Object Design Document

C04 – Esistere

Riferimento	C04_Esistere_ODD_V1.0
Versione	1.0
Data	
Destinatario	Prof.ssa Filomena Ferrucci, Prof.re Fabio Palomba
Presentato da	C04 Team Esistere: • Antonio D'Auria (AA) • Luca Casillo (LC) • Maria Giovanna Della Pietra (MGP) • Ogham If Dell'Erba (OE) • Raffaele Forte (RF) • Rosa Carotenuto (RC) • Valentino Dragone (VD)
Approvato da	Alessandra Parziale, Saverio Napolitano

Revision History

Data	Versione	Descrizione	Autori
17/12/2023	0.1.0	Aggiunta Object Design Goal	MGP, OE, RC, RF, VD
17/12/2023	0.2.0	Aggiunta Object Trade Offs	MGP, OE, RC, RF, VD
17/12/2023	0.3.0	Aggiunta Components off-the-shelf	AA, LC
17/12/2023	0.4.0	Aggiunta Design Pattern	AA, LC, MGP, OE, RF, RC, VD
23/12/2023	0.4.1	Aggiunta diagramma del design pattern DAO	RC
25/12/2023	0.4.2	Aggiunta diagramma del design pattern Facade	MGP
27/12/2023	0.5.0	Aggiunta Packages	AA, LC, MGP, OE, RF, RC, VD
30/12/2023	0.6.0	Aggiunta Class Diagram	AA, LC, MGP, OE, RF, RC, VD
31/12/2023	0.6.1	Revisione Architetturale	LC, RC
31/12/2023	0.6.2	Revisione Strutturale	AA, RF
01/12/2023	0.6.3	Revisione Generale	LC
23/01/2024	1.0.0	Revisione Generale	AA, LC, MGP, OE, RF, RC, VD

Team Members

Nome	Ruolo nel progetto	Acronimo	Informazione di controllo
Alessandra Parziale	Project Manager	AP	a.parziale8@studenti.unisa.it
Saverio Napolitano	Project Manager	SN	s.napolitano44@studenti.unisa.it
Antonio D'Auria	Team Member	AA	a.dauria123@studenti.unisa.it
Luca Casillo	Team Member	LC	l.casillo16@studenti.unisa.it
Maria Giovanna Della Pietra	Team Member	MGP	m.dellapietra10@studenti.unisa.it
Rosa Carotenuto	Team Member	RC	r.carotenuto16@studenti.unisa.it
Ogham If Dell'Erba	Team Member	OE	o.dellerba@studenti.unisa.it
Valentino Dragone	Team Member	VD	v.dragone5@studenti.unisa.it
Raffaele Forte	Team Member	RF	r.forte12@studenti.unisa.it

Sommario

1. Introduzione

Esistere si propone di semplificare le interazioni tra medici e caregiver e aiutare i familiari a stare più vicini ai pazienti col fine ultimo di dare supporto a tutte le figure coinvolte nella terapia contro l'Alzheimer.

In questa prima sezione del documento verranno descritti gli object design goal, i trade-offs, e le linee guida per la fase di implementazione, riguardanti la nomenclatura, la documentazione e la convenzione sui formati.

1.1 Object Design Goals

Rank	ID Design Goal	Descrizione	Origine
1	ODG_1 Manutenibilità	Il sistema dovrà garantire la presenza di codice leggibile che rispetta gli standard del linguaggio implementativo tramite l'utilizzo del tool ESLint	DG_9
2	ODG_2 Riusabilità	Il sistema dovrà garantire la presenza di codice riusabile anche grazie l'utilizzo di Design Pattern	DG_9, DG_10
3	ODG_3 Robustezza	Il sistema deve risultare robusto ed in caso di emergenza deve garantire la gestione del 80% degli errori attraverso apposite eccezioni	DG_2, DG_11
4	ODG_5 Tempi di risposta	Il sistema dovrebbe fornire una risposta al frontend, dopo una richiesta http, in meno di 3 secondi nell'80% dei casi	DG_1
5	ODG_6 Gestione degli errori	Il sistema deve rimanere attivo in caso di errore, fornendo schermate di errore personalizzate, senza perdere il controllo del flusso	DG_4, DG_14
6	ODG_7 Costi	Il sistema avrà un costo iniziale stimato di 107.560€, a cui si aggiungerà il costo della manutenzione, che ammonterà a circa 1500€ per pacchetto	DG_6, DG_7

1.2 Object Design Trade-Offs

Trade-off	Descrizione

Tempi di risposta Vs Robustezza	Per garantire un maggior livello di robustezza, il sistema effettuerà più controlli sugli input da parte degli utenti. Questo potrebbe impattare sui tempi di risposta allungandoli.
Robustezza Vs Costi	Il team si impegnerà a rientrare nel budget, anche a costo di ottenere un sistema meno robusto.

1.3 Definizioni, Acronimi e Abbreviazioni

- **ODG:** Object Design Goals
- **RDBMS:** DBMS Relazionale (Relational Database Managment System)
- **AWS:** Amazon Web Services
- **CI/CD:** Continuous Integration & Continuous Deployment
- **COTS:** Components Off-the-Shelf
- **DAO:** Data Access Object
- **CRUD =**
    - Rappresenta:
        - Create,
        - Read,
        - Update,
        - Delete.
- **HTTP:** Hypertext Transfer Protocol

Vengono riportati di seguito alcune definizioni presenti nel documento:

- **Package**: raggruppamento di classi, interface o file correlati;
- **Design Pattern**: template di soluzioni a problemi ricorrenti impiegati per ottenere riuso e flessibilità;
- **Interfaccia**: insieme di signature delle operazioni offerte dalla classe;

1.4 Riferimenti

1 *Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition* - Bernd Bruegge & Allen H. Dutoit

2 *System Design Goals*

3 *Requirement Analysis Document*

4 *Matrice di Tracciabilità*

1.5 Component Off-the-Shelf

Nell'implementazione del nostro sistema saranno utilizzati:

- Per l'implementazione lato Backend verranno utilizzati Node.js ed Express, il secondo è un framework per applicazioni web per Node.js utilizzato per la costruzione del WebServer
- Per l'implementazione lato Frontend verrà utilizzato React, una libreria open-source di JavaScript

Inoltre, verranno utilizzati alcuni COTS (Component Off-the-Shelf) per facilitare lo sviluppo:

- I dati verranno memorizzati tramite PostegreSQL, un DBMS relazionale open-source (RDBMS)
- La garanzia di manutenibilità e leggibilità del codice verrà assicurata dall'utilizzo del tool ESLint e dall'utilizzo di Prettier per la formattazione
- Utilizzo del servizio Amazon Lightsail, offerto da Amazon Web Services (AWS), che permette l'hosting del database e del WebServer
- L'utilizzo delle GithubActions per mantenere l'integrazione continua e il deployment continuo (CI/CD)

