intx): Hash

- blocked_response_message(text_result: string): string
- split_title_and_content(text_result: string, generationId: int): Hash
- sanitize_title(raw: string): string

5.1.2.16 GeneratedDatumSerializer



Classe serializer che si occupa di serializzare i dati delle generazioni dei contenuti. Viene utilizzata per serializzare i dati delle generazioni in un formato adatto per essere inviato al frontend$_G$, gestendo eventuali trasformazioni necessarie per la corretta rappresentazione dei dati.

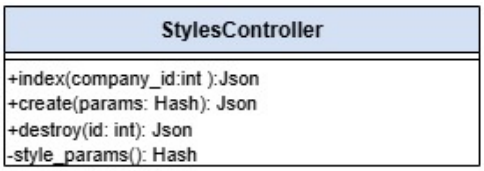
Attributi

- Nessuno

Metodi

- serialize_collection(generated_data: GeneratedDatum[]): Json
- image_path(generation:int): string
- serialize(generation: int): Json

5.1.2.17 StylesController



Controller dedicato alla gestione degli stili di generazione dei contenuti. Espone endpoint per la creazione, eliminazione e recupero degli stili disponibili.

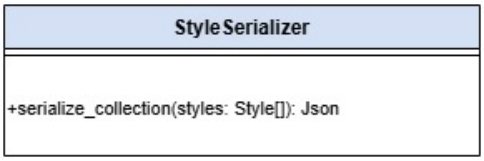
Attributi

- Nessuno

Metodi

- index(): Json
- create(params: Hash): Json
- destroy(id: int): Json
- style_params(): Hash

5.1.2.18 StylesSerializer



Classe serializer che si occupa di serializzare i dati degli stili di generazione dei contenuti. Viene utilizzata per serializzare i dati degli stili in un formato adatto per essere inviato al frontend$_G$, gestendo eventuali trasformazioni necessarie per la corretta rappresentazione dei dati.

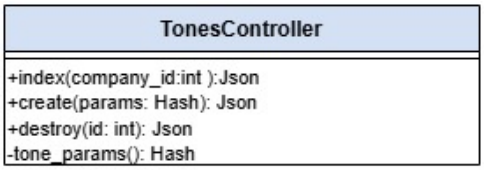
Attributi

- Nessuno

Metodi

- serialize_collection(styles: Style[]): Json

5.1.2.19 TonesController



Controller dedicato alla gestione dei toni di generazione dei contenuti. Espone endpoint per la creazione, eliminazione e recupero dei toni disponibili.

Attributi

- Nessuno

Metodi

- index(): Json
- create(params: Hash): Json
- destroy(id: int): Json
- tone_params(): Hash

5.1.2.20 TonesSerializer

Classe serializer che si occupa di serializzare i dati dei toni di generazione dei contenuti. Viene utilizzata per serializzare i dati dei toni in un formato adatto per essere inviato al frontend$_G$, gestendo eventuali trasformazioni necessarie per la corretta rappresentazione dei dati.

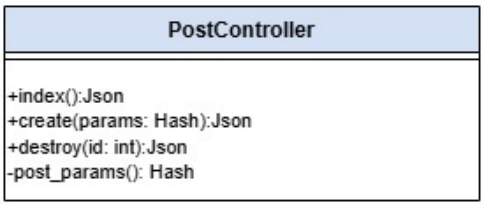
Attributi

- Nessuno

Metodi

- serialize_collection(tones: Tone[]): Json

5.1.2.21 PostsController



Controller dedicato alla gestione dei post generati. Espone endpoint per la creazione, eliminazione e recupero dei post disponibili, delegando la logica di business alla combinazione di *PostsSerializer* per la serializzazione dei dati e *PostCreatorService* per la logica di creazione dei post, con la sua logica specifica di validazione$_G$.

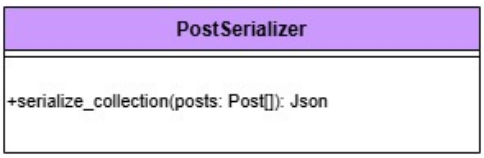
Attributi

- Nessuno

Metodi

- index(): Json
- create(params: Hash): Json
- destroy(id: int): Json
- post_params(): Hash

5.1.2.22 PostsSerializer



Classe serializer che si occupa di serializzare i dati dei post generati. Viene utilizzata per serializzare i dati dei post in un formato adatto per essere inviato al frontend$_G$, gestendo eventuali trasformazioni necessarie per la corretta rappresentazione dei dati.

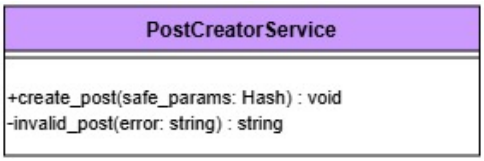
Attributi

- Nessuno

Metodi

- serialize_collection(posts: Post[]): Json

5.1.2.23 PostCreatorService



Classe service che si occupa della creazione dei post. Viene utilizzata per gestire la logica di creazione dei post, inclusa la validazione$_G$ e la persistenza dei dati. Il servizio si assicura che i dati forniti siano validi e completi prima di creare un nuovo post, restituendo eventuali errori riscontrati durante la validazione$_G$.

Attributi

- Nessuno

Metodi

- create_post(params: Hash): Post
- invalid_post(error: string): string

5.2 Modulo$_G$ AI CoPilot

Il modulo$_G$ AI CoPilot$_G$ è la componente del sistema dedicata al supporto dei Consulenti del Lavoro (CdL)$_G$ nella gestione e revisione documentale automatizzata. L'obiettivo principale è l'efficientamento del workflow$_G$ operativo attraverso l'impiego di modelli di intelligenza artificiale per l'estrazione di metadati e il riconoscimento anagrafico.

I diagrammi riportati di seguito rappresentano diagrammi delle classi parziali del blocco OCR$_G$, utili a mostrare i componenti principali senza appesantire la lettura con l'intero grafo delle dipendenze.

Il primo diagramma mostra *AIDocumentsController*, ossia il controller che gestisce tutto il flusso legato allo split, all'analisi documentale e alle eventuali modifiche manuali (ad esempio validazioni e riassegnazioni). Il controller delega la logica ai servizi applicativi e conclude l'orchestrazione con l'avvio dei job, che innescano asincronamente i flussi rappresentati dagli altri tre diagrammi delle classi parziali. Tutte le dipendenze sono iniettate tramite il container di sistema per massimizzare il disaccoppiamento; il dettaglio completo dell'iniezione non è rappresentato nei diagrammi per ragioni di comodità e chiarezza, ma rimane evidente dall'uso costante di componenti iniettati.

5.2.1 Diagrammi classi parziali del blocco OCR$_G$

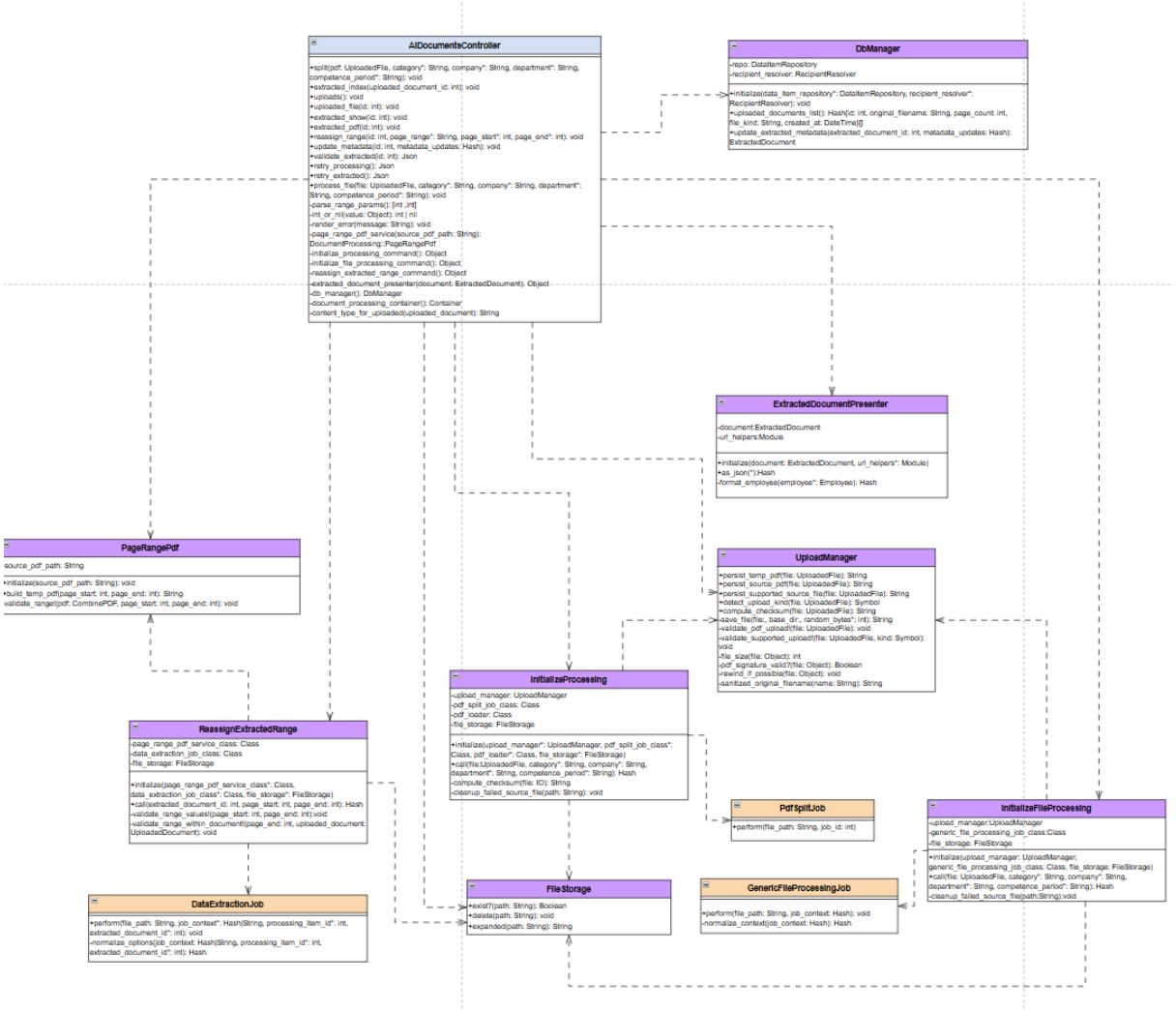


Figura 13: Diagramma classi parziale – Document Controller

Il diagramma mostra il controller di ingresso del modulo$_G$ OCR$_G$: riceve le richieste HTTP, valida i parametri e delega l'esecuzione ai servizi applicativi, fino all'avvio dei job asincroni.

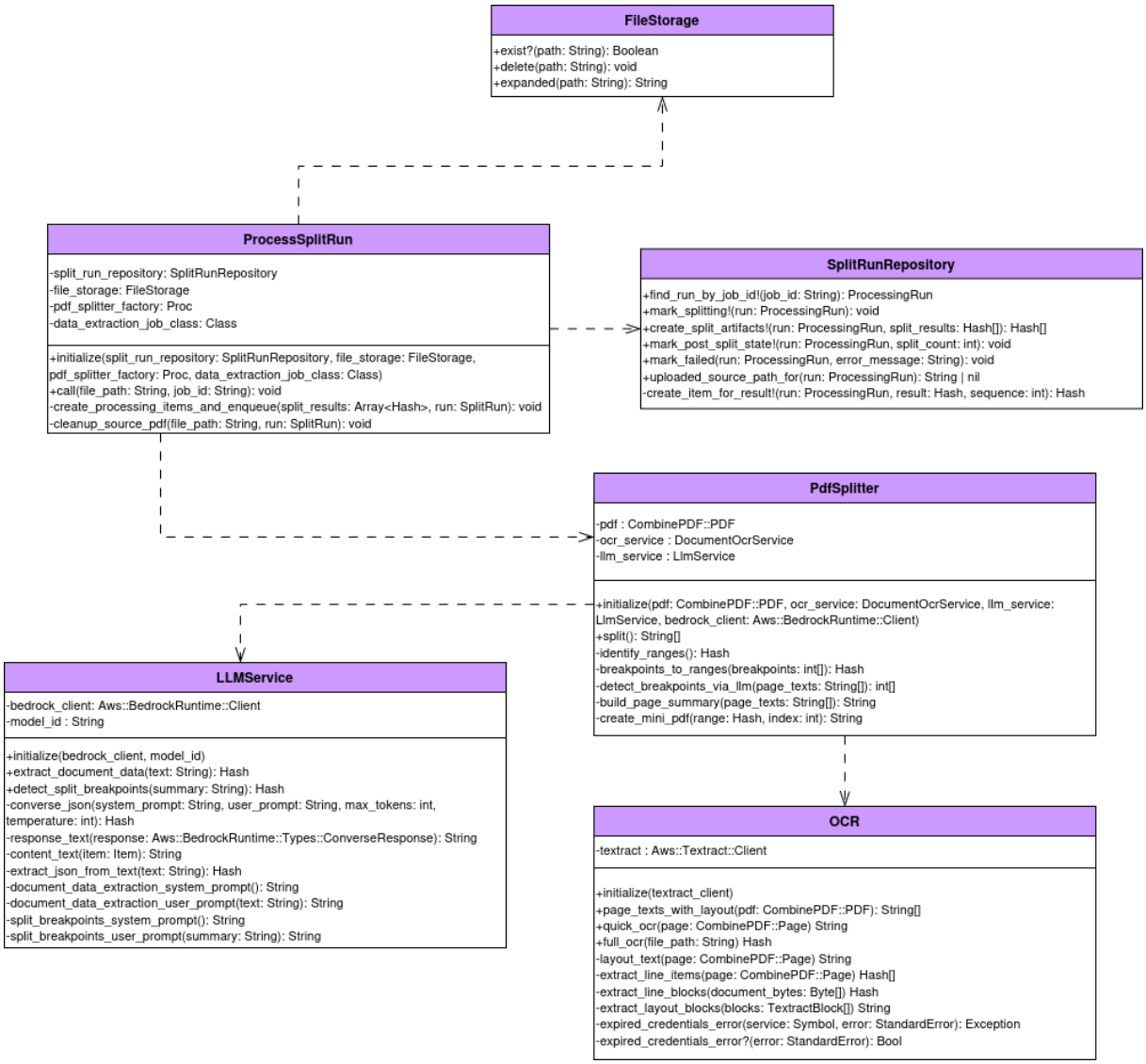


Figura 14: Diagramma classi parziale – Flusso di split PDF

Questo diagramma rappresenta il flusso asincrono di splitting dei PDF, dalla preparazione del file alla suddivisione in item di elaborazione e al dispatch$_G$ delle estrazioni.

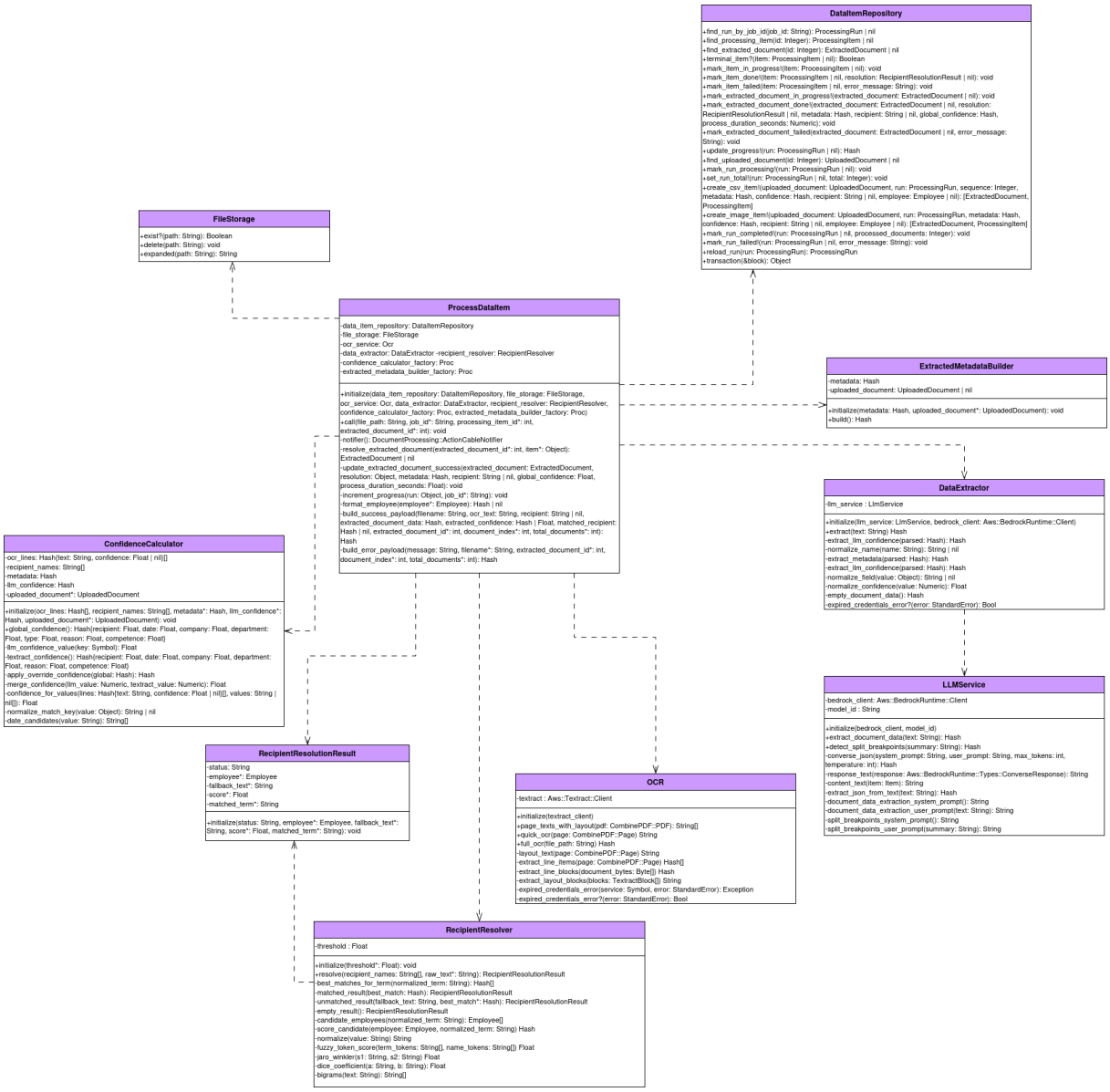


Figura 15: Diagramma classi parziale – Flusso di estrazione da PDF

Il diagramma evidenzia la catena di elaborazione dei documenti PDF estratti, con orchestrazione dei servizi OCR/LLM, persistenza dei metadati e aggiornamento dello stato dei processing item.

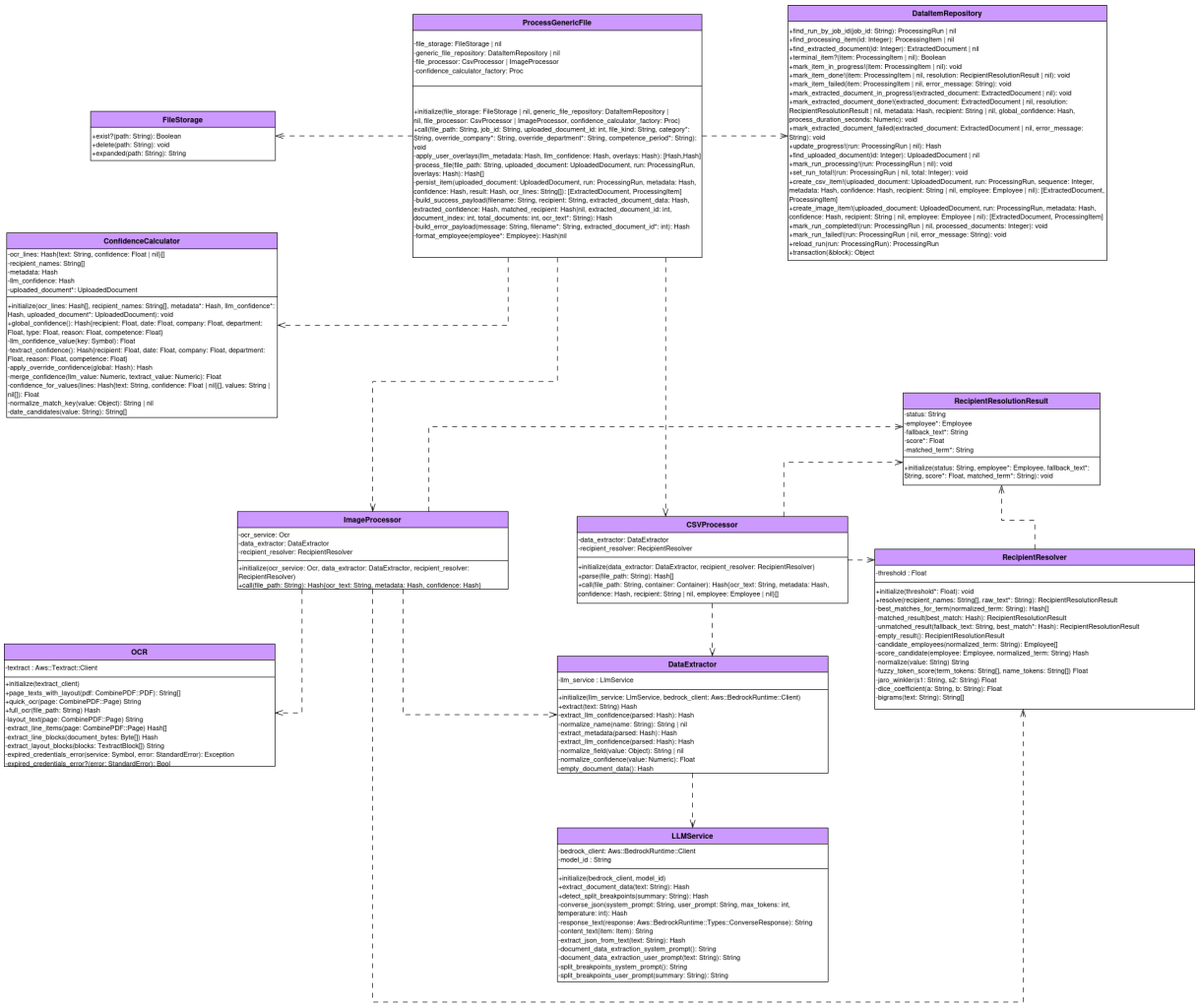


Figura 16: Diagramma classi parziale – Flusso file generici

Questo diagramma mostra il percorso dedicato ai file non PDF (ad esempio immagini o CSV), con inizializzazione specifica, job asincrono dedicato e normalizzazione del risultato.

5.2.2 Diagrammi di sequenza del blocco OCR$_G$

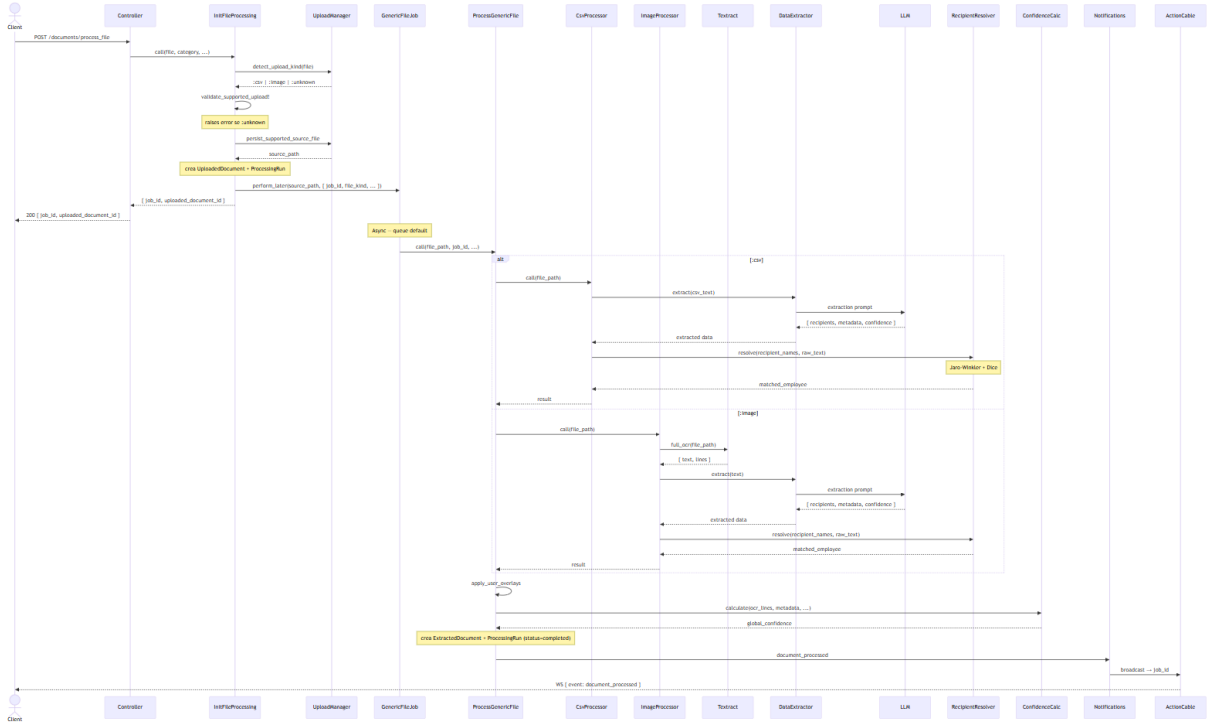


Figura 17: Diagramma di sequenza – Flusso Generic File

Il diagramma descrive la sequenza operativa per i file generici, attualmente immagini e csv: richiesta al controller, inizializzazione del processing dedicato, accodamento job e aggiornamento asincrono dello stato.

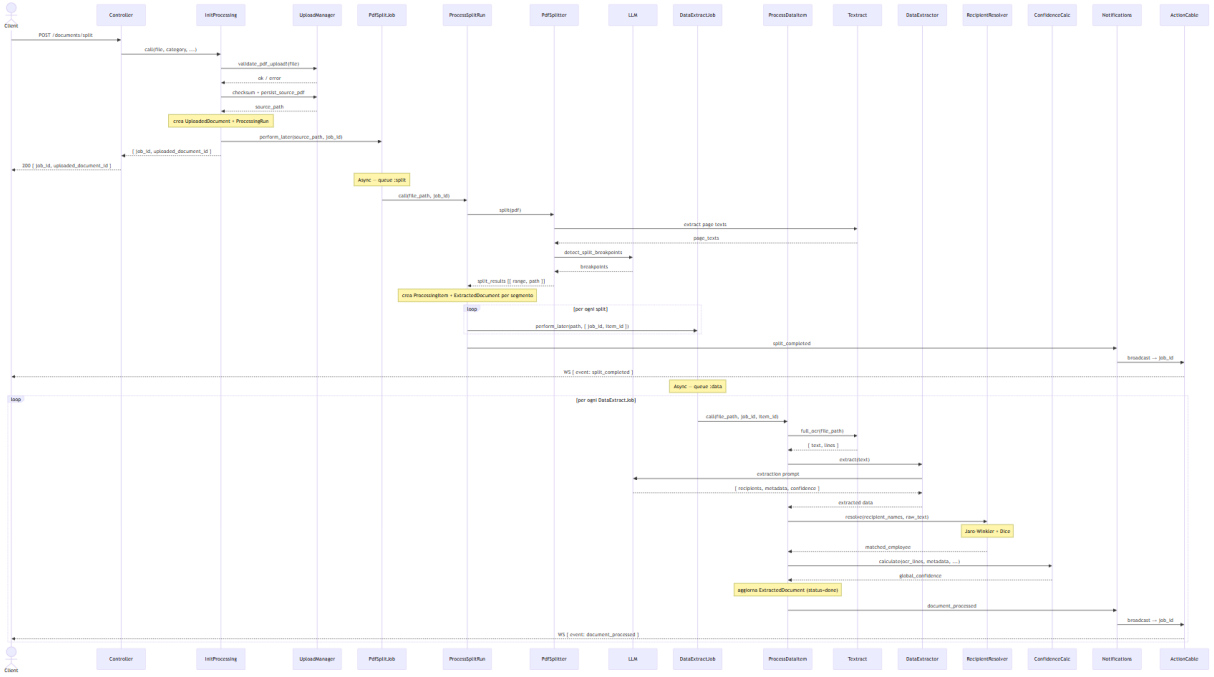


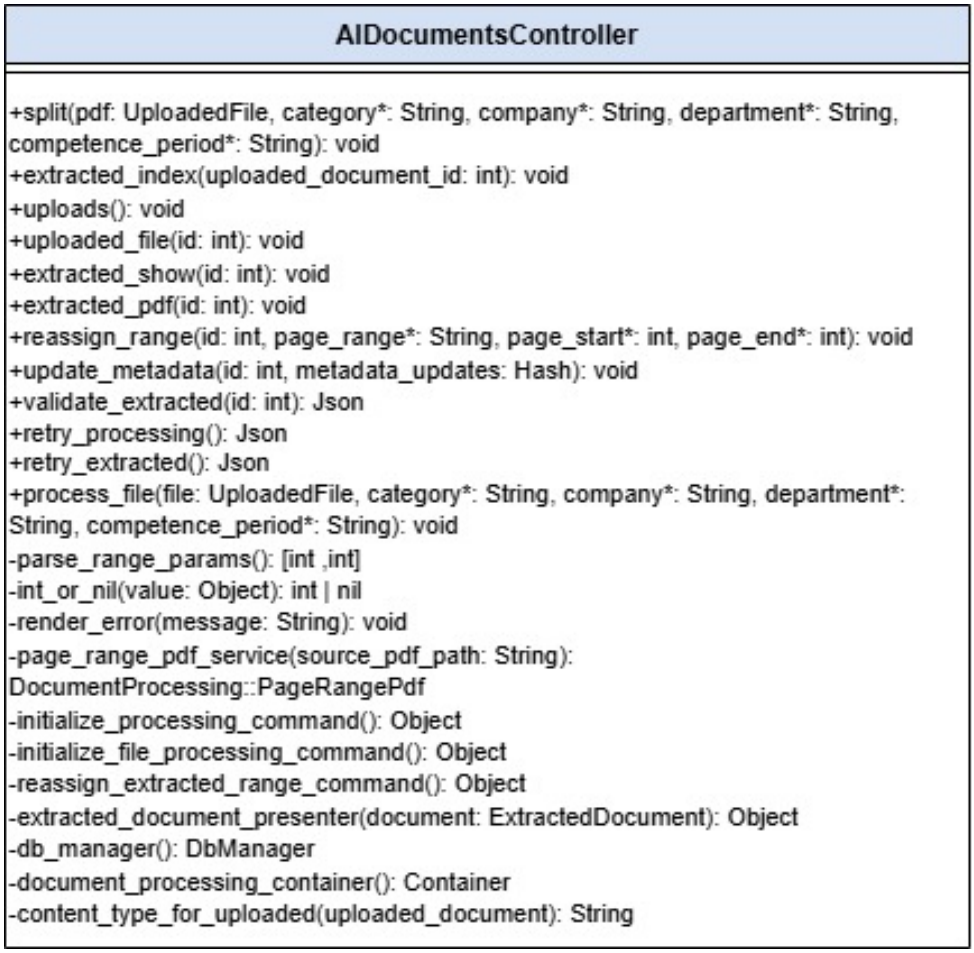
Figura 18: Diagramma di sequenza – Flusso PDF/OCR

Il diagramma mostra il flusso PDF con OCR$_G$: upload e split, orchestrazione dei servizi di estrazione, invocazione dei job asincroni e persistenza progressiva dei risultati estratti. Il flusso è separato dal percorso dei file generici, evidenziando la specificità del processo di splitting e dell'estrazione basata su pagine.

5.2.3 Tutte le classi del blocco OCR$_G$

Dopo aver mostrato i diagrammi parziali per i componenti principali, si riportano di seguito le descrizioni di tutte le classi coinvolte nel blocco OCR$_G$, con i dettagli di attributi e metodi per ciascuna classe.

5.2.3.1 AIDocumentsController



Classe di tipo Controller che espone gli endpoint per la gestione del ciclo di vita dei documenti nel modulo$_G$ AI CoPilot$_G$. Coordina le richieste HTTP provenienti dal client per operazioni quali il caricamento dei file, l'avvio del processo di splitting, la validazione$_G$ dei dati estratti e la riassegnazione manuale dei range di pagine.

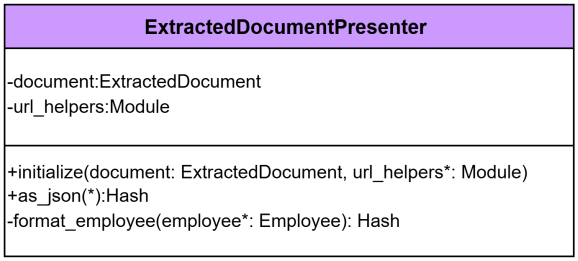
Attributi

- Nessuno

Metodi

- split(pdf: UploadedFile, category*: String, company*: String, department*: String, competence_period*: String): void
- test_data(pdf: UploadedFile): void
- extracted_index(uploaded_document_id: int): void
- uploads(): void
- uploaded_file(id: int): void
- extracted_show(id: int): void
- extracted_pdf(id: int): void
- reassign_range(id: int, page_range*: String, page_start*: int, page_end*: int): void
- update_metadata(id: int, metadata_updates: Hash): void
- validate_extracted(id: int): Json
- retry_processing(): Json
- retry_extracted(): Json
- process_file(file: UploadedFile, category*: String, company*: String, department*: String, competence_period*: String): void
- parse_range_params(): [int ,int]
- int_or_nil(value: Object): int | nil
- render_error(message: String): void
- page_range_pdf_service(source_pdf_path: String): DocumentProcessing::PageRangePdf
- initialize_processing_command(): Object
- initialize_file_processing_command(): Object
- reassign_extracted_range_command(): Object
- extracted_document_presenter(document: ExtractedDocument): Object
- db_manager(): DbManager
- document_processing_container(): Container
- content_type_for_uploaded(uploaded_document): String

5.2.3.2 ExtractedDocumentPresenter



Classe di tipo Presenter (Serializer) responsabile della trasformazione dei dati di un documento estratto in una struttura JSON ottimizzata per la presentazione lato frontend$_G$. Incapsula la logica di formattazione dei metadati, includendo la normalizzazione dei dati anagrafici del dipendente associato e la generazione di URL per il download del PDF estratto.

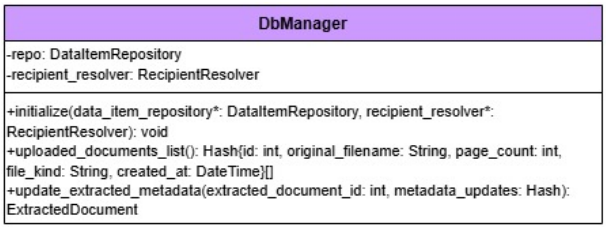
Attributi

- document: ExtractedDocument
- url_helpers: Module

Metodi

- initialize(document: ExtractedDocument, url_helpers*: Module): void
- as_json(*): Hash
- format_employee(employee*: Employee): Hash

5.2.3.3 DbManager



Componente delegato all'interazione con il database per il modulo$_G$ di elaborazione documentale. Agisce come un livello di astrazione per il recupero e l'aggiornamento dei metadati dei documenti estratti, incapsulando l'uso dei repository$_G$ e dei resolver per l'associazione dei destinatari.

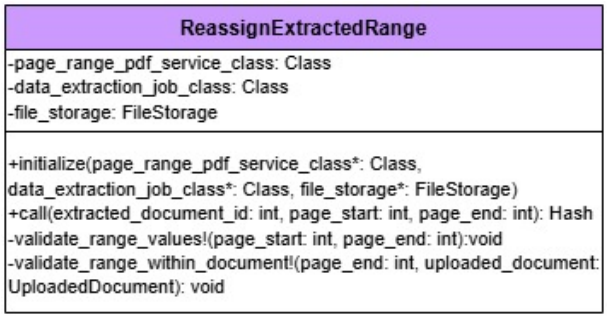
Attributi

- repo: DataItemRepository
- recipient_resolver: RecipientResolver

Metodi

- initialize(data_item_repository*: DataItemRepository, recipient_resolver*: RecipientResolver): void
- uploaded_documents_list(): Hash {id: int, original_filename: String, page_count: int, file_kind: String, created_at: DateTime} []
- update_extracted_metadata(extracted_document_id: int, metadata_updates: Hash): ExtractedDocument

5.2.3.4 ReassignExtractedRange



Servizio dedicato alla modifica del range di pagine associato a un documento già estratto. In caso di errore nel riconoscimento automatico, permette di ricalcolare i metadati eseguendo un nuovo job di estrazione dati basato sulle nuove pagine selezionate dall'utente.

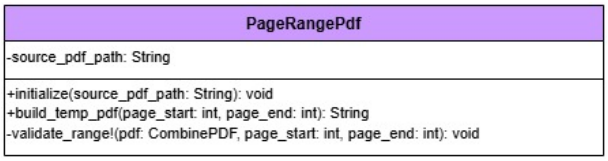
Attributi

- page_range_pdf_service_class: Class
- data_extraction_job_class: Class
- file_storage: FileStorage

Metodi

- initialize(page_range_pdf_service_class*: Class, data_extraction_job_class*: Class, file_storage*: FileStorage)
- call(extracted_document_id: int, page_start: int, page_end: int): Hash
- validate_range_values!(page_start: int, page_end: int):void
- validate_range_within_document!(page_end: int, uploaded_document: UploadedDocument): void

5.2.3.5 PageRangePdf



Servizio di utilità che si occupa della manipolazione fisica dei file PDF. È responsabile della creazione di file temporanei estrapolando un sottoinsieme specifico di pagine a partire da un documento sorgente più ampio.

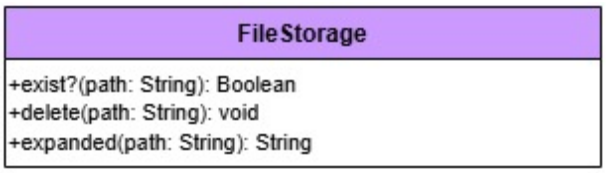
Attributi

- source_pdf_path: String

Metodi

- initialize(source_pdf_path: String): void
- build_temp_pdf(page_start: int, page_end: int): String
- validate_range!(pdf: CombinePDF, page_start: int, page_end: int): void

5.2.3.6 FileStorage



Interfaccia$_G$ di astrazione per la gestione del file system. Offre metodi standardizzati per verificare l'esistenza, eliminare ed espandere i percorsi dei file temporanei e persistenti utilizzati durante le fasi di caricamento e analisi.

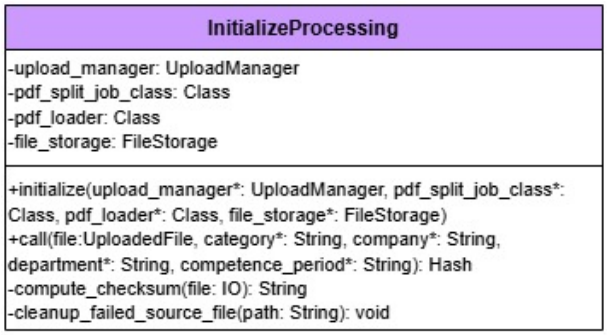
Attributi

- Nessuno

Metodi

- exist?(path: String): Boolean
- delete(path: String): void
- expanded(path: String): String

5.2.3.7 InitializeProcessing



Componente di orchestrazione che gestisce le fasi preparatorie all'elaborazione di un file,. Verifica$_G$ l'integrità del caricamento, calcola i checksum e innesca la catena di elaborazione asincrona (come lo splitting del PDF) tramite job dedicati.

Attributi

- upload_manager: UploadManager
- pdf_split_job_class: Class
- pdf_loader: Class

- file_storage: FileStorage

Metodi

- initialize(upload_manager*: UploadManager, pdf_split_job_class*: Class, pdf_loader*: Class, file_storage*: FileStorage)
- call(file:UploadedFile, category*: String, company*: String, department*: String, competence_period*: String): Hash
- compute_checksum(file: IO): String
- cleanup_failed_source_file(path: String): void

5.2.3.8 PdfSplitJob

Job in background responsabile dell'esecuzione asincrona del processo di divisione dei file PDF. Viene richiamato per alleggerire il carico sul thread principale, delegando la separazione logica dei documenti massivi in file individuali.