1.6 Design Patterns

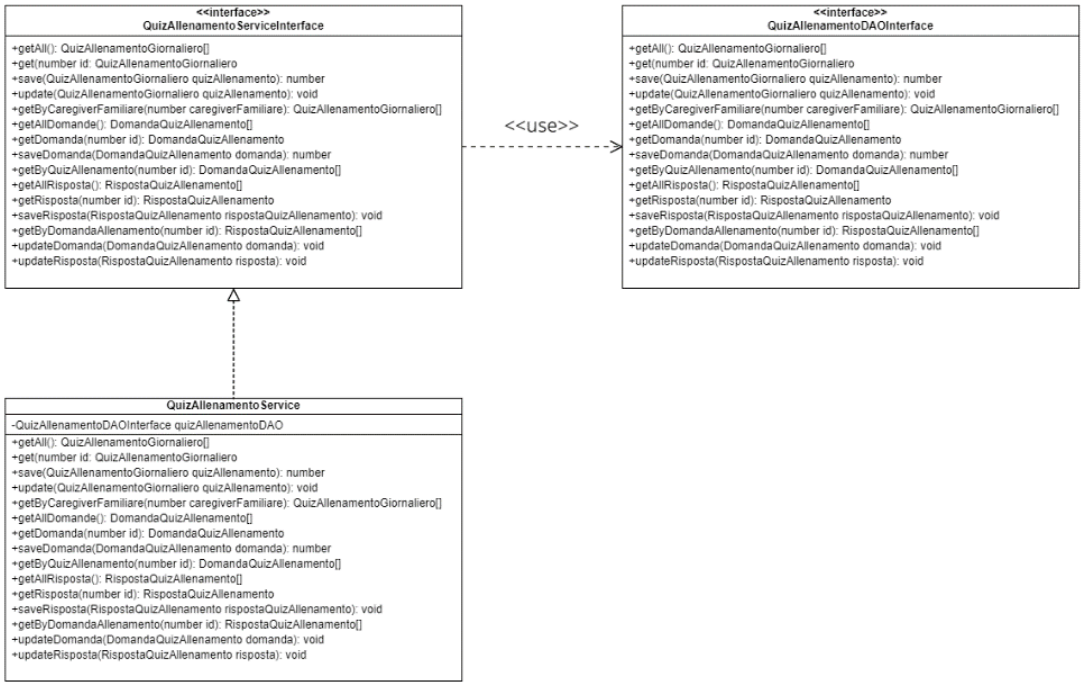
In questa sezione verranno descritti i design pattern utilizzati per lo sviluppo di Esistere.

Facade

Il design pattern Facade è un tipo di design pattern strutturale.

Lo scopo principale di questo design pattern è fornire un'interfaccia unificata e di più alto livello per un insieme di funzioni di un determinato sottosistema: semplificherà l'utilizzo delle interfacce, incapsulando la complessità del sottosistema.

L'utilizzo del design pattern permetterà il disaccoppiamento dei client dai dettagli implementativi del sottosistema, riducendo la dipendenza tra le classi, permettendo flessibilità e manutenibilità. Nel caso di Esistere, verrà utilizzato per interfacciare il Service Layer utilizzato nell'architettura con l'Application Layer e per interfacciare l'Application Layer con il livello superiore.



Adapter

Il design pattern Adapter è un tipo di design pattern strutturale.

Lo scopo principale di questo design pattern è facilitare la progettazione, semplificando le relazioni tra le entità. Infatti, permette ad oggetti con diverse interfacce di collaborare tra loro, attraverso la classe "adapter" che permette di convertire i dati in oggetti comprensibili dal sistema.

All'interno del sistema l'Adapter è stato utilizzato per facilitare la comunicazione tra back-end e front-end che in alcuni casi lavorano in un formato diverso: il back-end potrebbe lavorare con dati in un formato ottimizzato per l'archiviazione e la logica di business, mentre il front-end potrebbe voler richiedere i dati in un formato ottimizzato per la visualizzazione dell'interfaccia utente.

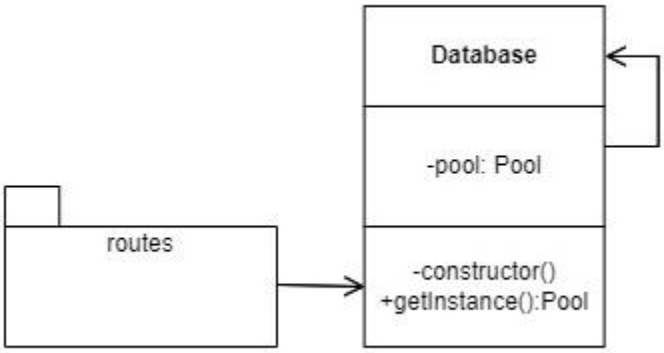
Singleton

Il design pattern Singleton è un tipo di design pattern di creazione.

Lo scopo principale di questo tipo di design pattern è garantire che una classe abbia una singola istanza e fornire un punto di accesso globale a questa istanza da qualsiasi punto dell'applicazione.

Il controllo dell'istanza viene permesso impostando il costruttore di default della classe privato e provvedendo a fornire il riferimento all'unica istanza di quella classe tramite un metodo.

Principalmente, questo viene usato per gestire l'accesso concorrente alle risorse condivise, nel caso specifico di Esistere verrà utilizzato per gestire le connessioni al database.



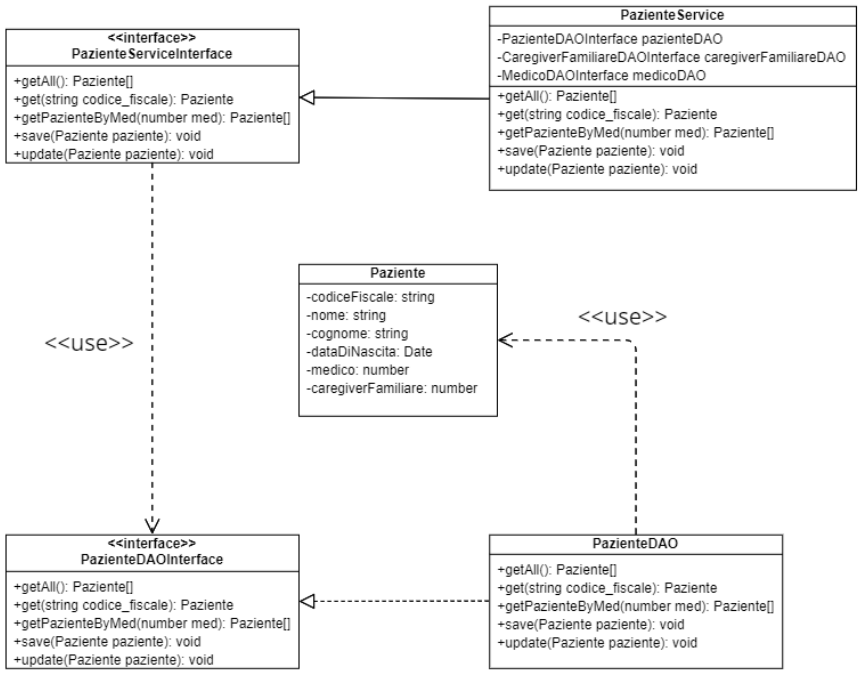
DAO

Il design pattern DAO (Data Access Object) è un tipo di design pattern architetturale utilizzato principalmente per fornire un'interfaccia per l'archiviazione e l'accesso ai dati persistenti, senza la necessità di conoscere i dettagli implementativi per effettuare queste operazioni.

Inoltre, è stato progettato per dividere la logica di business dalla logica di accesso ai dati: esso, infatti, fornisce metodi standardizzati finalizzati a nascondere i dettagli specifici del database consentendo cambiamenti nelle tecnologie, riducendo l'influenza che questi cambiamenti possano avere sul resto dell'applicazione.

Tutto ciò è stato scelto con lo scopo di migliorare la modularità e manutenibilità del sistema, rispettando i design goals che sono stati definiti.

Nell'applicazione in questione, provvederà inoltre a fornire quelli che sono le operazioni CRUD: Create, Read, Update e Delete (quindi Creazione, Lettura, Aggiornamento ed Eliminazione) collegati direttamente ad uno specifico metodo HTTP, rispettivamente POST, GET, PUT E DELETE



1.7 Linee guida per la documentazione delle interfacce

Le linee guida per la documentazione delle interfacce contengono una lista di regole che gli sviluppatori saranno tenuti a rispettare durante la progettazione delle interfacce. In particolare:

- Per le tecnologie selezionate verranno utilizzate le indicazioni riportate nel link sottostante:
  - **JavaScript Standard Style:** https://standardjs.com
- Per la semplicità di utilizzo da parte dei singoli utenti, la chiarezza delle interfacce e la documentazione legata a tali ambiti verranno utilizzate le indicazioni riportate nel link sottostante:
  - **Fundamental Usability Guidelines for User Interface Design:** https://ieeexplore.ieee.org/abstract/document/4561210

2. Packages

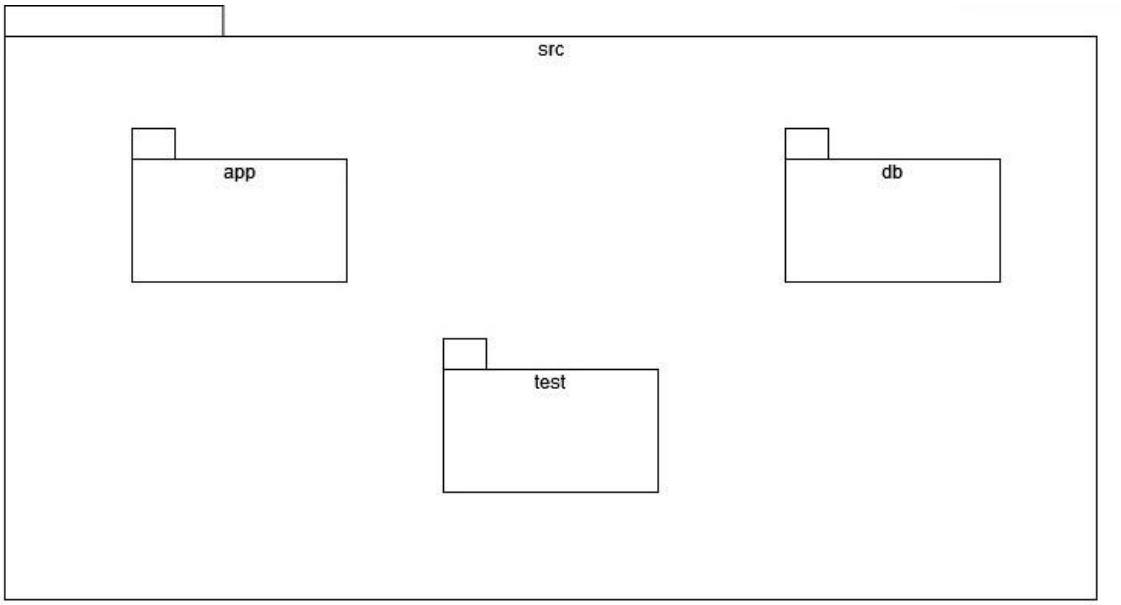
All'interno di questa sezione è mostrata la suddivisione del sistema in package, in base a come è stato definito nel documento di System Design. La suddivisione è motivata dalle scelte architetturali prese e definisce una divisione tra il lato back-end e il lato front-end in un modello Three-Tier:

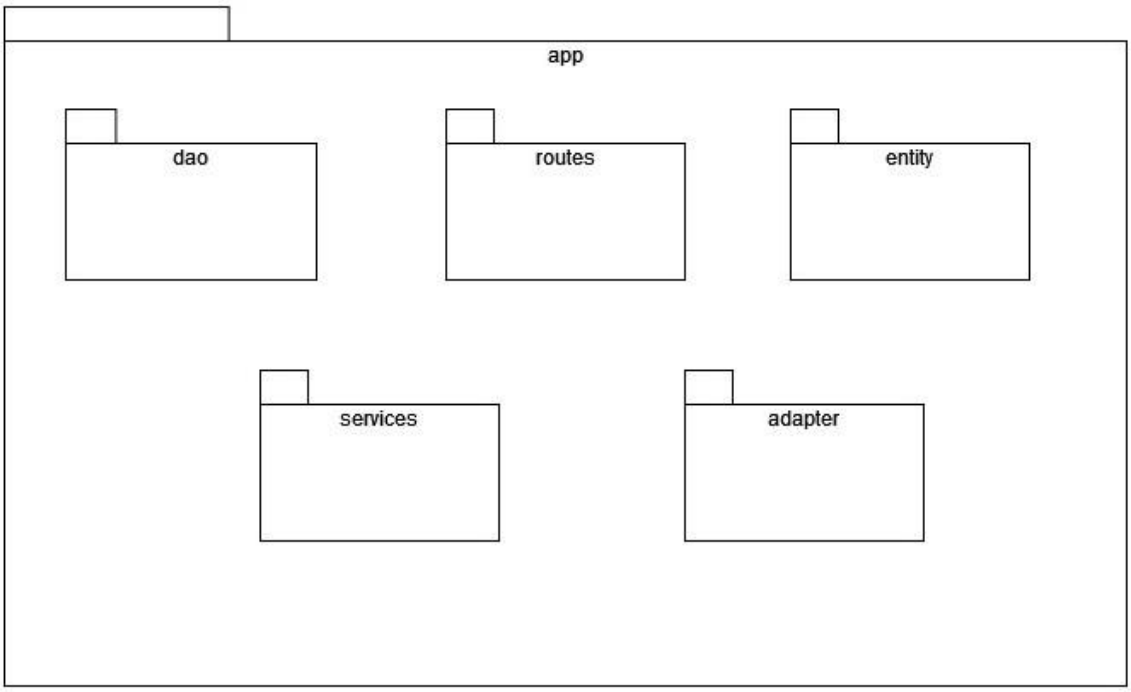
- **src:** contenente i file sorgente

- **client**: contiene tutto ciò che è relativo al lato front-end
  - **assets:** contiene risorse multimediali
  - **components:** contiene i componenti di React
  - **css:** contiene i fogli di stile
  - **interfaces:** contiene le interfacce delle Entity
  - **services:** contiene i services per l'interfacciamento col Web Service
- **server:** contiene tutto ciò che è relativo al lato back-end
  - **dao:** contiene i Data Access Object
  - **entity:** contiene le classi Entity
  - **routes:** contiene i controller dell'Application Layer
  - **services:** rappresenta il Service Layer
- **test:** contiene tutto ciò che è relativo al testing