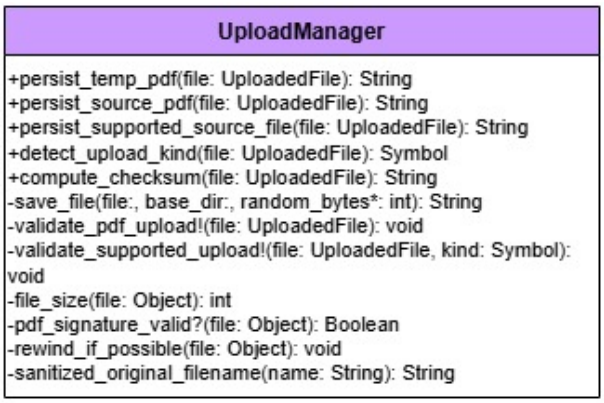
Attributi

- Nessuno

Metodi

- perform(file_path: String, job_id: int): void

5.2.3.9 UploadManager



Servizio centralizzato per la validazione$_G$ e la persistenza dei file in ingresso. Gestisce il salvataggio dei file sorgente, verifica$_G$ la correttezza del formato (es. firme PDF) e ne determina la tipologia per indirizzarli al corretto flusso di elaborazione.

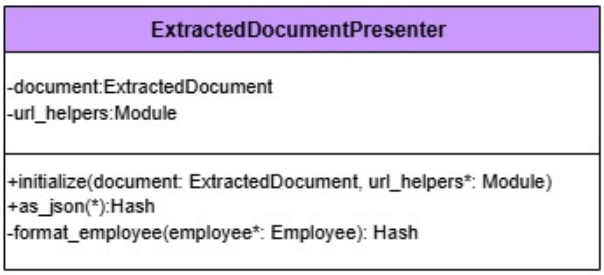
Attributi

- Nessuno

Metodi

- persist_temp_pdf(file: UploadedFile): String
- persist_source_pdf(file: UploadedFile): String
- persist_supported_source_file(file: UploadedFile): String
- detect_upload_kind(file: UploadedFile): Symbol
- compute_checksum(file: UploadedFile): String
- save_file(file: UploadedFile, base_dir: String, random_bytes: int): String
- validate_pdf_upload!(file: UploadedFile): void
- validate_supported_upload!(file: UploadedFile, kind: Symbol): void
- file_size(file: Object): int
- pdf_signature_valid?(file: Object): Boolean
- rewind_if_possible(file: Object): void
- sanitized_original_filename(name: String): String

5.2.3.10 ExtractedDocumentPresenter



Classe di presentazione (Serializer) responsabile della formattazione dei dati estratti. Trasforma le entità complesse, inclusi i dati anagrafici dei dipendenti associati, in una struttura JSON ottimizzata per la renderizzazione lato frontend$_G$.

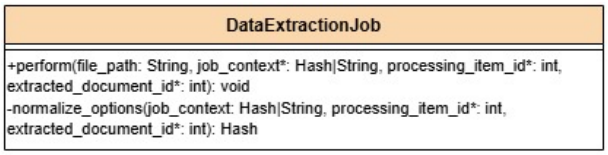
Attributi

- document:ExtractedDocument
- url_helpers:Module

Metodi

- initialize(document: ExtractedDocument, url_helpers*: Module)
- as_json(*):Hash
- format_employee(employee*: Employee): Hash

5.2.3.11 DataExtractionJob



Job in background che esegue materialmente il processo di estrazione dati su un singolo documento o frammento di esso. Normalizza il contesto di esecuzione e coordina le chiamate ai servizi AI$_G$ in modo asincrono.

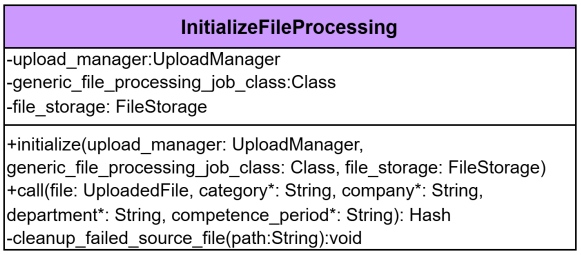
Attributi

- Nessuno

Metodi

- perform(file_path: String, job_context*: Hash|String, processing_item_id*: int, extracted_document_id*: int): void
- normalize_options(job_context: Hash|String, processing_item_id*: int, extracted_document_id*: int): Hash

5.2.3.12 InitializeFileProcessing



Servizio analogo a *InitializeProcessing*, ma specializzato per la gestione iniziale di file generici che non richiedono processi di splitting complessi (come le immagini o i CSV).

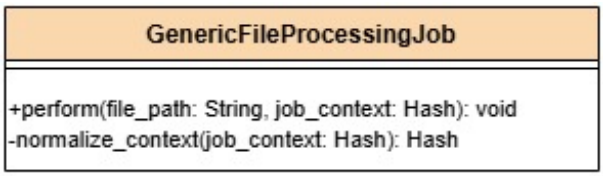
Attributi

- upload_manager: UploadManager
- generic_file_processing_job_class: Class
- file_storage: FileStorage

Metodi

- initialize(upload_manager: UploadManager, generic_file_processing_job_class: Class, file_storage: FileStorage)
- call(file: UploadedFile, category*: String, company*: String, department*: String, competence_period*: String): Hash
- compute_checksum(file: UploadedFile): String
- cleanup_failed_source_file(path:String):void

5.2.3.13 GenericFileProcessingJob



Job in background incaricato di processare file di formati eterogenei (es. file non PDF) in maniera asincrona, applicando la specifica logica di estrazione in base al tipo di file.

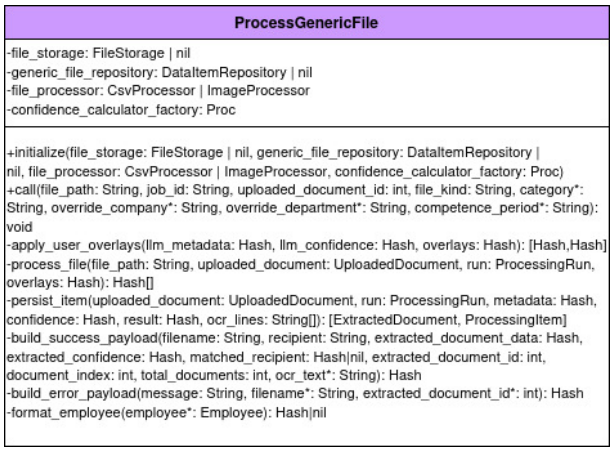
Attributi

- Nessuno

Metodi

- perform(file_path: String, job_context: Hash): void
- normalize_context(job_context: Hash): Hash

5.2.3.14 ProcessGenericFile



Servizio core per l'elaborazione dei file generici (immagini, CSV). Smista il documento al processore corretto, integra i punteggi di confidenza$_G$ restituiti dall'LLM e prepara i payload (sia in caso di successo che di errore) per l'aggiornamento in tempo reale dell'interfaccia$_G$.
Nonostante ha molti metodi la responsabilità è unica e ben definita: orchestrare il processo di estrazione dati per file generici, applicando le logiche di normalizzazione e integrazione dei risultati in un unico punto centralizzato, delegando tutto, mantenendo così la coesione del flusso di elaborazione specifico per questi formati.

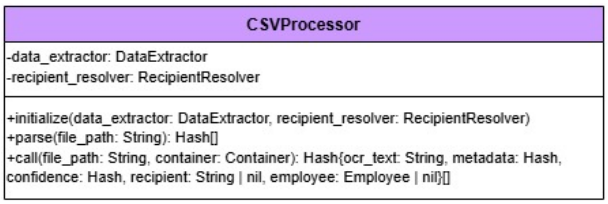
Attributi

- file_storage: FileStorage | nil
- generic_file_repository: DataItemRepository | nil
- file_processor: CsvProcessor | ImageProcessor
- confidence_calculator_factory: Proc

Metodi

- initialize(file_storage: FileStorage | nil, generic_file_repository: DataItemRepository | nil, file_processor: CsvProcessor | ImageProcessor, confidence_calculator_factory: Proc): void
- call(file_path: String, job_id: String, uploaded_document_id: int, file_kind: String, category*: String, override_company*: String, override_department*: String, competence_period*: String): void
- apply_user_overlays(llm_metadata: Hash, llm_confidence: Hash, overlays: Hash): [Hash,Hash]
- process_file(file_path: String, uploaded_document: UploadedDocument, run: ProcessingRun, overlays: Hash): Hash[]
- persist_item(uploaded_document: UploadedDocument, run: ProcessingRun, metadata: Hash, confidence: Hash, result: Hash, ocr_lines: String[]): [ExtractedDocument, ProcessingItem]
- build_success_payload(filename: String, recipient: String, extracted_document_data: Hash, extracted_confidence: Hash, matched_recipient: Hash|nil, extracted_document_id: int, document_index: int, total_documents: int, ocr_text*: String): Hash
- build_error_payload(message: String, filename*: String, extracted_document_id*: int): Hash
- format_employee(employee*: Employee): Hash|nil

5.2.3.15 CSVProcessor



Servizio specializzato nell'estrazione di dati da file CSV. Implementa logiche di parsing e normalizzazione specifiche per questo formato, estraendo i metadati strutturati e preparando i dati per la valutazione della confidence da parte dell'LLM$_G$.

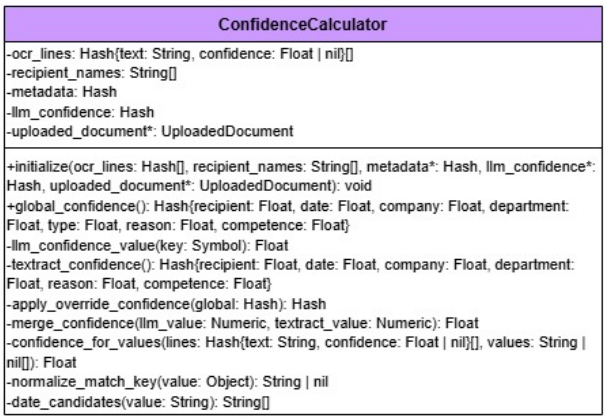
Attributi

- data_extractor: DataExtractor
- recipient_resolver: RecipientResolver

Metodi

- initialize(data_extractor: DataExtractor, recipient_resolver: RecipientResolver)
- parse(file_path: String): Hash[]
- call(file_path: String, container: Container): Hash{ocr_text: String, metadata: Hash, confidence: Hash, recipient: String | nil, employee: Employee | nil}[]

5.2.3.16 ConfidenceCalculator



Motore di calcolo dedicato alla determinazione del punteggio di affidabilità$_G$ (confidence score$_G$) dei dati estratti[cite: 817, 826]. Combina le metriche restituite dal motore OCR$_G$ con quelle dell'LLM$_G$, applicando logiche di fusione e regole di override in base a campi specifici (es. destinatari, date).
Anche qui ha solo una responsabilità, quella di calcolare i punteggi di confidenza$_G$ in maniera centralizzata.

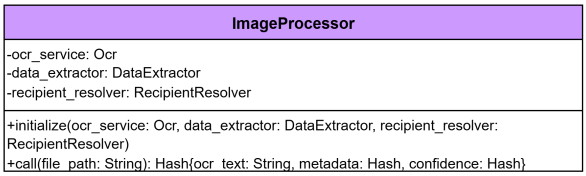
Attributi

- ocr_lines: Hash{text: String, confidence: Float | nil}[]
- recipient_names: String[]
- metadata: Hash
- llm_confidence: Hash
- uploaded_document*: UploadedDocument

Metodi

- initialize(ocr_lines: Hash[], recipient_names: String[], metadata*: Hash, llm_confidence*: Hash, uploaded_document*: UploadedDocument): void
- global_confidence(): Hash{recipient: Float, date: Float, company: Float, department: Float, type: Float, reason: Float, competence: Float}
- llm_confidence_value(key: Symbol): Float
- textract_confidence(): Hash{recipient: Float, date: Float, company: Float, department: Float, reason: Float, competence: Float}
- apply_override_confidence(global: Hash): Hash
- merge_confidence(llm_value: Numeric, textract_value: Numeric): Float
- confidence_for_values(lines: Hash{text: String, confidence: Float | nil}[], values: String | nil[]): Float
- normalize_match_key(value: Object): String | nil
- date_candidates(value: String): String[]

5.2.3.17 ImageProcessor



Servizio che coordina il flusso di estrazione dei dati a partire da formati immagine. Incapsula le chiamate al servizio OCR$_G$ per ottenere il testo grezzo e lo inoltra al modulo$_G$ di estrazione per identificare i metadati strutturati e i destinatari.

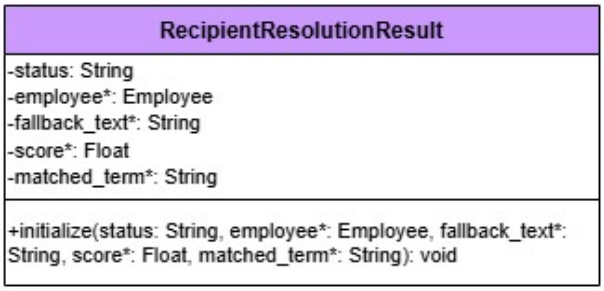
Attributi

- ocr_service: Ocr$_G$
- data_extractor: DataExtractor
- recipient_resolver: RecipientResolver

Metodi

- initialize(ocr_service: Ocr$_G$, data_extractor: DataExtractor, recipient_resolver: RecipientResolver)
- call(file_path: String): Hash{ocr_text: String, metadata: Hash, confidence: Hash}

5.2.3.18 RecipientResolutionResult



Data Transfer Object (DTO) che modella il risultato del processo di identificazione di un destinatario[cite: 846, 854]. Contiene lo stato della ricerca, il dipendente associato, eventuali testi di fallback e il punteggio di accuratezza del match.

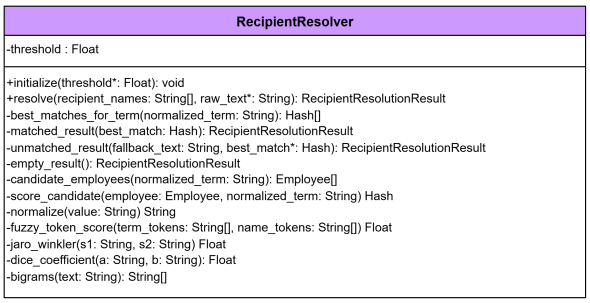
Attributi

- status: String
- employee*: Employee
- fallback_text*: String
- score*: Float
- matched_term*: String

Metodi

- initialize(status: String, employee*: Employee, fallback_text*: String, score*: Float, matched_term*: String): void

5.2.3.19 RecipientResolver



Motore di risoluzione anagrafica che associa i nomi estratti dal testo ai record reali dei dipendenti[cite: 855, 860]. Impiega algoritmi di string matching avanzati (come Jaro-Winkler e coefficiente di Dice) per valutare i candidati e restituire l'associazione più probabile.

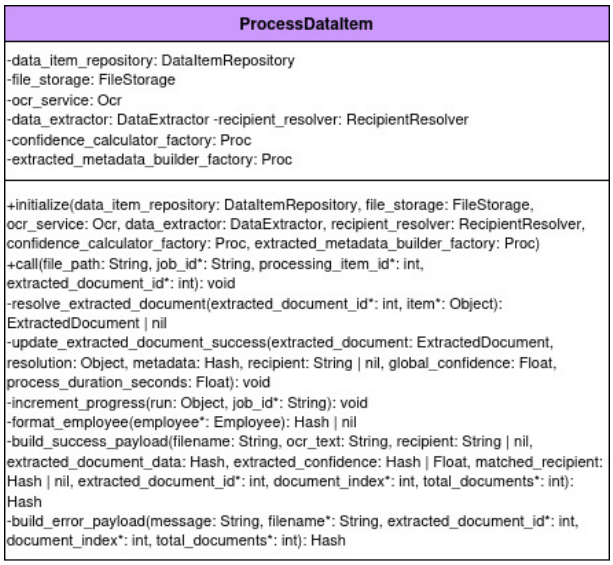
Attributi

- threshold : Float

Metodi

- initialize(threshold*: Float): void
- resolve(recipient_names: String[], raw_text*: String): RecipientResolutionResult
- best_matches_for_term(normalized_term: String): Hash[]
- matched_result(best_match: Hash): RecipientResolutionResult
- unmatched_result(fallback_text: String, best_match*: Hash): RecipientResolutionResult
- empty_result(): RecipientResolutionResult
- candidate_employees(normalized_term: String): Employee[]
- score_candidate(employee: Employee, normalized_term: String): Hash
- normalize(value: String): String
- fuzzy_token_score(term_tokens: String[], name_tokens: String[]): Float
- jaro_winkler(s1: String, s2: String): Float
- dice_coefficient(a: String, b: String): Float
- bigrams(text: String): String[]

5.2.3.20 ProcessDataItem



Servizio orchestratore per l'elaborazione end-to-end di un singolo elemento di dato. Coordina l'interazione tra OCR$_G$, Data Extractor, Recipient Resolver e gestisce l'aggiornamento dello stato del job, inviando notifiche WebSockets (ActionCable) al termine del processo.

Attributi

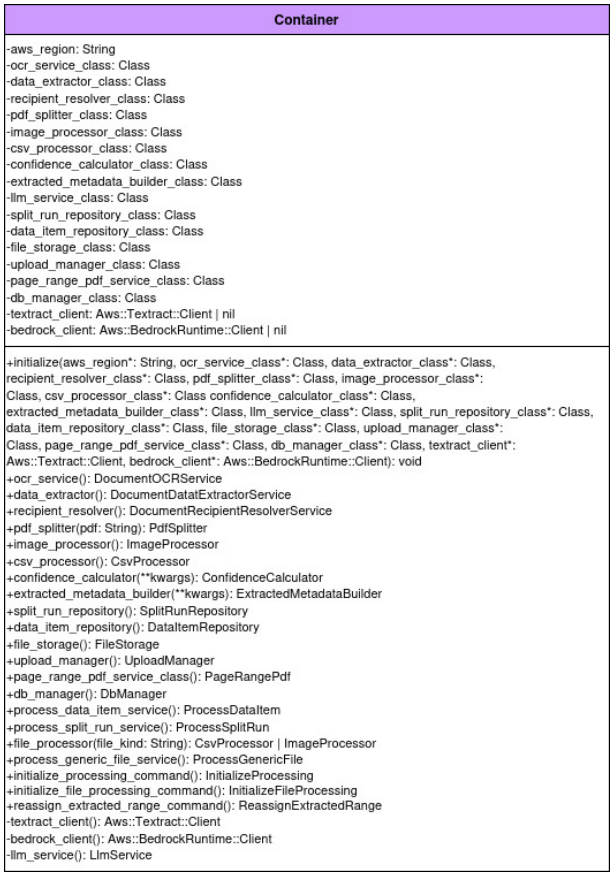
- data_item_repository: DataItemRepository
- file_storage: FileStorage
- ocr_service: Ocr$_G$
- data_extractor: DataExtractor
- recipient_resolver: RecipientResolver
- confidence_calculator_factory: Proc
- extracted_metadata_builder_factory: Proc

Metodi

- initialize(data_item_repository: DataItemRepository, file_storage: FileStorage, ocr_service: Ocr$_G$, data_extractor: DataExtractor, recipient_resolver: RecipientResolver, confidence_calculator_factory: Proc, extracted_metadata_builder_factory: Proc)
- call(file_path: String, job_id*: String, processing_item_id*: int, extracted_document_id*: int): void
- resolve_extracted_document(extracted_document_id*: int, item*: Object): ExtractedDocument | nil
- update_extracted_document_success(extracted_document: ExtractedDocument, resolution: Object, metadata: Hash, recipient: String | nil, global_confidence: Float, process_duration_seconds: Float): void
- increment_progress(run: Object, job_id*: String): void

- format_employee(employee*: Employee): Hash | nil
- build_success_payload(filename: String, ocr_text: String, recipient: String | nil, extracted_document_data: Hash, extracted_confidence: Hash | Float, matched_recipient: Hash | nil, extracted_document_id*: int, document_index*: int, total_documents*: int): Hash
- build_error_payload(message: String, filename*: String, extracted_document_id*: int, document_index*: int, total_documents*: int): Hash

5.2.3.21 Container



Modulo$_G$ di Dependency Injection (DI) del sistema. Centralizza la configurazione e l'istanziazione di tutte le classi di servizio, repository$_G$ e client esterni (es. AWS$_G$ Bedrock$_G$, AWS$_G$ Textract) garantendo una corretta inversione delle dipendenze.

Attributi

- aws_region: String
- ocr_service_class: Class
- data_extractor_class: Class
- recipient_resolver_class: Class
- pdf_splitter_class: Class
- image_processor_class: Class
- csv_processor_class: Class

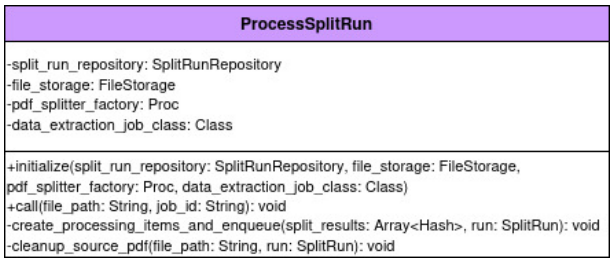
- confidence_calculator_class: Class
- extracted_metadata_builder_class: Class
- llm_service_class: Class
- split_run_repository_class: Class
- data_item_repository_class: Class
- file_storage_class: Class
- upload_manager_class: Class
- page_range_pdf_service_class: Class
- db_manager_class: Class
- textract_client: Aws::Textract::Client | nil
- bedrock_client: Aws::BedrockRuntime::Client | nil

Metodi

- initialize(aws_region*: String, ocr_service_class*: Class, data_extractor_class*: Class, recipient_resolver_class*: Class, pdf_splitter_class*: Class, image_processor_class*: Class, csv_processor_class*: Class, confidence_calculator_class*: Class, extracted_metadata_builder_class*: Class, llm_service_class*: Class, split_run_repository_class*: Class, data_item_repository_class*: Class, file_storage_class*: Class, upload_manager_class*: Class, page_range_pdf_service_class*: Class, db_manager_class*: Class, textract_client*: Aws::Textract::Client bedrock_client*: Aws::BedrockRuntime::Client): void
- ocr_service(): DocumentOCRService
- data_extractor(): DocumentDataExtractorService
- recipient_resolver(): DocumentRecipientResolverService
- pdf_splitter(pdf: String): PdfSplitter
- image_processor(): ImageProcessor
- csv_processor(): CsvProcessor
- confidence_calculator(**kwargs): ConfidenceCalculator
- extracted_metadata_builder(**kwargs): ExtractedMetadataBuilder
- split_run_repository(): SplitRunRepository
- data_item_repository(): DataItemRepository
- file_storage(): FileStorage
- upload_manager(): UploadManager
- page_range_pdf_service_class(): PageRangePdf
- db_manager(): DbManager
- process_data_item_service(): ProcessDataItem
- process_split_run_service(): ProcessSplitRun

- file_processor(file_kind: String): CsvProcessor | ImageProcessor
- process_generic_file_service(): ProcessGenericFile
- initialize_processing_command(): InitializeProcessing
- initialize_file_processing_command(): InitializeFileProcessing
- reassign_extracted_range_command(): ReassignExtractedRange

5.2.3.22 ProcessSplitRun



Servizio che orchestra il ciclo di vita post-split di un documento PDF. Recupera i segmenti separati, crea i relativi item di elaborazione nel database e ne avvia la messa in coda per l'estrazione asincrona dei dati.

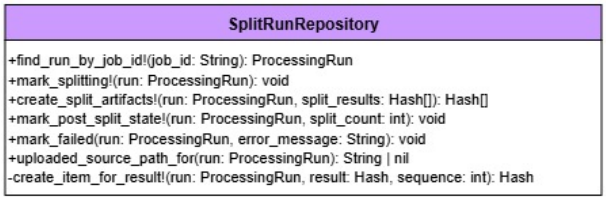
Attributi

- split_run_repository: SplitRunRepository
- file_storage: FileStorage
- pdf_splitter_factory: Proc
- data_extraction_job_class: Class

Metodi

- initialize(split_run_repository: SplitRunRepository, file_storage: FileStorage, pdf_splitter_factory: Proc, data_extraction_job_class: Class)
- call(file_path: String, job_id: String): void
- create_processing_items_and_enqueue(split_results: Hash[], run: SplitRun): void
- cleanup_source_pdf(file_path: String, run: SplitRun): void

5.2.3.23 SplitRunRepository



Classe di tipo Repository$_G$ dedicata alla gestione dello stato per le esecuzioni di splitting. Fornisce l'interfaccia$_G$ di persistenza per tracciare i progressi dell'elaborazione massiva, segnandone successi, artefatti creati o eventuali fallimenti.

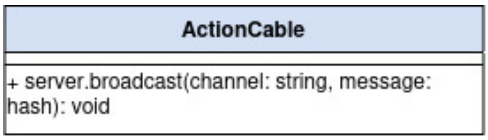
Attributi

- Nessuno

Metodi

- find_run_by_job_id!(job_id: String): ProcessingRun
- mark_splitting!(run: ProcessingRun): void
- create_split_artifacts!(run: ProcessingRun, split_results: Hash[]): Hash[]
- mark_post_split_state!(run: ProcessingRun, split_count: int): void
- mark_failed(run: ProcessingRun, error_message: String): void
- uploaded_source_path_for(run: ProcessingRun): String | nil
- create_item_for_result!(run: ProcessingRun, result: Hash, sequence: int): Hash

5.2.3.24 ActionCable



Classe nativa di rails per la gestione delle connessioni WebSocket. Nel contesto del modulo$_G$ AI Copilot$_G$, viene utilizzata per inviare aggiornamenti in tempo reale al frontend$_G$ riguardo lo stato dell'analisi in corso.

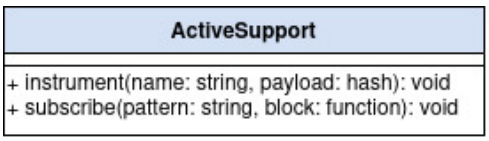
Attributi

- Nessuno

Metodi

- server.broadcast(channel: string, message: Json): void

5.2.3.25 ActiveSupport



Classe nativa di rails per la gestione delle connessioni WebSocket. Nel contesto del modulo$_G$ AI Copilot$_G$, viene utilizzata insieme ad altri componenti per gestire le comunicazioni in tempo reale. Recupera le notifiche inviate dai servizi di elaborazione e le distribuisce ai client sottoscritti, facilitando l'aggiornamento dinamico dell'interfaccia$_G$ utente.

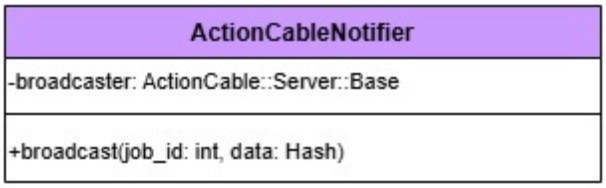
Attributi

- Nessuno

Metodi

- instrument(name: string, payload: hash): void
- subscribe(pattern: string, block: function): void

5.2.3.26 ActionCableNotifier



Servizio per la gestione delle notifiche in tempo reale. Permette al backend$_G$ di comunicare al frontend$_G$ via WebSockets lo stato di avanzamento dei processi asincroni (job) per un aggiornamento fluido della UI.

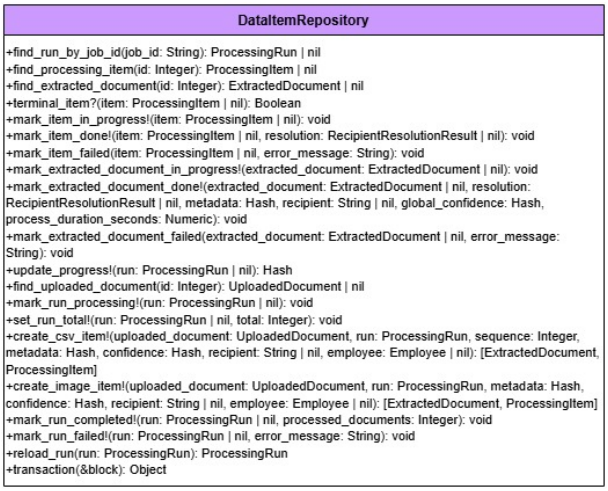
Attributi

- broadcaster: ActionCable::Server::Base

Metodi

- broadcast(job_id: int, data: Hash)

5.2.3.27 DataItemRepository



Repository$_G$ che gestisce la persistenza degli oggetti legati alla singola elaborazione e all'estrazione documentale. Regola le transazioni sul database e la transizione di stato degli item da "in progress" a "completato" o "fallito".

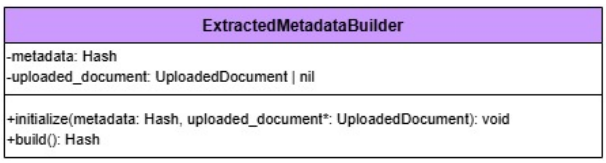
Attributi

- Nessuno

Metodi

- find_run_by_job_id(job_id: String): ProcessingRun | nil
- find_processing_item(id: Integer): ProcessingItem | nil
- find_extracted_document(id: Integer): ExtractedDocument | nil
- terminal_item?(item: ProcessingItem | nil): Boolean
- mark_item_in_progress!(item: ProcessingItem | nil): void
- mark_item_done!(item: ProcessingItem | nil, resolution: RecipientResolutionResult | nil): void
- mark_item_failed(item: ProcessingItem | nil, error_message: String): void
- mark_extracted_document_in_progress!(extracted_document: ExtractedDocument | nil): void
- mark_extracted_document_done!(extracted_document: ExtractedDocument | nil, resolution: RecipientResolutionResult | nil, metadata: Hash, recipient: String | nil, global_confidence: Hash, process_duration_seconds: Numeric): void
- mark_extracted_document_failed(extracted_document: ExtractedDocument | nil, error_message: String): void
- update_progress!(run: ProcessingRun | nil): Hash
- find_uploaded_document(id: Integer): UploadedDocument | nil
- mark_run_processing!(run: ProcessingRun | nil): void
- set_run_total!(run: ProcessingRun | nil, total: Integer): void
- create_csv_item!(uploaded_document: UploadedDocument, run: ProcessingRun, sequence: Integer, metadata: Hash, confidence: Hash, recipient: String | nil, employee: Employee | nil): [ExtractedDocument, ProcessingItem]
- create_image_item!(uploaded_document: UploadedDocument, run: ProcessingRun, metadata: Hash, confidence: Hash, recipient: String | nil, employee: Employee | nil): [ExtractedDocument, ProcessingItem]
- mark_run_completed!(run: ProcessingRun | nil, processed_documents: Integer): void
- mark_run_failed!(run: ProcessingRun | nil, error_message: String): void
- reload_run(run: ProcessingRun): ProcessingRun
- transaction(&block): Object

5.2.3.28 ExtractedMetadataBuilder



Classe di utilità responsabile della costruzione e formattazione finale dell'oggetto dei metadati estratti, preparandolo per essere associato in maniera coerente al modello del database.

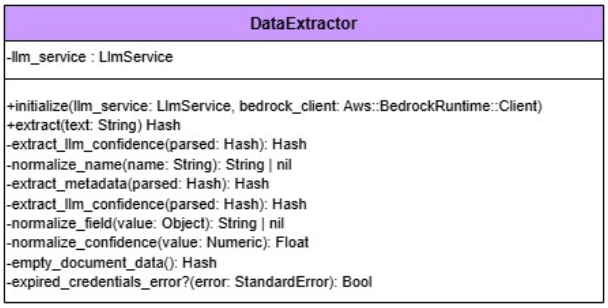
Attributi

- metadata: Hash
- uploaded_document: UploadedDocument | nil

Metodi

- initialize(metadata: Hash, uploaded_document: UploadedDocument): void
- build(): Hash

5.2.3.29 DataExtractor



Servizio delegato all'interfacciamento con il modello LLM$_G$ per estrapolare metadati strutturati dal testo grezzo. Include logiche per la pulizia del risultato JSON e la normalizzazione dei campi estratti prima del salvataggio.

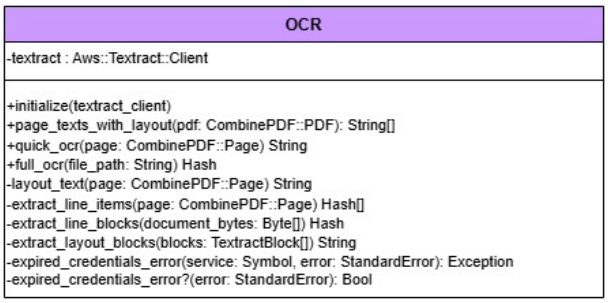
Attributi

- llm_service : LlmService

Metodi

- initialize(llm_service: LlmService, bedrock_client: Aws::BedrockRuntime::Client)
- extract(text: String): Hash
- extract_llm_confidence(parsed: Hash): Hash
- normalize_name(name: String): String | nil
- extract_metadata(parsed: Hash): Hash
- extract_llm_confidence(parsed: Hash): Hash
- normalize_field(value: Object): String | nil
- normalize_confidence(value: Numeric): Float
- empty_document_data(): Hash
- expired_credentials_error?(error: StandardError): Bool

5.2.3.30 OCR



Servizio che wrappa le chiamate al motore OCR$_G$ (AWS$_G$ Textract). È responsabile della conversione di PDF o immagini in blocchi di testo strutturati, analizzando i layout per fornire un input coerente all'intelligenza artificiale generativa.

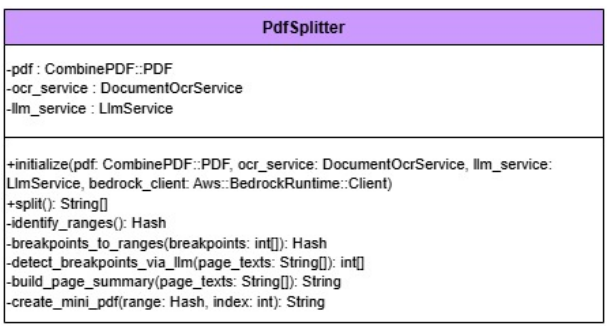
Attributi

- textract : Aws::Textract::Client

Metodi

- initialize(textract_client)
- page_texts_with_layout(pdf: CombinePDF::PDF): String[]
- quick_ocr(page: CombinePDF::Page) String
- full_ocr(file_path: String) Hash
- layout_text(page: CombinePDF::Page) String
- extract_line_items(page: CombinePDF::Page) Hash[]
- extract_line_blocks(document_bytes: Byte[]) Hash
- extract_layout_blocks(blocks: TextractBlock[]) String
- expired_credentials_error(service: Symbol, error: StandardError): Exception
- expired_credentials_error?(error: StandardError): Bool

5.2.3.31 PdfSplitter



Motore di divisione dei PDF basato su intelligenza artificiale. Identifica i confini logici tra documenti diversi all'interno dello stesso file (breakpoints) basandosi su riassunti del testo di ogni pagina analizzati tramite l'LLM$_G$.

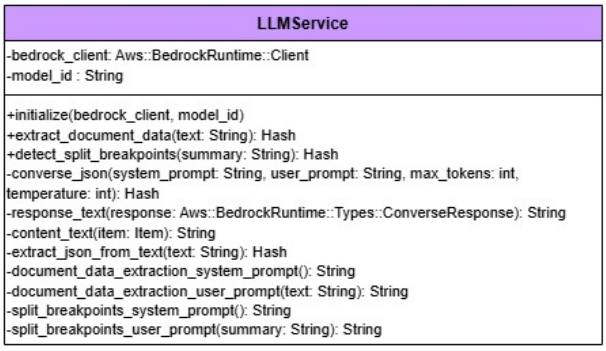
Attributi

- pdf : CombinePDF::PDF
- ocr_service : DocumentOcrService
- llm_service : LlmService

Metodi

- +initialize(pdf: CombinePDF::PDF, ocr_service: DocumentOcrService, llm_service: LlmService, bedrock_client: Aws::BedrockRuntime::Client)
- split(): String[]
- identify_ranges(): Hash
- breakpoints_to_ranges(breakpoints: int[]): Hash
- detect_breakpoints_via_llm(page_texts: String[]): int[]
- build_page_summary(page_texts: String[]): String
- create_mini_pdf(range: Hash, index: int): String

5.2.3.32 LLMService



Servizio dedicato alla gestione della comunicazione con i foundation models (es. tramite AWS$_G$ Bedrock$_G$ Runtime). Prepara i prompt$_G$ di sistema e utente per l'estrazione dati e il rilevamento dei breakpoint, gestendo la corretta esecuzione delle richieste AI$_G$ e il parsing delle risposte JSON.

Attributi

- bedrock_client: Aws::BedrockRuntime::Client
- model_id : String

Metodi

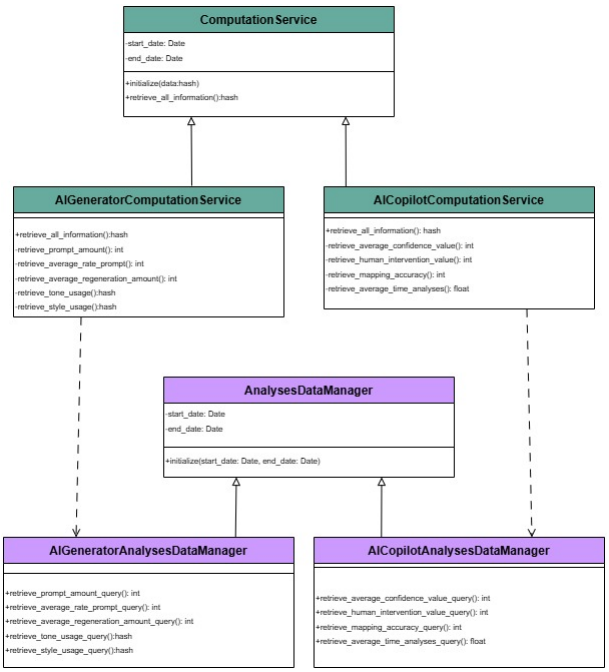
- initialize(bedrock_client, model_id)
- extract_document_data(text: String): Hash

- detect_split_breakpoints(summary: String): Hash
- converse_json(system_prompt: String, user_prompt: String, max_tokens: int, temperature: int): Hash
- response_text(response: Aws::BedrockRuntime::Types::ConverseResponse): String
- content_text(item: Item): String
- extract_json_from_text(text: String): Hash
- document_data_extraction_system_prompt(): String
- document_data_extraction_user_prompt(text: String): String
- split_breakpoints_system_prompt(): String
- split_breakpoints_user_prompt(summary: String): String

5.3 Modulo$_G$ Data analisi

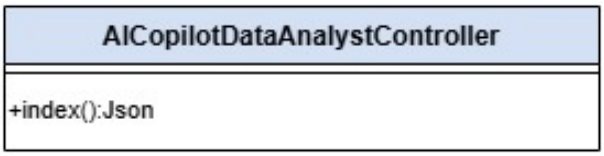
Il modulo$_G$ *Data analisi* espone i componenti fondamentali per l'estrazione, il calcolo e l'esposizione delle metriche di utilizzo e delle statistiche relative alle due funzionalità$_G$ principali del sistema: l'AI Co-Pilot$_G$ per i documenti e l'AI-assistant$_G$ generativo. Il modulo$_G$ segue un'architettura a livelli, separando la gestione delle richieste (Controller), la logica di business e calcolo (Service) e l'interazione con il database (Data Manager).

5.3.1 Diagramma delle classi blocco modulo$_G$ Data analisi



Questo diagramma illustra le classi principali del modulo$_G$ di Data Analisi, evidenziando le loro responsabilità e le relazioni tra di esse. Lo scopo di questo modulo$_G$ è fornire un'interfaccia$_G$ coerente per l'estrazione e la presentazione delle metriche chiave, consentendo al front-end$_G$ di visualizzare dashboard$_G$ informative sull'utilizzo e le performance dell'AI Co-Pilot$_G$ e dell'AI-assistant$_G$ generativo.

5.3.1.1 AICopilotDataAnalystController



Classe di tipo Controller responsabile dell'esposizione degli endpoint API$_G$ relativi alle statistiche dell'AI Co-Pilot$_G$. Gestisce le richieste HTTP provenienti dal front-end$_G$, validando i parametri di input (es. i filtri temporali). Delega la complessa logica di calcolo delle metriche (come confidenza$_G$ media e accuratezza) al livello di servizio sottostante, per poi impacchettare i risultati in un formato standard (es. JSON) da restituire al client.

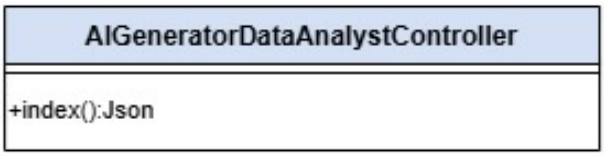
Attributi

- Nessuno

Metodi

- +index():Json

5.3.1.2 AIGeneratorDataAnalystController



Classe di tipo Controller che gestisce l'interfaccia$_G$ API$_G$ per le analytics dell'AI-assistant$_G$ generativo. Riceve e processa le richieste relative ai dati di generazione (prompt$_G$ effettuati, rigenerazioni, feedback degli utenti). Come il suo omologo per il Co-Pilot, funge da punto di ingresso per le richieste, interfacciandosi con i servizi di computazione per elaborare i dati e restituire le risposte al client in base ai parametri temporali specificati.

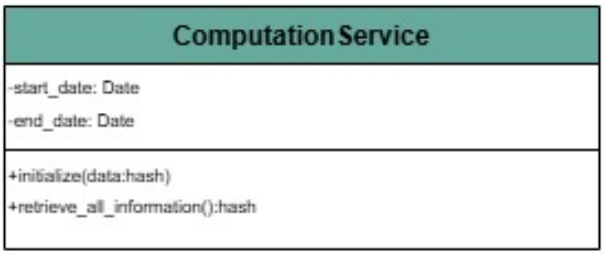
Attributi

- Nessuno

Metodi

- +index():Json

5.3.1.3 ComputationService



Interfaccia$_G$ o classe base astratta che definisce il contratto standard per i servizi di calcolo delle metriche di analytics. Fornisce i metodi comuni e le firme necessarie per l'elaborazione dei dati grezzi, garantendo uniformità e polimorfismo tra i diversi servizi di computazione implementati nel modulo$_G$.

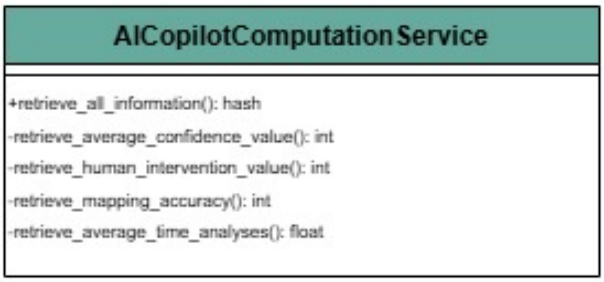
Attributi

- start_date: Date
- end_date: Date

Metodi

- initialize(data:hash)
- retrieve_all_information():hash

5.3.1.4 AiCopilotComputationService



Classe concreta che implementa la logica di business per l'elaborazione delle metriche specifiche dell'AI Co-Pilot$_G$. Riceve i dati grezzi dal livello dati e applica gli algoritmi necessari per calcolare indicatori chiave quali: la percentuale di confidenza$_G$ media delle estrazioni, la frequenza dell'intervento umano per le correzioni, l'accuratezza strutturale del mapping dei dati e i tempi medi di elaborazione dei documenti.

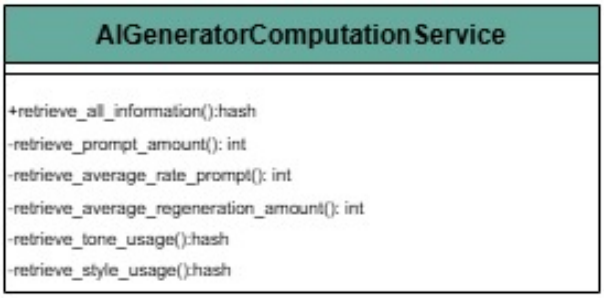
Attributi

- Nessuno

Metodi

- retrieve_all_information():hash
- retrieve_prompt_amount(): int
- retrieve_average_rate_prompt(): int
- retrieve_average_regeneration_amount(): int
- retrieve_tone_usage():hash
- retrieve_style_usage():hash

5.3.1.5 AiGeneratorComputationService



Classe concreta contenente la logica di business per la generazione delle statistiche relative all'AI-assistant$_G$. Si occupa di aggregare e calcolare le metriche di utilizzo dell'assistente, tra cui il volume totale delle generazioni, la valutazione media fornita dagli utenti, il tasso di rigenerazione dei prompt$_G$ e la distribuzione percentuale dell'utilizzo dei vari toni e stili di comunicazione.

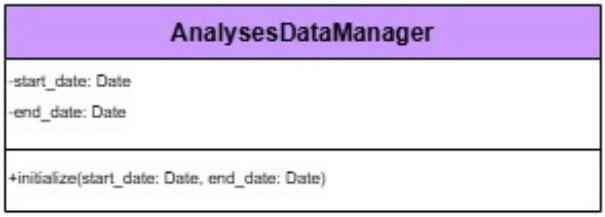
Attributi

- Nessuno

Metodi

- retrieve_all_information(): hash
- retrieve_average_confidence_value(): int
- retrieve_human_intervention_value(): int
- retrieve_mapping_accuracy(): int
- retrieve_average_time_analyses(): float

5.3.1.6 AnalysesDataManager



Interfaccia$_G$ o classe base per il livello di persistenza (Data Access Object / Repository) dedicato alle analytics. Definisce le operazioni comuni per l'interrogazione del database al fine di estrarre lo storico delle interazioni, i log di sistema e i metadati senza esporre la complessità delle query SQL (o ORM) ai livelli superiori.

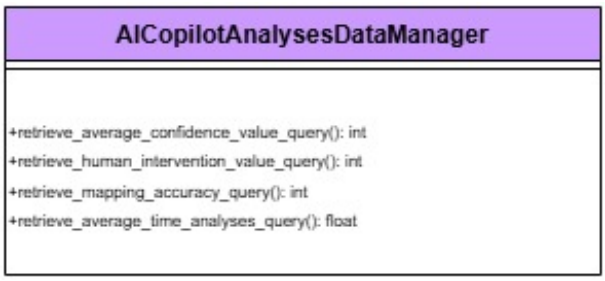
Attributi

- start_date: Date
- end_date: Date

Metodi

- initialize(start_date: Date, end_date: Date)

5.3.1.7 AiCopilotAnalysesDataManager



Implementazione$_G$ specifica del Data Manager responsabile del recupero dei dati storici legati al modulo$_G$ AI Co-Pilot$_G$. Interroga le tabelle del database pertinenti (es. storico dei documenti caricati, audit log delle modifiche manuali, timestamp di elaborazione) estraendo dataset filtrati che verranno poi passati al *AiCopilotComputationService* per le elaborazioni statistiche.

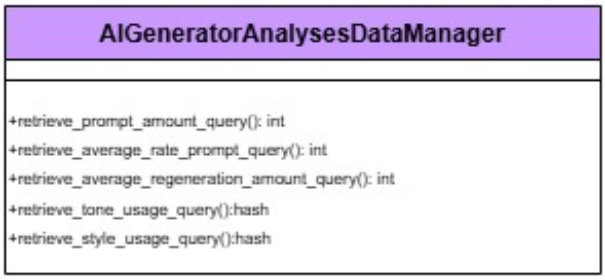
Attributi

- Nessuno

Metodi

- retrieve_prompt_amount_query(): int
- retrieve_average_rate_prompt_query(): int
- retrieve_average_regeneration_amount_query(): int
- retrieve_tone_usage_query():hash
- retrieve_style_usage_query():hash

5.3.1.8 AiGeneratorAnalysesDataManager



Implementazione$_G$ specifica del Data Manager per il recupero dei dati relativi all'AI-assistant$_G$ generativo. Si occupa di eseguire query sul database per estrarre lo storico dei prompt$_G$ inviati, i parametri utilizzati (tono e stile), i feedback registrati (rating) e i conteggi delle richieste di rigenerazione, fornendo il set di dati grezzi al *AiGeneratorComputationService*.

Attributi

- Nessuno

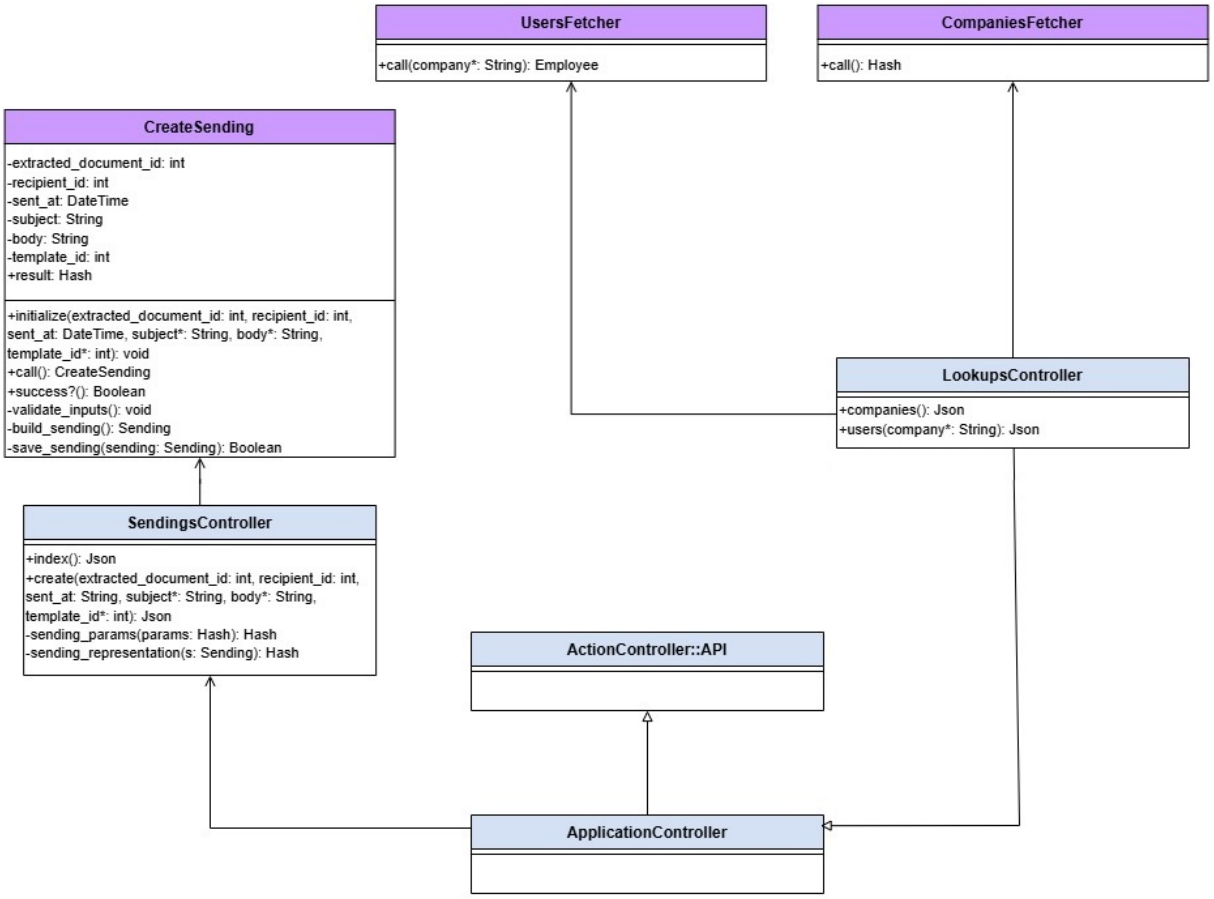
Metodi

- retrieve_average_confidence_value_query(): int
- retrieve_human_intervention_value_query(): int
- retrieve_mapping_accuracy_query(): int
- retrieve_average_time_analyses_query(): float

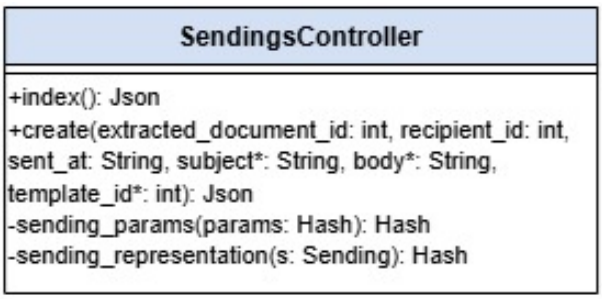
5.4 Classi condivise

Questa sezione include classi che, pur non appartenendo a un modulo$_G$ specifico, forniscono funzionalità$_G$ trasversali utilizzate sia dal modulo$_G$ di AI Co-Pilot$_G$ che da quello di AI-assistant$_G$ generativo.

5.4.1 Diagramma delle classi blocco modulo$_G$ Classi condivise



5.4.1.1 SendingsController



Questa classe di tipo Controller gestisce le richieste HTTP relative alla creazione e gestione delle "sendings", ovvero le comunicazioni inviate ai destinatari in seguito all'estrazione dei dati. Fornisce endpoint per visualizzare le sendings esistenti e per crearne di nuove, delegando la logica di business alla classe *CreateSending*. Inoltre, include metodi di utilità per formattare i parametri di invio e rappresentare le sendings in un formato adatto alla risposta JSON.

Attributi

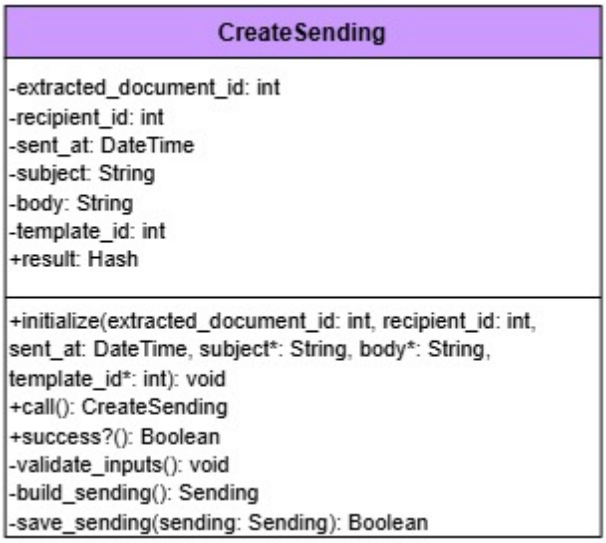
- Nessuno

Metodi

- index(): Json
- create(extracted_document_id: int, recipient_id: int, sent_at: String, subject*: String, body*: String, template_id*: int): Json
- sending_params(params: Hash): Hash
- sending_representation(s: Sending): Hash

5.4.1.2 CreateSending

Questa classe di tipo Command incapsula la logica necessaria per creare una nuova sending. Si occupa di validare i parametri di input, costruire l'oggetto Sending da salvare nel database e gestire eventuali errori durante il processo di creazione.



Attributi

- extracted_document_id: int
- recipient_id: int
- sent_at: DateTime
- subject: String
- body: String
- template_id: int
- result: Hash

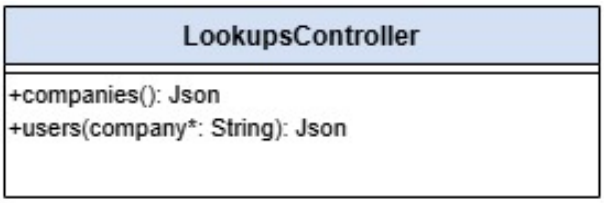
Metodi

- initialize(extracted_document_id: int, recipient_id: int, sent_at: DateTime, subject*: String, body*: String, template_id*: int): void
- call(): CreateSending
- success?(): Boolean
- validate_inputs(): void

- build_sending(): Sending
- save_sending(sending: Sending): Boolean

5.4.1.3 LookupsController

Questa classe di tipo Controller fornisce endpoint API$_G$ per recuperare dati di riferimento condivisi, come la lista delle aziende e degli utenti associati a ciascuna azienda. Questi dati sono utilizzati sia dal modulo$_G$ di AI Co-Pilot$_G$ che da quello di AI-assistant$_G$ generativo per popolare dropdown, assegnare responsabilità o filtrare le informazioni visualizzate. Il controller delega la logica di recupero dei dati a classi di servizio dedicate (es. *UsersFetcher* e *CompaniesFetcher*).



Attributi

- Nessuno

Metodi

- companies(): Json
- users(company*: String): Json

5.4.1.4 UsersFetcher

Questa classe di tipo Service Object è responsabile del recupero degli utenti associati a una specifica azienda. Fornisce un'interfaccia$_G$ semplice per ottenere la lista degli employee da un database o da un servizio esterno, filtrando per azienda. Viene utilizzata principalmente dal *LookupsController* per popolare le dropdown di selezione degli utenti nel front-end$_G$.

Attributi

- Nessuno

Metodi

- call(company*: String): Employee

5.4.1.5 CompaniesFetcher

Questa classe di tipo Service Object è responsabile del recupero della lista delle aziende disponibili. Fornisce un'interfaccia$_G$ semplice per ottenere l'elenco delle aziende da un database o da un servizio esterno, restituendo i dati in un formato adatto alla serializzazione JSON. Viene utilizzata principalmente dal *LookupsController* per popolare le dropdown di selezione delle aziende nel front-end$_G$.

Attributi

- Nessuno

Metodi

- call(): Hash

6 Architettura Frontend$_G$

6.1 Principi di progettazione

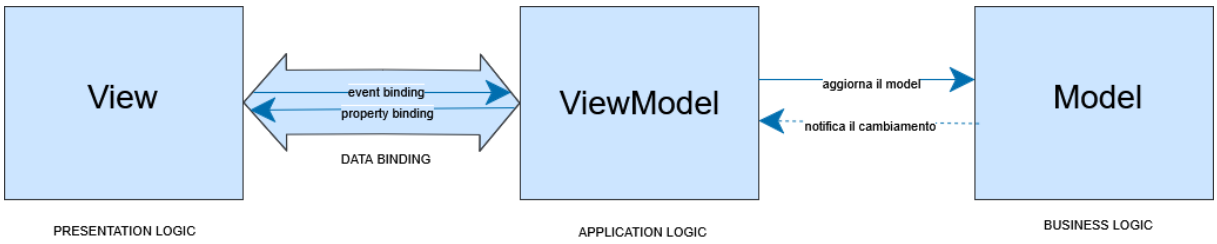
L'architettura frontend$_G$ è organizzata in modo modulare, separando i componenti UI dai servizi di comunicazione con il backend$_G$ e dagli altri servizi di utilità.

Per semplicità e manutenibilità$_G$ i dati principali (risultati di generazione, risultati di estrazione e riconoscimento) sono modellati tramite *interface*. Un chiaro vantaggio di raccogliere i dati in queste strutture è il passaggio di oggetti semplici e leggeri tra i componenti e la manutenibilità$_G$ del codice: se in futuro si volessero aggiungere campi o modificare la struttura dei dati, basterebbe aggiornare l'interfaccia$_G$ senza dover modificare il passaggio dei dati tra i componenti. L'uso di interface invece di classi è una scelta consapevole. Innanzitutto, le interface scompaiono completamente durante la compilazione a JavaScript, risparmiando spazio e accelerando il download e il parsing iniziale.
In secondo luogo, i modelli del progetto$_G$ contengono esclusivamente dati puri senza alcuna logica di business, validazione$_G$ o comportamento: una classe aggiungerebbe solo complessità inutile e overhead di istanziazione. Questa scelta riflette una chiara separazione delle responsabilità: i modelli rimangono semplici contenitori di dati, mentre la validazione$_G$ e la trasformazione rimangono delegate ai servizi e serializer appropriati.

6.2 Il pattern MVVM

Nonostante Angular non obblighi rigidamente l'utilizzo del pattern Model-View-ViewModel (MVVM), questo è riconosciuto universalmente come l'approccio migliore ed è quello che è stato adottato nel progetto.
Infatti, anche se tecnicamente è possibile implementare un'architettura di tipo MVP, il pattern MVVM offre il vantaggio di una chiara separazione tra *presentation logic* e *application logic*, garantito dal *data binding* (si veda la Sezione 6.2.2). Qui di seguito è riportata un'illustrazione

dell'architettura MVVM e di come avviene la comunicazione tra i vari componenti del pattern architetturale.



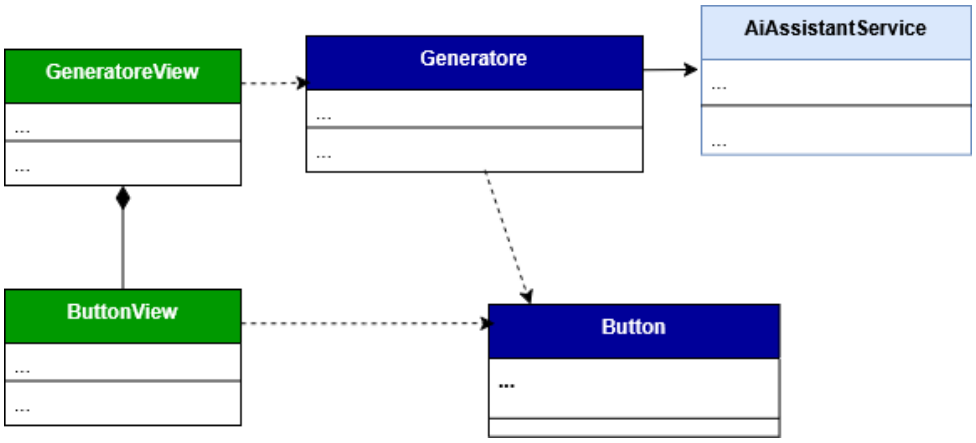
- **Model:** Il Model rappresenta i dati e la logica di business dell'applicazione. In Angular, questo è spesso rappresentato da servizi o classi che gestiscono i dati, eseguono operazioni e interagiscono con le API (per esempio tramite chiamate http o connessioni websocket). Il Model si occupa di recuperare, aggiornare e manipolare i dati.
- **View:** I template Angular fungono da View, visualizzando i dati e raccogliendo l'input dell'utente. Il data binding (particolari direttive di attributo o di evento) e altri comandi (come quelli che iniziano con '@') vengono utilizzati per collegare la View al ViewModel.
- **ViewModel:** Nel contesto di Angular, il ViewModel è spesso rappresentato dai Componenti e dalle relative classi TypeScript. La classe del Componente contiene la logica applicativa e interagisce con i servizi (quindi con il model) per recuperare e manipolare i dati. Espone proprietà e metodi a cui la View può collegarsi tramite data binding, consentendo un flusso di dati continuo tra ViewModel (Componente) e View (template).

Corrispondenza nel progetto

- **Model:** Sono tutte le classi Typerscript che si trovano dentro le cartelle: `services` (servizi come AiAssistantService e AiCopilotService), `serializers` (che serializzano e deserializzano dati di risposte e chiamate al backend) e `models` (che rappresentano i *DTO*, si veda Sezione 6.3)
- **View:** Sono i template HTML e i file di stile associati di ogni componente UI (dentro la cartella `components` oppure nelle singole cartelle che rappresentano le intere pagine)
- **ViewModel:** Sono tutte le relative classi Typescript delle componenti UI. Esse reagiscono agli eventi della View (tramite data-binding) e comunicano con i servizi per aggiornare i dati del model.

6.2.1 Esempio di pattern MVVM nel progetto

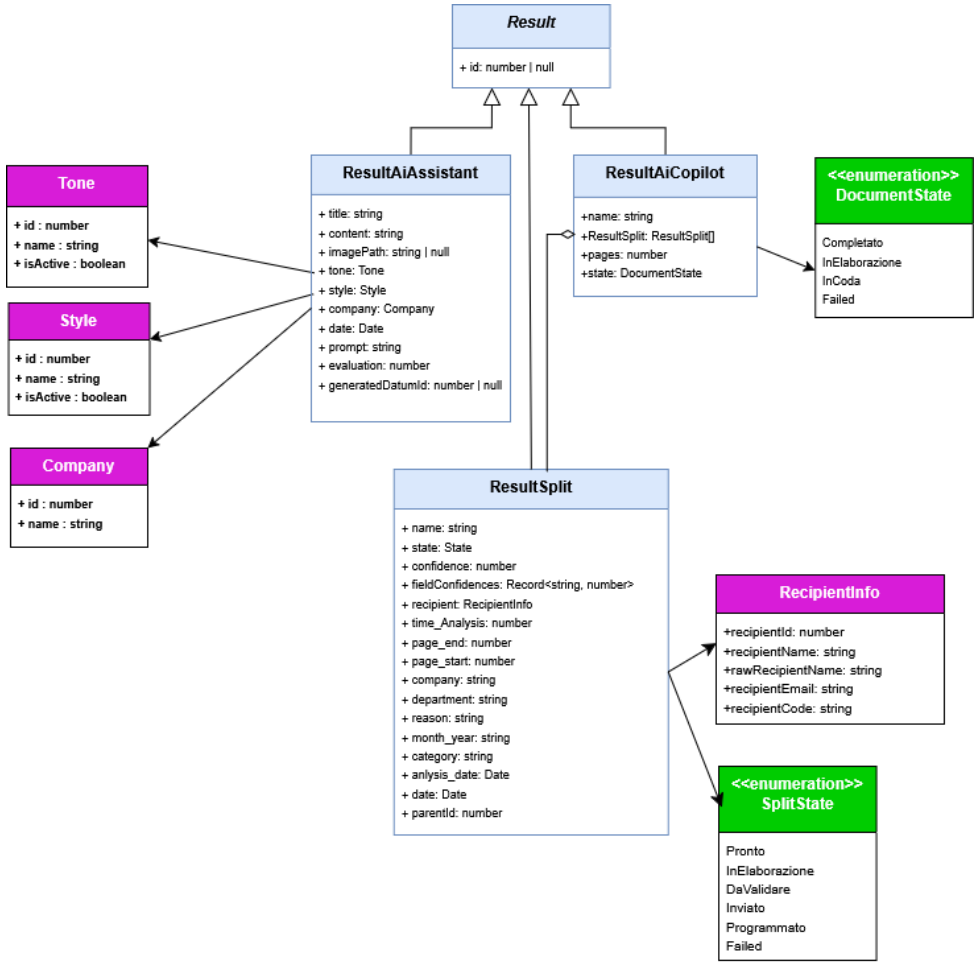
L'immagine in figura 6.2.1 rappresenta un diagramma a classi UML che vuole illustrare il pattern MVVM nel progetto.
Prende l'esempio della pagina di generazione e solamente una delle componenti riutilizzabili UI, perchè ai fini della dimostrazione l'illustrazione è stata semplificata.
La classe `GeneratoreView` (che rappresenta la View) utilizza proprietà e metodi del componente `Generatore` e si aggiorna tramite *property binding*. La classe `Generatore` (che rappresenta il ViewModel) risponde agli eventi della View (tramite event binding) e modifica di conseguenza il Model chiamando le operazioni esposte da `AiAssistantService`.
Per completezza si mostra anche la relazione con la sotto-componente `Button` (componente riutilizzabile) ma il ragionamento è il medesimo con qualsiasi altra componente. `Generatore` importa il pulsante nei metadata e lo usa come child-component, creando un collegamento di @Input @Output.
Invece `GeneratoreView` è composto dal template HTML del pulsante.



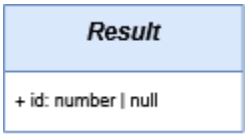
6.2.2 *Two way data binding*

Se il *ViewModel* è progettato correttamente, le modifiche apportate ad esso si riflettono **automaticamente** alla *View*, e viceversa. I due infatti sono strettamente collegati dal *data binding*. Modellare un MVP sarebbe tecnicamente possibile, ma comporterebbe a mescolare *presentation logic* e *application logic*, perché il *Presenter* dovrebbe gestire anche la logica di aggiornamento della *View* (per esempio agendo direttamente sul DOM, cosa possibile perché nativamente il ViewModel ha un riferimento diretto alla View).
Ciò andrebbe contro le *best practices* di Angular che offre nativamente il *two-way data binding* gestito, sotto al cofano, da due pattern observer: uno sul *ViewModel* e uno sulla *View*.

6.3 Definizione dei modelli di dati



6.3.1 Result



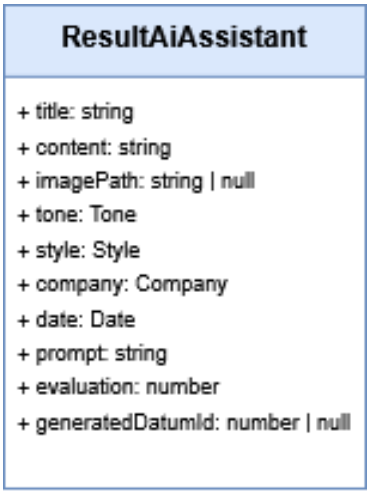
Interface che rappresenta un singolo risultato di estrazione o generazione, definisce il campio *id* comune a tutti i risultati.

Campi

- id: number | null - Identificatore univoco del risultato, può essere null in caso di risultati non ancora persistiti o generati.

Nota: nel diagramma UML è rappresentata come una classe astratta per coerenza con la notazione del modello, ma nel codice TypeScript$_G$ viene implementata come interface, perché per natura del linguaggio e per le esigenze del progetto$_G$ è sufficiente definire la struttura dei dati senza introdurre una classe reale o comportamento astratto.

6.3.2 ResultAiAssistant



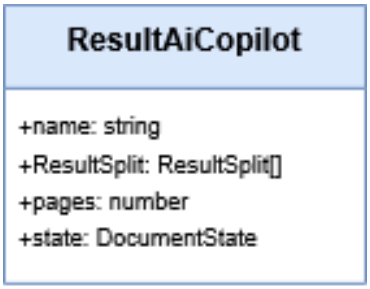
Interface che estende *Result* e rappresenta un risultato generato dall'assistente AI$_G$.

Campi

- id: number | null - Identificatore univoco del post ereditato da Result, può essere null se la generazione non è un post.
- title: string - Titolo o breve descrizione del risultato generato.
- content: string - Contenuto completo del risultato generato dall'assistente AI$_G$.
- imagePath: string | null - Percorso dell'immagine associata al risultato, se presente.
- tone: Tone - Tono utilizzato per generare il risultato (es. formale, amichevole, tecnico).
- style: Style - Stile di scrittura utilizzato per il risultato (es. procedurale, dettagliato, narrativo).
- company: Company - Azienda associata al risultato.
- date: Date - Data e ora in cui il risultato è stato generato.
- prompt: string - Prompt$_G$ di input utilizzato per generare il risultato.
- evaluation: number - Valutazione numerica fornita dall'utente per il risultato (es. da 1 a 5 stelle).
- generatedDatumId: number | null - Identifica l'id di un risultato che non è necessariamente anche un post.

Nota: nel diagramma UML è rappresentata come una classe per coerenza con la notazione del modello, ma nel codice TypeScript$_G$ viene implementata come interface, perché per natura del linguaggio e per le esigenze del progetto$_G$ è sufficiente definire la struttura dei dati senza introdurre una classe reale.

6.3.3 ResultAiCopilot



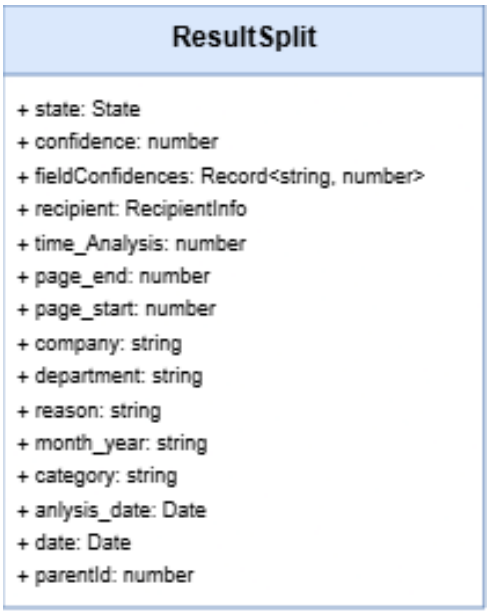
Interface che estende *Result* e rappresenta un risultato di estrazione dati da un documento.

Campi

- id: number | null - Identificatore univoco ereditato da Result; può essere null nelle fasi iniziali o in assenza di persistenza.
- name: string - Nome logico del documento padre caricato.
- ResultSplit: ResultSplit[] - Collezione dei documenti estratti/segmentati associati al documento padre.
- pages: number - Numero totale di pagine del documento padre.
- state: DocumentState - Stato complessivo del documento padre nel flusso di elaborazione.

Nota: nel diagramma UML può essere rappresentata come entità di dominio, ma nel codice TypeScript$_G$ è implementata come interface, in modo da descrivere solo la struttura dati senza introdurre comportamento applicativo.

6.3.4 ResultSplit



Interface che estende *Result* e rappresenta un risultato di divisione di un documento in più parti.

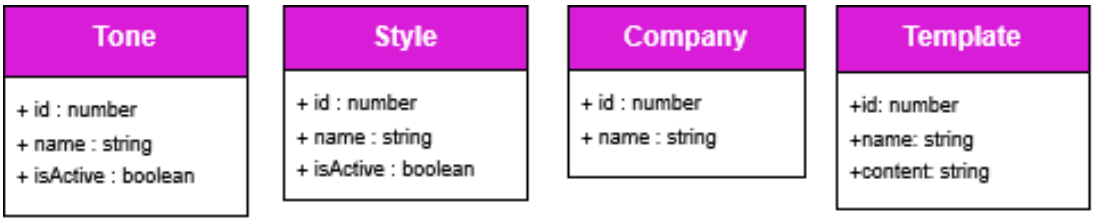
Campi

- id: number | null - Identificatore univoco ereditato da Result per il documento estratto.
- state: State - Stato corrente del documento estratto nel workflow$_G$ operativo.
- confidence: number - Confidenza$_G$ complessiva dell'estrazione/analisi.
- fieldConfidences: Record<string, number> - Confidenze puntuali per singolo campo estratto.
- recipient: RecipientInfo - Informazioni del destinatario associato (id, nome normalizzato, nome raw, email, codice).
- time_Analysis: number - Tempo impiegato per l'analisi del documento estratto.
- page_start: number - Pagina iniziale dell'intervallo analizzato.
- page_end: number - Pagina finale dell'intervallo analizzato.
- company: string - Azienda associata al documento.
- department: string - Reparto associato al documento.
- reason: string - Causale/motivazione estratta dal contenuto.
- month_year: string - Competenza temporale (mese/anno) associata al documento.
- category: string - Categoria documentale assegnata.
- analysis_date: Date - Data di creazione del record estratto.
- date: Date - Data interna al documento, se disponibile.
- parentId: number - Identificatore del documento padre (ResultAiCopilot) da cui deriva l'estratto.

Nota: anche questa entità è modellata come interface TypeScript$_G$; rappresenta un contratto dati usato da service e componenti senza comportamento interno.

6.3.5 Altre strutture di supporto

6.3.5.1 Tone, Style, Company, Template

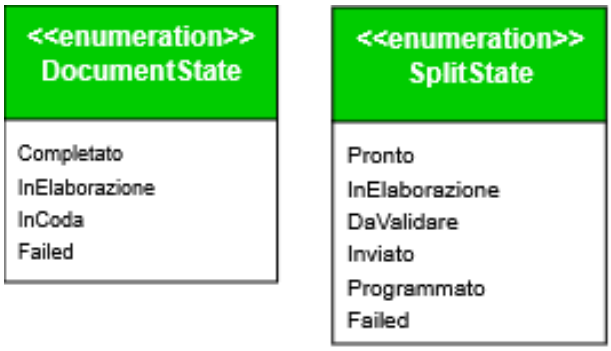


Interface usate per lo stesso motivo dei result, e in particolare per modellare le opzioni di toni, stili, aziende e template$_G$.

Campi presenti nelle opzioni

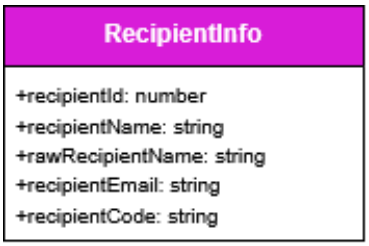
- id: number - Identificatore univoco dell'opzione.
- name: string - Nome visualizzato dell'opzione.
- isActive: boolean - Indica se l'opzione è attiva o è stata eliminata. Presente per permettere all'utente di visualizzare anche le opzioni eliminate.
- content: string - Contenuto associato all'opzione, se necessario (es. messaggio del template$_G$).

6.3.5.2 DocumentState e SplitState



Enum usati per definire gli stati che i documenti processati e estratti possono assumere.

6.3.5.3 RecipientInfo

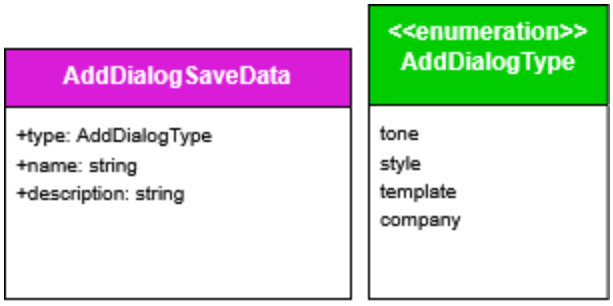


Interface che rappresenta le informazioni del destinatario associate a un documento estratto, usata principalmente per visualizzare i dati del destinatario in interfaccia$_G$

- recepientId: number - Id del destinatario nel database
- recepientName: string - Nome del destinatario trovato nel db
- recepientNameRaw: string - Nome del destinatario estratto dal documento
- recepientEmail: string - Email del destinatario trovato nel db
- recepientCode: string - Codice fiscale trovato nel db

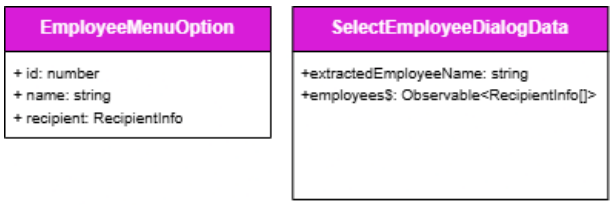
6.3.6 AddDialogSaveData e AddDialogType

Interface e *Enum* usati per gestire i dati e i tipi di dialog di salvataggio dei risultati generati dall'assistente AI$_G$.



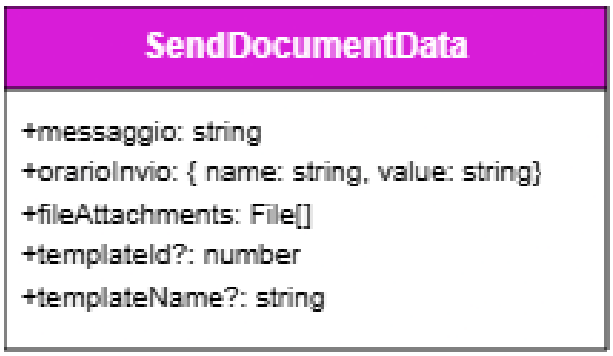
6.3.7 EmployeeMenuOption e SelectEmployeeDialogData

Interface usate per gestire i dati e i tipi di dialog di selezione dei dipendenti.



6.3.8 SendDocumentData

Interface usata per gestire i dati relativi all'invio di documenti.