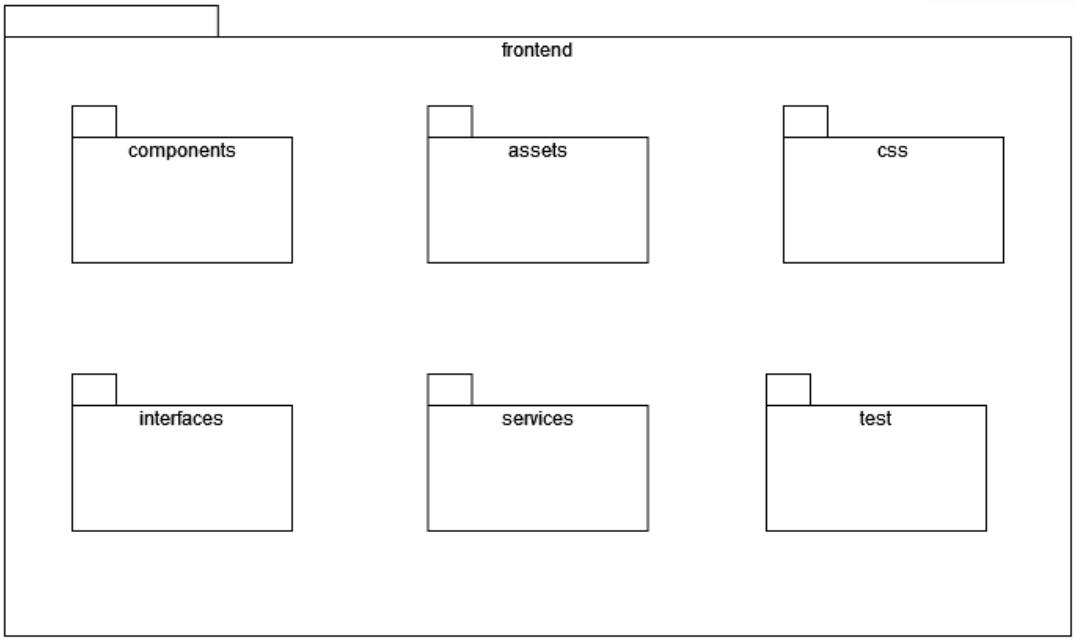
Package Esistere

Package backend

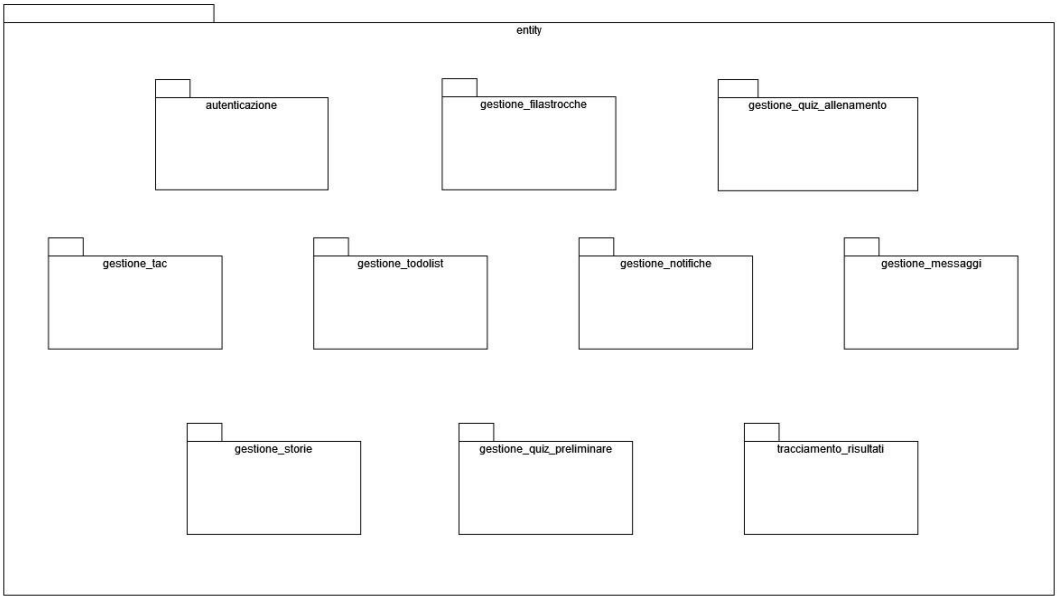


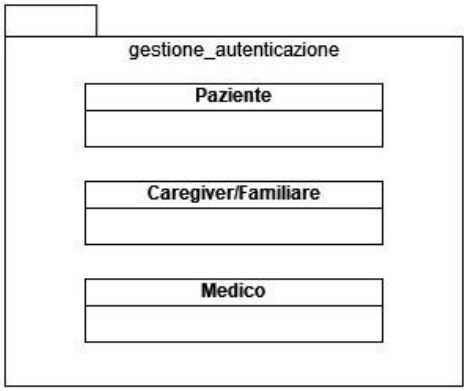

app
dao
routes
entity
services
adapter

Package frontend

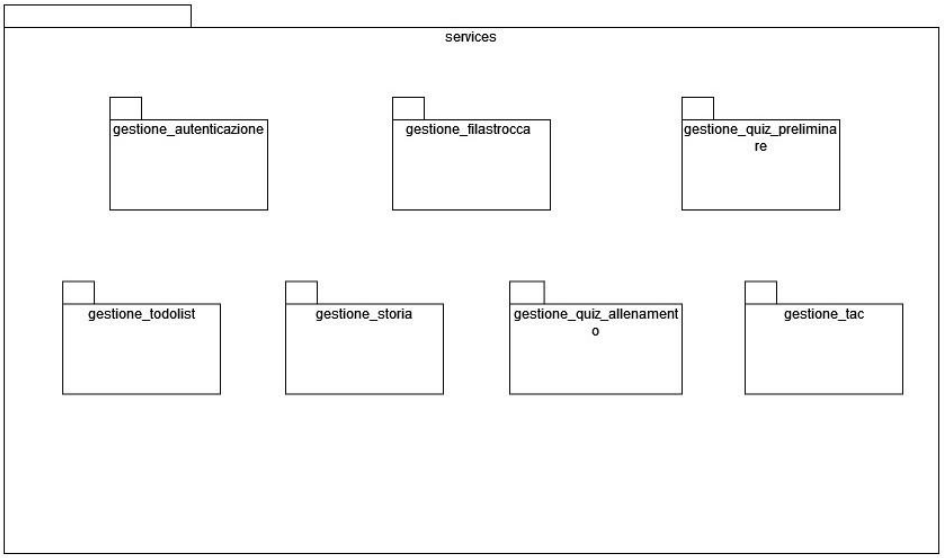


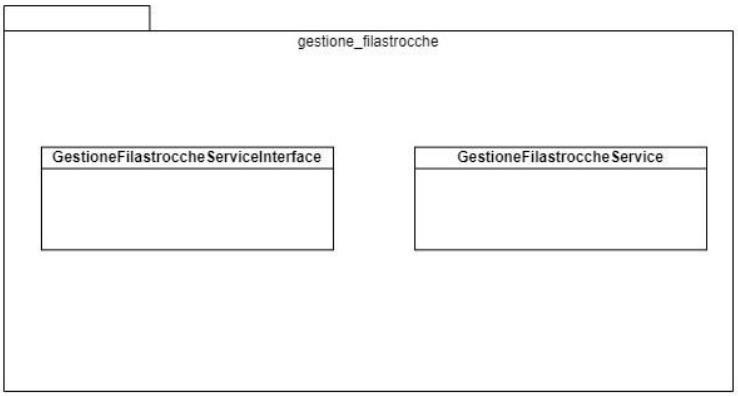
Package Entity



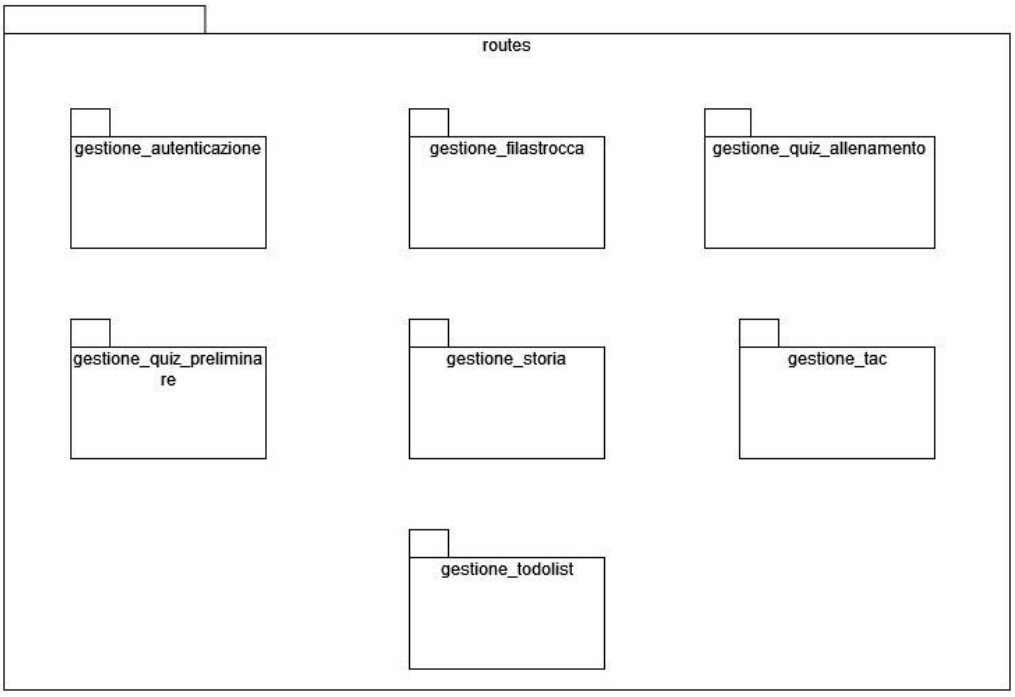


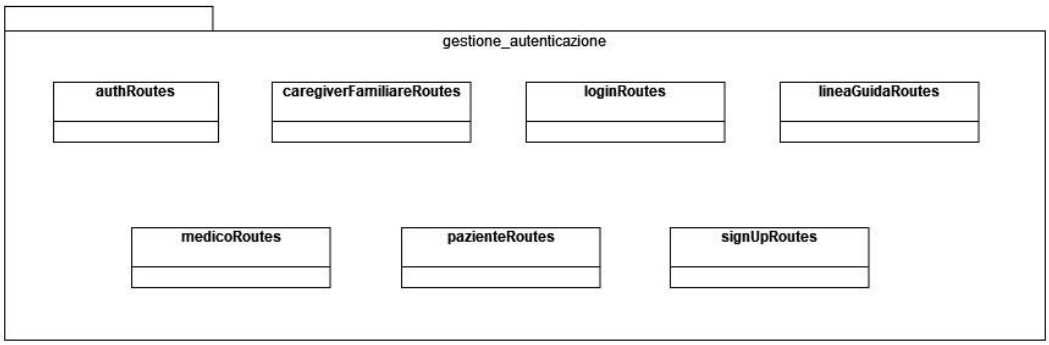
Package Service



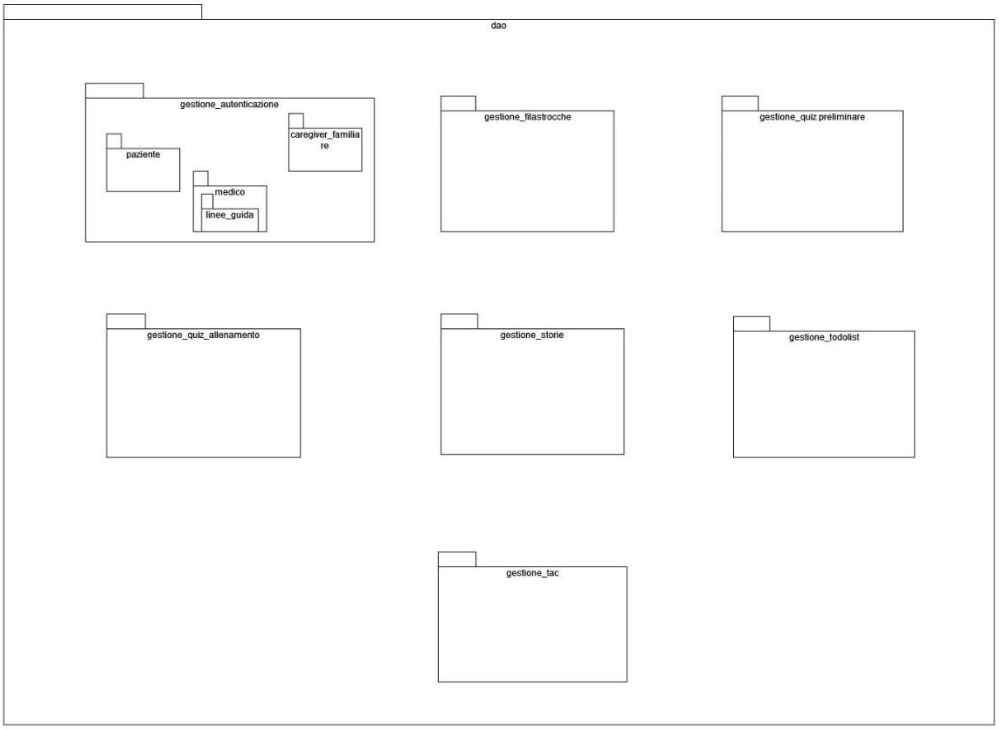


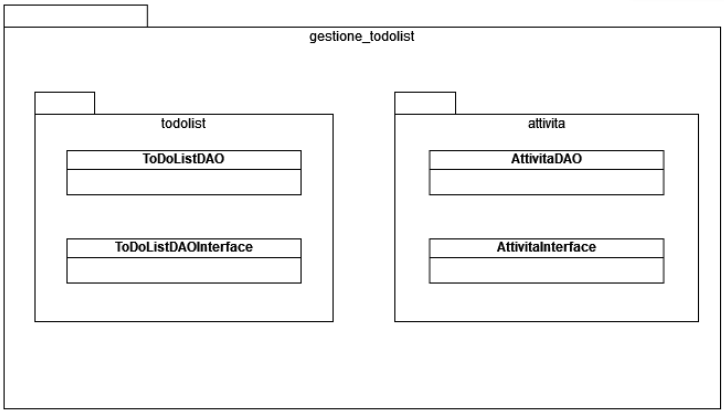
Package routes




gestione_autenticazione
authRoutes
caregiverFamiliareRoutes
loginRoutes
lineaGuidaRoutes
medicoRoutes
pazienteRoutes
signUpRoutes

Package dao


dao
gestione_autenticazione
paziente
caregiver_familiare
medico
linee_guida
gestione_filastrocche
gestione_quiz preliminare
gestione_quiz_allenamento
gestione_storie
gestione_todolist
gestione_tac



3. Class Interfaces

Di seguito sono presentate le interfacce di ciascun package:

1. Package gestione_autenticazione
2. Package gestione_quiz_allenamento
3. Package gestione_todolist
4. Package gestione_quiz_preliminare
5. Package gestione_filastrocche
6. Package gestione_storie
7. Package gestione_tac

3.1 Package gestione_autenticazione

Nome Classe	PazienteService
Descrizione	**Permette di gestire le operazioni relative ad un paziente**
Metodi	+getAll(): Promise<Paziente[]> +get(string codice_fiscale): Promise <Paziente> +save(Paziente paziente): void +update(Paziente paziente): void +getCgFamByPaziente(number id): Promise<CaregiverFamiliare> +getMedByPaziente(number id): Promise<Medico>
Invariante di classe	N/A

Nome Metodo	+getAll(): Promise <Paziente[]>
Descrizione	**Permette di recuperare i dati di tutti i pazienti**
Pre-condizione	N/A
Post-condizione	**context**: PazienteService:: getAll():Promise<Paziente[]> **post:** paziente[] <> null

Nome Metodo	+get(string codice_fiscale): Promise <Paziente>
Descrizione	**Permette di recuperare i dati di un determinato paziente dato il suo codice fiscale**