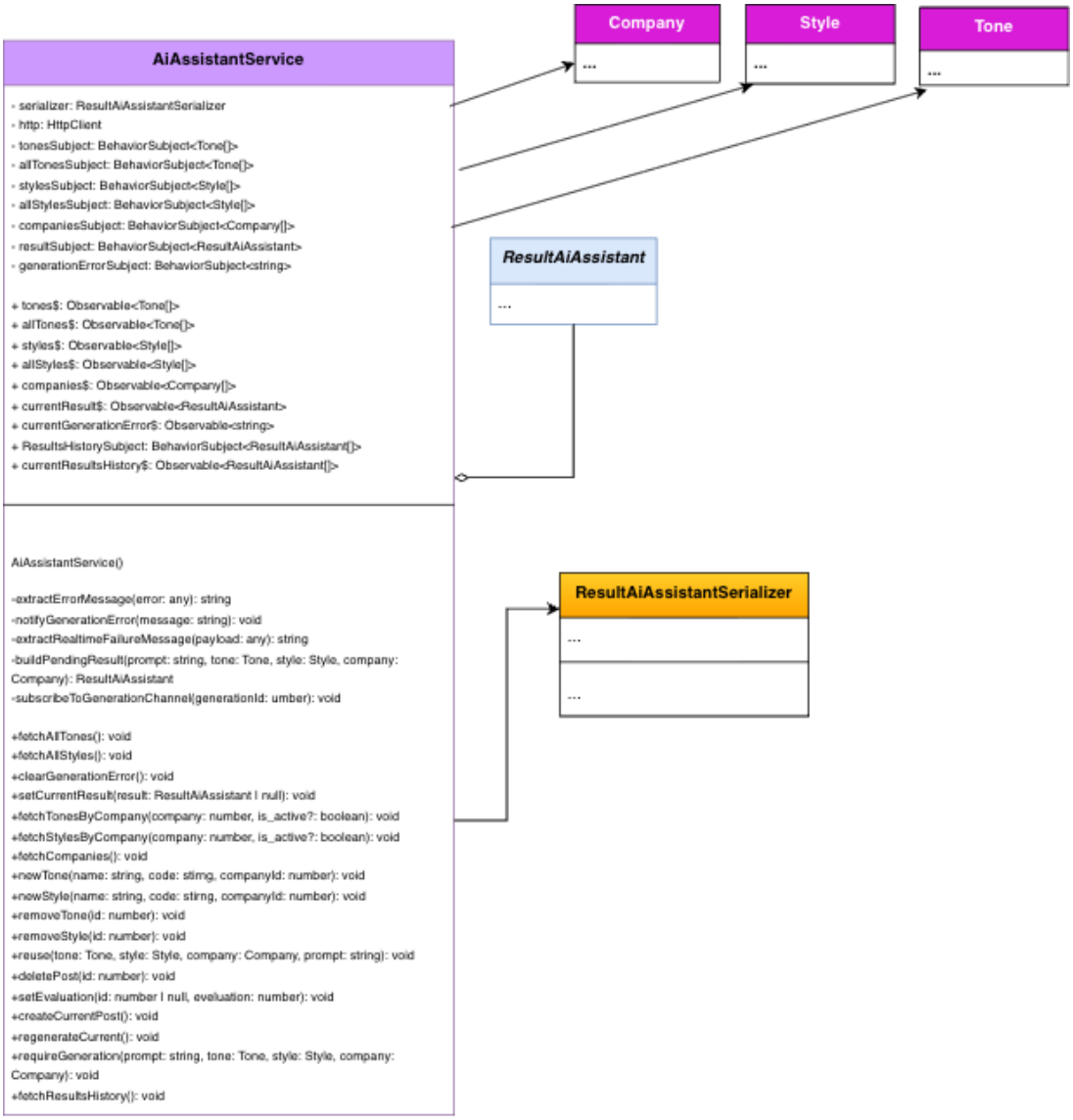


6.4 Servizi di comunicazione con il backend$_G$

Costanti di configurazione usate nei servizi:

- API_BASE: string - URL base delle API$_G$ HTTP backend$_G$.
- WS_URL: string - URL del canale WebSocket ActionCable.

6.4.1 AiAssistantService



È il servizio frontend$_G$ che centralizza l'integrazione con l'assistente AI$_G$: gestisce invio di richieste al backend$_G$ e ricezione delle risposte (incluse eventuali informazioni di stato/errore). Utilizza il wrapper HttpClient di Angular$_G$ per la comunicazione HTTP mentre usa le API$_G$ native dei browser per la comunicazione websocket.

Attributi

- Comunicazione con backend$_G$: gestisce la serializzazione dei dati e l'invio delle richieste HTTP verso il backend$_G$.
  - serializer: AiAssistantSerializer
  - http: HttpClient
- Observable delle opzioni readonly: espongono al frontend$_G$ gli aggiornamenti delle opzioni selezionabili, raggruppando tutte le varianti di tono, stile e azienda.
  - tones$: Observable<Tone[]>
  - allTones$: Observable<Tone[]>
  - styles$: Observable<Style[]>
  - allStyles$: Observable<Style[]>

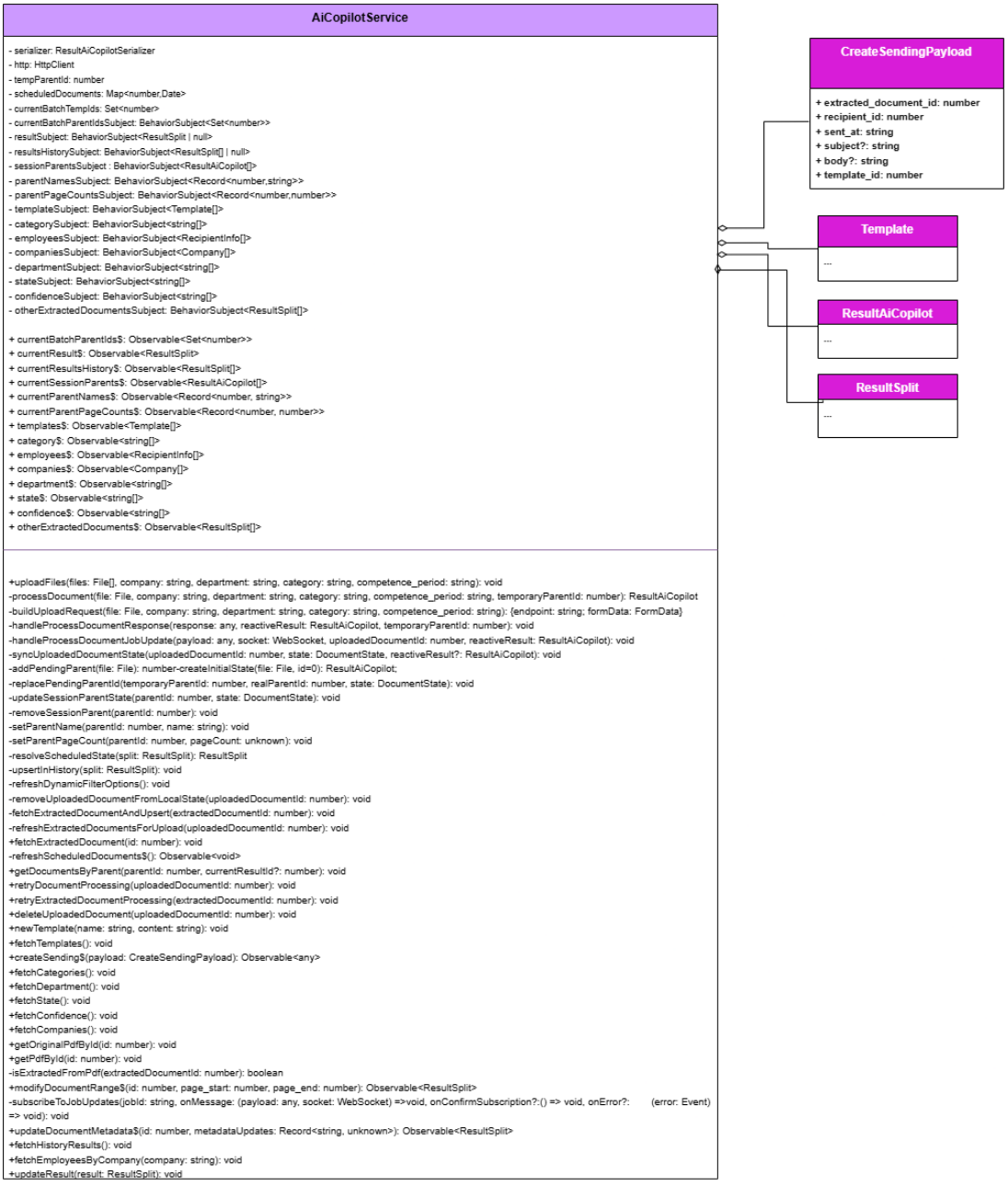
  - companies$: Observable<Company[]>
- Subject delle opzioni: mantengono lo stato interno delle opzioni prima della loro esposizione tramite gli observable corrispondenti.
  - tonesSubject: BehaviorSubject<Tone[]>
  - allTonesSubject: BehaviorSubject<Tone[]>
  - stylesSubject: BehaviorSubject<Style[]>
  - allStylesSubject: BehaviorSubject<Style[]>
  - companiesSubject: BehaviorSubject<Company[]>
- Stato del risultato corrente: conserva e pubblica il risultato generato dall'assistente AI$_G$ attualmente visualizzato.
  - resultSubject: BehaviorSubject<ResultAiAssistant>
  - result$: Observable<ResultAiAssistant>
- Stato degli errori di generazione: conserva e pubblica l'ultimo errore verificatosi durante la generazione del risultato.
  - generationErrorSubject: BehaviorSubject<string | null>
  - currentGenerationError$: Observable<string | null>
- Cronologia dei risultati: mantiene l'elenco dei risultati già generati e lo rende disponibile ai componenti sottoscritti.
  - resultsHistorySubject: BehaviorSubject<ResultAiAssistant[]>
  - resultsHistory$: Observable<ResultAiAssistant[]>

Metodi

- Gestione degli errori: normalizza e pubblica i messaggi di errore provenienti da HTTP e WebSocket.
  - extractErrorMessage(error: any): string
  - extractRealtimeFailureMessage(payload: any): string
  - notifyGenerationError(message: string): void
  - clearGenerationError(): void
- Gestione dello stato corrente: aggiorna il risultato AI$_G$ attualmente visualizzato.
  - setCurrentResult(result: ResultAiAssistant | null): void
  - buildPendingResult(prompt: string, tone: Tone, style: Style, company: Company): ResultAiAssistant
- Recupero delle opzioni: carica da backend$_G$ toni, stili e aziende disponibili.
  - fetchAllTones(): void
  - fetchAllStyles(): void
  - fetchTonesByCompany(company: number, is_active?: boolean): void
  - fetchStylesByCompany(company: number, is_active?: boolean): void
  - fetchCompanies(): void
- Gestione dei toni e degli stili: creazione e rimozione.
  - newTone(name: string, code: string, companyId: number): void
  - newStyle(name: string, code: string, companyId: number): void
  - removeTone(id: number): void
  - removeStyle(id: number): void
- Generazione AI$_G$: avvia e gestisce il flusso di generazione/rigenerazione del contenuto.
  - reuse(tone: Tone, style: Style, company: Company, prompt: string): void
  - requireGeneration(prompt: string, tone: Tone, style: Style, company: Company): void
  - regenerateCurrent(): void
  - subscribeToGenerationChannel(generationId: number): void
- Gestione post e storico: salva, elimina e recupera i risultati generati.
  - createCurrentPost(): void
  - deletePost(id: number): void
  - fetchResultsHistory(): void

- Valutazione del risultato: invia la valutazione del contenuto generato al backend$_G$.
  - setEvaluation(id: number | null, evaluation: number): void

6.4.2 AiCopilotService



Attributi

- Comunicazione con backend$_G$: gestisce la serializzazione dei dati e l'invio delle richieste HTTP verso il backend$_G$.
  - http: HttpClient . Client HTTP Angular$_G$ per chiamate REST.
  - serializer: ResultAiCopilotSerializer . Conversione payload backend$_G$
- **Attributi di stato upload/sessione**
  - tempParentId: number Contatore per ID temporanei negativi dei documenti in coda.
  - scheduledDocuments: Map<number, Date> Mappa documento estratto → data di invio programmata.

  - currentBatchTempIds: Set<number> ID temporanei della batch corrente.
  - currentBatchParentIdsSubject: BehaviorSubject<Set<number>> Stato interno ID parent (temporanei/reali).
  - currentBatchParentIds$: Observable<Set<number>> Observable readonly degli ID parent della batch.
  - sessionParentsSubject: BehaviorSubject<ResultAiCopilot[]> Stato interno documenti parent caricati in sessione.
  - currentSessionParents$: Observable<ResultAiCopilot[]> Observable readonly dei parent della sessione.
  - parentNamesSubject: BehaviorSubject<Record<number, string>> Subject dei nomi dei parent per ID.
  - currentParentNames$: Observable<Record<number, string>> Observable readonly dei nomi parent.
  - parentPageCountsSubject: BehaviorSubject<Record<number, number>> Subject del numero di pagine del parent per ID.
  - currentParentPageCounts$: Observable<Record<number, number>> Observable readonly conteggio pagine parent.
- **Attributi di stato risultati/documenti**
  - resultSubject: BehaviorSubject<ResultSplit | null> Stato del documento estratto corrente.
  - currentResult$: Observable<ResultSplit | null> Observable readonly del risultato corrente.
  - resultsHistorySubject: BehaviorSubject<ResultSplit[] | null> Storico risultati estratti in memoria frontend$_G$.
  - currentResultsHistory$: Observable<ResultSplit[] | null> Observable readonly della history risultati.
  - otherExtractedDocumentsSubject: BehaviorSubject<ResultSplit[]> Stato interno documenti fratelli del parent corrente.
  - otherExtractedDocuments$: Observable<ResultSplit[]> Observable readonly dei documenti fratelli.
- **Attributi di lookup/filtri/template**
  - templatesSubject: BehaviorSubject<TemplateOption[]> Stato interno template$_G$.
  - templates$: Observable<TemplateOption[]> Observable readonly template$_G$.
  - categorySubject: BehaviorSubject<string[]> Stato interno categorie filtro.
  - category$: Observable<string[]> Observable readonly categorie.
  - employeesSubject: BehaviorSubject<RecipientInfo[]> Stato interno destinatari.
  - employees$: Observable<RecipientInfo[]> Observable readonly destinatari.
  - companiesSubject: BehaviorSubject<Company[]> Stato interno aziende.
  - companies$: Observable<Company[]> Observable readonly aziende.
  - departmentSubject: BehaviorSubject<string[]> Stato interno reparti filtro.
  - department$: Observable<string[]> Observable readonly reparti.
  - stateSubject: BehaviorSubject<string[]> Stato interno opzioni stato documento.
  - state$: Observable<string[]> Observable readonly stati.
  - confidenceSubject: BehaviorSubject<string[]> Stato interno classi di confidenza$_G$.
  - confidence$: Observable<string[]> Observable readonly confidenza$_G$.

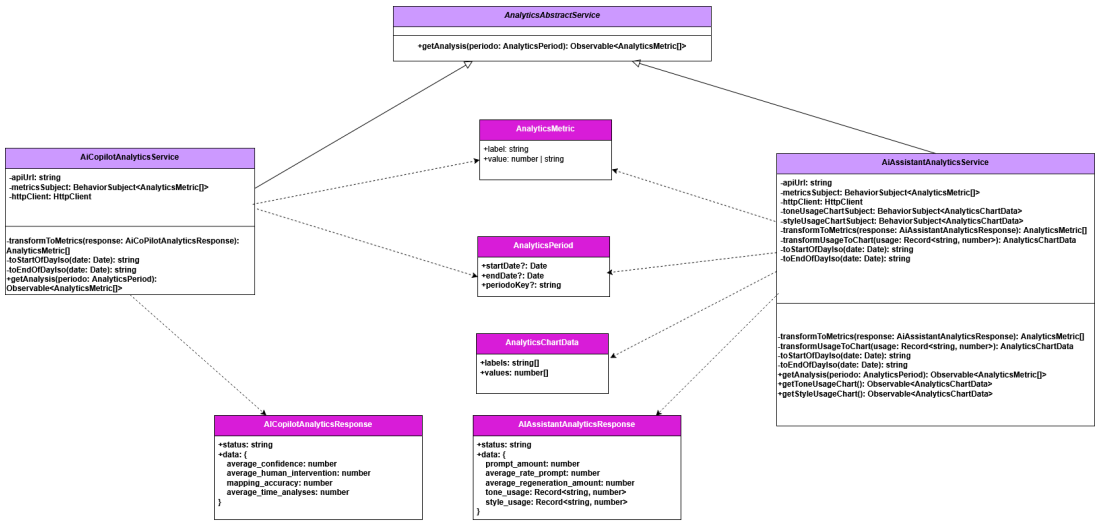
Metodi

- **Gestione upload e orchestrazione processing**
  - uploadFiles(files: File[], company: string, department: string, category: string, competence_period: string): void Avvia il caricamento batch, crea parent temporanei in coda e delega il processing per file.
  - processDocument(file: File, company: string, department: string, category: string, competence_period: string, temporaryParentId: number): ResultAiCopilot Si occupa di delegare le operazioni per creare lo stato iniziale, costruire la request e delegare la gestine della

risposta.
  - buildUploadRequest(file: File, company: string, department: string, category: string, competence_period: string): endpoint: string, formData: FormData Decide endpoint e in base al tipo di file processa o fa spit.
  - handleProcessDocumentResponse(response: any, reactiveResult: ResultAiCopilot, temporaryParentId: number): void Gestisce la risposta http iniziale, imposta l'id reale del documento, o decide se il job e completato o ava aperta la subscription per il websocker.
  - handleProcessDocumentJobUpdate(payload: any, socker: WebSocket, uplodedDocumentId: number, reactiveResult: ResultAiCopilot): void Gestisce gli eventi websocket
  - syncUplodedDocumentState(uploadedDocumentId: number, state: DocumentState, reactiveResult?: ResultAiCopilot): void Aggiorna lo stato del risultato locale e storico.
  - subscribeToJobUpdates(jobId: string, onMessage: (payload: any, socket: WebSocket) void, onConfirmSubscription?: () → void, onError?: (error: Event) → void): void Centralizza la sottoscrizione ActionCable per ricevere aggiornamenti del job.
- **Gestione stato parent e mapping locale**
  - addPendingParent(file: File): number Crea un parent temporaneo con stato In Coda e lo aggiunge alla sessione.
  - replacePendingParentId(temporaryParentId: number, realParentId: number, state: DocumentState): void Sostituisce ID temporaneo con ID reale backend$_G$ e aggiorna mappe correlate.
  - updateSessionParentState(parentId: number, state: DocumentState): void Aggiorna lo stato del parent nella lista sessione.
  - removeSessionParent(parentId: number): void Rimuove un parent dalla sessione (tipicamente in caso di errore upload).
  - setParentName(parentId: number, name: string): void Imposta/aggiorna il nome del documento parent.
  - setParentPageCount(parentId: number, pageCount: unknown): void Normalizza e salva il numero pagine del parent.
  - removeUploadedDocumentFromLocalState(uploadedDocumentId: number): void Elimina da stato locale parent, figli e metadati correlati senza refresh pagina.
- **Gestione risultati estratti e history**
  - resolveScheduledState(split: ResultSplit): ResultSplit Converte lo stato Inviato in Programmato se presente un invio futuro in mappa.
  - upsertInHistory(split: ResultSplit): void Inserisce o aggiorna un risultato nella history mantenendo coerenza di stato.
  - refreshDynamicFilterOptions(): void Rigenera opzioni dinamiche di filtro (categorie e reparti).
  - fetchExtractedDocumentAndUpsert(extractedDocumentId: number): void Recupera un estratto dal backend$_G$ e lo sincronizza in history.
  - refreshExtractedDocumentsForUpload(uploadedDocumentId: number): void Ricarica tutti gli estratti di un parent e aggiorna metadati parent.
  - fetchExtractedDocument(id: number): void Recupera il dettaglio di un estratto dopo sincronizzazione degli invii programmati.
  - refreshScheduledDocuments$(): Observable<void> Aggiorna la mappa locale dei documenti programmati partendo da /sendings.
  - getDocumentsByParent(parentId: number, currentResultId?: number): void Carica i documenti fratelli (stesso parent), escludendo opzionalmente quello corrente.
  - fetchHistoryResults(): void Carica la history iniziale: upload, metadati parent ed estratti associati.
  - updateResult(result: ResultSplit): void Sincronizza il risultato corrente e lo persiste in history locale.

- **Retry, modifica e cancellazione documenti**
  - retryDocumentProcessing(uploadedDocumentId: number): void Riavvia il processamento di un documento parent già caricato.
  - retryExtractedDocumentProcessing(extractedDocumentId: number): void Riavvia la rianalisi di un singolo documento estratto.
  - deleteUploadedDocument(uploadedDocumentId: number): void Elimina un parent e i suoi figli dal backend$_G$ e dallo stato locale.
  - modifyDocumentRange$(id: number, page_start: number, page_end: number): Observable<ResultSplit> Richiede riassegnazione intervallo pagine, segue eventi async e restituisce il risultato aggiornato.
  - updateDocumentMetadata$(id: number, metadataUpdates: Record<string, unknown>): Observable<ResultSplit> Aggiorna metadati di un estratto e propaga il nuovo stato in UI e history.
- **Template, invio e lookup**
  - newTemplate(name: string, content: string): void Crea un nuovo template$_G$ e aggiorna lo stato locale template$_G$.
  - fetchTemplates(): void Recupera elenco template$_G$e dettagli completi di ciascun template$_G$.
  - $(payload: CreateSendingPayload): Observable<any> Crea un invio e registra eventuale scheduling futuro del documento.
  - fetchCategories(): void Estrae categorie uniche dalla history risultati.
  - fetchDepartment(): void Estrae reparti unici dalla history risultati.
  - fetchState(): void Popola i valori di filtro stato a partire dall'enum State.
  - fetchConfidence(): void Popola gli intervalli di confidenza$_G$ predefiniti.
  - fetchCompanies(): void Recupera le aziende disponibili dal backend$_G$.
  - fetchEmployeesByCompany(company: string): void Recupera utenti/destinatari filtrati per azienda.
- **Apertura documenti**
  - getOriginalPdfById(id: number): void Apre in nuova finestra il file originale del parent.
  - getPdfById(id: number): void Apre il PDF dell'estratto con cache-buster temporale per evitare versioni non aggiornate.

6.4.3 AiAnalyticsService



6.4.3.1 AiAssistantAnalyticsService

Attributi

- Configurazione endpoint backend$_G$: definisce l'URL per recuperare le metriche dell'assistente.
  - apiUrl: string
- Stato metriche principali: mantiene i KPI principali mostrati in UI.
  - metricsSubject: BehaviorSubject<AnalyticsMetric[]>
- Stato grafico uso toni: conserva labels e valori del grafico relativo ai toni.
  - toneUsageChartSubject: BehaviorSubject<AnalyticsChartData>
- Stato grafico uso stili: conserva labels e valori del grafico relativo agli stili.
  - styleUsageChartSubject: BehaviorSubject<AnalyticsChartData>
- Client HTTP: gestisce l'invocazione delle API REST.
  - httpClient: HttpClient

Metodi

- Recupero analytics: invia la richiesta HTTP con filtro temporale e aggiorna metriche e grafici.
  - getAnalysis(periodo: AnalyticsPeriod): Observable<AnalyticsMetric[]>
- Esposizione grafici: pubblica gli stream dei dati chart per toni e stili.
  - getToneUsageChart(): Observable<AnalyticsChartData>
  - getStyleUsageChart(): Observable<AnalyticsChartData>
- Mapping risposta backend$_G$: converte i KPI principali in metriche frontend$_G$.
  - transformToMetrics(response: AiAssistantAnalyticsResponse): AnalyticsMetric[]
- Conversione usage in chart: trasforma mappe key-value in labels/values.
  - transformUsageToChart(usage: Record<string, number>): AnalyticsChartData
- Normalizzazione date filtro: prepara i timestamp ISO per inizio e fine giornata.
  - toStartOfDayIso(date: Date): string
  - toEndOfDayIso(date: Date): string

6.4.3.2 AiCoPilotAnalyticsService

Attributi

- Configurazione endpoint backend$_G$: definisce l'URL per recuperare le metriche analytics del co-pilot.
  - apiUrl: string
- Stato metriche analytics: mantiene l'ultimo snapshot delle metriche da mostrare in UI.
  - metricsSubject: BehaviorSubject<AnalyticsMetric[]>
- Client HTTP: gestisce l'invocazione delle API REST.
  - httpClient: HttpClient

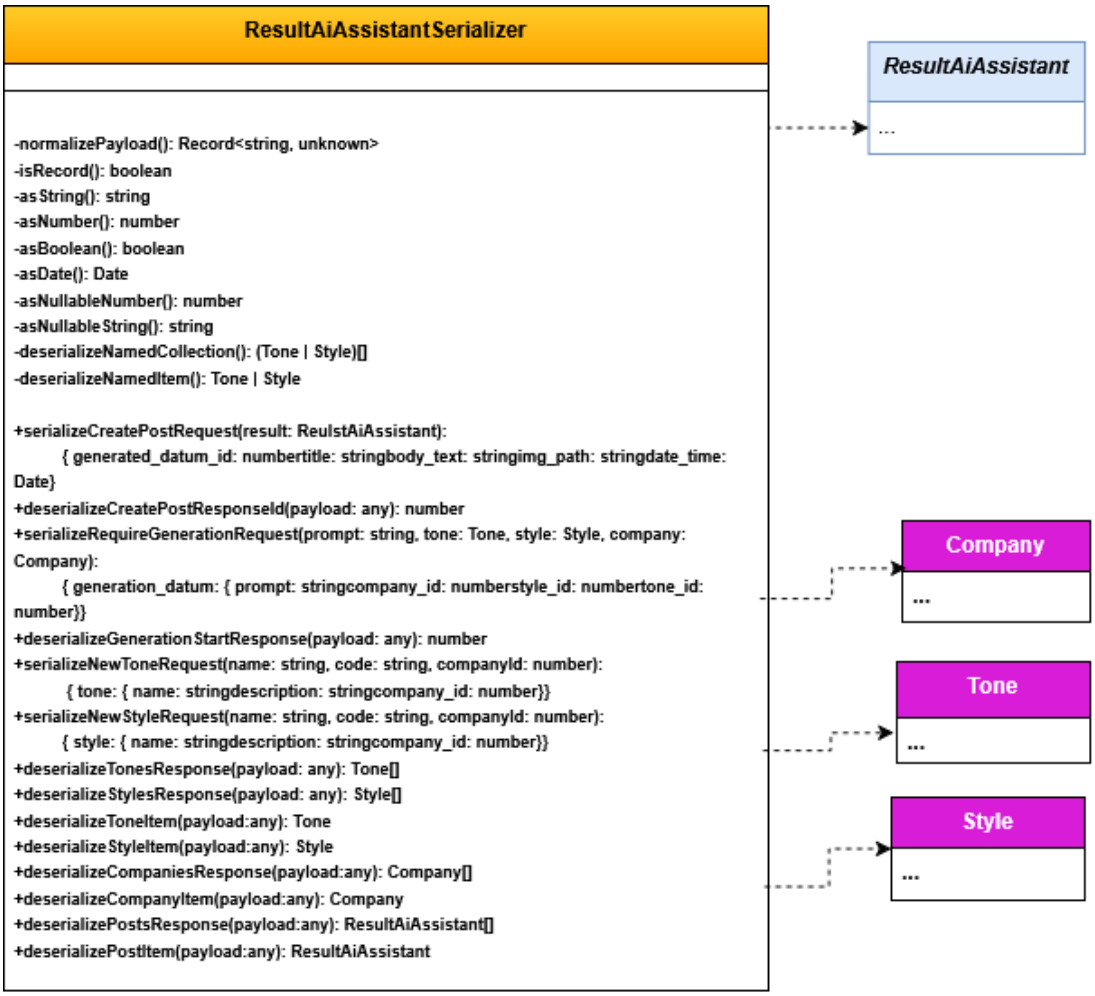
Metodi

- Recupero analytics: invia la richiesta HTTP con eventuale filtro temporale e aggiorna lo stream metriche.
  - getAnalysis(periodo: AnalyticsPeriod): Observable<AnalyticsMetric[]>
- Mapping risposta backend$_G$: converte il payload API in metriche leggibili dal frontend$_G$.
  - transformToMetrics(response: AiCoPilotAnalyticsResponse): AnalyticsMetric[]
- Normalizzazione date filtro: prepara i timestamp ISO per inizio e fine giornata.
  - toStartOfDayIso(date: Date): string
  - toEndOfDayIso(date: Date): string

6.5 Adapter: Serializzazione/deserializzazione dei dati

Benchè in Typescript$_G$ è semplice convertire dati strutturati json in interfacce, le classi sono state utilizzate per non dipendere dalla struttura esatta dei dati restituiti dal backend$_G$ e conformali alle esigenze del frontend$_G$.
Inoltre questi serializer fungono da punto centrale per la gestione di eventuali trasformazioni, normalizzazioni o arricchimenti dei dati, rispettando il principio di single responsibility e mantenendo il codice dei servizi di comunicazione più pulito e focalizzato sulla logica di comunicazione piuttosto che sulla manipolazione dei dati.

6.5.1 ResultAiAssistantSerializer

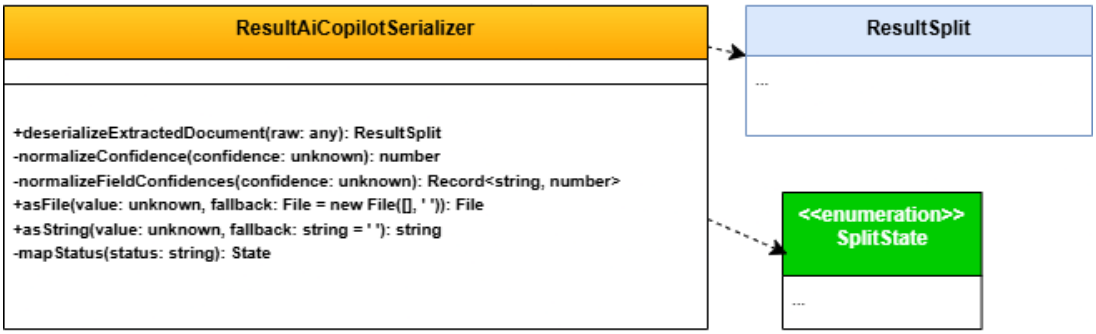


Questa classe funge da adapter per la serializzazione e deserializzazione dei dati relativi al modulo$_G$ dell'Ai Assistant Generativo. È responsabile di convertire i dati ricevuti dal backend$_G$ (in formato JSON) in valori di tipo ResultAiAssistant, Tone, Style o Company.
Metodi

- Serializzazione richieste verso backend$_G$.
  - serializeCreatePostRequest(result: ResultAiAssistant): { generated_datum_id: number | null; title: string; body_text: string; img_path: string | null; date_time: Date; }
  - serializeRequireGenerationRequest(prompt: string, tone: Tone, style: Style, company: Company): { generation_datum: { prompt: string; company_id: number; style_id: number; tone_id: number } }

  - serializeNewToneRequest(name: string, code: string, companyId: number): { tone: { name: string; description: string; company_id: number } }
  - serializeNewStyleRequest(name: string, code: string, companyId: number): { style: { name: string; description: string; company_id: number } }
- Deserializzazione risposte backend$_G$.
  - deserializeCreatePostResponseId(payload: any): number
  - deserializeGenerationStartResponse(payload: any): number
  - deserializeTonesResponse(payload: any): Tone[]
  - deserializeStylesResponse(payload: any): Style[]
  - deserializeToneItem(payload: any): Tone
  - deserializeStyleItem(payload: any): Style
  - deserializeCompaniesResponse(payload: any): Company[]
  - deserializeCompanyItem(payload: any): Company
  - deserializePostsResponse(payload: any): ResultAiAssistant[]
  - deserializePostItem(payload: any): ResultAiAssistant
- Metodi di supporto e validazione$_G$ interna (privati).
  - isRecord(value: unknown): value is Record<string, unknown>
  - asString(value: unknown, fallback = ‘’): string
  - asNumber(value: unknown, fallback = 0): number
  - asBoolean(value: unknown, defaultValue: boolean): boolean
  - asDate(value: unknown): Date
  - asNullableNumber(value: unknown): number | null
  - asNullableString(value: unknown): string | null
  - deserializeNamedCollection(payload: any, key: ’tones’ | ’styles’): Array<Tone | Style>
  - deserializeNamedItem(payload: any): Tone | Style

6.5.2 ResultAiCopilotSerializer

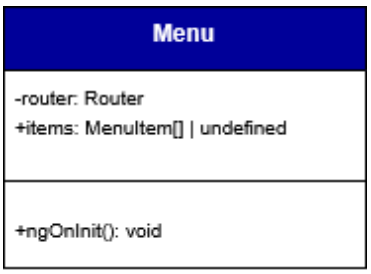


Metodi

- Deserializzazione risposte backend$_G$ (documenti estratti).
  - deserializeExtractedDocument(raw: any): ResultSplit
- Normalizzazione confidence e mapping dei campi (privati).
  - normalizeConfidence(confidence: unknown): number
  - normalizeFieldConfidences(confidence: unknown): Record<string, number>
- Metodi di supporto e conversione tipi.
  - asFile(value: unknown, fallback: File = new File([], ‘’)): File
  - asString(value: unknown, fallback: string = ‘’): string
- mapStatus(status: string): State Mapping stato backend$_G$ stato frontend$_G$.

6.6 Micro componenti UI

6.6.1 Menu



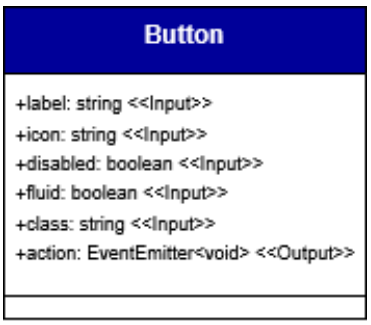
Componente UI che rappresenta il menù di navigazione laterale.
Attributi

- Servizi utilizzati dal componente per navigazione e gestione messaggi.
  - router: Router Servizio Angular$_G$ usato per leggere la rotta corrente ed effettuare navigazioni programmatiche.
- items: MenuItem[] | undefined Collezione delle sezioni e delle voci del menu, inizializzata nel ciclo di vita del componente.

Metodi

- ngOnInit(): void Inizializza l'array items con le sezioni principali del menu (AI Assistant$_G$ Generativo, AI Co-Pilot$_G$ per CDL$_G$, Analytics), configurando routerLink o command per ciascuna voce.

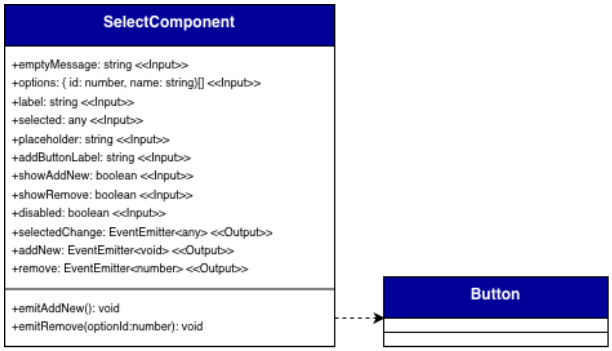
6.6.2 Button



Componente UI riutilizzabile che rappresenta un pulsante con etichetta e icona personalizzabili.
Attributi

- Attributi di input: configurano aspetto e comportamento del pulsante dal componente padre.
  - label: string
  - icon: string
  - disabled: boolean
  - fluid: boolean
  - class: string
- Attributi di output: espone l'evento emesso dal pulsante verso il componente padre.
  - action: EventEmitter<void>

6.6.3 SelectComponent



Componente UI che rappresenta un menu a tendina con ricerca e pulsante per aggiungere opzioni. Viene utilizzato per gestire le scelte di toni, stili le aziende o i template$_G$.

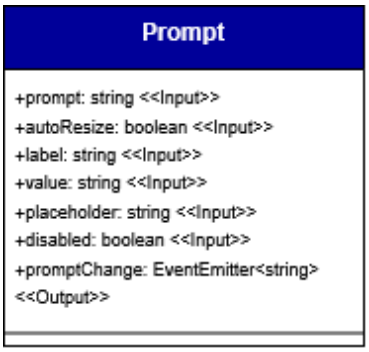
Attributi

- attributi di input: definiscono i dati mostrati e il comportamento del menu a tendina.
  - emptyMessage: string Testo mostrato quando non ci sono opzioni da visualizzare.
  - options: { id: number; name: string }[] | null | undefined Elenco delle opzioni selezionabili visualizzate nel menu.
  - label: string Etichetta testuale mostrata sopra la select.
  - selected: any Valore attualmente selezionato nel menu.
  - placeholder: string Testo mostrato quando nessuna opzione è stata selezionata.
  - addButtonLabel: string Etichetta del pulsante usato per aggiungere una nuova opzione.
  - showAddNew: boolean Abilita o nasconde il pulsante per inserire una nuova opzione.
  - showRemove: boolean Abilita o nasconde il pulsante per rimuovere un'opzione.
  - disabled: boolean Disabilita il menu rendendolo non modificabile e non interagibile.
- Attributi di output: comunicano al componente padre le azioni eseguite dall'utente.
  - selectedChange: EventEmitter<any> Notifica al componente padre quando cambia l'opzione selezionata.
  - addNew: EventEmitter<void> Segnala la richiesta di creazione di una nuova opzione.
  - remove: EventEmitter<number> Segnala la richiesta di eliminazione di un'opzione, passando il suo identificativo.

Metodi

- Gestione inserimento e rimozione opzione: emette l'evento di richiesta creazione o eliminazione elemento.
  - emitAddNew(): void
  - emitRemove(optionId: number): void

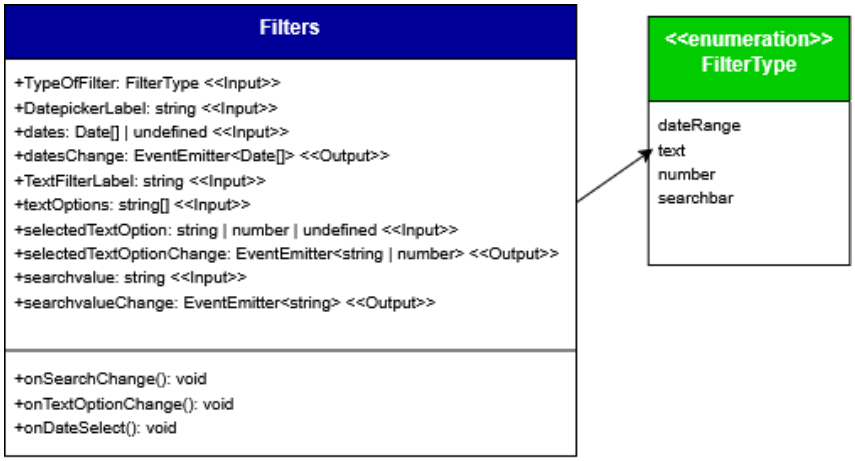
6.6.4 Prompt



Componente UI che rappresenta un'area di testo per l'inserimento del prompt$_G$ da inviare all'assistente AI$_G$, con funzionalità$_G$ di invio tramite pulsante o tasto.
Attributi

- Attributi di input: definiscono il contenuto e il comportamento del campo di testo.
  - prompt: string Contiene il testo del prompt$_G$ inserito o visualizzato nel campo.
  - autoResize: boolean Abilita o disabilita il ridimensionamento automatico dell'area di testo.
  - label: string Etichetta testuale mostrata sopra il campo di input.
  - value: string Valore corrente associato al contenuto del campo.
  - placeholder: string Testo mostrato quando il campo è vuoto.
  - disabled: boolean Disabilita il campo rendendolo non modificabile e non focalizzabile.
- Attributi di output: comunicano al componente padre le modifiche del testo inserito.
  - promptChange: EventEmitter<string> Emette il nuovo valore del prompt$_G$ quando il contenuto del campo cambia.

6.6.5 Filters



Componente UI che rappresenta dei semplici filtri.
Attributi

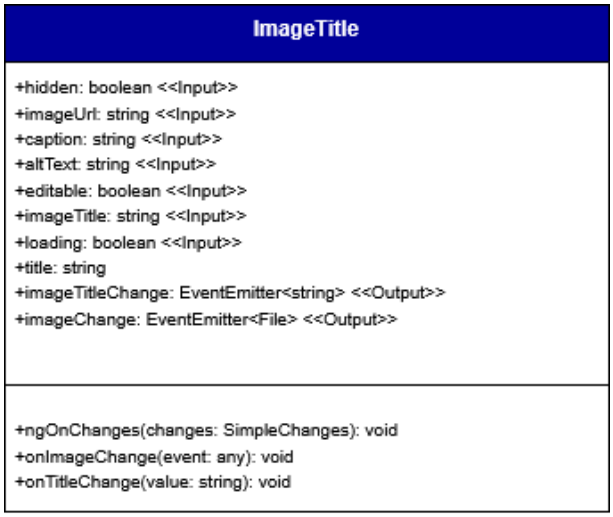
- Attributi di input: definiscono il tipo di filtro e i dati associati.
  - TypeOfFilter: FilterType Specifica il tipo di filtro da visualizzare: dateRange, text, number o searchbar.
  - DatepickerLabel: string Etichetta del filtro per selezione data.
  - dates: Date[] | undefined Array di date selezionate dal componente datepicker.

  - TextFilterLabel: string Etichetta del filtro per selezione testuale.
  - textOptions: string[] Elenco delle opzioni testuali disponibili per il filtro.
  - selectedTextOption: number | undefined | string Opzione testuale attualmente selezionata.
  - searchvalue: string Valore attuale inserito nel campo di ricerca.
- Attributi di output: comunicano al componente padre i cambiamenti dei filtri.
  - datesChange: EventEmitter<Date[]> Emette l'array di date quando cambia la selezione nel datepicker.
  - selectedTextOptionChange: EventEmitter<number | string> Emette l'opzione testuale selezionata quando cambia la scelta.
  - searchvalueChange: EventEmitter<string> Emette il valore di ricerca quando cambia il contenuto della searchbar.

Metodi

- Gestione cambiamenti filtri: notificano al componente padre le modifiche dei valori immesse dall'utente.
  - onSearchChange(): void Emette il valore corrente della ricerca quando cambia il contenuto della searchbar.
  - onTextOptionChange(): void Emette l'opzione testuale selezionata quando cambia la scelta.
  - onDateSelect(): void Emette l'array di date selezionate quando cambia la selezione nel datepicker.

6.6.6 ImageTitle



Componente UI che rappresenta immagine di generazione e titolo della generazione.

Attributi

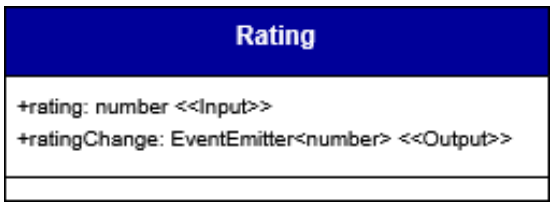
- Attributi di input: configurano l'aspetto e il comportamento del componente immagine con titolo.
  - hidden: boolean Nasconde o visualizza il componente.
  - imageUrl: string URL dell'immagine da visualizzare.
  - caption: string Didascalia testuale associata all'immagine.
  - altText: string Testo alternativo mostrato se l'immagine non carica.
  - editable: boolean Abilita la modalità di modifica per titolo e immagine.
  - imageTitle: string Titolo dell'immagine visualizzato e modificabile.
  - loading: boolean Mostra uno spinner di caricamento durante l'upload.
- Attributi di output: comunicano al componente padre i cambiamenti di titolo e immagine.

  - imageTitleChange: EventEmitter<string> Emette il nuovo titolo quando cambia il valore inserito.
  - imageChange: EventEmitter<File> Emette il file immagine selezionato quando viene caricato un nuovo file.
- Attributi interni: mantengono lo stato locale del componente.
  - title: string Titolo sull'immagine sincronizzato con l'input imageTitle.

Metodi

- Gestione ciclo di vita: sincronizza i cambiamenti degli input con lo stato interno.
  - ngOnChanges(changes: SimpleChanges): void Aggiorna il titolo interno quando cambia l'input imageTitle.
- Gestione modifica immagine e titolo: notificano al componente padre le variazioni.
  - onImageChange(event: any): void Estrae il file dall'evento di upload e lo emette verso il componente padre.
  - onTitleChange(value: string): void Aggiorna il titolo interno e notifica il componente padre della modifica.

6.6.7 Rating

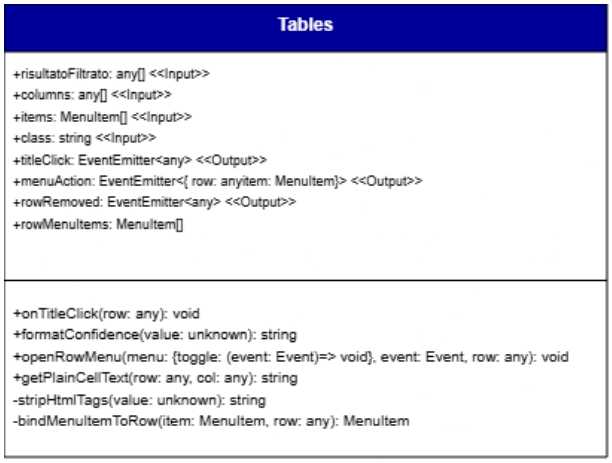


Componente UI che rappresenta un sistema di valutazione a stelle, utilizzato per valutare i risultati generati dall'assistente AI$_G$.

Attributi

- Attributi di input: configurano il valore iniziale e il comportamento del componente di valutazione.
  - rating: number Valore della valutazione corrente visualizzato dal componente.
- Attributi di output: comunicano al componente padre le variazioni della valutazione.
  - ratingChange: EventEmitter<number> Emette il nuovo valore selezionato dall'utente quando la valutazione cambia.

6.6.8 Tables



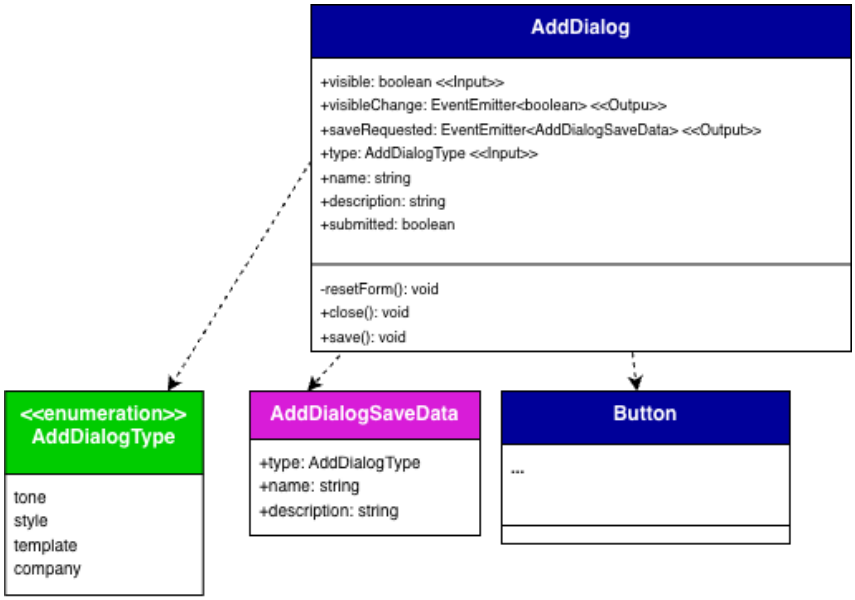
Attributi

- Attributi di input: configurano dati e comportamento della tabella.
  - risultatoFiltrato: any[] Array dei dati filtrati da visualizzare nelle righe della tabella.
  - columns: any[] Definisce la struttura e le colonne della tabella.
  - items: MenuItem[] Elenco delle voci di menu contestuale disponibili per ogni riga.
  - class: string Classi CSS aggiuntive da applicare al componente tabella.
- Attributi di output: comunicano al componente padre le interazioni sulla tabella.
  - titleClick: EventEmitter<any> Emette la riga quando l'utente clicca sul titolo.
  - menuAction: EventEmitter<{ row: any; item: MenuItem }> Emette la riga e la voce di menu selezionata quando viene eseguita un'azione dal menu contestuale.
  - rowRemoved: EventEmitter<any> Emette la riga eliminata quando viene selezionata l'azione di rimozione (retrocompatibilità).
  - rowMenuItems: MenuItem[] Voci di menu associate alla riga attualmente selezionata, sincronizzate con le condizioni della riga.

Metodi

- Gestione clic sulla tabella: notificano al componente padre le interazioni dell'utente.
  - onTitleClick(row: any): void Emette la riga quando viene cliccato il titolo.
- Gestione menu contestuale: sincronizzano le voci del menu con lo stato della riga.
  - openRowMenu(menu: { toggle: (event: Event) => void }, event: Event, row: any): void Prepara le voci di menu per la riga selezionata applicando le condizioni di visibilità e apre il menu.
  - bindMenuItemToRow(item: MenuItem, row: any): MenuItem Associa una voce di menu a una riga, gestendo visibilità condizionale e comandi personalizzati.
- Estrazione e normalizzazione testo: preparano il testo delle celle eliminando formattazione HTML.
  - getPlainCellText(row: any, col: any): string Estrae il valore testuale di una cella eliminando i tag HTML.
  - stripHtmlTags(value: unknown): string Rimuove tag HTML, entità HTML e normalizza gli spazi dal testo fornito.
- Gestione e formattazione confidenza$_G$.
  - formatConfidence(value: unknown): string Normalizza e formatta il valore di confidenza$_G$.

6.6.9 AddDialog



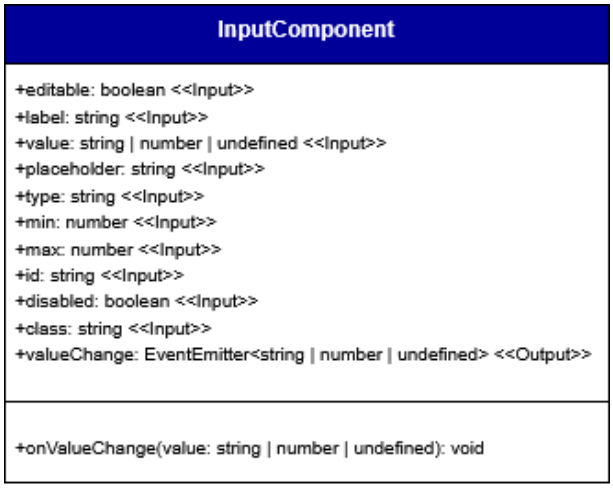
Attributi

- Attributi di input: configurano lo stato e il tipo di dialog.
  - visible: boolean Abilita o disabilita la visualizzazione del dialog.
  - type: AddDialogType Specifica il tipo di elemento da aggiungere: tone, style, template$_G$ o company.
- Attributi di output: comunicano al componente padre le azioni del dialog.
  - visibleChange: EventEmitter<boolean> Notifica il componente padre quando il dialog viene chiuso.
  - saveRequested: EventEmitter<AddDialogSaveData> Emette i dati inseriti (nome, descrizione e tipo) quando viene richiesto il salvataggio.
- Attributi interni: mantengono lo stato del form all'interno del dialog.
  - name: string Nome dell'elemento inserito nel campo di input.
  - description: string Descrizione dell'elemento inserita nel campo di input.
  - submitted: boolean Flag che traccia se il form è stato inviato, usato per la visualizzazione degli errori di validazione$_G$.

Metodi

- Attributi calcolati: determinano dinamicamente il titolo e l'etichetta del dialog in base al tipo.
  - get dialogTitle(): string Ritorna il titolo del dialog personalizzato secondo il tipo di elemento.
  - get saveLabel(): string Ritorna l'etichetta del pulsante di salvataggio personalizzata secondo il tipo di elemento.
- Gestione chiusura dialog: chiude il dialog e resetta il form interno.
  - close(): void Nasconde il dialog, notifica il componente padre e resetta i dati del form.
- Gestione salvataggio: valida e salva i dati inseriti.
  - save(): void Valida l'input, normalizza i dati (trim), emette l'evento di salvataggio verso il padre e chiude il dialog.
- Supporto interno: resetta lo stato del form.
  - resetForm(): void Reimposta i campi del form e il flag di invio a valori iniziali.

6.6.10 InputComponent



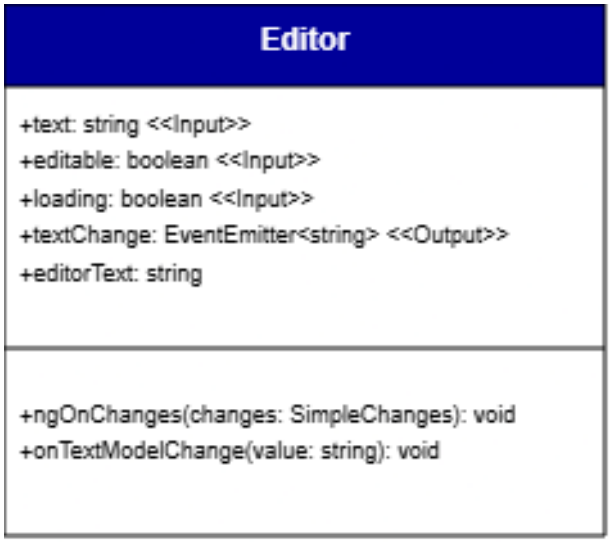
Attributi

- Attributi di input: configurano l'aspetto e il comportamento del campo di input.
  - editable: boolean Abilita o disabilita la modificabilità del campo.
  - label: string Etichetta testuale mostrata sopra il campo.
  - value: string | number | undefined Valore corrente contenuto nel campo.
  - placeholder: string Testo mostrato quando il campo è vuoto.
  - type: string Tipo di input HTML: text, number, email, password, ecc.
  - min: number Valore minimo accettato (rilevante per input di tipo numero).
  - max: number Valore massimo accettato (rilevante per input di tipo numero).
  - id: string Identificativo univoco del campo HTML.
  - disabled: boolean Disabilita il campo rendendolo non modificabile e non focalizzabile.
  - class: string Classi CSS aggiuntive da applicare al componente.
- Attributi di output: comunicano al componente padre le modifiche del valore.
  - valueChange: EventEmitter<string | number | undefined> Emette il nuovo valore quando il contenuto del campo cambia.

Metodi

- Gestione modifica valore: notifica il componente padre delle variazioni.
  - onValueChange(value: string | number | undefined): void Emette il nuovo valore verso il componente padre quando il campo viene modificato.

6.6.11 Editor



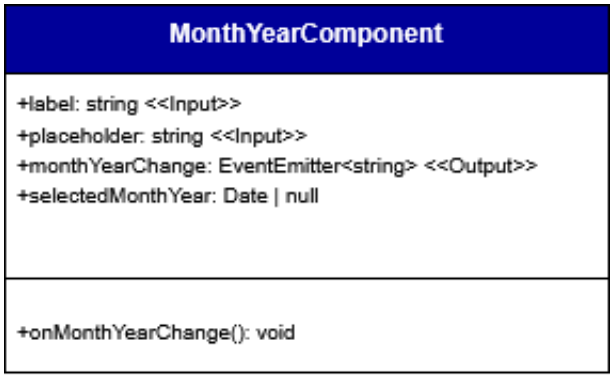
Attributi

- Attributi di input: configurano contenuto e stato dell'editor.
  - text: string Contenuto testuale mostrato nell'editor proveniente dal componente padre.
  - editable: boolean Abilita o disabilita la modifica del contenuto.
  - loading: boolean Indica se il componente è in stato di caricamento.
- Attributi di output: comunicano al componente padre le modifiche del contenuto.
  - textChange: EventEmitter<string> Emette il nuovo testo quando il contenuto dell'editor cambia.
- Attributi interni: mantengono lo stato locale del testo nell'editor.
  - editorText: string Copia locale del testo usata per sincronizzare il modello dell'editor.

Metodi

- Gestione ciclo di vita: sincronizza gli input con lo stato interno.
  - ngOnChanges(changes: SimpleChanges): void Aggiorna editorText quando cambia il valore dell'input text dal padre (es. quando arriva il risultato di una generazione).
- Gestione modifica testo: aggiorna il modello locale e notifica il componente padre.
  - onTextModelChange(value: string): void Se non è in loading, aggiorna editorText ed emette textChange con il nuovo contenuto.

6.6.12 MonthYearComponent



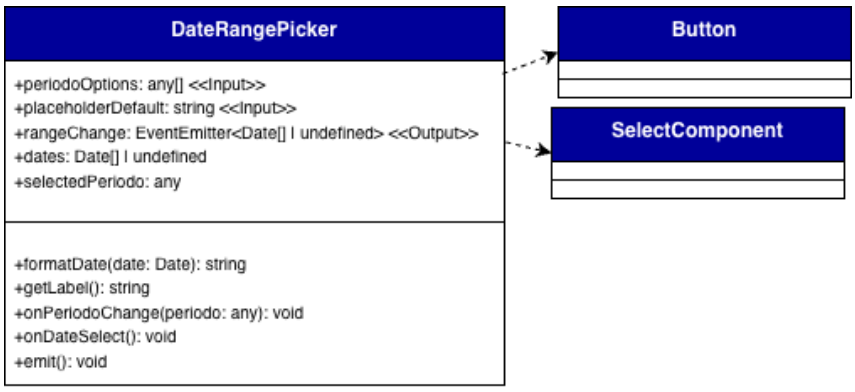
Attributi

- Attributi di input: configurano l'etichetta e il testo segnaposto del selettore mese/anno.
  - label: string Etichetta visualizzata sopra il controllo.
  - placeholder: string Testo mostrato quando non è stato ancora selezionato alcun mese/anno.
- Attributi di output: comunicano al componente padre il valore selezionato.
  - monthYearChange: EventEmitter<string> Emette il mese/anno selezionato nel formato YYYY-MM.
- Attributi interni: mantengono lo stato locale del selettore.
  - selectedMonthYear: Date | null Valore data selezionato nel datepicker, usato per ricavare mese e anno.

Metodi

- Gestione selezione mese/anno: converte la data scelta nel formato richiesto e la emette al padre.
  - onMonthYearChange(): void Legge selectedMonthYear, estrae mese e anno e li emette come stringa nel formato YYYY-MM; se non c'è alcun valore selezionato, emette una stringa vuota.

6.6.13 DateRangePicker



Attributi

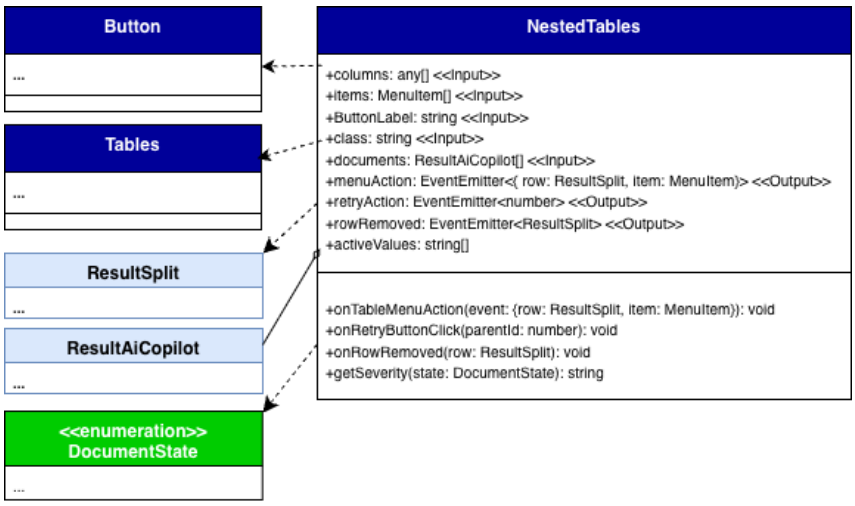
- Attributi di input: configurano il comportamento del selettore di intervallo.
  - periodoOptions: any[] Opzioni disponibili per il selettore di periodo.
  - placeholderDefault: string Testo mostrato quando non è stato ancora selezionato alcun periodo.
- Attributi di output: comunicano al componente padre i valori selezionati.
  - rangeChange: EventEmitter<Date[] | undefined> Emette l'intervallo di date selezionato nel formato YYYY-MM-DD.
- Attributi interni: mantengono lo stato locale del selettore di intervallo.
  - dates: Date[] | undefined Valore di intervallo selezionato nel datepicker.
  - selectedPeriodo: Date | null Mostra il risultato del periodo selezionato

Metodi

- Gestione selezione intervallo: converte le date scelte nel formato richiesto e le emette al padre.
  - onPeriodoChange(periodo: any): void Legge il selectedPeriodo, calcola le date di inizio e fine del periodo selezionato e le emette.
  - onDateSelect(): void Gestisce la selezione manuale dell'intervallo di date nel datepicker.
  - formatDate(date: Date): string Prende un oggetto Date e restituisce una stringa formattata nel formato italiano (DD/MM/YYYY).

- getLabel(): string Decide che testo mostrare come etichetta del selettore in base alle date selezionate o al periodo predefinito scelto.
- emit(): void Emette l'intervallo di date selezionato al componente padre.

6.6.14 NestedTables



Componente UI che rappresenta una riga espandibile che contiene a sua volta una tabella. Viene utilizzato per visualizzare i dettagli dei documenti estratti, dove ogni riga rappresenta un documento e l'espansione mostra i dati estratti associati a uno split di quel documento.

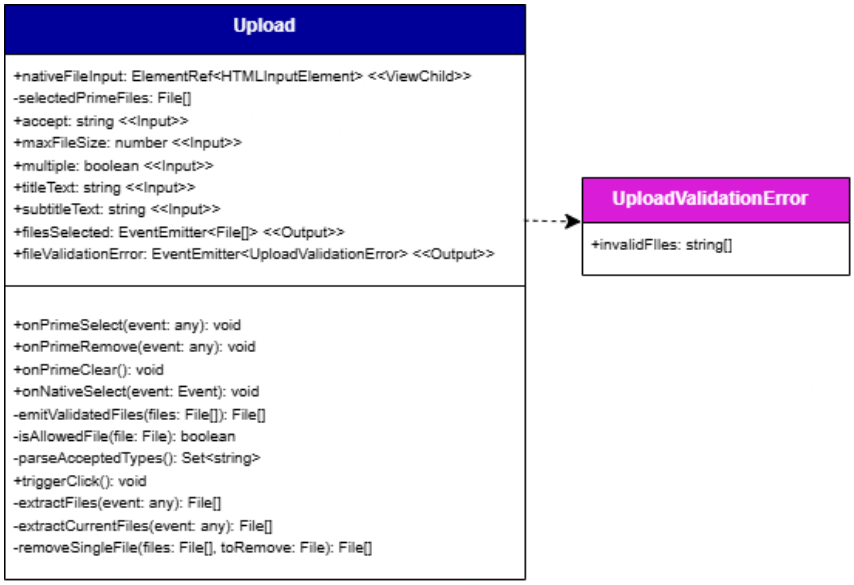
Attributi

- Attributi di input: configurano i dati e il comportamento della tabella annidata.
  - columns: any[] Definisce le colonne da visualizzare nelle tabelle nidificate.
  - items: MenuItem[] Elenco delle azioni disponibili nel menu contestuale per ciascuna riga.
  - ButtonLabel: string Testo mostrato nel pulsante associato all'azione di retry.
  - class: string Classe CSS aggiuntiva per personalizzare l'aspetto del componente.
  - documents: ResultAiCopilot[] Collezione dei documenti gestiti e mostrati dal componente.
- Attributi di output: comunicano al componente padre le interazioni e le azioni eseguite dall'utente.
  - menuAction: EventEmitter{ row: ResultSplit; item: MenuItem } Emette l'azione di menu selezionata insieme alla riga di riferimento.
  - retryAction: EventEmitter<number> Emette l'identificativo del record padre quando viene richiesto un nuovo tentativo.
  - rowRemoved: EventEmitter<ResultSplit> Notifica al componente padre la rimozione di una riga.
- Attributi interni: mantengono lo stato locale del componente e gestiscono le condizioni di visualizzazione.
  - activeValues: string[] Mantiene lo stato degli elementi attivi nell'interfaccia$_G$ annidata.

Metodi

- onTableMenuAction(event: { row: ResultSplit; item: MenuItem }): void Inoltra verso l'output menuAction l'evento di menu ricevuto dalla tabella interna.
- onRetryButtonClick(parentId: number): void Propaga l'identificativo del padre tramite retryAction quando l'utente avvia il retry.
- onRowRemoved(row: ResultSplit): void Propaga tramite rowRemoved la riga eliminata dal livello tabellare interno.
- getSeverity(state: DocumentState): string Restituisce la classe di severità CSS coerente con lo stato del documento per la resa grafica.

6.6.15 Upload



Componente UI che rappresenta un'area di drop per il caricamento di file, con supporto per selezione manuale tramite dialog di sistema. Viene utilizzato per consentire agli utenti di caricare documenti da processare tramite OCR$_G$ e estrazione dati.
Attributi

- Attributi di input: configurano il comportamento e il testo del componente di upload.
  - accept: string Definisce le estensioni di file accettate dal selettore di caricamento.
  - maxFileSize: number Imposta la dimensione massima consentita per ciascun file, espressa in byte.
  - multiple: boolean Abilita o disabilita la selezione multipla dei file.
  - titleText: string Testo principale mostrato nell'area di caricamento.
  - subtitleText: string Testo secondario visualizzato sotto il titolo del componente.
- Attributi di output: comunicano al componente padre i file selezionati e gli eventuali errori di validazione$_G$.
  - filesSelected: EventEmitter<File[]> Emette l'insieme dei file validi selezionati dall'utente.
  - fileValidationError: EventEmitter<UploadValidationError> Emette l'elenco dei file non validi quando la validazione$_G$ fallisce.

- Attributi interni: mantengono lo stato locale e gestiscono l'accesso al controllo nativo.
  - nativeFileInput: ElementRef<HTMLInputElement> Riferimento all'input file nativo utilizzato per avviare la selezione da codice.
  - selectedPrimeFiles: File[] Conserva l'insieme corrente dei file gestiti dal componente PrimeNG.

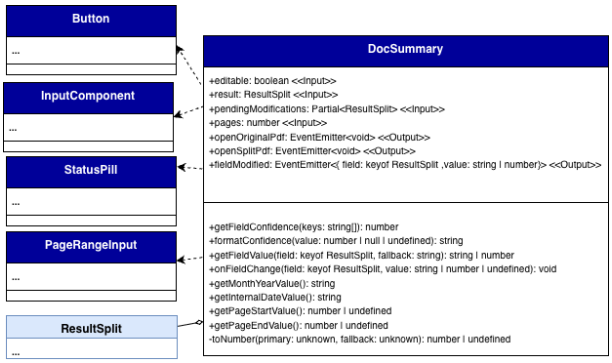
Metodi

- onPrimeSelect(event: any): void Gestisce la selezione dei file tramite il componente PrimeNG, valida i file e aggiorna lo stato interno.
- onPrimeRemove(event: any): void Gestisce la rimozione di un file o dell'intera selezione dal controllo PrimeNG e aggiorna lo stato interno.
- onPrimeClear(): void Resetta completamente la selezione corrente e notifica il componente padre con una lista vuota.
- onNativeSelect(event: Event): void Gestisce la selezione effettuata tramite input nativo, valida i file e resetta il valore dell'input per consentire una nuova selezione dello stesso file.
- emitValidatedFiles(files: File[]): File[] Filtra i file validi, emette gli errori per quelli non ammessi e pubblica l'elenco dei file accettati tramite filesSelected.
- isAllowedFile(file: File): boolean Verifica$_G$ se un file rispetta le estensioni configurate nella proprietà accept.
- parseAcceptedTypes(): Set<string> Converte la stringa di configurazione accept in un insieme di estensioni normalizzate e confrontabili.
- triggerClick(): void Simula il click sull'input file nativo per aprire la finestra di selezione.
- extractFiles(event: any): File[] Estrae i file dall'evento ricevuto supportando le diverse strutture fornite dai componenti di upload.
- extractCurrentFiles(event: any): File[] | null Recupera la lista corrente dei file quando è disponibile nel payload dell'evento.
- removeSingleFile(files: File[], toRemove: File): File[] Rimuove un file specifico dall'elenco confrontandolo per nome, dimensione e ultima modifica.

UploadValidationError

- invalidFiles: string[] Nome dei file che hanno causato l'errore di validazione$_G$.

6.6.16 DocSummary



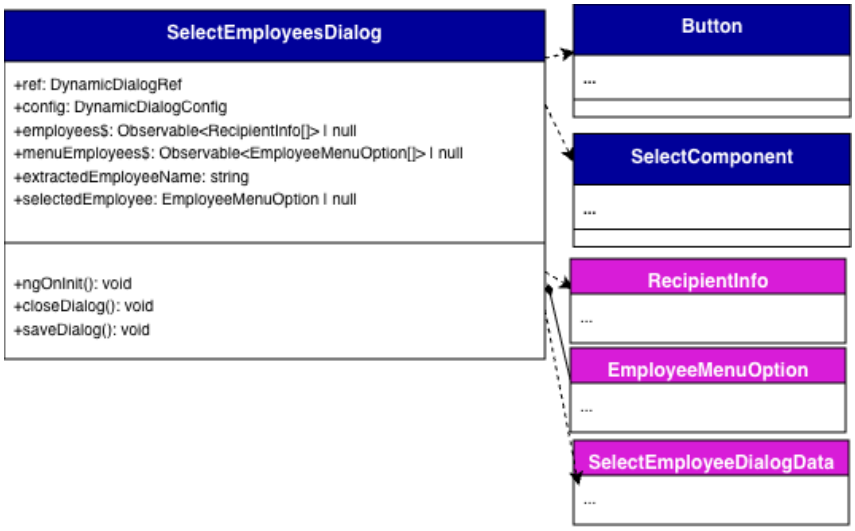
Componente UI che rappresenta un riepilogo di un documento, mostrando informazioni chiave estratte (o compilate).
Attributi

- Attributi di input: configurano lo stato iniziale e il contenuto visualizzato dal componente.
  - editable: boolean Indica se i campi del riepilogo documento sono modificabili dall'utente.
  - result: ResultSplit | null
    Contiene i dati del documento da visualizzare e, se necessario, modificare.
  - pendingModifications: Partial<ResultSplit>
    Memorizza le modifiche locali non ancora confermate sul documento.
  - pages: number
    Indica il numero totale di pagine associate al documento.
- Attributi di output: comunicano al componente padre le azioni eseguite dall'utente.
  - openOriginalPdf: EventEmitter<void> Notifica la richiesta di apertura del PDF originale.
  - openSplitPdf: EventEmitter<void> Notifica la richiesta di apertura del PDF suddiviso.
  - fieldModified: EventEmitter{ field: keyof ResultSplit; value: string | number | undefined }
    Emite la modifica di un campo del riepilogo con il nuovo valore inserito.

Metodi

- getFieldConfidence(...keys: string[]): number | null Restituisce il livello di confidenza$_G$ del primo campo disponibile tra quelli indicati.
- formatConfidence(value: number | null | undefined): string Normalizza e formatta il valore di confidenza$_G$.
- getFieldValue(field: keyof ResultSplit, fallback: string = 'Non trovato'): string | number Recupera il valore di un campo considerando prima le modifiche pendenti e poi il dato originale, con fallback configurabile.
- onFieldChange(field: keyof ResultSplit, value: string | number | undefined): void Propaga verso l'esterno la modifica di un campo del documento.
- getMonthYearValue(): string Normalizza il valore del campo data nel formato mese/anno, gestendo diversi formati di input.
- getPageStartValue(): number | undefined Restituisce il valore della pagina iniziale considerando prima le modifiche pendenti e poi i dati originali.
- getPageEndValue(): number | undefined Restituisce il valore della pagina finale considerando prima le modifiche pendenti e poi i dati originali.
- toNumber(primary: unknown, fallback: unknown): number | undefined Converte un valore in numero, usando un fallback se il valore principale non è disponibile o non valido.

6.6.17 SelectEmployeeDialog



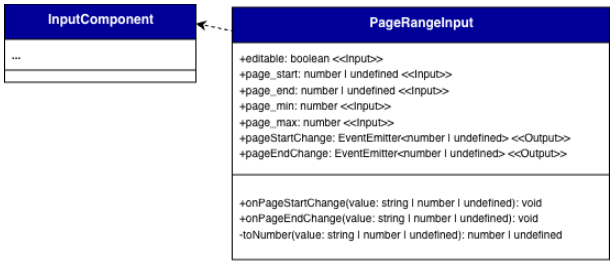
Componente UI che rappresenta un dialog per la selezione di un dipendente dalla lista di dipendenti dell'azienda inserita o estratta, con possibilità di ricerca.
Attributi

- Attributi interni: mantengono lo stato locale del dialog e gestiscono l'integrazione con PrimeNG Dynamic Dialog.
  - ref: DynamicDialogRef Riferimento al dialog corrente, usato per chiudere la finestra e restituire eventualmente un payload.
  - config: DynamicDialogConfig Configurazione del dialog, utilizzata per leggere i dati passati in apertura.
  - employees$: Observable<EmployeeOption[]> | null Stream contenente la lista dei dipendenti selezionabili nel dialog.
  - menuEmployees$: Observable<EmployeeMenuOption[]> | null Stream contenente la lista dei dipendenti formattati per essere mostrati nel menu.
  - extractedEmployeeName: string Nome dipendente estratto automaticamente e mostrato come riferimento all'utente.
  - selectedEmployee: EmployeeOption | null Dipendente attualmente selezionato dall'utente nel menu a tendina.

Metodi

- ngOnInit(): void Inizializza lo stato del componente leggendo `extractedEmployeeName` ed `employees$` dai dati del dialog.
- closeDialog(): void Chiude il dialog senza restituire alcun risultato di selezione.
- saveDialog(): void Se è presente una selezione valida, costruisce il payload `SelectEmployeeDialogResult` e chiude il dialog restituendolo al chiamante.

6.6.18 PageRangeInput



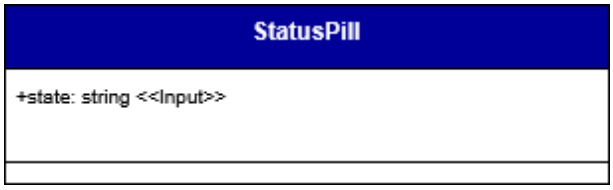
Componente UI che rappresenta un campo di input per l'inserimento di un intervallo di pagine, con validazione$_G$ e parsing del formato "start-end".
Attributi

- Attributi di input: configurano l'editabilità e i limiti dell'intervallo pagine.
  - editable: boolean Indica se i campi dell'intervallo pagine possono essere modificati dall'utente.
  - page_start: number | undefined Rappresenta la pagina iniziale dell'intervallo selezionato.
  - page_end: number | undefined Rappresenta la pagina finale dell'intervallo selezionato.
  - page_min: number Definisce il valore minimo consentito per le pagine selezionabili.
  - page_max: number Definisce il valore massimo consentito per le pagine selezionabili.
- Attributi di output: comunicano al componente padre le variazioni dell'intervallo.
  - pageStartChange: EventEmitter<number | undefined> Emette il nuovo valore della pagina iniziale quando viene modificata.
  - pageEndChange: EventEmitter<number | undefined> Emette il nuovo valore della pagina finale quando viene modificata.

Metodi

- onPageStartChange(value: string | number | undefined): void Normalizza e aggiorna la pagina iniziale; se necessario riallinea la pagina finale per garantire coerenza dell'intervallo.
- onPageEndChange(value: string | number | undefined): void Normalizza e aggiorna la pagina finale, impedendo che sia minore della pagina iniziale.
- get pageEndMin(): number Restituisce il minimo consentito per page_end considerando il vincolo del limite inferiore e la pagina iniziale corrente.
- private toNumber(value: string | number | undefined): number | undefined Converte un valore eterogeneo in numero, restituendo undefined in caso di input vuoto o non valido.

6.6.19 StatusPill

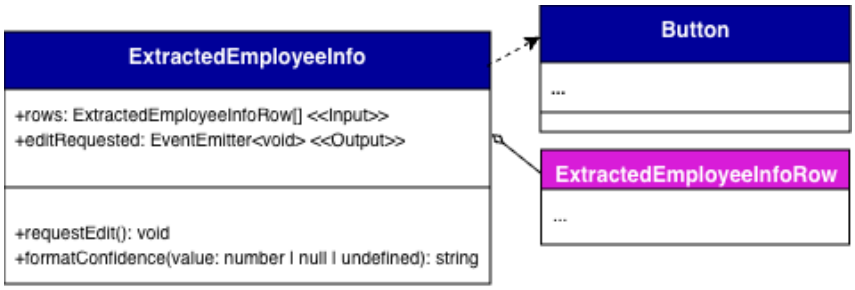


Componente UI che rappresenta una "pillola" di stato, visualizzando un'etichetta testuale con uno stile che varia in base al contenuto.

Attributi

- state: string Definisce lo stato testuale da rappresentare nella pillola di stato.

6.6.20 ExtractedEmployeeInfo



Attributi

- Attributi di input: configurano i dati mostrati nella tabella delle informazioni dipendente estratte.
  - rows: ExtractedEmployeeInfoRow[] Elenco delle righe con i dati estratti e di matching del dipendente.
- Attributi di output: comunicano al componente padre le azioni richieste dall'utente.
  - editRequested: EventEmitter<void> Notifica la richiesta di apertura/modifica dei dati estratti.

Metodi

- requestEdit(): void Emette l'evento `editRequested` per segnalare al componente padre la volontà di modificare i dati.
- requestRowRemoval(rowIndex: number): void Emette l'indice della riga da rimuovere tramite `rowRemoved`.

ExtractedEmployeeInfoRow (interfaccia$_G$ di supporto)

Attributi

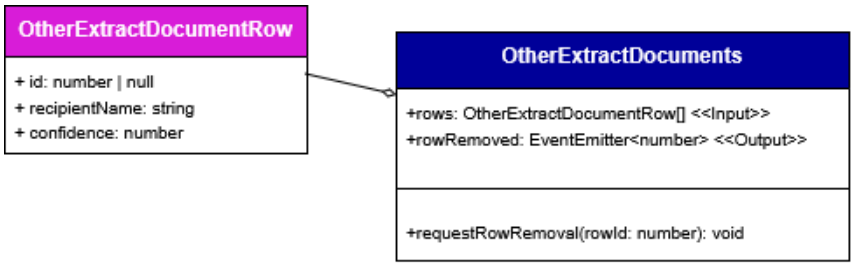
- Tipo dati di supporto: non è un componente Angular$_G$, ma un'interfaccia$_G$ TypeScript$_G$ usata da `ExtractedEmployeeInfo`.
  - recipient: RecipientInfo Oggetto con le informazioni del destinatario/dipendente associato alla riga.
  - rawName: string Nome originale estratto prima della normalizzazione.
  - hasMatch: boolean Indica se è stata trovata una corrispondenza valida con un dipendente noto.
  - recipientConfidence: number | null Livello di confidenza$_G$ dell'estrazione del destinatario; nullo se non disponibile.

RecipientInfo (interfaccia$_G$ di supporto)

Attributi

- Tipo dati di supporto: interfaccia$_G$ TypeScript$_G$ che rappresenta i dati completi del destinatario restituiti dal backend$_G$.
  - recipientId: number Identificativo univoco del destinatario.
  - recipientName: string Nome normalizzato del destinatario/dipendente.
  - rawRecipientName: string Nome grezzo estratto prima della normalizzazione.
  - recipientEmail: string Indirizzo email associato al destinatario.
  - recipientCode: string Codice identificativo del dipendente.

6.6.21 OtherExtractDocuments



Componente UI che rappresenta gli altri documenti estratti oltre quello corrente, dallo stesso file originale splittato.

Attributi

- Attributi di input: configurano i dati mostrati nella tabella dei documenti estratti.
  - rows: OtherExtractDocumentRow[] Elenco delle righe da visualizzare, ciascuna contenente identificativo, nominativo destinatario e confidenza$_G$.
- Attributi di output: comunicano al componente padre le azioni utente sulle righe.
  - rowRemoved: EventEmitter<number> Emette l'identificativo della riga che l'utente richiede di rimuovere.

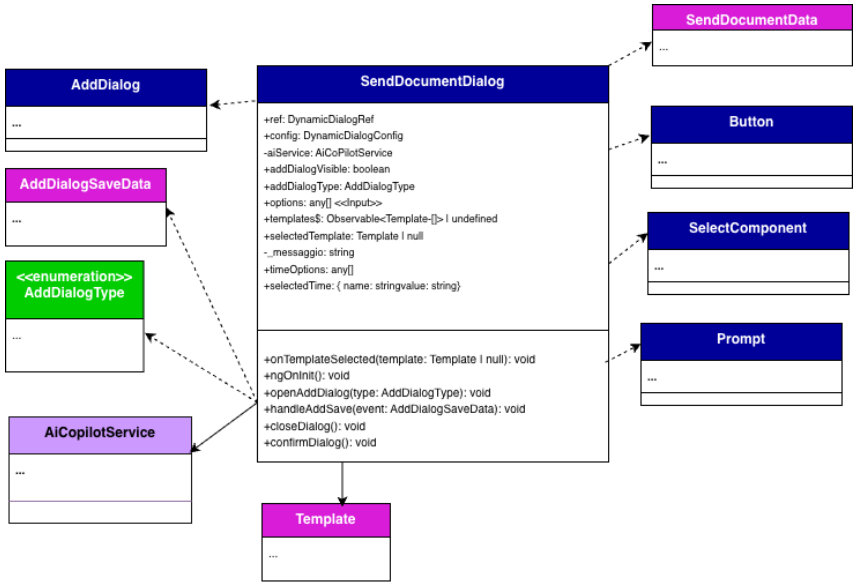
Metodi

- requestRowRemoval(rowId: number): void Inoltra verso l'esterno la richiesta di rimozione di una riga emettendo il relativo identificativo tramite `rowRemoved`.