Pre-condizione	**context**: PazienteService:: get(string codice_fiscale):Promise<Paziente> **pre:** codice_fiscale <> null
Post-condizione	**context**: PazienteService:: get(string codice_fiscale):Promise<Paziente> **post**: paziente <> null

	+save(Paziente paziente): void
Descrizione	**Permette di salvare i dati di un Paziente**
Pre-condizione	**context**: PazienteService:: save(Paziente paziente): void **pre:** paziente <> null and not get(paziente.codice_fiscale)
Post-condizione	N/A

	+update(Paziente paziente):void
Descrizione	**Permette di modificare i dati di un Paziente**
Pre-condizione	**context**: PazienteService:: update(Paziente paziente): void **pre:** paziente <> null
Post-condizione	N/A

Nome Metodo	+getCgFamByPaziente(number id): Promise<CaregiverFamiliare>
Descrizione	**Permette di recuperare i dati di un determinato Caregiver dato il suo paziente**
Pre-condizione	**context:** PazienteService:: getCgFamByPaziente(number id): Promise<CaregiverFamiliare> **pre:** id <> null
Post-condizione	**context:** PazienteService:: getCgFamByPaziente(number id): Promise<CaregiverFamiliare> **post:** caregiverFamiliare <> null

Nome Metodo	+getMedByPaziente(number id): Promise<Medico>
Descrizione	**Permette di recuperare i dati di un determinato Medico dato il suo paziente**
Pre-condizione	**context:** PazienteService:: getMedByPaziente(number id): Promise<Medico> **pre:** id <> null
Post-condizione	**context:** PazienteService:: getMedByPaziente(number id): Promise<Medico> **post:** medico <> null

Nome Classe	CaregiverFamiliareService
Descrizione	**Permette di gestire le operazioni relative ad un Caregiver/Familiare**
Metodi	+getAll(): Promise<CaregiverFamiliare[]> +get(string || number codice): Promise <CaregiverFamiliare> +save(CaregiverFamiliare caregiverFamiliare): Promise <number> +update(CaregiverFamiliare caregiverFamiliare): void
Invariante di classe	N/A