OtherExtractDocumentRow
Attributi

- Tipo dati di supporto: non è un componente Angular$_G$, ma un'interfaccia$_G$ TypeScript$_G$ usata da `OtherExtractDocuments`.
  - id: number | null Identificativo univoco della riga; può essere nullo quando il record non è ancora persistito o identificato.
  - recipientName: string Nome del destinatario associato al documento estratto.
  - confidence: number Valore numerico di confidenza$_G$ dell'estrazione associata alla riga.

6.6.22 SendDocumentDialog



Componente UI che rappresenta un dialog per la conferma dell'invio di un documento. Mostra un campo di testo per inserire un messaggio personalizzato o la selezione di un template$_G$ per un messaggio predefinito. Inoltre mostra la possibilità di inviare il messaggio immediatamente o programmarlo a orari prestabiliti. Infine il pulsante di invio.

Attributi

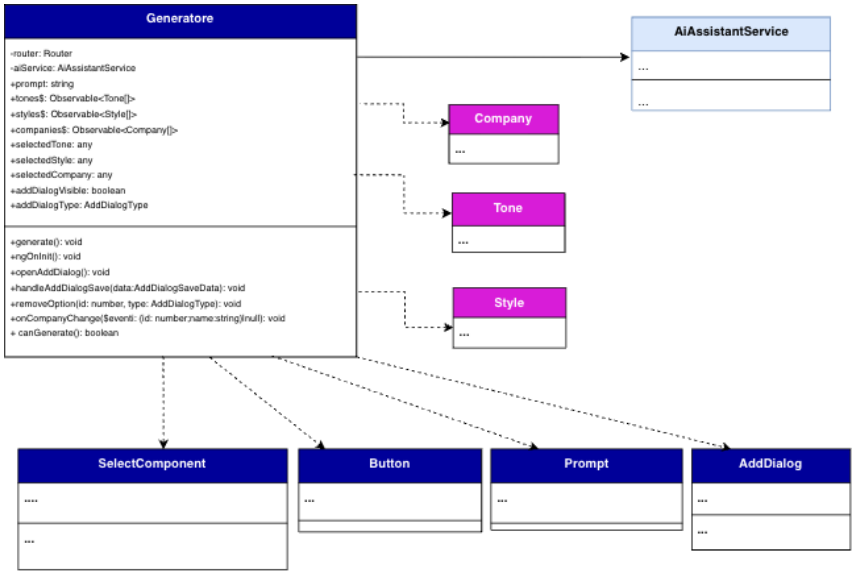
- Attributi di input: configurano i dati opzionali passati al componente.
  - options: any[] Collezione generica di opzioni configurabili ricevute dal componente padre.
- Attributi interni: gestiscono lo stato locale del dialog.
  - ref: DynamicDialogRef Riferimento al dialog corrente, usato per chiudere la finestra con o senza payload.
  - config: DynamicDialogConfig Configurazione del dialog, usata per leggere i dati passati in apertura.
  - aiService: AiCoPilotService Servizio applicativo usato per la creazione di nuovi template$_G$.
  - addDialogVisible: boolean Indica se il dialog secondario di aggiunta è visibile.
  - addDialogType: AddDialogType Specifica il tipo di dialog secondario da aprire (es. template/tone).
  - templates$: Observable<TemplateOption[]> | undefined Stream dei template$_G$ disponibili da mostrare nella selezione.
  - selectedTemplate: TemplateOption | null Template$_G$ attualmente selezionato dall'utente.
  - _messaggio: string Buffer interno del testo messaggio, usato dagli accessor della proprietà messaggio.
  - timeOptions: any[] Elenco delle opzioni temporali disponibili per la pianificazione dell'invio.
  - selectedTime: { name: string; value: string } Opzione temporale attualmente selezionata per l'invio.
- Attributi calcolati:
  - messaggio: string (Accessor get/set) Espone il contenuto del messaggio: in lettura usa `_messaggio` oppure il contenuto del template$_G$ selezionato; in scrittura aggiorna `_messaggio`.

Metodi

- get messaggio(): string Restituisce il testo del messaggio privilegiando il valore inserito manualmente, con fallback al contenuto del template$_G$ selezionato.
- set messaggio(value: string): void Aggiorna il contenuto interno del messaggio.
- onTemplateSelected(template: TemplateOption | null): void Aggiorna il template$_G$ selezionato e sincronizza il messaggio con il contenuto del template$_G$ scelto.
- ngOnInit(): void Inizializza il componente leggendo dallo stato del dialog lo stream dei template$_G$ disponibili.
- openAddDialog(type: AddDialogType): void Imposta il tipo di dialog secondario da aprire e ne abilita la visualizzazione.
- handleAddSave(event: AddDialogSaveData): void Gestisce il salvataggio dal dialog secondario e, se il tipo è template$_G$, invoca il servizio per crearne uno nuovo.
- closeDialog(): void Chiude il dialog senza restituire dati di conferma.
- confirmDialog(): void Costruisce il payload `SendDocumentData` con messaggio, orario, allegati e template$_G$ selezionato, quindi chiude il dialog restituendolo.

6.7 Macro componenti UI (pagine)

6.7.1 Pagina di generazione



Pagina principale per la generazione di documenti, che utilizza i componenti SelectComponent, Button, Prompt$_G$, AddDialog e riceve segnali da questi per fare operazioni con l'AiAssistantService.

Attributi

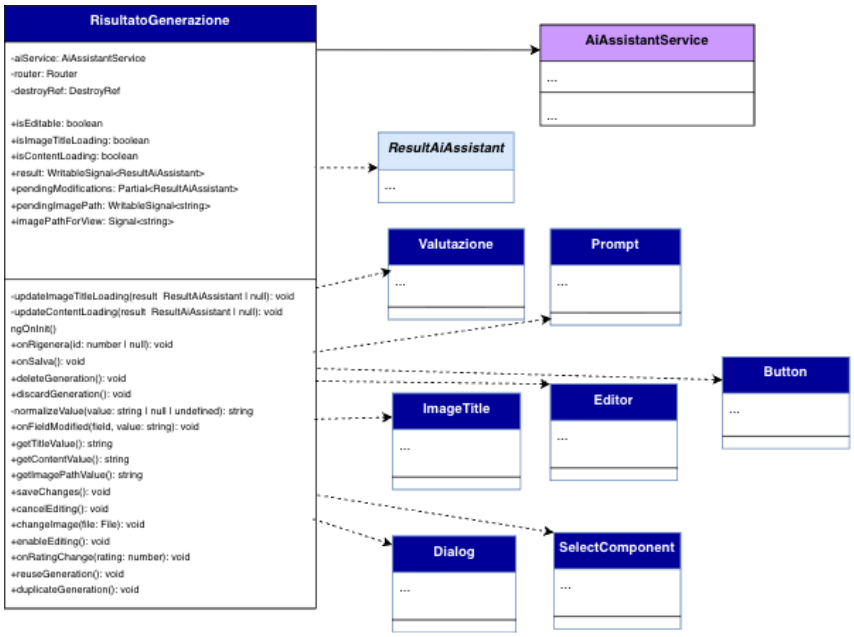
- router: Router Iniettato per la navigazione verso la pagina dei risultati generati.
- aiService: AiAssistantService Servizio applicativo iniettato per la gestione delle richieste di generazione, il recupero di toni/stili/aziende e la creazione di nuove opzioni.
- prompt: string Contiene il testo del prompt$_G$ inserito dall'utente, inizializzato dallo stato di navigazione.
- tones$: Observable Stream reattivo dei toni disponibili, sempre sincronizzato con il servizio.
- styles$: Observable Stream reattivo degli stili disponibili, sempre sincronizzato con il servizio.
- companies$: Observable Stream reattivo delle aziende disponibili, sempre sincronizzato con il servizio.

- selectedTone: any Tono attualmente selezionato dall'utente, inizializzato dallo stato di navigazione.
- selectedStyle: any Stile attualmente selezionato dall'utente, inizializzato dallo stato di navigazione.
- selectedCompany: any Azienda attualmente selezionata; il suo cambio innesca il recupero di toni e stili associati.
- addDialogVisible: boolean Indica se il dialog di aggiunta di nuovo tono/stile è visibile.
- addDialogType: AddDialogType Specifica il tipo di dialog secondario da aprire (tone o style).

Metodi

- get canGenerate(): boolean (getter) Valida lo stato di completamento del modulo$_G$: verifica$_G$ la selezione di tono, stile e la lunghezza minima del prompt$_G$.
- generate(): void Invia la richiesta di generazione al servizio con prompt$_G$, tono, stile e azienda selezionati. Sottoscrive il risultato e naviga verso la pagina dei risultati.
- ngOnInit(): void Inizializza il componente eseguendo il recupero delle aziende e, se un'azienda è preselezionata, carica i toni e gli stili ad essa associati.
- openAddDialog(type: AddDialogType): void Prepara l'apertura del dialog secondario impostando il tipo (tono o stile) e rendendo visibile la finestra di dialogo.
- handleAddDialogSave(data: AddDialogSaveData): void Gestisce il salvataggio dal dialog secondario: invoca il servizio per creare un nuovo tono o stile associato all'azienda selezionata.
- removeOption(id: number, type: AddDialogType): void Rimuove un'opzione (tono o stile) identificata da id, invocando il metodo appropriato del servizio in base al tipo.
- onCompanyChange(event: { id: number; name: string } | null): void Aggiorna l'azienda selezionata e sincronizza il caricamento di toni e stili relativi; reimposta le selezioni precedenti di tono e stile.

6.7.2 Risultato generazione



Componente UI che rappresenta la pagina di visualizzazione del risultato di una generazione, riutilzzata sia per la visualizzazione immediata dopo la generazione, sia per la visualizzazione di una generazione salvata. Permette inoltre di modificare, rigenerare, salvare, valutare e cancellare

il risultato tramite interazioni con AiAssistantService.

Attributi

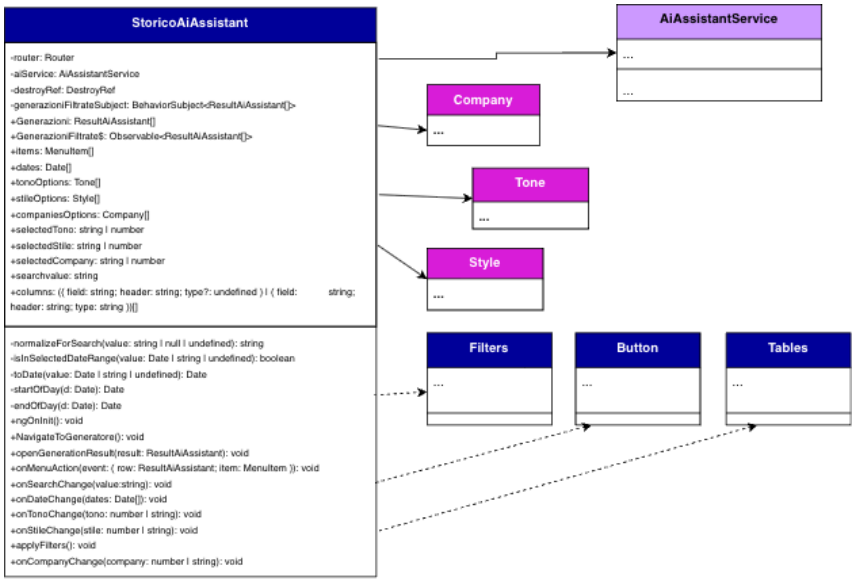
- aiService: AiAssistantService Servizio applicativo iniettato per la gestione della rigenerazione, salvataggio, valutazione e operazioni sugli articoli generati.
- router: Router Iniettato per la navigazione verso altre pagine (ad esempio, ritorno al generatore in caso di errore).
- destroyRef: DestroyRef Iniettato per la gestione della pulizia delle sottoscrizioni RxJS al distruggersi del componente.
- isEditable: boolean Indica se il componente è in modalità di modifica; abilita/disabilita i campi di input per la modifica del contenuto.
- isImageTitleLoading: boolean Indica se titolo e immagine sono ancora in fase di generazione da parte dell'IA.
- isContentLoading: boolean Indica se il contenuto principale è ancora in fase di generazione da parte dell'IA.
- result: Signal<ResultAiAssistant | null> Signal che contiene il risultato principale della generazione; viene aggiornato via sottoscrizione al servizio e sincronizzato nello stato di navigazione.
- pendingModifications: Partial<ResultAiAssistant> Oggetto parziale che traccia le modifiche in sospeso (title, content, imagePath) prima del salvataggio definitivo.
- pendingImagePath: signal<string | null> Signal che mantiene il percorso Base64 dell'immagine in modifica prima del salvataggio.
- imagePathForView: computed Computed Signal che restituisce il percorso immagine da visualizzare: priorità al pendingImagePath, fallback a quello del result.

Metodi

- ngOnInit(): void Inizializza il componente: ripristina il result dallo stato di navigazione, si sottoscrive agli aggiornamenti del servizio (result e errori), e aggiorna gli stati di caricamento.
- hasPendingModifications: boolean (getter) Restituisce true se ci sono pendingModifications non vuote; usato per abilitare il pulsante di salvataggio.
- updateImageTitleLoading(result: ResultAiAssistant | null): void Aggiorna isImageTitleLoading controllando se il result è completo e se titolo e immagine hanno valori non vuoti.
- updateContentLoading(result: ResultAiAssistant | null): void Aggiorna isContentLoading controllando se il result è completo e se il contenuto ha un valore non vuoto.
- onRigenera(id: number | null): void Invoca il servizio per rigenerare il result corrente mantenendone l'id.
- onSalva(): void Invoca il servizio per creare e salvare un nuovo post dal result corrente.
- deleteGeneration(): void Elimina il post tramite il servizio e naviga verso la pagina generatore.
- discardEvalutation(): void Inserisce la valutazione minima (1) e naviga verso la pagina generatore.
- normalizeValue(value: string | null | undefined): string Converte un valore stringa nullable in stringa vuota se non definito; usato per normalizzare i confronti.
- onFieldModified(field: 'title' | 'content', value: string): void Traccia la modifica di un campo (titolo o contenuto) in pendingModifications; rimuove la modifica se il valore è ripristinato a quello originale.
- getTitleValue(): string Restituisce il titolo da visualizzare: priorità a pendingModifications.title, fallback al result().title.
- getContentValue(): string Restituisce il contenuto da visualizzare: priorità a pendingModifications.content, fallback al result().content.
- getImagePathValue(): string Restituisce il percorso immagine formattato per la visualizzazione; aggiunge il base URL localhost se il percorso non è Base64 né URL remoto.
- saveChanges(): void Salva le modifiche pendenti: sostituisce il result con il merge di result e pendingModifications, sincronizza il servizio, svuota le modifiche e disabilita l'editing.

- cancelEditing(): void Annulla tutte le modifiche in sospeso: reimposta pendingModifications e pendingImagePath e disabilita l'editing.
- changeImage(file: File): void Converte un file immagine selezionato in Base64 e lo traccia in pendingModifications e pendingImagePath.
- enableEditing(): void Abilita la modalità di modifica svuotando le modifiche precedenti e impostando isEditable.
- onRatingChange(rating: number): void Invoca il servizio per registrare la valutazione dell'utente sul generatedDatumId del result corrente.
- reuseGeneration(): void Prepara il riutilizzo della generazione inviando al servizio tono, stile, azienda e prompt$_G$ dal result corrente.
- duplicateGeneration(): void Naviga verso il generatore passando lo stato di tono, stile, azienda e prompt$_G$ dal result corrente per la duplicazione.

6.7.3 Pagina di storico dei risultati dell'Ai Assistant



Pagina che mostra lo storico dei risultati salvati come post.
Attributi

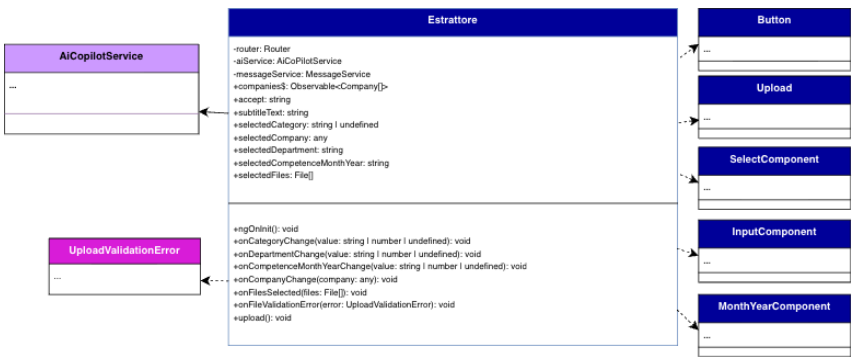
- router: Router Iniettato per la navigazione verso il generatore e verso la pagina del risultato di generazione.
- aiService: AiAssistantService Servizio applicativo iniettato per recuperare lo storico, gestire le azioni sui risultati e caricare i dati di supporto per i filtri.
- destroyRef: DestroyRef Utilizzato per interrompere automaticamente le sottoscrizioni quando il componente viene distrutto.
- Generazioni: ResultAiAssistant[] Collezione completa dei risultati di generazione caricati dallo storico.
- generazioniFiltrateSubject: BehaviorSubject<ResultAiAssistant[]> Subject interno che mantiene l'elenco filtrato dei risultati e notifica gli aggiornamenti alla vista.
- GenerazioniFiltrate$: Observable<ResultAiAssistant[]> Stream osservabile dell'elenco filtrato, esposto alla vista per il binding reattivo.
- items: MenuItem[] Voci del menu contestuale associate alle azioni disponibili su ogni riga.
- dates: Date[] | undefined Intervallo di date selezionato per filtrare lo storico.
- tonoOptions: Tone[] Elenco delle opzioni di tono disponibili per il filtro.
- stileOptions: Style[] Elenco delle opzioni di stile disponibili per il filtro.

- companiesOptions: Company[] Elenco delle aziende disponibili per il filtro.
- selectedTono: number | string | undefined Tono attualmente selezionato nei filtri.
- selectedStile: number | string | undefined Stile attualmente selezionato nei filtri.
- selectedCompany: number | string | undefined Azienda attualmente selezionata nei filtri.
- searchvalue: string Valore testuale inserito nella ricerca libera.
- columns: { field: string; header: string; type?: string }[] Configurazione delle colonne mostrate nella tabella dello storico.

Metodi

- ngOnInit(): void Inizializza il componente caricando aziende, toni, stili e storico delle generazioni; sottoscrive gli stream del servizio e aggiorna i filtri quando arrivano nuovi dati.
- NavigateToGeneratore(): void Naviga alla pagina del generatore.
- openGenerationResult(result: ResultAiAssistant): void Imposta il risultato selezionato come corrente nel servizio e apre la pagina del dettaglio di generazione.
- onMenuAction(event: { row: ResultAiAssistant; item: MenuItem }): void Gestisce le azioni del menu contestuale sulla riga: elimina, riutilizza o duplica la generazione selezionata.
- onSearchChange(value: string): void Aggiorna il testo di ricerca e riapplica i filtri.
- onDateChange(dates: Date[]): void Aggiorna il range di date selezionato e riapplica i filtri.
- onTonoChange(tono: number | string): void Aggiorna il tono selezionato e riapplica i filtri.
- onStileChange(stile: number | string): void Aggiorna lo stile selezionato e riapplica i filtri.
- applyFilters(): void Applica tutti i criteri di filtro disponibili su storico, ricerca testuale, tono, stile, azienda e intervallo di date, quindi pubblica il risultato filtrato.
- normalizeForSearch(value: string | null | undefined): string Normalizza una stringa per la ricerca rimuovendo markup HTML, spazi multipli e maiuscole, così da rendere i confronti consistenti.
- isInSelectedDateRange(value: Date | string | undefined): boolean Verifica$_G$ se una data rientra nell'intervallo selezionato; se non ci sono date impostate, considera il record valido.
- toDate(value: Date | string | undefined): Date | null Converte un valore in una istanza Date valida oppure restituisce null se la conversione non è possibile.
- startOfDay(d: Date): Date Restituisce l'inizio del giorno della data passata.
- endOfDay(d: Date): Date Restituisce la fine del giorno della data passata.
- onCompanyChange(company: number | string): void Aggiorna l'azienda selezionata e riapplica i filtri alla tabella.

6.7.4 Estrattore



Componente UI che rappresenta la pagina per estrarre i dati da un documento, mostrando il documento estratto, i dati associati e permettendo di modificare o confermare i dati estratti.

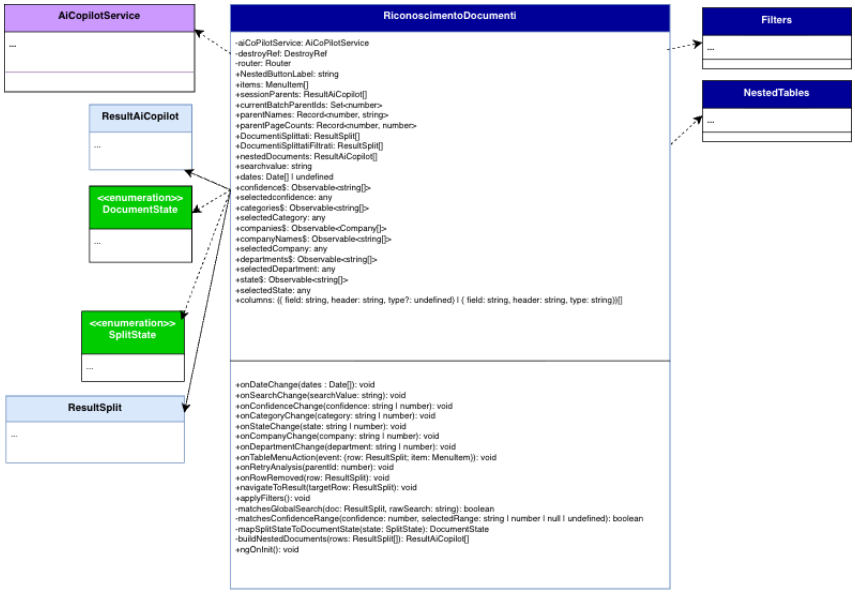
Attributi

- router: Router Iniettato per la navigazione verso la pagina di riconoscimento documenti dopo l'upload.
- aiService: AiCoPilotService Servizio applicativo iniettato per recuperare le aziende disponibili e inviare i file al backend.
- messageService: MessageService Servizio usato per mostrare notifiche toast, in particolare in caso di file non validi.
- companies$: Observable<Company[]> Stream osservabile delle aziende, esposto dal servizio e usato dalla vista.
- accept: string Estensioni consentite per l'upload (pdf, csv, jpg, jpeg, png).
- subtitleText: string Testo descrittivo mostrato sotto il titolo del componente di upload.
- selectedCategory: string | undefined Categoria attualmente selezionata.
- selectedCompany: any Azienda attualmente selezionata.
- selectedDepartment: string Reparto selezionato.
- selectedCompetenceMonthYear: string Mese/anno di competenza selezionato.
- selectedFiles: File[] Elenco dei file validi selezionati e pronti per l'upload.
- canUpload: boolean (getter) Indica se è possibile avviare l'upload; è true quando è presente almeno un file selezionato.

Metodi

- ngOnInit(): void Inizializza la pagina caricando l'elenco aziende tramite servizio.
- onCategoryChange(value: string | number | undefined): void Aggiorna la categoria selezionata convertendo il valore in stringa.
- onDepartmentChange(value: string | number | undefined): void Aggiorna il reparto selezionato convertendo il valore in stringa.
- onCompetenceMonthYearChange(value: string | number | undefined): void Aggiorna il mese/anno di competenza convertendo il valore in stringa.
- onCompanyChange(company: any): void Imposta l'azienda selezionata e resetta il reparto dipendente dall'azienda.
- onFilesSelected(files: File[]): void Salva la lista dei file validati ricevuta dal componente di upload.
- onFileValidationError(error: UploadValidationError): void Mostra un toast di warning con i nomi dei file non validi scartati in fase di validazione.
- upload(): void Se ci sono file selezionati, invia la batch al servizio con i metadati correnti e naviga alla pagina di riconoscimento mantenendo la sessione.

6.7.5 Riconoscimento documenti



Componente UI che rappresenta la pagina di riconoscimento dei documenti caricati, che mostra i documenti riconosciuti e splittati, con filtri avanzati, vista nested per documento padre e azioni operative (modifica, elimina, retry analisi).

Attributi

- aiCoPilotService: AiCoPilotService Servizio applicativo iniettato per caricare filtri, storico risultati, dati di sessione e invocare le azioni sui documenti.
- destroyRef: DestroyRef Utilizzato per interrompere automaticamente le sottoscrizioni quando il componente viene distrutto.
- router: Router Iniettato per la navigazione verso la pagina di anteprima/modifica documento.
- NestedButtonLabel: string Etichetta del pulsante per rilanciare l'analisi sul documento padre.
- items: MenuItem[] Voci del menu contestuale della tabella (azioni Modifica/Elimina).
- sessionParents: ResultAiCopilot[] Elenco dei documenti padre correnti legati alla sessione.
- currentBatchParentIds: Set<number> Insieme degli id padre appartenenti alla batch corrente, usato per limitare la vista.
- parentNames: Record<number, string> Mappa id padre → nome documento padre.
- parentPageCounts: Record<number, number> Mappa id padre → numero pagine del documento padre.
- DocumentiSplittati: ResultSplit[] Collezione completa degli split caricati dallo storico corrente.
- DocumentiSplittatiFiltrati: ResultSplit[] Collezione degli split dopo applicazione dei filtri attivi.
- nestedDocuments: ResultAiCopilot[] Struttura nested finale per la visualizzazione: documento padre con figli associati.
- searchvalue: string Valore testuale della ricerca globale.
- dates: Date[] | undefined Intervallo date selezionato per il filtro temporale.
- confidence$: Observable<any> Stream delle opzioni confidence.
- selectedconfidence: any Valore/intervallo confidence selezionato (inizializzato anche da history.state).
- categories$: Observable<any> Stream delle categorie disponibili.
- selectedCategory: any Categoria selezionata (eventualmente inizializzata da history.state).
- companies$: Observable<Company[]> Stream delle aziende disponibili.
- companyNames$: Observable<string[]> Stream derivato con i soli nomi azienda.

- selectedCompany: any Azienda selezionata (eventualmente inizializzata da history.state).
- departments$: Observable<any> Stream dei reparti disponibili.
- selectedDepartment: any Reparto selezionato (eventualmente inizializzato da history.state).
- state$: Observable<any> Stream degli stati disponibili.
- selectedState: any Stato selezionato (eventualmente inizializzato da history.state).
- columns: { field: string; header: string; type?: string }[] Configurazione delle colonne mostrate nella tabella documenti.

Metodi

- ngOnInit(): void Inizializza filtri e dati della pagina: richiama le fetch del servizio, sottoscrive gli stream di storico/sessione/mappe e costruisce il menu contestuale.
- onDateChange(dates: Date[]): void Aggiorna il range date selezionato e riapplica i filtri.
- onSearchChange(searchValue: string): void Aggiorna la ricerca globale testuale e riapplica i filtri.
- onConfidenceChange(confidence: string | number): void Aggiorna il filtro confidence e riapplica i filtri.
- onCategoryChange(category: string | number): void Aggiorna il filtro categoria e riapplica i filtri.
- onStateChange(state: string | number): void Aggiorna il filtro stato e riapplica i filtri.
- onCompanyChange(company: string | number): void Aggiorna il filtro azienda e riapplica i filtri.
- onDepartmentChange(department: string | number): void Aggiorna il filtro reparto e riapplica i filtri.
- onTableMenuAction(event: { row: ResultSplit; item: MenuItem }): void Gestisce le azioni del menu sulla riga: naviga alla modifica oppure elimina il documento padre.
- onRetryAnalysis(parentId: number): void Richiede il retry dell'elaborazione per il documento padre indicato.
- onRowRemoved(row: ResultSplit): void Rimuove localmente la riga eliminata e ricalcola la vista filtrata.
- navigateToResult(targetRow: ResultSplit): void Naviga alla pagina di anteprima documento passando la riga selezionata e il numero pagine del padre.
- applyFilters(): void Applica tutti i filtri (ricerca, date, confidence, categoria, stato, azienda, reparto), ricostruisce i raggruppamenti padre-figli e aggiorna la lista nested finale.
- matchesGlobalSearch(doc: ResultSplit, rawSearch: string): boolean Verifica se la riga soddisfa la ricerca globale confrontando il testo su campi principali, inclusi nomi documento padre da storico/sessione.
- matchesConfidenceRange(confidence: number, selectedRange: string | number | null | undefined): boolean Controlla se la confidence rientra nel valore/intervallo selezionato (es. "40 - 70").
- mapSplitStateToDocumentState(state: State): DocumentState Converte lo stato del documento split nello stato visualizzato per il documento padre.
- buildNestedDocuments(rows: ResultSplit[]): ResultAiCopilot[] Raggruppa gli split per parentId e costruisce la struttura nested con metadati del padre (nome, pagine, stato).

6.7.6 Anteprima documento

Componente UI che rappresenta la pagina che mostra l'anteprima di un documento estratto, con gestione modifica metadati/range pagine, invio documento e apertura PDF originale o splittato.

Attributi

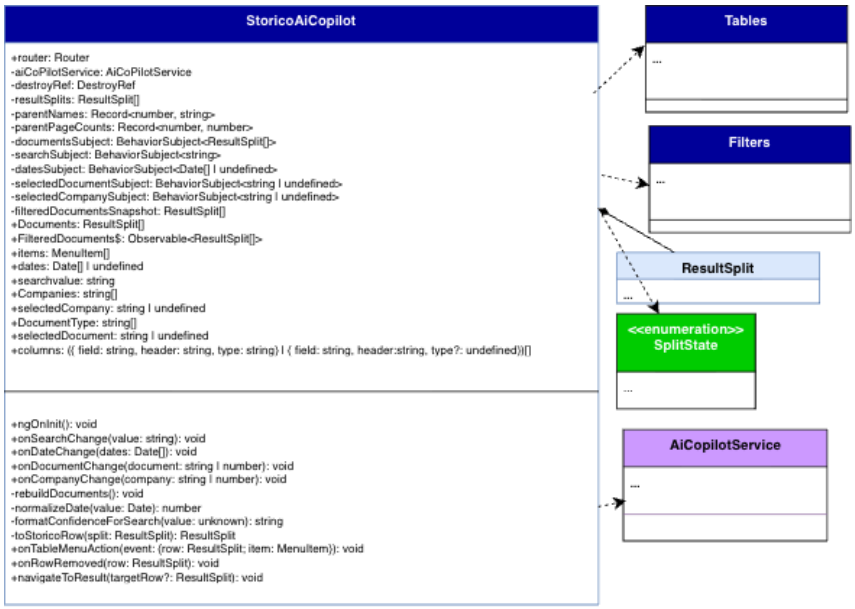
- aiService: AiCoPilotService Servizio applicativo iniettato per recuperare template, dettaglio documento, documenti correlati e inviare aggiornamenti al backend.
- destroyRef: DestroyRef Utilizzato per interrompere automaticamente le sottoscrizioni quando il componente viene distrutto.
- isEditable: boolean Flag che indica se la pagina e in modalita modifica.
- result$: BehaviorSubject<ResultSplit | null> Subject che mantiene il risultato corrente mostrato in pagina (inizializzato da history.state).
- extractedEmployeeRows$: BehaviorSubject<ExtractedEmployeeInfoRow[]> Subject che contiene le righe pronte per la sezione dipendente estratto.
- dialogService: DialogService Servizio usato per aprire dialog dinamici (selezione dipendente e invio documento).
- messageService: MessageService Servizio usato per notifiche toast di successo/errore/info.
- ref: DynamicDialogRef | null Riferimento al dialog aperto, utile per ascoltare l'evento di chiusura.
- templates$: Observable<any> Stream dei template disponibili per il dialog di invio.
- result: ResultSplit | null (getter/setter) Accessor del risultato corrente sincronizzato con result$.
- pendingModifications: Partial<ResultSplit> Mappa delle modifiche locali non ancora salvate.
- pages: number Numero pagine documento padre passato tramite stato di navigazione.
- otherExtractedDocumentRows$: Observable<OtherExtractDocumentRow[]> Stream con gli altri documenti estratti appartenenti allo stesso padre.
- extractedEmployeeRows: ExtractedEmployeeInfoRow[] (getter/setter) Accessor delle righe dipendente sincronizzato con extractedEmployeeRows$.
- hasPendingModifications: boolean (getter) Indica se esistono modifiche locali pendenti da salvare.
- hasRecipientMatch: boolean (getter) Indica se il destinatario estratto e associato a un dipendente reale (recipientId > 0).

Metodi

- ngOnInit(): void Inizializza i template, costruisce le righe dipendente dal risultato corrente, ricarica il dettaglio dal backend, sottoscrive gli stream di aggiornamento e popola la lista documenti correlati.

- handleOpenOriginalPdf(): void Apre il PDF originale del documento padre; mostra errore se parentId non disponibile.
- handleOpenSplitPdf(): void Apre il PDF dello split corrente; mostra errore se id documento non disponibile.
- handleEditExtractedEmployeeInfo(): void Apre il dialog di selezione dipendente, aggiorna i metadati destinatario sul backend e applica il risultato aggiornato in pagina.
- buildExtractedEmployeeRows(result: ResultSplit | null): ExtractedEmployeeInfoRow[] Costruisce la riga informativa del dipendente (recipient, raw name, match, confidence) partendo dal risultato corrente.
- applyIncomingResult(updated: ResultSplit): void Aggiorna lo stato locale con il risultato ricevuto, rigenera le righe dipendente e sincronizza lo stato di navigazione nel browser.
- showDialog(): void Apre il dialog di invio documento, crea il sending con scheduling scelto e aggiorna lo stato del documento (Inviato/Programmato).
- resolveSentAt(optionValue: string): Date Converte l'opzione di scheduling selezionata (now, tomorrow_9am, day_after_9am, monday_9am) nella data/ora effettiva di invio.
- enableEditing(): void Abilita la modalita modifica azzerando eventuali modifiche pendenti.
- cancelEditing(): void Annulla la modalita modifica, scarta le modifiche locali e mostra notifica informativa.
- normalizeValue(value: string | number | undefined | null): string Normalizza valori eterogenei in stringa per confronti consistenti.
- onFieldModified(event: { field: keyof ResultSplit; value: string | number | undefined }): void Registra o rimuove una modifica locale confrontando il nuovo valore con quello originale del campo.
- saveChanges(): void Valida e salva le modifiche pendenti: invia aggiornamento range pagine e/o metadati, aggiorna lo stato locale e notifica l'esito.
- buildMetadataUpdates(modifications: Partial<ResultSplit>): Record<string, unknown> Costruisce il payload metadati conforme all'endpoint backend (mappando i campi del modello frontend).
- buildRangeUpdates(modifications: Partial<ResultSplit>, current: ResultSplit): page_start: number; page_end: number | null Estrae e converte le modifiche sul range pagine, restituendo null se assenti o non numeriche.
- isRangeValid(range: { page_start: number; page_end: number }, current: ResultSplit): boolean Verifica la validita del range pagine rispetto ai vincoli (interi, ordine corretto, limiti documento).

6.7.7 Storico Ai CoPilot



Componente UI che rappreseta la pagina che mostra lo storico dei documenti processati da AI Copilot, con filtri (ricerca, date, categoria, azienda) e azioni contestuali su ciascuna riga.
Attributi

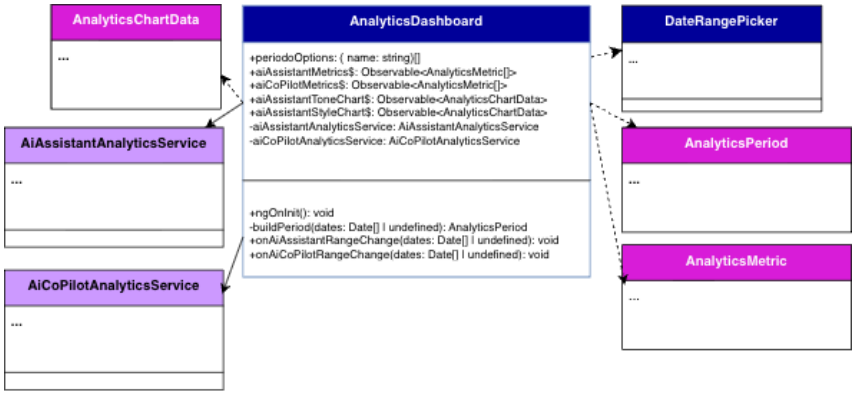
- router: Router Iniettato per la navigazione verso la pagina di anteprima del documento selezionato.
- aiCoPilotService: AiCoPilotService Servizio applicativo iniettato per caricare storico risultati, aziende e gestire le azioni sui documenti.
- destroyRef: DestroyRef Utilizzato per interrompere automaticamente le sottoscrizioni quando il componente viene distrutto.
- resultSplits: ResultSplit[] Collezione completa degli split ricevuti dal servizio (storico sorgente).
- parentNames: Record<number, string> Mappa id documento padre → nome documento originale, usata per arricchire le righe tabella.
- parentPageCounts: Record<number, number> Mappa id documento padre → numero pagine, usata in navigazione verso anteprima.
- documentsSubject: BehaviorSubject<ResultSplit[]> Subject interno con la lista documenti da sottoporre alla pipeline di filtro.
- searchSubject: BehaviorSubject<string> Subject interno per il testo di ricerca globale.
- datesSubject: BehaviorSubject<Date[] | undefined> Subject interno per il range date selezionato.
- selectedDocumentSubject: BehaviorSubject<string | undefined> Subject interno per il filtro tipologia documento/categoria.
- selectedCompanySubject: BehaviorSubject<string | undefined> Subject interno per il filtro azienda.
- filteredDocumentsSnapshot: ResultSplit[] Snapshot locale dell'ultima lista filtrata, usato come fallback in apertura dettaglio.
- Documents: ResultSplit[] Lista documenti mostrabile in tabella (derivata da resultSplits con mapping dedicato).
- FilteredDocuments$: Observable<ResultSplit[]> Stream osservabile della lista filtrata, ottenuto da combineLatest dei subject filtro.
- items: MenuItem[] Voci del menu contestuale per riga (Modifica, Elimina, Riprova).
- dates: Date[] | undefined Intervallo date selezionato nel filtro.

- searchvalue: string Valore testuale inserito nella ricerca globale.
- Companies: string[] Elenco nomi azienda disponibili nel filtro.
- selectedCompany: string | undefined Azienda attualmente selezionata nei filtri.
- DocumentType: string[] Elenco tipologie documento/categorie disponibili nel filtro.
- selectedDocument: string | undefined Tipologia documento/categoria attualmente selezionata.
- columns: { field: string; header: string; type?: string }[] Configurazione colonne della tabella storico.

Metodi

- ngOnInit(): void Inizializza menu e caricamento dati (storico e aziende), sottoscrive gli stream del servizio e mantiene aggiornato lo snapshot filtrato.
- onSearchChange(value: string): void Aggiorna il testo di ricerca e notifica il relativo subject per ricalcolare i filtri.
- onDateChange(dates: Date[]): void Aggiorna il range date e notifica il relativo subject per ricalcolare i filtri.
- onDocumentChange(document: string | number): void Aggiorna la tipologia documento/categoria selezionata e notifica il subject dedicato.
- onCompanyChange(company: string | number): void Aggiorna l'azienda selezionata e notifica il subject dedicato.
- rebuildDocuments(): void Ricostruisce la lista righe tabella a partire dagli split sorgente e aggiorna le opzioni filtro tipologia.
- normalizeDate(value: Date): number Normalizza una data al solo giorno (azzerando ora/minuti) per confronti consistenti nei filtri range.
- formatConfidenceForSearch(value: unknown): string Converte/confina la confidenza in stringa normalizzata (una cifra decimale) per la ricerca testuale.
- toStoricoRow(split: ResultSplit): any Mappa un ResultSplit nel formato riga usato dalla tabella storico, includendo il nome documento originale.
- onTableMenuAction(event: { row: ResultSplit; item: MenuItem }): void Gestisce azioni menu su riga: Modifica (apre anteprima), Elimina (cancella documento padre), Riprova (solo se stato Failed).
- onRowRemoved(row: ResultSplit): void Rimuove la riga dalla lista locale e aggiorna lo stream documenti.
- navigateToResult(targetRow?: ResultSplit): void Naviga alla pagina anteprima passando il risultato selezionato (o il primo filtrato) e il numero pagine del documento padre.

6.7.8 Analytics dashboard



Componente UI che rappresenta la pagina che mostra la dashboard analytics con metriche aggregate per AI Assistant e AI CoPilot, includendo grafici di utilizzo tono/stile e filtri per intervallo

temporale.
Attributi

- periodoOptions: { name: string }[] Elenco delle opzioni di periodo predefinite disponibili nella UI (Sempre, mese corrente, ultime settimane, ultimi mesi, ultimo anno).
- aiAssistantMetrics$: Observable<AnalyticsMetric[]> Stream delle metriche aggregate dell'area AI Assistant, aggiornato in base al periodo selezionato.
- aiCoPilotMetrics$: Observable<AnalyticsMetric[]> Stream delle metriche aggregate dell'area AI CoPilot, aggiornato in base al periodo selezionato.
- aiAssistantToneChart$: Observable<AnalyticsChartData> Stream dati per il grafico di utilizzo dei toni nell'area AI Assistant.
- aiAssistantStyleChart$: Observable<AnalyticsChartData> Stream dati per il grafico di utilizzo degli stili nell'area AI Assistant.
- aiAssistantAnalyticsService: AiAssistantAnalyticsService Servizio iniettato per recuperare metriche e dati grafici relativi ad AI Assistant.
- aiCoPilotAnalyticsService: AiCoPilotAnalyticsService Servizio iniettato per recuperare metriche relative ad AI CoPilot.

Metodi

- constructor(aiAssistantAnalyticsService: AiAssistantAnalyticsService, aiCoPilotAnalyticsService: AiCoPilotAnalyticsService) Inizializza i servizi analytics e collega subito gli stream dei grafici tono/stile di AI Assistant.
- ngOnInit(): void Carica le metriche iniziali senza filtro temporale, impostando il periodo di default a "sempre" per entrambi i domini analytics.
- buildPeriod(dates: Date[] | undefined): AnalyticsPeriod Costruisce il payload periodo da inviare ai servizi: se il range non e completo restituisce "sempre", altrimenti usa startDate ed endDate.
- onAiAssistantRangeChange(dates: Date[] | undefined): void Aggiorna le metriche AI Assistant quando cambia il date range; ignora aggiornamenti con range incompleto.
- onAiCoPilotRangeChange(dates: Date[] | undefined): void Aggiorna le metriche AI CoPilot quando cambia il date range; ignora aggiornamenti con range incompleto.

DESIGN PATTERN

7 Design Pattern

7.1 Design Pattern Backend

La progettazione del sistema si è avvalsa di consolidati design pattern, suddivisi in due categorie: quelli implementati esplicitamente nella logica di dominio e quelli ereditati intrinsecamente dal framework Ruby on Rails$_G$.

L'architettura del backend$_G$ riflette una separazione netta tra le responsabilità: la creazione e composizione degli oggetti è centralizzata nel *Container* tramite dependency injection, che fornisce ai *Service Object* tutte le dipendenze necessarie senza che questi debbano istanziarle direttamente.
Quando la logica varia in base al tipo di input, il pattern *Strategy* seleziona dinamicamente l'algoritmo appropriato; quando occorre produrre famiglie di oggetti correlati, *Abstract Factory* isola la costruzione dal client.
La persistenza, quando diventa complessa, è gestita dai *Repository*, che centralizzano le transizioni di stato e garantiscono la correttezza in presenza di elaborazioni parallele tramite pessimistic locking.
Il livello di presentazione è affidato ai *Serializer*, che trasformano i modelli in strutture JSON

disaccoppiate dai dettagli di persistenza.
Il framework contribuisce con *Active Record* per il mapping oggetti-tabelle e con *MVC* come struttura portante dell'intera applicazione.

La tabella 18 riassume tutti i pattern documentati in questa sezione.

Pattern	**Categoria**	**GoF / POEAA**	**Dove applicato**
Service Object	Comportamentale	—	`ProcessDataItem`, `ProcessSplitRun`, `ProcessGenericFile`, `AIGeneratorService`
Repository	Architetturale	POEAA	`DataItemRepository`, `SplitRunRepository`
Container / DI	Architetturale	—	`DocumentProcessing::Container`, `AiGeneratorContainer`
Strategy	Comportamentale	GoF	`file_processor` (CSV / Image)
Abstract Factory	Creazionale	GoF	`SetterFactory`
Serializer	Strutturale	—	`PostSerializer`, `ToneSerializer`, `StyleSerializer`, `GeneratedDatumSerializer`
Active Record	Architetturale	POEAA	Tutti i modelli Rails (`UploadedDocument`, `ExtractedDocument`, `Company`, `Employee`, ecc.)
MVC	Architetturale	—	Intera applicazione Rails (Controller, Model, View JSON)

Tabella 16: Riepilogo dei design pattern adottati

7.1.1 Service Object

Intent: Incapsulare un'operazione di business complessa in una classe dedicata con un'unica responsabilità, esponendo un'interfaccia$_G$ uniforme `call(...)`*, o, in alternativa, un set di metodi pubblici che rappresentano le fasi dell'operazione. Nella sua variante* ***orchestratore****, il Service Object coordina più collaboratori iniettati senza conoscerne i dettagli interni, mantenendo comunque una singola responsabilità di alto livello.*

Il pattern *Service Object* costituisce l'ossatura della logica applicativa. Ogni servizio ha una responsabilità precisa e non porta stato persistente tra invocazioni distinte.

Esempi:

7.1.1.1 ProcessDataItem

Orchestratore centrale della pipeline OCR$_G$: coordina riconoscimento ottico dei caratteri, estrazione dati tramite LLM$_G$, calcolo della confidenza$_G$, risoluzione del destinatario e persistenza del risultato. Nonostante sia un metodo che ha tante dipendenze e passi, la sua responsabilità è univoca: processare un documento e estrarre i dati, delegando ogni fase a collaboratori dedicati iniettati tramite il costruttore.

Listing 1: ProcessDataItem – struttura del metodo `call`

```
# app/services/document_processing/process_data_item.rb
module DocumentProcessing
  class ProcessDataItem
    def initialize(
      data_item_repository:,
      file_storage:,
      ocr_service:,
      data_extractor:,
      recipient_resolver:,
      confidence_calculator_factory:,
      extracted_metadata_builder_factory:
    )
      @data_item_repository = data_item_repository
      @file_storage = file_storage
      @ocr_service = ocr_service
      @data_extractor = data_extractor
      @recipient_resolver = recipient_resolver
      @confidence_calculator_factory = confidence_calculator_factory
      @extracted_metadata_builder_factory = extracted_metadata_builder_factory
    end

    def call(file_path:, job_id:, processing_item_id: nil, extracted_document_id: nil)
      start_time = Time.now
      run = data_item_repository.find_run_by_job_id(job_id)
      item = processing_item_id ? data_item_repository.find_processing_item(
          processing_item_id) : nil
      extracted_document = resolve_extracted_document(extracted_document_id, item)

      return if data_item_repository.terminal_item?(item)

      data_item_repository.mark_item_in_progress!(item)
      data_item_repository.mark_extracted_document_in_progress!(extracted_document)

      ocr_result = ocr_service.full_ocr(file_path)
      full_text = ocr_result[:text]
      ocr_lines = ocr_result[:lines]

      extracted_data = data_extractor.extract(full_text)
      recipient_names = extracted_data[:recipients]
      # normalizza al destinatario singolo
      recipient = Array(recipient_names).compact.first
      extracted_document_data = extracted_data[:metadata]
      llm_confidence = extracted_data[:llm_confidence]
      final_global_confidence = confidence_calculator_factory.call(
        ocr_lines: ocr_lines,
        recipient_names: recipient_names,
        metadata: extracted_document_data,
        llm_confidence: llm_confidence,
        uploaded_document: extracted_document&.uploaded_document
      ).global_confidence

      resolution = recipient_resolver.resolve(recipient_names:, raw_text: full_text)

      data_item_repository.mark_item_done!(item:, resolution:)
      duration = Time.now - start_time
      update_extracted_document_success(extracted_document, resolution,
          extracted_document_data, recipient, final_global_confidence, duration)
```

```
    if job_id.present?
      ActiveSupport::Notifications.instrument("document_processing.lifecycle",
        job_id: job_id,
        **build_success_payload(
          filename: File.basename(file_path),
          ocr_text: full_text,
          recipient: recipient,
          extracted_document_data: extracted_document_data,
          extracted_confidence: final_global_confidence,
          matched_recipient: format_employee(resolution.employee),
          extracted_document_id: extracted_document&.id,
          document_index: item&.sequence,
          total_documents: run&.total_documents
        )
      )
    end
  rescue StandardError => e
    data_item_repository.mark_item_failed(item:, error_message: e.message)
    data_item_repository.mark_extracted_document_failed(extracted_document:,
        error_message: e.message)

    if job_id.present?
      ActiveSupport::Notifications.instrument("document_processing.lifecycle",
        job_id: job_id,
        **build_error_payload(
          message: e.message,
          filename: file_path ? File.basename(file_path) : nil,
          extracted_document_id: extracted_document&.id,
          document_index: item&.sequence,
          total_documents: run&.total_documents
        )
      )
    end
  ensure
    increment_progress(run, job_id)
    file_storage.delete(file_path) if file_path && file_storage.exist?(file_path)
  end
 end
end
```

7.1.1.2 ProcessSplitRun

Gestisce la fase di splitting di un PDF: divide il documento in sotto-PDF, crea i record `ProcessingItem` / `ExtractedDocument` associati e accoda un job `DataExtractionJob` per ciascuna porzione.

7.1.1.3 ProcessGenericFile

Gestisce file non-PDF (CSV, immagini) tramite il pattern Strategy (si veda la sezione 7.1.4): seleziona il processore appropriato in base al `file_kind` e applica gli override utente con confidenza$_G$ 1.0.

7.1.1.4 AiGeneratorService

SI occupa di fare le chiamate al modello LLM$_G$ per generare i testi e immagini in uscita, applicando i toni e stili selezionati dall'utente. Orchestra la costruzione dei prompt$_G$, l'invocazione del modello e il salvataggio del risultato, restituendo l'output della richiesta.

7.1.2 Repository

Intent (POEAA – Fowler): Mediare tra il dominio e il livello di mapping dati usando un'interfaccia$_G$ simile a una collezione per accedere agli oggetti del dominio.

I Repository$_G$ centralizzano tutte le transizioni di stato e le query di persistenza, isolando il resto dell'applicazione da ActiveRecord. Rails ha uno degli ORM piú completi sul mercato e per query semplici è possibile utilizzare direttamente i modelli, in quanto le operazioni sono native e non ha senso introdurre un ulteriore livello di astrazione che non aggiunge valore. Tuttavia, per operazioni più complesse (es. transizioni di stato con lock, query aggregate) è consigliabile incapsulare la logica in un Repository$_G$ dedicato, evitando duplicazioni e garantendo la correttagestione della concorrenza, permettendo di separare nettamente la logica di dominio dalla persistenza.

Listing 2: DataItemRepository – transizioni di stato con pessimistic locking

```
# app/services/document_processing/persistence/data_item_repository.rb
module DocumentProcessing
  module Persistence
    class DataItemRepository

      # Trova un ProcessingItem per id
      def find_processing_item(id)
        ProcessingItem.find(id)
      end

      # Transizione atomica in_progress (lock pessimistico)
      def mark_item_in_progress!(item)
        item.with_lock do
          item.reload
          return if item.done? || item.failed?
          item.update!(status: "in_progress")
        end
      end

      # Transizione done con aggiornamento employee
      def mark_item_done!(item:, resolution:)
        item.with_lock do
          item.update!(status: "done")
          if resolution.matched?
            item.extracted_document
                .update!(matched_employee: resolution.employee)
          end
        end
      end

      # Aggiorna il progresso del ProcessingRun
      def update_progress!(run)
        run.with_lock do
          total = run.processing_items.count
          done  = run.processing_items.where(status: %w[done failed]).count
```

```
        pct   = total.positive? ? (done.to_f / total * 100).round : 0
        run.update!(progress_percentage: pct)
        run.update!(status: "completed") if done == total
      end
    end

    def transaction(&block)
      ApplicationRecord.transaction(&block)
    end
  end
 end
end
```

Il *Pessimistic Locking* (`with_lock`) è fondamentale per garantire la correttezza delle transizioni di stato quando più job elaborano documenti in parallelo: senza lock, due worker potrebbero leggere lo stesso stato `queued` e procedere simultaneamente.

Listing 3: SplitRunRepository – creazione artefatti di splitting

```
# app/services/document_processing/persistence/split_run_repository.rb
def create_split_artifacts!(run:, split_results:)
  split_results.each_with_index do |result, idx|
    extracted = ExtractedDocument.create!(
      uploaded_document: run.uploaded_document,
      sequence:  idx + 1,
      page_start: result[:page_start],
      page_end:  result[:page_end],
      status:    "queued"
    )
    ProcessingItem.create!(
      processing_run:  run,
      extracted_document: extracted,
      sequence:        idx + 1,
      status:          "queued"
    )
  end
end
```

7.1.3 Container e Dependency Injection

Intent: Centralizzare la creazione e la composizione delle dipendenze in un oggetto dedicato (IoC Container), permettendo ai client di ricevere le dipendenze senza istanziarle direttamente.

7.1.3.1 DocumentProcessing::Container

Il Container è il cuore dell'architettura: gestisce l'intero grafo di dipendenze del modulo$_G$ OCR$_G$, inizializzando i servizi con lazy memoization e supportando l'override di qualsiasi componente per i test$_G$.

Listing 4: DocumentProcessing::Container – composizione delle dipendenze

```
# app/services/document_processing/container.rb
module DocumentProcessing
  class Container
    def initialize(
      ocr_service_class:     DocumentProcessing::OcrService,
```

```
    data_extractor_class:   DocumentProcessing::DataExtractor,
    pdf_splitter_class:     DocumentProcessing::PdfSplitter,
    job_class:              PdfSplitJob,
    **overrides
  )
    @ocr_service_class  = ocr_service_class
    @data_extractor_class = data_extractor_class
    @pdf_splitter_class = pdf_splitter_class
    @job_class          = job_class
    @overrides          = overrides
  end

  # Memoizzazione lazy: il client Textract viene creato
  # solo al primo accesso
  def ocr_service
    @ocr_service ||= @ocr_service_class.new(
      client: Aws::Textract::Client.new(region: aws_region)
    )
  end

  # Factory Method: seleziona il processore corretto per tipo file
  def file_processor(file_kind)
    case file_kind.to_s
    when "csv"  then csv_processor
    when "image" then image_processor
    else raise ArgumentError, "file_kind non supportato: #{file_kind}"
    end
  end

  # Composizione complessa: process_data_item_service aggrega
  # 8 dipendenze diverse
  def process_data_item_service
    @process_data_item_service ||= ProcessDataItem.new(
      ocr_service:              ocr_service,
      data_extractor:           data_extractor,
      recipient_resolver:       recipient_resolver,
      confidence_calculator_factory: method(:confidence_calculator),
      data_item_repository:     data_item_repository,
      metadata_builder:         metadata_builder,
      file_storage:             file_storage
    )
  end

  def initialize_processing_command
    Commands::InitializeProcessing.new(
      upload_manager:     upload_manager,
      file_storage:       file_storage,
      data_item_repository: data_item_repository,
      job_class:          @job_class
    )
  end

  private

  def aws_region
    ENV.fetch("AWS_REGION", "eu-west-1")
  end
end
```

```
end
```

Questo approccio consente di testare qualsiasi servizio sostituendo una singola dipendenza tramite override del costruttore, senza modificare il codice di produzione:

Listing 5: Utilizzo del Container nei test – override delle dipendenze

```
# test/controllers/documents_controller_test.rb
def fake_container(overrides = {})
  init_cmd = fake_command(
    overrides.fetch(:init_result,
      { status: "queued", job_id: "j1", uploaded_document_id: 1 })
  )
  Object.new.tap do |c|
    c.define_singleton_method(:initialize_processing_command) { init_cmd }
    c.define_singleton_method(:file_storage) {
      overrides.fetch(:file_storage, DocumentProcessing::Persistence::FileStorage.new)
    }
    c.define_singleton_method(:db_manager) {
      overrides.fetch(:db_manager, real_db_manager)
    }
  end
end

# Nel test: il Container reale viene sostituito con il fake
stub_new(DocumentProcessing::Container, fake_container) do
  post split_documents_path, params: { pdf: "dummy" }
end
```

7.1.3.2 AiGenerator::AiGeneratorContainer

Container analogo per il modulo$_G$ di generazione AI$_G$: gestisce i servizi Bedrock$_G$ per testo e immagini, il data manager e la factory dei parametri.

Listing 6: AiGeneratorContainer – servizi Bedrock$_G$ con configurazione da ENV

```
# app/services/ai_generator/ai_generator_container.rb
module AiGenerator
class AiGeneratorContainer
  REGION_FALLBACK = ENV.fetch("AWS_REGION", "us-east-1")

  def aiGeneratorService
    imgGenerator = self.imgGenerator
    textGenerator = self.textGenerator
    aiGeneratorDataManager = self.aiGeneratorDataManager
    setterFactory = self.setterFactory
    textResponseValidator = self.textResponseValidator
    @aiGeneratorService ||= AIGeneratorService.new(imgGenerator, textGenerator,
        aiGeneratorDataManager, setterFactory, textResponseValidator)
  end

  private

  def aiGeneratorDataManager
    @aiGeneratorDataManager ||= AIGeneratorDataManager.new
  end

  def setterFactory
```

```
    @setterFactory ||= SetterFactory.new
  end

  def textResponseValidator
    @textResponseValidator ||= TextResponseValidator.new
  end

  def textGenerator
    @textGenerator ||= TextGeneratorService.new(region: text_generation_region)
  end

  def imgGenerator
    @imgGenerator ||= ImageGeneratorService.new(region: image_generation_region)
  end

  def text_generation_region
    ::BEDROCK_CONFIG_GENERATION["region"] || REGION_FALLBACK
  end

  def image_generation_region
    ::BEDROCK_CONFIG_IMAGE_GENERATION["region"] || REGION_FALLBACK
  end
end
end
```

7.1.4 Strategy

Intent (GoF): Definire una famiglia di algoritmi, incapsulare ciascuno e renderli intercambiabili. Lo Strategy permette di variare l'algoritmo indipendentemente dai client che lo utilizzano.

Il metodo `file_processor` del Container implementa la selezione dinamica della strategia di elaborazione in base al tipo di file:

Listing 7: Strategy – selezione del processore per tipo file

```
# In DocumentProcessing::Container
def file_processor(file_kind)
  case file_kind.to_s
  when "csv"  then csv_processor # Strategia CSV
  when "image" then image_processor # Strategia immagine
  else raise ArgumentError, "file_kind non supportato: #{file_kind}"
  end
end

def csv_processor
  @csv_processor ||= CsvProcessor.new(
    recipient_resolver: recipient_resolver,
    metadata_builder: metadata_builder
  )
end

def image_processor
  @image_processor ||= ImageProcessor.new(
    ocr_service:      ocr_service,
    data_extractor:   data_extractor,
    recipient_resolver: recipient_resolver
  )
end
```

Entrambi i processori condividono la stessa interfaccia$_G$ `call(file_path)` e restituiscono un hash uniforme con `metadata`, `confidence`, `recipient` e `ocr_text`. Il chiamante (`ProcessGenericFile`) non conosce quale strategia stia usando:

Listing 8: ProcessGenericFile – polimorfismo tramite Strategy

```
# Il servizio riceve il processore dal Container: non conosce
# se e' CSV o immagine
def call(file_path:, job_id:, uploaded_document_id:, ...)
  uploaded = UploadedDocument.find(uploaded_document_id)
  # file_processor e' gia' la strategia corretta
  result = @file_processor.call(file_path)

  # Override utente: confidenza\G{} 1.0 (certezza massima)
  result[:metadata].merge!(override_company: override_company) if override_company
  create_extracted_document(uploaded, result, ...)
end
```

7.1.5 Abstract Factory

Intent (GoF): Fornire un'interfaccia$_G$ per creare famiglie di oggetti correlati o dipendenti senza specificarne le classi concrete.

`SetterFactory` è la factory concreta del modulo$_G$ AI$_G$: produce una coppia di oggetti correlati (`TextParamsSetterService` e `ImageParamsSetterService`) che condividono la stessa interfaccia$_G$ (`valid?`, `getData`, metodo `build*`) e collaborano all'interno della stessa pipeline di generazione.

Listing 9: SetterFactory – creazione della famiglia testo/immagine

```
# app/services/ai_generator/setter_factory.rb
module AiGenerator
  class SetterFactory
    def create_text_setter(params_data)
      TextParamsSetterService.new(params_data)
    end

    def create_image_setter(params_data)
      ImageParamsSetterService.new(params_data)
    end
  end
end
```

Il client `AIGeneratorService` riceve la factory via dependency injection e invoca i due metodi senza conoscere le classi concrete:

Listing 10: AIGeneratorService – utilizzo della factory senza dipendenza diretta

```
# app/services/ai_generator/ai_generator_service.rb
def create_content(generationID)
  # ...recupero dati...

  textSetter = @setterFactory.create_text_setter({ prompt:, toneDescription:,
                styleDescription:, companyName:, companyDescription: })
  imageSetter = @setterFactory.create_image_setter({ prompt:,
                width: nil, height: nil, seed: nil })

  validate_setters!(textSetter, imageSetter) # stessa interfaccia valid?/getData

  textResult = createText(textSetter, generationData.prompt)
```

```
  imageResult = createImage(imageSetter, textResult)
  # ...
end
```

Sostituire `SetterFactory` con una variante (es. per test$_G$ o per un modello diverso) non richiede alcuna modifica al client: è sufficiente iniettare una factory alternativa nel costruttore di `AIGeneratorService`.

7.1.6 Serializer

Intent: Separare la logica di presentazione/formattazione dal modello dati, producendo rappresentazioni JSON coerenti e disaccoppiate dai dettagli di persistenza.

I Serializer producono la struttura JSON delle collezioni REST con chiavi camelCase coerenti con il frontend$_G$:

Listing 11: PostSerializer – serializzazione con join eagerly loaded

```
# app/serializers/post_serializer.rb
module PostSerializer
  def self.serialize_collection(posts)
    {
      posts: posts.map do |post|
        gd = post.generated_datum
        {
          id:             post.id,
          title:          post.title,
          postText:       post.body_text,
          imgPath:        post.img_path,
          dateTime:       post.date_time,
          generatedDatumId: post.generated_datum_id,
          toneId:         gd&.tone_id,
          toneName:       gd&.tone&.name,
          is_tone_active: gd&.tone&.is_active,
          styleId:        gd&.style_id,
          styleName:      gd&.style&.name,
          is_style_active: gd&.style&.is_active,
          companyId:      gd&.company_id,
          companyName:    gd&.company&.name,
          rating:         gd&.rating
        }
      end
    }
  end
end
```

7.1.7 Pattern ereditati dal framework

7.1.7.1 Model-View-Controller (MVC)

L'architettura dell'intera applicazione è fondata sul pattern MVC imposto da Ruby on Rails$_G$:

- **Model**: gestisce la logica dei dati, le validazioni e le associazioni tra entità (es. `ExtractedDocument`, `UploadedDocument`, `Company`).
- **View**: in questo ecosistema API-only, la view è rappresentata da un risultato JSON serializzato.

- **Controller**: orchestra il flusso, validando i parametri HTTP e delegando ai Command o ai Service Object la logica di business.

Non si tratta del tradizionale approccio MVC basato su template$_G$HTML generati lato server — in questo caso, infatti, la "view" è rappresentata da dati in formato JSON — ma la struttura di base rimane comunque riconoscibile. I controller continuano a svolgere il ruolo di intermediari tra le richieste HTTP e la logica applicativa, che è incapsulata nei servizi e nei modelli.
Ci troviamo però di fronte a un approccio più moderno, caratterizzato da una netta separazione tra backend$_G$ e frontend$_G$. Il backend$_G$ si limita a esporre API$_G$ RESTful, senza occuparsi del rendering o della presentazione dei dati. Questo consente di ottenere un'architettura più modulare, scalabile e manutenibile.
Il rendering viene invece delegato ad Angular$_G$, che gestisce la visualizzazione lato client (client-side rendering), evitando di sovraccaricare il server. Questo si differenzia dal classico MVC, in cui il rendering avviene lato server (server-side rendering).

Questo modello dunque rappresenta oggi uno standard per applicazioni sviluppate con Ruby on Rails$_G$ e offre diversi vantaggi: maggiore scalabilità$_G$, flessibilità nella scelta del frontend$_G$, interfacce più reattive grazie al paradigma delle Single Page Application (SPA), distribuzione del carico sul client e possibilità di riutilizzare il backend$_G$ per applicazioni mobile o integrazioni con servizi di terze parti.

Listing 12: DocumentsController – Controller MVC che delega ai Command

```
# app/controllers/documents_controller.rb
class DocumentsController < ActionController::Base
  skip_before_action :verify_authenticity_token

  # POST /documents/split
  def split
    result = initialize_processing_command.call(
      file:           params[:pdf],
      company:        params[:company],
      category:       params[:category],
      competence_period: params[:competence_period]
    )
    if result.is_a?(Hash) && result[:ok] == false
      return render_error(result[:message])
    end
    render json: {
      status:            result[:status],
      job_id:            result[:job_id],
      uploaded_document_id: result[:uploaded_document_id]
    }
  rescue StandardError => e
    render_error("Errore: #{e.message}")
  end

  # DELETE /documents/uploads/:id
  def destroy_upload
    uploaded = UploadedDocument.find(params[:id])
    file_storage.delete(uploaded.storage_path) if
      file_storage.exist?(uploaded.storage_path)
    uploaded.destroy! # cascade -> ExtractedDocument rimossi
    render json: { status: "ok", message: "Documento eliminato" }
  rescue ActiveRecord::RecordNotFound
    render json: { status: "error",
```

```
                message: "Documento sorgente non trovato" },
         status: :not_found
  end

  private

  # Factory Method: il Container costruisce il Command
  def initialize_processing_command
    document_processing_container.initialize_processing_command
  end

  def document_processing_container
    @document_processing_container ||= DocumentProcessing::Container.new
  end
end
```

7.1.7.2 Active Record

Ogni classe Modello mappa una tabella del database (pattern *Active Record* di Martin Fowler), nascondendo la complessità SQL e fornendo un'interfaccia$_G$ orientata agli oggetti per operazioni CRUD, relazioni e validazioni:

Listing 13: Modelli principali – associazioni, validazioni e predicate methods

```
# app/models/uploaded_document.rb
class UploadedDocument < ApplicationRecord
  has_many :extracted_documents, dependent: :destroy # cascade delete
  belongs_to :employee, optional: true

  validates :original_filename, presence: true
  validates :storage_path,   presence: true
  validates :checksum,       presence: true, uniqueness: true
  validates :file_kind,      inclusion: { in: %w[pdf csv image] }, allow_nil: true
end

# app/models/extracted_document.rb
class ExtractedDocument < ApplicationRecord
  belongs_to :uploaded_document
  belongs_to :matched_employee, class_name: 'User', optional: true

  STATUSES = %w[queued in_progress done failed sent validated].freeze

  validates :sequence,           presence: true
  validates :page_start, :page_end, presence: true,
                                 numericality: { greater_than: 0 }
  validates :status, inclusion: { in: STATUSES }, allow_nil: true
  validate :end_page_must_be_after_start

  # Predicate methods per leggibilita'
  def done?     = status == "done"
  def failed?   = status == "failed"
  def validated? = status == "validated"

  private
  def end_page_must_be_after_start
    return if page_start.blank? || page_end.blank?
    return if page_end >= page_start
```

```
    errors.add(:page_end, 'deve essere maggiore o uguale a page_start')
  end
end

# app/models/employee.rb
class Employee < ApplicationRecord
  belongs_to :user
  belongs_to :company
end

# app/models/company.rb
class Company < ApplicationRecord
  has_many :tones,        dependent: :destroy
  has_many :styles,       dependent: :destroy
  has_many :generated_data, dependent: :destroy
  validates :description, length: { maximum: 500 }
end
```

Le associazioni `dependent: :destroy` implementano implicitamente la gestione delle chiavi esterne a livello applicativo: la cancellazione di un `UploadedDocument` propaga automaticamente la rimozione di tutti gli `ExtractedDocument` collegati.

7.2 Design Pattern Frontend

Il frontend$_G$ Angular adotta un'architettura modulare basata su componenti e servizi per la gestione dello stato, seguendo i principi di separazione delle responsabilità e riusabilità.
I componenti UI sono progettati per essere il più possibile presentazionali, delegando la logica di business ai servizi iniettati, che a loro volta interagiscono con API$_G$ RESTful esposte dal backend$_G$.
Il pattern *Observer* è ampiamente utilizzato tramite RxJS$_G$: i servizi espongono stream di dati (Observable) a cui i componenti si sottoscrivono per ricevere aggiornamenti in tempo reale, mantenendo sincronizzata la UI con lo stato dell'applicazione.
Il pattern *Facade* è implementato dai servizi che incapsulano la complessità delle chiamate HTTP e della gestione dello stato, offrendo un'interfaccia$_G$ semplice e coerente ai componenti.

La tabella 18 riassume tutti i pattern documentati in questa sezione.

Pattern	**Categoria**	**GoF / POEAA**	**Dove applicato**
Facade	Strutturale	GoF	AiCopilotService, AiAssistantService
MVVM	Architetturale		Intera applicazione Angular

Tabella 18: Riepilogo dei design pattern adottati

7.2.1 Facade

Intent (GoF): fornire ai componenti un'interfaccia$_G$ semplificata per svolgere operazioni complesse, nascondendo la complessità sottostante.
`AiCopilotService` e `AiAssistantService` fungono da Facade per i componenti Angular, incapsulando la logica di interazione con le API$_G$ RESTful del backend$_G$.
Tali service offrono metodi ad alto livello (es. (fetchResultsHistory), (uploadFiles)) che orchestrano chiamate HTTP, gestione degli errori e aggiornamento dello stato, permettendo ai

componenti di concentrarsi esclusivamente sulla presentazione dei dati e sull'interazione con l'utente, senza doversi preoccupare dei dettagli di implementazione delle API$_G$ o della gestione dello stato globale.

7.2.2 Model-View-ViewModel (MVVM)

Si rimanda alla spiegazione nella sezione 6.

8 Database

8.1 Schema ER

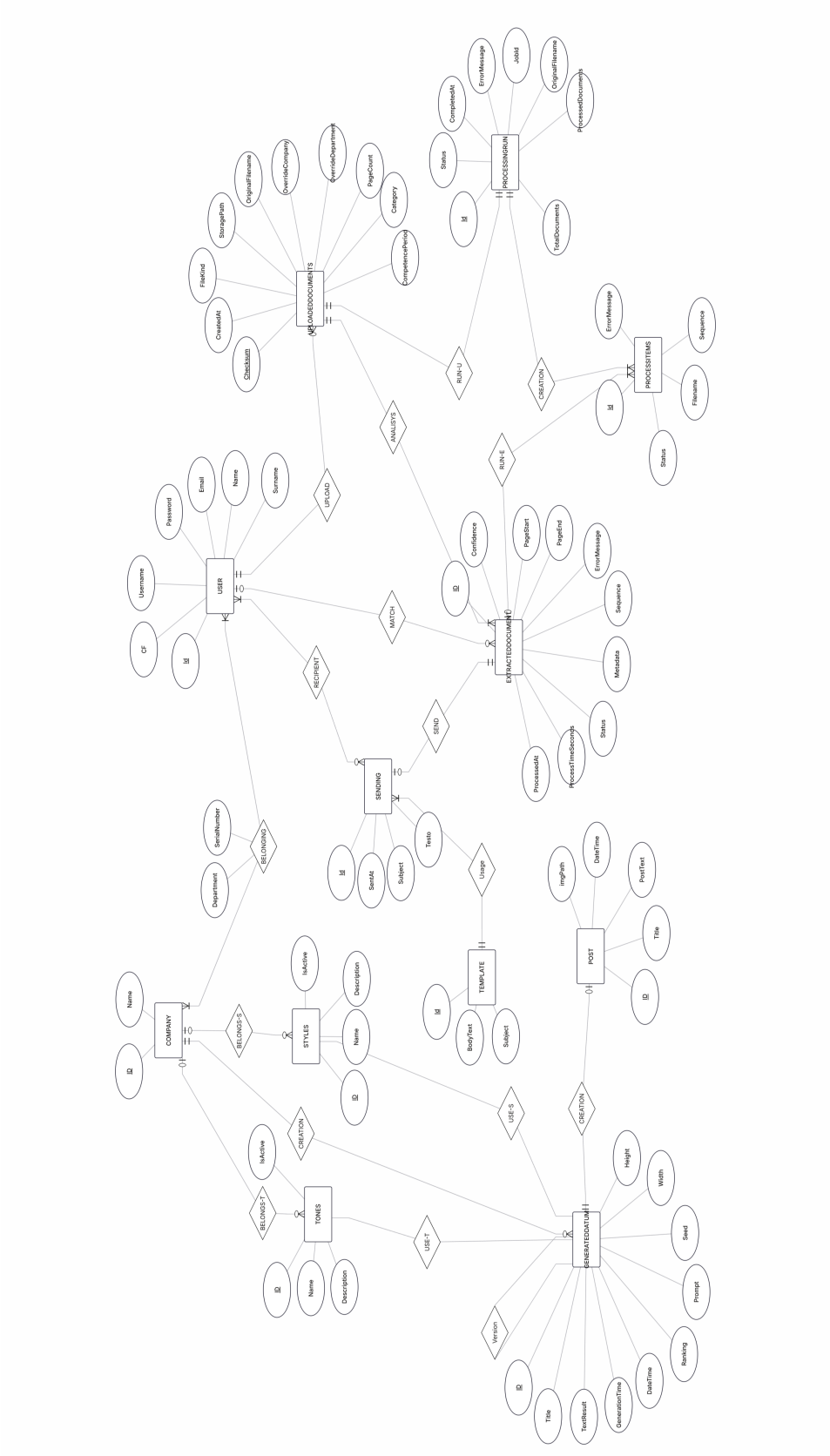


Figura 19: Schema ER

8.2 Schema RS

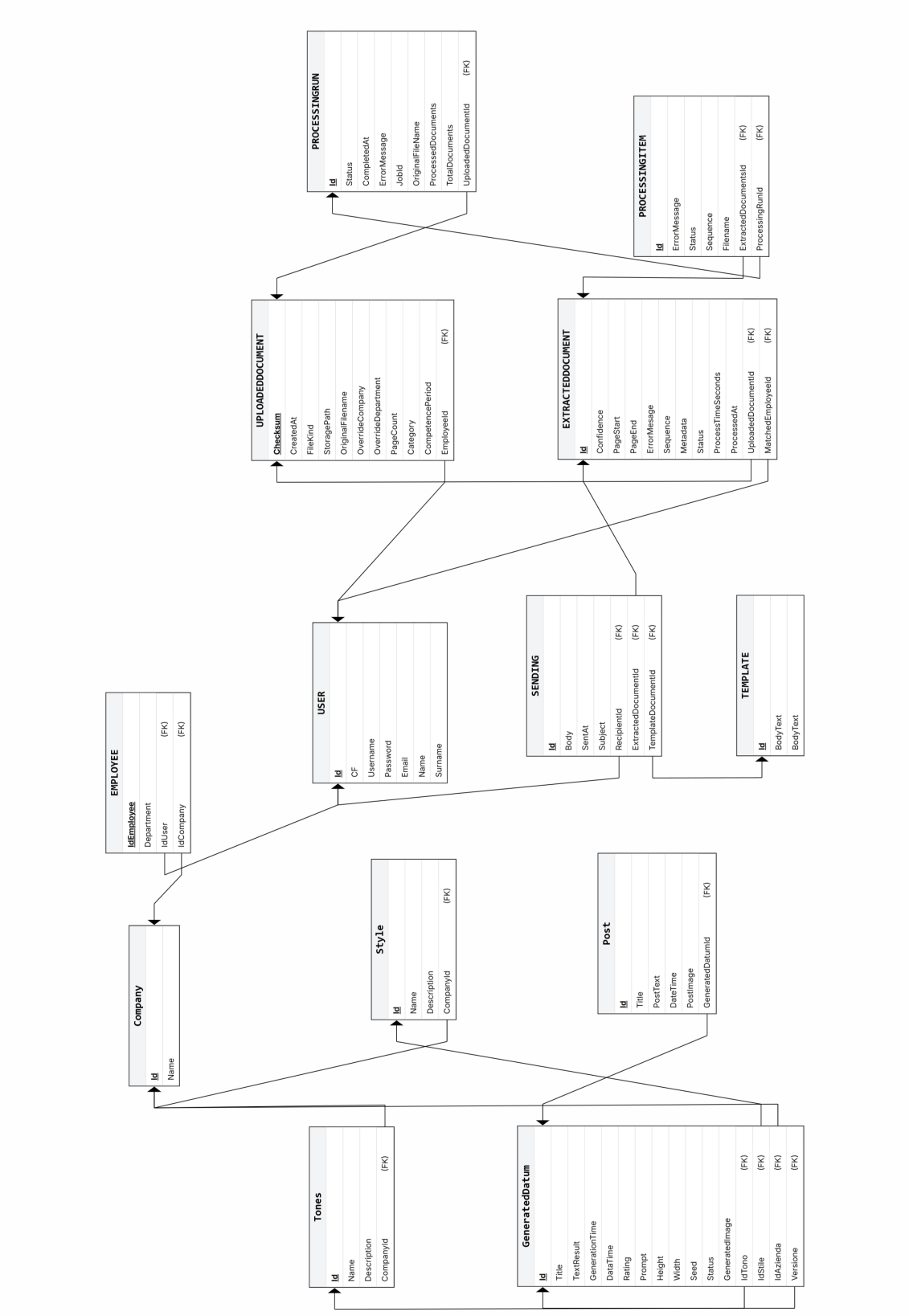


Figura 20: Schema RS

8.3 Schema ERD (Entity-Relationship Diagram)

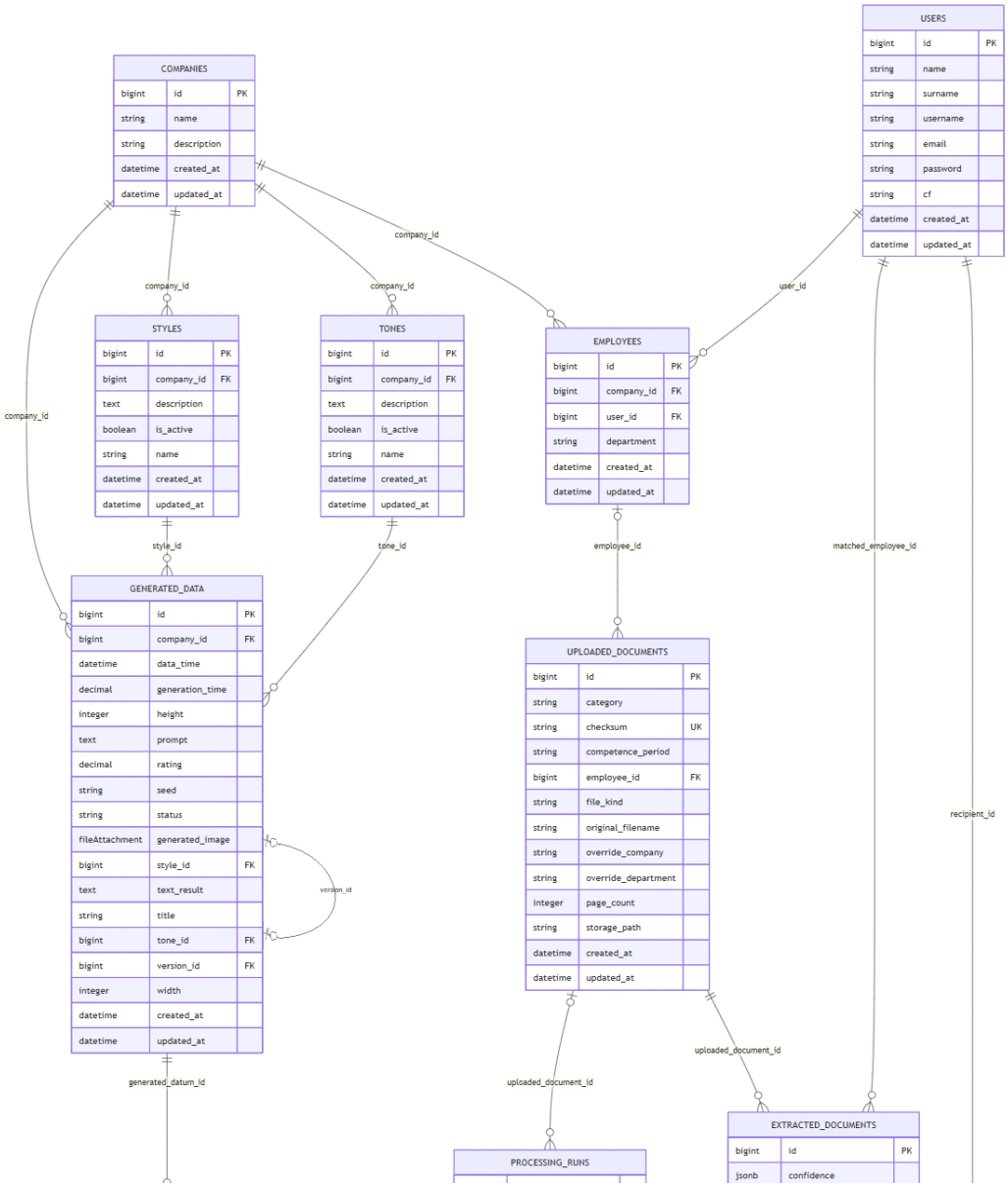


Figura 21: Schema ERD1

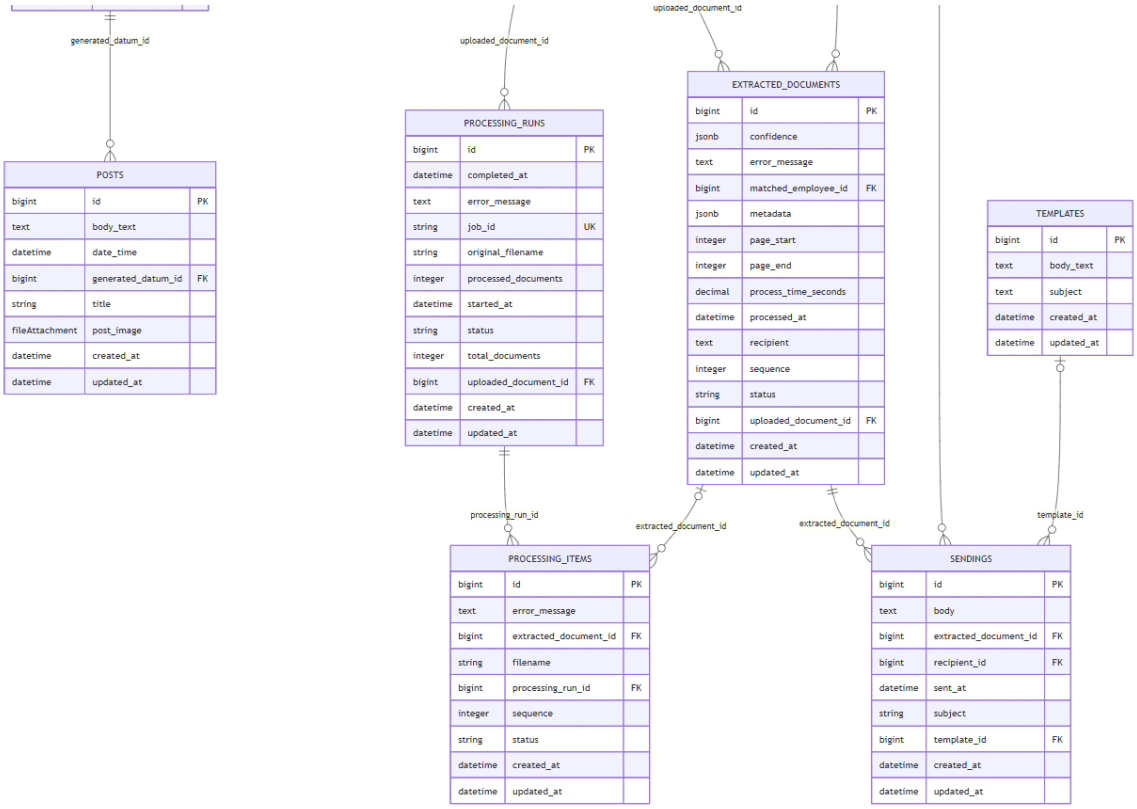


Figura 22: Schema ERD2

8.4 Descrizione tabelle

8.4.1 User

Costituisce l'entità anagrafica principale del sistema e gestisce l'accesso all'applicativo. Memorizza i dati personali dell'utente (codice fiscale, nome, cognome) e le credenziali di autenticazione (username, email, password). In un'architettura API-only, questa tabella è il punto di riferimento per l'emissione e la convalida dei token di sicurezza$_G$.