Nome Metodo	+getAll(): Promise <CaregiverFamiliare[]>
Descrizione	**Permette di recuperare i dati di tutti i Caregiver/Familiari**
Pre-condizione	N/A
Post-condizione	**context:** CaregiverFamiliareService:: getAll(): Promise<CaregiverFamiliare[]> **post:** caregiverFamiliare[] <> null

Nome Metodo	+get(string | | number codice): Promise <CaregiverFamiliare>
Descrizione	**Permette di recuperare i dati di un determinato Caregiver/Familiare dato il suo codice identificativo oppure data la sua email**
Pre-condizione	**context:** CaregiverFamiliareService:: get(string | | number codice):Promise<CaregiverFamiliare> **pre:** codice<>null and email<>null
Post-condizione	**context:** CaregiverFamiliareService:: get(string| | number codice): Promise<CaregiverFamiliare> **post:** caregiverFamiliare <> null

Nome Metodo	+save(CaregiverFamiliare caregiverFamiliare): Promise <number>
Descrizione	**Permette di salvare i dati di un Caregiver/Familiare e restituire il codice_identificativo assegnatogli**
Pre-condizione	**context:** CaregiverFamilareService::save(CaregiverFamiliare caregiverFamiliare): Promise <number> **pre:** caregiverFamiliare <> null and not get(caregiver.email)
Post-condizione	**context**: CaregiverFamiliareService:: save(CaregiverFamiliare caregiverFamiliare) : Promise <number> **post:** codice_identificativo <> null

Nome Metodo	+update(CaregiverFamiliare caregiverFamiliare): void
Descrizione	**Permette di modificare i dati di un Caregiver/Familiare**
Pre-condizione	**context:** PazienteService::update(CaregiverFamiliare caregiverFamiliare): void **pre:** caregiverFamiliare<> null
Post-condizione	N/A

Nome Classe	MedicoService
Descrizione	**Permette di gestire le operazioni relative al Medico**
Metodi	+getAll(): Promise<Medico[]> +get(string | | number codice): Promise <Medico>

	+save(Medico medico): void +update(Medico medico): void +getPazientiByMed(number med): Promise<Paziente[]> +getPazienteByMed(string cf): Promise<Paziente>
Invariante di classe	N/A

Nome Metodo	**+getAll(): Promise <Medico[]>**
Descrizione	**Permette di recuperare i dati del medico**
Pre-condizione	N/A
Post-condizione	**context:** MedicoService:: getAll: Promise<Medico > **post:** Medico[] <> null

Nome Metodo	**+get(string ||number codice): Promise <Medico>**
Descrizione	**Permette di recuperare i dati di un determinato medico dato il suo codice identificativo oppure data la sua email**
Pre-condizione	**context**: MedicoService:: get(string || number codice):Promise<Medico > **pre:** codice_identificativo<>null and email<>null
Post-condizione	**context:** MedicoService:: get(string|| number codice): Promise<Medico> **post:** Medico<>null

	+save(Medico medico): void
Descrizione	**Permette di salvare i dati di un Medico**
Pre-condizione	**context:** MedicoService:: save(Medico medico): void **pre:** medico <> null and not get(medico.codice_fiscale)
Post-condizione	N/A

	+update(Medico medico):void
Descrizione	**Permette di modificare i dati di un Medico**
Pre-condizione	**context**: MedicoService:: update(Medico medico): void **pre:** medico <> null

Post-condizione	N/A

Nome Metodo	+getPazientiByMed(number med): Promise<Paziente[]>
Descrizione	**Permette di recuperare i dati dei pazienti dato il loro medico**
Pre-condizione	**context:** MedicoService:: getPazientiByMed(number med): Promise<Paziente[]> **pre:** med<>null
Post-condizione	**context:** MedicoService:: getPazientiByMed(number med): Promise<Paziente[]> **post:** Paziente[]<>null

Nome Metodo	+getPazienteByMed(string cf): Promise<Paziente>
Descrizione	**Permette di recuperare i dati di un determinato paziente dato il medico**
Pre-condizione	**context:** MedicoService:: getPazienteByMed(string cf): Promise<Paziente> **pre:** cf<>null
Post-condizione	**context:** MedicoService:: getPazienteByMed(string cf): Promise<Paziente> **post:** Paziente<>null

Nome Classe	LineeGuidaService
Descrizione	**Permette di gestire le operazioni relative alle linee guida**
Metodi	+getAll(): Promise<LineaGuida[]> +get(number id): Promise<LineaGuida> +getByMed(number medico): Promise<LineaGuida> +save(LineaGuida lineeguida): void +update(LineaGuida lineeguida): void
Invariante di classe	N/A

Nome Metodo	+getAll(): Promise <LineaGuida[]>
Descrizione	**Permette di recuperare tutte le linee guida**
Pre-condizione	N/A

Post-condizione	**context:** LineeGuidaService:: getAll: Promise<LineeGuida[]> **post:** LineeGuida[] <> null

Nome Metodo	**+ get(number id): Promise<LineaGuida>**
Descrizione	**Permette di recuperare una serie di linee guida dato il loro id**
Pre-condizione	**context**: LineeGuidaService:: get(number id): Promise<LineaGuida> **pre:** id<>null
Post-condizione	**context:** LineeGuidaService:: get(number id): Promise<LineaGuida> **post:** LineaGuida<>null

Nome Metodo	**getByMed(number medico): Promise<LineaGuida>**
Descrizione	**Permette di recuperare le linee guida di uno specifico medico**
Pre-condizione	**context**: LineeGuidaService:: getByMed(number medico): Promise<LineaGuida> **pre:** medico<>null
Post-condizione	**context:** LineeGuidaService:: getByMed(number medico): Promise<LineaGuida> **post:** LineaGuida<>null

	+ save(LineaGuida lineeguida): void
Descrizione	**Permette di salvare delle linee guida**
Pre-condizione	**context:** LineeGuidaService:: save(LineaGuida lineeguida): void **pre:** lineeguida <> null
Post-condizione	N/A

	+ update(LineaGuida lineeguida): void
Descrizione	**Permette di modificare i dati di un Medico**
Pre-condizione	**context**: LineeGuidaService:: update(LineaGuida lineeguida): void **pre:** lineeguida <> null
Post-condizione	N/A

3.4 Package gestione_quiz_allenamento

Nome Classe	QuizAllenamentoService
Descrizione	**Permette di gestire le operazioni relative al Quiz di Allenamento Giornaliero**
Metodi	+getAll(): Promise <QuizAllenamentoGiornaliero[]> +get(number id): Promise <QuizAllenamentoGiornaliero> +save(QuizAllenamentoGiornaliero quizAllenamento): Promise <number> +update(QuizAllenamentoGiornaliero quizAllenamento): Promise<void> +getByCaregiverFamiliare(number caregiverFamiliare): Promise<QuizAllenamentoGiornaliero[]> +getAllDomande(): Promise <DomandaQuizAllenamento[]> +getDomanda(number id): Promise <DomandaQuizAllenamento> +saveDomanda(DomandaQuizAllenamento domanda):Promise<number> +getByQuizAllenamento(number id):Promise<DomandaQuizAllenamento[]> +getAllRisposta(): Promise <RispostaQuizAllenamento[]> +getRisposta(number id): Promise<RispostaQuizAllenamento> +saveRisposta(RispostaQuizAllenamento rispostaQuizAllenamento): void +getByDomandaAllenamento(number id):Promise<RispostaQuizAllenamento[]> +updateRisposta(RispostaQuizAllenamento risposta): Promise<void> +updateDomanda(DomandaQuizAllenamento domanda): Promise<void> + createQuizAllenamento(QuizAllenamentoGiornaliero quizAllenamento, Map<DomandaQuizAllenamento, RispostaQuizAllenamento[]> domandeRisposte): Promise<void> +getDomandeRisposte(number quizAllenamento): Promise<{ [key: string]: DomandeRisposte }>;
)Invariante di classe	N/A