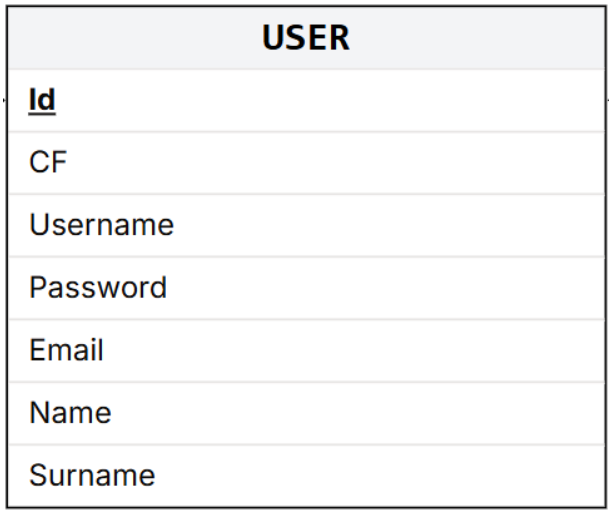


Figura 23: Tabella User

8.4.2 Company

Rappresenta l'organizzazione, l'azienda cliente o la divisione aziendale all'interno del sistema. Funge da aggregatore principale a cui vengono associati sia i dipendenti sia i parametri di personalizzazione per la generazione dei contenuti (toni e stili).

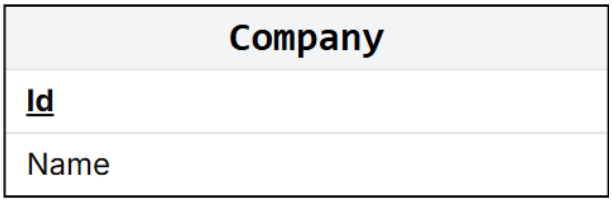


Figura 24: Tabella Company

8.4.3 Employee

Funge da tabella di raccordo (join table con attributi) per mappare la relazione molti-a-molti tra *User* e *Company*. Permette di contestualizzare un utente all'interno di una specifica azienda, assegnandogli un dipartimento di competenza (*Department*).

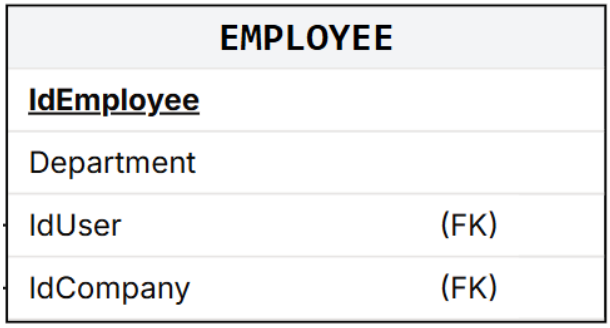


Figura 25: Tabella Employee

8.4.4 Tones

Tabella di dizionario dedicata al modulo$_G$ AI$_G$ Generator. Definisce le direttive sul "tono di voce" (es. Professionale, Amichevole) da fornire in input ai modelli generativi. Ogni tono è vincolato a una specifica azienda (*CompanyId*), garantendo una personalizzazione della generazione isolata per ogni tenant.

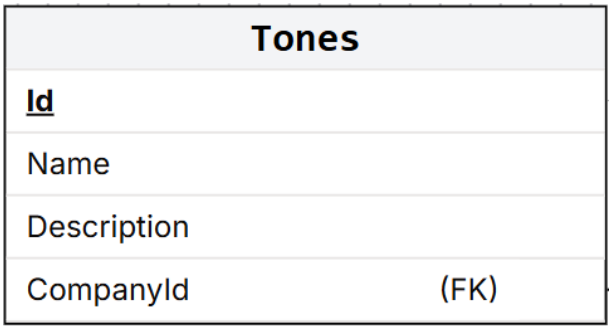


Figura 26: Tabella Tones

8.4.5 Style

Simile all'entità *Tones*, questa tabella definisce gli stili di scrittura o formattazione (es. Conciso, Dettagliato). Anche in questo caso, la relazione con *Company* permette di compartimentare i preset stilistici a livello aziendale.

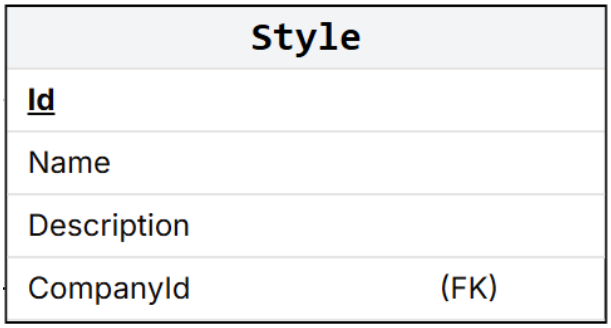


Figura 27: Tabella Style

8.4.6 GeneratedDatum

È l'entità centrale del modulo$_G$ AI$_G$ Generator. Traccia in modo esaustivo ogni interazione con il modello di intelligenza artificiale, persistendo l'input (*Prompt*), l'output testuale (*TextResult*) o visivo (*ImgPath*, *Seed*, *Height*, *Width*), le performance (*GenerationTime*) e il feedback utente (*Rating*). Presenta una chiave esterna autoreferenziale (*Versione*) cruciale per il tracciamento delle rigenerazioni successive di uno stesso contenuto base.

GeneratedDatum	
Id	
Title	
TextResult	
GenerationTime	
DataTime	
Rating	
Prompt	
Height	
Width	
Seed	
Status	
GeneratedImage	
IdTono	(FK)
IdStile	(FK)
IdAzienda	(FK)
Versione	(FK)

Figura 28: Tabella GeneratedDatum

8.4.7 Post

Entità destinata a rappresentare l'utilizzo finale di un contenuto generato dal sistema. Collega direttamente un output testuale e/o grafico approvato proveniente da *GeneratedDatum* alle logiche di pubblicazione o schedulazione esterne, memorizzandone titolo, corpo del testo e data di effettivo utilizzo.

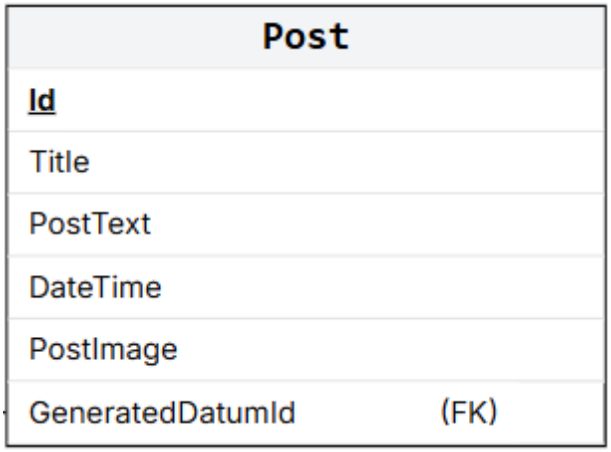


Figura 29: Tabella Post

8.4.8 UploadedDocument

Rappresenta il punto di ingresso dei file fisici nel modulo$_G$ AI Copilot$_G$. Oltre a mantenere i riferimenti di archiviazione (*StoragePath*, *OriginalFilename*, *Checksum*), traccia gli eventuali interventi umani di forzatura dei dati (*OverrideCompany*, *OverrideDepartment*), permettendo di misurare l'affidabilità$_G$ del sistema automatico.

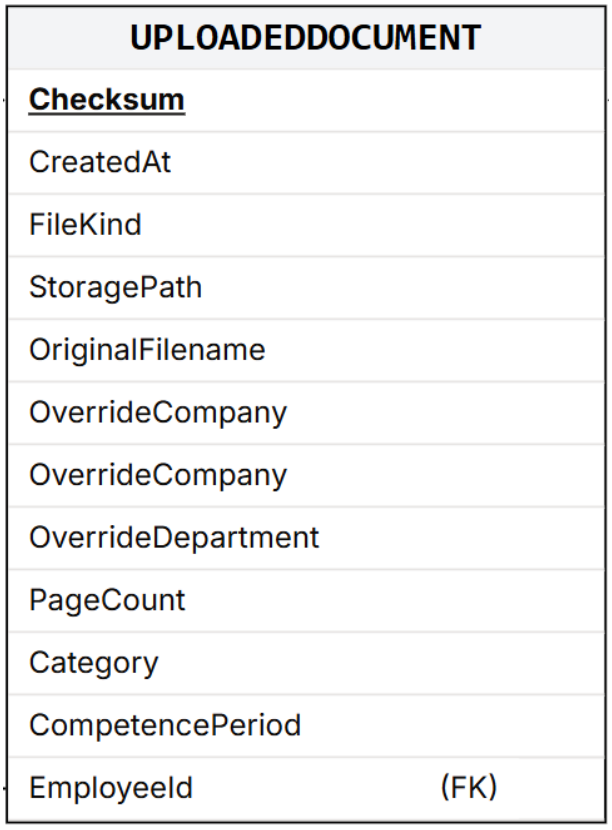


Figura 30: Tabella UploadedDocument

8.4.9 ExtractedDocument

Consolida i dati di business estrapolati dall'intelligenza artificiale a partire dal file originario (*UploadedDocument*). Memorizza le metriche di affidabilità$_G$ dell'estrazione (*Confidence*), i tempi di elaborazione e mappa il risultato all'entità di business corretta (*MatchedEmployeeId*).

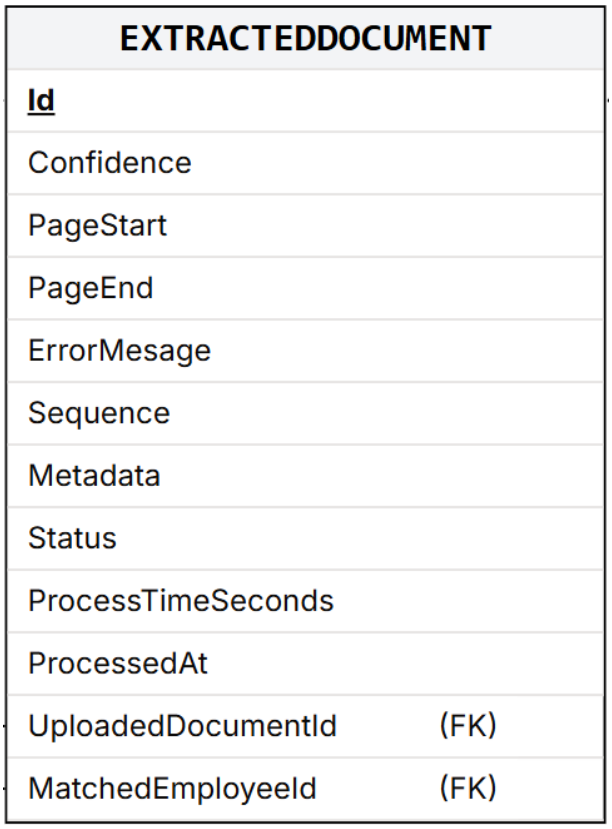


Figura 31: Tabella ExtractedDocument

8.4.10 ProcessingRun

Entità deputata alla gestione dell'elaborazione asincrona (Background Jobs). Mappa un'esecuzione di sistema (batch o singola) per il processamento dei documenti caricati, esponendo lo stato globale dell'operazione (*Status*, *CompletedAt*, *JobId*) e conteggiando i volumi elaborati (*TotalDocuments*, *ProcessedDocuments*).

PROCESSINGRUN	
Id	
Status	
CompletedAt	
ErrorMessage	
JobId	
OriginalFileName	
ProcessedDocuments	
TotalDocuments	
UploadedDocumentId	(FK)

Figura 32: Tabella ProcessingRun

8.4.11 ProcessingItem

Gestisce la granularità delle operazioni asincrone. Ogni *ProcessingRun* è suddiviso in molteplici *ProcessingItem*, permettendo al sistema di tracciare in dettaglio lo stato di avanzamento (*Status*, *Sequence*) e i potenziali errori (*ErrorMessage*) a livello di singolo documento o operazione isolata.

PROCESSINGITEM	
Id	
ErrorMessage	
Status	
Sequence	
Filename	
ExtractedDocumentsId	(FK)
ProcessingRunId	(FK)

Figura 33: Tabella ProcessingItem

8.4.12 Template

Archivio dei modelli standard precompilati. Contiene strutture di testo riutilizzabili per formattare gli output di sistema o le comunicazioni in uscita in modo omogeneo e standardizzato.

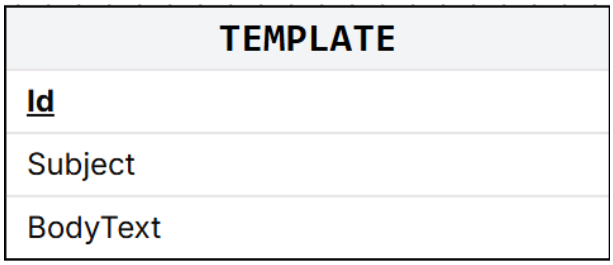


Figura 34: Tabella Template

8.4.13 Sending

Log di tracciamento per le comunicazioni in uscita. Questa entità collega in modo relazionale un destinatario (*RecipientId*, mappato verso *User*), il contenuto estrapolato dal sistema (*ExtractedDocument*) e l'eventuale formato utilizzato (*Template*), registrando l'effettivo payload della comunicazione (*Subject*, *Body*) e il timestamp di invio (*SentAt*).

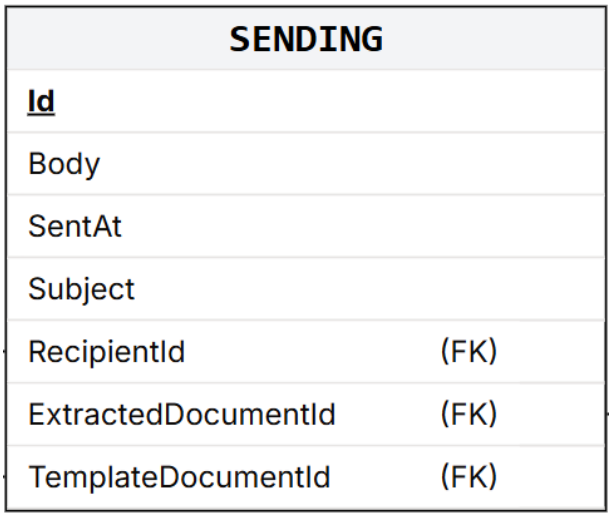


Figura 35: Tabella Sending

9 Requisiti Funzionali Soddisfatti

Il periodo di analisi dei requisiti$_G$ ha portato all'identificazione di un set di requisiti funzionali$_G$ che il prodotto$_G$ software deve soddisfare. Di seguito viene fornita una panoramica dei principali requisiti soddisfatti nel sistema, con riferimento alle sezioni del documento dove sono descritti in dettaglio. I requisiti funzionali$_G$ si dividono in obbligatori, desiderabili e opzionali, a seconda della loro importanza e priorità$_G$ per il progetto$_G$. In particolare sono stati identificati 131 requisiti funzionali$_G$, di cui 97 obbligatori, 0 desiderabili e 34 opzionali. I requisiti obbligatori hanno come focus l'implementazione$_G$ dei tre moduli principali del sistema, ovvero il modulo$_G$ di analisi documenti, il modulo$_G$ di generazione testo e il modulo$_G$ di raccolta dati, che rappresentano le funzionalità$_G$ core del prodotto$_G$ software. Ciascuna di queste aree non era presente all'interno dell'applicativo Nexum quindi i rappresentanti dell'azienda (Eggon) hanno ritenuto fondamentale implementare queste funzionalità$_G$ per soddisfare le esigenze degli utenti finali e garantire il successo del progetto$_G$. I requisiti opzionali, invece, si concentrano su funzionalità$_G$ come la gestione delle utenze e la distribuzione dei documenti, che invece sono già presenti all'interno

dell'applicativo Nexum e quindi non rappresentano un elemento di differenziazione significativo per il prodotto$_G$ software che stiamo sviluppando.

9.1 Completamento requisiti

- **Requisiti Obbligatori**: 97/97 requisiti funzionali$_G$ soddisfatti, con focus sui moduli di analisi documenti, generazione testo e raccolta dati.
- **Requisiti Desiderabili**: 0 requisiti desiderabili identificati.
- **Requisiti Opzionali**: 0/34 requisiti opzionali soddisfatti, principalmente relativi alla gestione utenze e distribuzione documenti.

9.2 Tabella di tracciamento dei requisiti

Tabella 19: Tabella dei Requisiti funzionali$_G$

Codice	Descrizione	Stato	Priorità$_G$
RF-001	Il sistema deve permettere all'utente non autenticato di effettuare la registrazione di un nuovo account.	Non implementato	Opzionale$_G$
RF-002	Il sistema deve permettere all'utente di inserire il proprio indirizzo email in fase di registrazione.	Non implementato	Opzionale$_G$
RF-003	Il sistema deve permettere all'utente di inserire una password in fase di registrazione.	Non implementato	Opzionale$_G$
RF-004	Il sistema deve permettere all'utente di inserire un username in fase di registrazione.	Non implementato	Opzionale$_G$
RF-005	Il sistema deve permettere all'utente di inserire il proprio nome in fase di registrazione.	Non implementato	Opzionale$_G$
RF-006	Il sistema deve permettere all'utente di inserire il proprio cognome in fase di registrazione.	Non implementato	Opzionale$_G$
RF-007	Il sistema deve permettere all'utente di inserire la propria matricola in fase di registrazione.	Non implementato	Opzionale$_G$
RF-008	Il sistema deve permettere di visualizzare un messaggio di errore se il formato dell'email inserita non è valido.	Non implementato	Opzionale$_G$
RF-009	Il sistema deve permettere di visualizzare un messaggio di errore se la password non rispetta i criteri di sicurezza$_G$.	Non implementato	Opzionale$_G$
RF-010	Il sistema deve impedire la registrazione se l'email inserita è già associata a un account esistente.	Non implementato	Opzionale$_G$

Continua nella prossima pagina...

Tabella 19 – continua dalla pagina precedente

Codice	Descrizione	Stato	Priorità$_G$
RF-011	Il sistema deve impedire la registrazione se lo username inserito è già utilizzato.	Non implementato	Opzionale$_G$
RF-012	Il sistema deve impedire la registrazione se la matricola inserita è già presente nel sistema.	Non implementato	Opzionale$_G$
RF-013	Il sistema deve permettere di visualizzare un messaggio di errore se il formato della matricola non è valido.	Non implementato	Opzionale$_G$
RF-014	Il sistema deve permettere all'utente registrato di effettuare il login (autenticazione).	Non implementato	Opzionale$_G$
RF-015	Il sistema deve notificare l'errore in caso di tentativo di login con email non registrata o password associata all'email errata.	Non implementato	Opzionale$_G$
RF-016	Il sistema deve consentire all'utente di poter inserire i dati necessari al login.	Non implementato	Opzionale$_G$
RF-017	Il sistema deve permettere all'utente di visualizzare i dati del proprio profilo.	Non implementato	Opzionale$_G$
RF-018	Il sistema deve mostrare l'email associata al profilo utente.	Non implementato	Opzionale$_G$
RF-019	Il sistema deve mostrare (o permettere la gestione del) la password del profilo utente.	Non implementato	Opzionale$_G$
RF-020	Il sistema deve mostrare lo username associato al profilo utente.	Non implementato	Opzionale$_G$
RF-021	Il sistema deve mostrare il nome associato al profilo utente.	Non implementato	Opzionale$_G$
RF-022	Il sistema deve mostrare il cognome associato al profilo utente.	Non implementato	Opzionale$_G$
RF-023	Il sistema deve mostrare la matricola associata al profilo utente.	Non implementato	Opzionale$_G$
RF-024	Il sistema deve permettere all'utente di effettuare il logout terminando la sessione.	Non implementato	Opzionale$_G$
RF-025	Il sistema deve gestire l'uscita dalla modifica profilo senza salvare i cambiamenti.	Non implementato	Opzionale$_G$
RF-026	Il sistema deve permettere all'Amministratore$_G$ di visualizzare la lista degli utenti registrati.	Non implementato	Opzionale$_G$
RF-027	Il sistema deve permettere la visualizzazione del dettaglio di un singolo utente dalla lista.	Non implementato	Opzionale$_G$
RF-028	Il sistema deve mostrare il ruolo associato a un utente registrato.	Non implementato	Opzionale$_G$
RF-029	Il sistema deve mostrare il nome di un utente registrato.	Non implementato	Opzionale$_G$

Continua nella prossima pagina...

Tabella 19 – continua dalla pagina precedente

Codice	Descrizione	Stato	Priorità$_G$
RF-030	Il sistema deve mostrare il cognome di un utente registrato.	Non implementato	Opzionale$_G$
RF-031	Il sistema deve permettere all'Amministratore$_G$ di modificare il ruolo di un utente registrato.	Non implementato	Opzionale$_G$
RF-032	Il sistema deve generare contenuti testuali tramite AI Assistant$_G$ in base a prompt$_G$ e parametri.	Implementato	Obbligatorio$_G$
RF-033	Il sistema deve permettere l'inserimento di un prompt$_G$ testuale per la generazione.	Implementato	Obbligatorio$_G$
RF-034	Il sistema deve permettere la selezione del tono per la generazione del contenuto.	Implementato	Obbligatorio$_G$
RF-035	Il sistema deve permettere la selezione dello stile per la generazione del contenuto.	Implementato	Obbligatorio$_G$
RF-036	Il sistema deve permettere la visualizzazione dello storico delle generazioni AI$_G$.	Implementato	Obbligatorio$_G$
RF-037	Il sistema deve notificare l'assenza di elementi se lo storico delle generazioni è vuoto.	Implementato	Obbligatorio$_G$
RF-038	Il sistema deve mostrare i dettagli completi di un elemento selezionato dallo storico.	Implementato	Obbligatorio$_G$
RF-039	Il sistema deve permettere di visualizzare lo stile utilizzato per un contenuto nello storico.	Implementato	Obbligatorio$_G$
RF-040	Il sistema deve permettere di visualizzare il testo del risultato generato nello storico.	Implementato	Obbligatorio$_G$
RF-041	Il sistema deve permettere di visualizzare il timestamp (data/ora) della generazione nello storico.	Implementato	Obbligatorio$_G$
RF-042	Il sistema deve permettere di visualizzare la valutazione assegnata dall'utente al contenuto nello storico.	Implementato	Obbligatorio$_G$
RF-043	Il sistema deve permettere di visualizzare il prompt$_G$ originale utilizzato per un contenuto nello storico.	Implementato	Obbligatorio$_G$
RF-044	Il sistema deve permettere di visualizzare il tono utilizzato per un contenuto nello storico.	Implementato	Obbligatorio$_G$
RF-045	Il sistema deve mostrare un'anteprima del contenuto generato dall'AI$_G$.	Implementato	Obbligatorio$_G$
RF-046	Il sistema deve permettere di modificare l'immagine associata al contenuto generato.	Implementato	Obbligatorio$_G$

Continua nella prossima pagina...

Tabella 19 – continua dalla pagina precedente

Codice	Descrizione	Stato	Priorità$_G$
RF-047	Il sistema deve notificare l'utente quando tenta di caricare un file immagine non valido.	Implementato	Obbligatorio$_G$
RF-048	Il sistema deve permettere di modificare il titolo del contenuto generato.	Implementato	Obbligatorio$_G$
RF-049	Il sistema deve permettere di modificare il testo del corpo del contenuto generato.	Implementato	Obbligatorio$_G$
RF-050	Il sistema deve permettere di annullare le modifiche apportate al contenuto generato.	Implementato	Obbligatorio$_G$
RF-051	Il sistema deve permettere di riutilizzare i parametri di un contenuto dello storico per una nuova generazione.	Implementato	Obbligatorio$_G$
RF-052	Il sistema deve permettere di duplicare un contenuto dallo storico per modificarne i parametri.	Implementato	Obbligatorio$_G$
RF-053	Il sistema deve permettere di filtrare la lista delle generazioni nello storico.	Implementato	Obbligatorio$_G$
RF-054	Il sistema deve permettere di visualizzare la lista dello storico aggiornata in base ai filtri applicati.	Implementato	Obbligatorio$_G$
RF-055	Il sistema deve permettere di rigenerare un contenuto tramite AI$_G$ mantenendo i parametri.	Implementato	Obbligatorio$_G$
RF-056	Il sistema deve permettere all'utente di valutare (rating) il contenuto generato.	Implementato	Obbligatorio$_G$
RF-057	Il sistema deve permettere di scartare il contenuto generato e pulire l'interfaccia$_G$.	Implementato	Obbligatorio$_G$
RF-058	Il sistema deve permettere di salvare il contenuto generato nel database.	Implementato	Obbligatorio$_G$
RF-059	Il sistema deve permettere all'utente l'inserimento di un nuovo tono per la generazione di contenuti.	Implementato	Obbligatorio$_G$
RF-060	Il sistema deve permettere all'utente l'eliminazione di un tono per la generazione di contenuti.	Implementato	Obbligatorio$_G$
RF-061	Il sistema deve permettere all'utente l'inserimento di un nuovo stile per la generazione di contenuti.	Implementato	Obbligatorio$_G$
RF-062	Il sistema deve permettere all'utente l'eliminazione di uno stile per la generazione di contenuti.	Implementato	Obbligatorio$_G$
RF-063	Il sistema deve permettere l'analisi di documenti tramite il modulo$_G$ AI Co-Pilot$_G$.	Implementato	Obbligatorio$_G$

Continua nella prossima pagina...

Tabella 19 – continua dalla pagina precedente

Codice	Descrizione	Stato	Priorità$_G$
RF-064	Il sistema deve permettere l'inserimento/selezione della categoria del documento.	Implementato	Obbligatorio$_G$
RF-065	Il sistema deve permettere l'inserimento del mese/anno di competenza del documento.	Implementato	Obbligatorio$_G$
RF-066	Il sistema deve permettere l'inserimento dell'azienda associata al documento.	Implementato	Obbligatorio$_G$
RF-067	Il sistema deve permettere l'inserimento del reparto associato al documento.	Implementato	Obbligatorio$_G$
RF-068	Il sistema deve permettere di controllare la correttezza del formato del file inserito.	Implementato	Obbligatorio$_G$
RF-069	Il sistema deve permettere di controllare che lo stesso file non sia già stato analizzato.	Implementato	Obbligatorio$_G$
RF-070	Il sistema deve permettere lo split di documenti diversi all'interno dello stesso file.	Implementato	Obbligatorio$_G$
RF-071	Il sistema deve permettere di visualizzare la lista dei documenti analizzati.	Implementato	Obbligatorio$_G$
RF-072	Il sistema deve notificare l'utente se nessun documento è stato riconosciuto dall'analisi.	Implementato	Obbligatorio$_G$
RF-073	Il sistema deve permettere di visualizzare i dettagli di un singolo documento dalla lista.	Implementato	Obbligatorio$_G$
RF-074	Il sistema deve permettere di visualizzare la competenza (periodo) del documento analizzato.	Implementato	Obbligatorio$_G$
RF-075	Il sistema deve permettere di visualizzare l'azienda associata al documento analizzato.	Implementato	Obbligatorio$_G$
RF-076	Il sistema deve permettere di visualizzare la causale del documento analizzato.	Implementato	Obbligatorio$_G$
RF-077	Il sistema deve permettere di visualizzare la lingua rilevata nel documento.	Implementato	Obbligatorio$_G$
RF-078	Il sistema deve permettere di visualizzare il numero di pagine del documento.	Implementato	Obbligatorio$_G$
RF-079	Il sistema deve permettere di visualizzare il nome originale del file del documento.	Implementato	Obbligatorio$_G$
RF-080	Il sistema deve permettere di visualizzare la data di redazione/caricamento del documento.	Implementato	Obbligatorio$_G$
RF-081	Il sistema deve permettere di visualizzare il codice identificativo del documento.	Implementato	Obbligatorio$_G$
RF-082	Il sistema deve permettere di visualizzare la tipologia del documento.	Implementato	Obbligatorio$_G$
RF-083	Il sistema deve mostrare l'anteprima visiva del documento analizzato.	Implementato	Obbligatorio$_G$

Continua nella prossima pagina...

Tabella 19 – continua dalla pagina precedente

Codice	Descrizione	Stato	Priorità$_G$
RF-084	Il sistema deve permettere di modificare il destinatario associato al documento.	Implementato	Obbligatorio$_G$
RF-085	Il sistema deve permettere di modificare la tipologia del documento.	Implementato	Obbligatorio$_G$
RF-086	Il sistema deve ricalcolare la percentuale di confidenza$_G$ dopo modifiche manuali.	Implementato	Obbligatorio$_G$
RF-087	Il sistema deve permettere di visualizzare la lista delle informazioni sui destinatari estratti.	Implementato	Obbligatorio$_G$
RF-088	Il sistema deve permettere di visualizzare i dettagli di un singolo destinatario in lista.	Implementato	Obbligatorio$_G$
RF-089	Il sistema deve permettere di visualizzare il codice fiscale del destinatario.	Implementato	Obbligatorio$_G$
RF-090	Il sistema deve permettere di visualizzare la matricola del destinatario.	Implementato	Obbligatorio$_G$
RF-091	Il sistema deve permettere di visualizzare il reparto del destinatario.	Implementato	Obbligatorio$_G$
RF-092	Il sistema deve permettere di visualizzare il nome/cognome del destinatario.	Implementato	Obbligatorio$_G$
RF-093	Il sistema deve notificare se nessun destinatario è stato riconosciuto.	Implementato	Obbligatorio$_G$
RF-094	Il sistema deve permettere di visualizzare lo storico completo dei documenti processati.	Implementato	Obbligatorio$_G$
RF-095	Il sistema deve notificare l'assenza di documenti nello storico.	Implementato	Obbligatorio$_G$
RF-096	Il sistema deve permettere di visualizzare i dettagli di un elemento nello storico documenti.	Implementato	Obbligatorio$_G$
RF-097	Il sistema deve permettere di visualizzare la percentuale di confidenza$_G$ dell'analisi nello storico.	Implementato	Obbligatorio$_G$
RF-098	Il sistema deve permettere di visualizzare l'appartenenza alle liste di distribuzione.	Implementato	Obbligatorio$_G$
RF-099	Il sistema deve permettere di visualizzare lo stato di elaborazione del documento.	Implementato	Obbligatorio$_G$
RF-100	Il sistema deve permettere di caricare un template$_G$di messaggio esistente.	Implementato	Obbligatorio$_G$
RF-101	Il sistema deve permettere di modificare l'oggetto del messaggio.	Implementato	Obbligatorio$_G$
RF-102	Il sistema deve permettere di modificare il testo del corpo del messaggio.	Implementato	Obbligatorio$_G$
RF-103	Il sistema deve permettere di salvare il messaggio corrente come nuovo template$_G$.	Implementato	Obbligatorio$_G$

Continua nella prossima pagina...

Tabella 19 – continua dalla pagina precedente

Codice	Descrizione	Stato	Priorità$_G$
RF-104	Il sistema deve permettere di eliminare un template$_G$di messaggio.	Implementato	Obbligatorio$_G$
RF-105	Il sistema deve permettere di visualizzare la lista dei template$_G$di messaggio disponibili.	Implementato	Obbligatorio$_G$
RF-106	Il sistema deve permettere di visualizzare un elemento della lista dei template$_G$di messaggio disponibili.	Implementato	Obbligatorio$_G$
RF-107	Il sistema deve permettere di visualizzare l'oggetto del template$_G$.	Implementato	Obbligatorio$_G$
RF-108	Il sistema deve permettere di visualizzare il testo del template$_G$.	Implementato	Obbligatorio$_G$
RF-109	Il sistema deve permettere di visualizzare il codice del template$_G$.	Implementato	Obbligatorio$_G$
RF-110	Il sistema deve permettere l'invio del documento e del messaggio associato.	Non implementato	Opzionale$_G$
RF-111	Il sistema deve permettere di allegare ulteriore contenuto al messaggio.	Non implementato	Opzionale$_G$
RF-112	Il sistema deve permettere di pianificare l'invio del documento e del messaggio associato.	Non implementato	Opzionale$_G$
RF-113	Il sistema deve permettere il filtraggio della lista dei documenti analizzati.	Implementato	Obbligatorio$_G$
RF-114	Il sistema deve mostrare la lista dei documenti aggiornata in base ai filtri.	Implementato	Obbligatorio$_G$
RF-115	Il sistema deve permettere il filtraggio della lista dei destinatari.	Implementato	Obbligatorio$_G$
RF-116	Il sistema deve mostrare la lista dei destinatari aggiornata in base ai filtri.	Implementato	Obbligatorio$_G$
RF-117	Il sistema deve permettere il filtraggio della lista dello storico documenti.	Implementato	Obbligatorio$_G$
RF-118	Il sistema deve mostrare la lista dello storico documenti aggiornata in base ai filtri.	Implementato	Obbligatorio$_G$
RF-119	Il sistema deve permettere di mostrare l'audit di un documento nello storico.	Implementato	Obbligatorio$_G$
RF-120	Il sistema deve permettere di visualizzare la dashboard$_G$ con i dati di analytics per l'AI Assistant$_G$.	Implementato	Obbligatorio$_G$
RF-121	Il sistema deve permettere di visualizzare il numero totale di prompt$_G$ generati.	Implementato	Obbligatorio$_G$
RF-122	Il sistema deve permettere di visualizzare il rating medio dei prompt$_G$ generati.	Implementato	Obbligatorio$_G$
RF-123	Il sistema deve permettere di visualizzare il numero di rigenerazioni effettuate.	Implementato	Obbligatorio$_G$

Continua nella prossima pagina...

Tabella 19 – continua dalla pagina precedente

Codice	Descrizione	Stato	Priorità$_G$
RF-124	Il sistema deve permettere di visualizzare le statistiche sui toni più utilizzati.	Implementato	Obbligatorio$_G$
RF-125	Il sistema deve permettere di visualizzare le statistiche sugli stili più utilizzati.	Implementato	Obbligatorio$_G$
RF-126	Il sistema deve permettere di visualizzare la dashboard$_G$ con i dati di analytics per l'AI Co-Pilot$_G$.	Implementato	Obbligatorio$_G$
RF-127	Il sistema deve permettere di visualizzare la confidenza$_G$ media delle analisi documenti.	Implementato	Obbligatorio$_G$
RF-128	Il sistema deve permettere di visualizzare la percentuale di interventi manuali necessari.	Implementato	Obbligatorio$_G$
RF-129	Il sistema deve permettere di visualizzare l'accuratezza del mapping dei dati.	Implementato	Obbligatorio$_G$
RF-130	Il sistema deve permettere di visualizzare i tempi medi di analisi dei documenti.	Implementato	Obbligatorio$_G$
RF-131	Il sistema deve permettere di filtrare i dati di analytics per periodo temporale.	Implementato	Obbligatorio$_G$