Nome Metodo	+getAll(): Promise <QuizAllenamentoGiornaliero[]>
Descrizione	**Permette di recuperare tutti i quiz di allenamento giornaliero**
Pre-condizione	N/A
Post-condizione	**context**: QuizAllenamentoService:: getAll(): Promise <QuizAllenamentoGiornaliero[]> **post**: quizAllenamentoGiornaliero[] <> null

Nome Metodo	+get(number id): Promise <QuizAllenamentoGiornaliero>
Descrizione	**Permette di recuperare un quiz di allenamento giornaliero dato un id**
Pre-condizione	**context**: QuizAllenamentoService:: get(number id): Promise <QuizAllenamentoGiornaliero> **pre:** id <> null
Post-condizione	**context**: QuizAllenamentoService:: get(number id): Promise <QuizAllenamentoGiornaliero> **post:** quizAllenaementoGiornaliero <> null

Nome Metodo	+save(QuizAllenamentoGiornaliero quiz): Promise<number>
Descrizione	**Permette di creare un quiz di allenamento giornaliero**
Pre-condizione	**context**: QuizAllenamentoService:: save(QuizAllenamentoGiornaliero quiz): Promise <number> **pre:** quiz <> null
Post-condizione	**context**: QuizAllenamentoService:: save(QuizAllenamentoGiornaliero quiz): Promise <number> **post**: number <> null

Nome Metodo	+update(QuizAllenamentoGiornaliero quizAllenamento): Promise<void>
Descrizione	**Permette di modificare un quiz di allenamento giornaliero**
Pre-condizione	**context**: QuizAllenamentoService:: update(QuizAllenamentoGiornaliero quizAllenamento): Promise<void> **pre:** quizAllenamento <> null
Post-condizione	**N/A**

Nome Metodo	+getByCaregiverFamiliare(number caregiverFamiliare): Promise<QuizAllenamentoGiornaliero[]>
Descrizione	**Permette di recuperare tutti i quiz di allenamento giornaliero dato un determinato caregiver/familiare**
Pre-condizione	**context**: QuizAllenamentoService:: getByCaregiverFamiliare(number caregiverFamiliare): Promise<QuizAllenamentoGiornaliero[]> **pre:** caregiverFamiliare <> null

Post-condizione	**context**: QuizAllenamentoService:: getByCaregiverFamiliare(number caregiverFamiliare): Promise<QuizAllenamentoGiornaliero[]> **post**: quiz <> null

Nome Metodo	**+getAllDomande(): Promise <DomandaQuizAllenamento[]>**
Descrizione	**Permette di recuperare tutte le domande**
Pre-condizione	N/A
Post-condizione	**context**: QuizAllenamentoService:: getAllDomande(): Promise <DomandaQuizAllenamento[]> **post**: domande <> null

Nome Metodo	**+getDomanda(number id): Promise <Domanda>**
Descrizione	**Permette di recuperare un quiz di allenamento giornaliero dato un id**
Pre-condizione	**context**: QuizAllenamentoService:: getDomanda(number id): Promise <Domanda > **pre:** id <> null
Post-condizione	**context**: QuizAllenamentoService:: get(number id): Promise <QuizAllenamentoGiornaliero[]> **post:** quizAllenaementoGiornaliero <> null

Nome Metodo	**+saveDomanda(DomandaQuizAllenamento domanda): Promise<number>**
Descrizione	**Permette di aggiungere una domanda ad un quiz di allenamento giornaliero**
Pre-condizione	**context**: QuizAllenamentoService:: saveDomanda(DomandaQuizAllenamento domanda): Promise<number> **pre:** domanda <> null
Post-condizione	**context**: QuizAllenamentoService:: saveDomanda(DomandaQuizAllenamento domanda): Promise<number> **pos**t: number <> null

Nome Metodo	+getByQuizAllenamento (number quizAllenamento): Promise<DomandaQuizAllenamento[]>
Descrizione	**Permette di recuperare tutte le domande in base ad un quiz di allenamento**
Pre-condizione	**context**: QuizAllenamentoService:: getByCaregiverFamiliare(number caregiverFamiliare): Promise<QuizAllenamentoGiornaliero[]> **pre:** caregiverFamiliare <> null
Post-condizione	**context**: QuizAllenamentoService:: getByQuizAllenamento (number quizAllenamento): Promise<DomandaQuizAllenamento[]> **post**: quizAllenamento <> null

Nome Metodo	+getAllRisposte(): Promise <RispostaQuizAllenamento[]>
Descrizione	**Permette di recuperare tutti i quiz di allenamento giornaliero**
Pre-condizione	N/A
Post-condizione	**context**: QuizAllenamentoService:: getAllRisposta(): Promise <RispostaQuizAllenamento[]> **post**: risposte <> null

Nome Metodo	+getRisposta(number id): Promise <RispostaQuizAllenamento>
Descrizione	**Permette di recuperare una risposta in base al suo id**
Pre-condizione	**context**: QuizAllenamentoService:: getRisposta(number id): Promise <RispostaQuizAllenamento> **pre:** id <> null
Post-condizione	**context**: QuizAllenamentoService:: get(number id): Promise <RispostaQuizAllenamento> **post:** risposta<> null

Nome Metodo	+saveRisposta(RispostaQuizAllenamento risposta): void
Descrizione	**Permette di salvare una risposta al quiz di allenamento**
Pre-condizione	**context**: QuizAllenamentoService:: saveRisposta(RispostaQuizAllenamento risposta): void **pre:** risposta <> null

Post-condizione	N/A

Nome Metodo	**+getByDomandaAllenamento(number domanda): Promise<RispostaQuizAllenamento[]>**
Descrizione	**Permette di recuperare tutte le risposte di una determinata domanda di un quiz di allenamento**
Pre-condizione	**context**: QuizAllenamentoService:: getByDomandaAllenamento(number domanda): Promise<RispostaQuizAllenamento> **pre:** domanda<> null
Post-condizione	**context**: QuizAllenamentoService:: getByDomandaAllenamento(number domanda): Promise<RispostaQuizAllenamento> **post**: risposte <> null

Nome Metodo	**+updateRisposta(RispostaQuizAllenamento risposta): Promise<void>**
Descrizione	**Permette di modificare una risposta nel quiz di allenamento giornaliero**
Pre-condizione	**context**: QuizAllenamentoService:: updateRisposta(RispostaQuizAllenamento risposta): Promise<void> **pre:** risposta<> null
Post-condizione	N/A

Nome Metodo	**+updateDomanda(DomandaQuizAllenamento domanda): Promise<void>**
Descrizione	**Permette di modificare una domanda nel quiz di allenamento giornaliero**
Pre-condizione	**context**: QuizAllenamentoService:: updateDomanda(DomandaQuizAllenamento domanda): Promise<void> **pre:** domanda<> null
Post-condizione	N/A

Nome Metodo	**createQuizAllenamento(QuizAllenamentoGiornaliero quizAllenamento, Map<DomandaQuizAllenamento, RispostaQuizAllenamento[]> domandeRisposte): Promise<void>**
Descrizione	**Permette di creare un quiz di allenamento giornaliero**

Pre-condizione	**context**: QuizAllenamentoService:: createQuizAllenamento(QuizAllenamentoGiornaliero quizAllenamento, Map<DomandaQuizAllenamento, RispostaQuizAllenamento[]> domandeRisposte): Promise<void> **pre:** quizAllenamento **<>** null and domandeRisposte <> null
Post-condizione	N/A

Nome Metodo	**+getDomandeRisposte(number quizAllenamento): Promise<{ [key: string]: DomandeRisposte }>**
Descrizione	**Permette di recuperare le domande e le risposte di un quiz di allenamento giornaliero**
Pre-condizione	**context**: QuizAllenamentoService:: getDomandeRisposte(number quizAllenamento): Promise<{ [key: string]: DomandeRisposte }> **pre:** quizAllenamento **<>** null
Post-condizione	**context**: QuizAllenamentoService:: getDomandeRisposte(number quizAllenamento): Promise<{ [key: string]: DomandeRisposte }> **post:** DomandeRisposte **<>** null

3.5 Package gestione_todolist

Nome Classe	**ToDoListService**
Descrizione	**Permette di gestire le operazioni relative alla ToDoList**
Metodi	+getAll(): Promise <ToDoList[]> +getByMed(number medico): Promise<ToDoList[]> +getByPaziente(string paziente): Promise<ToDoList[]> +getByMedAndPaz(number medico,string paziente): Promise<ToDoList[]> +get(number id): Promise <ToDoList> +getAllAttivitaByToDoList(number toDoList): Promise<Attivita[]> +getAttivita(number id): Promise<Attivita> +updateAttivita(Attivita attivita): void +saveAttivita(Attivita attivita): Promise<number> +createToDoList(ToDoList toDoList,Attivita[] attività): Promise<void>

	+update(ToDoList toDoList): void +save(ToDoList toDoList): void
Invariante di classe	N/A

Nome Metodo	**+getAll(): Promise <ToDoList[]>**
Descrizione	**Permette di recuperare tutte le ToDoList**
Pre-condizione	**N/A**
Post-condizione	**Context:** ToDoListService::getAll(): Promise<ToDoLIst[]> **Post:** ToDoList[]<>null

Nome Metodo	**+getByMed(number medico): Promise <ToDoList>**
Descrizione	**Permette di recuperare tutte le ToDoList di un determinato Medico**
Pre-condizione	**context:** ToDoListService::getByMed(number medico): Promise<ToDoLIst[]> **pre:** med <> null
Post-condizione	**context:** ToDoListService::getByMed(number medico): Promise<ToDoLIst[]> **post:** ToDoList[]<>null

Nome Metodo	**+getByPaziente(number paz): Promise <ToDoList[]>**
Descrizione	**Permette di recuperare tutte le ToDoList di un determinato Paziente**
Pre-condizione	**context:** ToDoListService::getByPaziente(string paziente): Promise<ToDoLIst[]> **pre:** cg <> null
Post-condizione	**context:** ToDoListService::getByPaziente(string paziente): Promise<ToDoLIst[]> **post:** ToDoList[]<>null

Nome Metodo	**+getByMedAndPaz(number medico,string paziente): Promise<ToDoList[]>**
Descrizione	**Permette di recuperare tutte le ToDoList dato un determinato medico e paziente**

Pre-condizione	**context:** ToDoListService:: getByMedAndPaz(number medico,string paziente): Promise<ToDoList[]> **pre:** medico <> null and paziente <> null
Post-condizione	**context:** ToDoListService:: getByMedAndPaz(number medico,string paziente): Promise<ToDoList[]> **post:** ToDoList[]<>null

Nome Metodo	**+get(number id): Promise <ToDoList>**
Descrizione	**Permette di recuperare una specifica ToDoList**
Pre-condizione	**context:** ToDoListService::get(number id): Promise<ToDoLIst> **pre:** Id<>null
Post-condizione	**context:** ToDoListService::get(number id): Promise<ToDoLIst> **post:** ToDoList<>null

Nome Metodo	**+getAllAttivitaByToDoList(number toDoList): Promise<Attivita[]>**
Descrizione	**Permette di recuperare tutte le attività di una ToDoList**
Pre-condizione	**context:** ToDoListService::getAllAttivitaByToDoList(number toDoList): Promise<Attivita[]> **pre:** toDoList<>null
Post-condizione	**context:** ToDoListService::getAllAttivitaByToDoList(number toDoList): Promise<Attivita[]> **post:** Attivita[]<>null

Nome Metodo	**+getAttivita(number id): Promise<Attivita>**
Descrizione	**Permette di recuperare una specifica attività**
Pre-condizione	**context:** ToDoListService:: getAttivita(number id): Promise<Attivita> **pre:** id<>null
Post-condizione	**context:** ToDoListService:: getAttivita(number id): Promise<Attivita> **post:** attivita<>null

Nome Metodo	+updateAttivita(Attivita attivita): void
Descrizione	**Permette di modificare una attività**
Pre-condizione	**context:** ToDoListService:: updateAttivita(Attivita attivita): void **pre:** attivita<>null
Post-condizione	**N/A**

Nome Metodo	+saveAttivita(Attivita attivita): Promise<number>
Descrizione	**Permette di aggiungere una attività**
Pre-condizione	**context:** ToDoListService:: saveAttivita(Attivita attivita): Promise<number> **pre:** attivita<>null
Post-condizione	**N/A**

Nome Metodo	+createToDoList(ToDoList toDoList, Attivita[] attività): Promise<void>
Descrizione	**Permette di creare una ToDoList**
Pre-condizione	**context**: ToDoListService:: createToDoList(ToDoList toDoList, Attivita[] attività): Promise<void> **pre:** toDoList <> null and attività <> null
Post-condizione	N/A

Nome Metodo	+update(ToDoList toDoList): void
Descrizione	**Permette di modificare una ToDoList**
Pre-condizione	**context:** ToDoListService::update(ToDoList toDoList): void **pre:** toDoList<>null
Post-condizione	N/A

Nome Metodo	+save(ToDoList toDoList): void
Descrizione	**Permette di aggiungere una ToDoList**

Pre-condizione	**context:** ToDoListService::save(ToDoList toDoList): void **pre:** toDoList<>null
Post-condizione	N/A

3.6 Package gestione_quiz_preliminare

Nome Classe	QuizPreliminareService
Descrizione	**Permette di gestire le operazioni relative al Quiz Preliminare**
Metodi	+getAll(): Promise<QuizPreliminare[]> +getByMed(number medico): Promise<QuizPreliminare[]> +getByPaziente(string paziente):Promise<QuizPreliminare> +get(number id): Promise <QuizPreliminare> +save(QuizPreliminare quizPreliminare): Promise<number> +update(QuizPreliminare quizPreliminare): void +getAllDomande(): Promise<DomandaQuizPreliminare[]> +getDomanda(number id):Promise<DomandaQuizPreliminare> +saveDomanda(DomandaQuizPreliminare domanda): Promise<number> +updateDomanda(DomandaQuizPreliminare domanda): void +getByQuizPreliminare(number quizPreliminare): Promise<DomandaQuizPreliminare[]> +getRispostaByPaziente(string paziente, number id): Promise< RispostaQuizPreliminare> +getRisposta (number id):Promise< RispostaQuizPreliminare> +saveRisposta(RispostaQuizPreliminare risposta): void +updateRisposta(RispostaQuizPreliminare risposta): void +getDomandeByQuizPreliminare(number id): Promise<DomandaQuizPreliminare[]> +saveQuizPreliminare(QuizPreliminare quizPreliminare, Map<DomandaQuizPreliminare, RispostaQuizPreliminare> domandeRisposte): void +getDomandeRisposte(number quizPreliminare, string paziente): Promise<{ [key:string]: DomandeRisposte }>
Invariante di classe	N/A

Nome Metodo	+getAll():Promise <QuizPreliminare[]>
Descrizione	**Permette di recuperare tutti i quiz preliminari**
Pre-condizione	N/A
Post-condizione	**context:** QuizPreliminareService:: getAll():Promise<QuizPreliminare[]> **post:** quizPreliminare[]<>null

Nome Metodo	+getByMed(number medico):Promise <QuizPreliminare[]>
Descrizione	**Permette di recuperare tutti i quiz preliminari di un determinato medico**
Pre-condizione	**context:** QuizPreliminareService::getByMed(number medico): Promise <QuizPreliminare[]> **pre:** medico<>null
Post-condizione	**context:** QuizPreliminareService::getByMed(number medico): Promise<QuizPreliminare[]> **post:** QuizPreliminare[]<>null

Nome Metodo	+getByPaziente(string paziente):Promise <QuizPreliminare>
Descrizione	**Permette di recuperare tutti i quiz preliminari di un determinato paziente**
Pre-condizione	**context:** QuizPreliminareService::getByPaziente(string paziente): Promise <QuizPreliminare[]> **pre:** paziente<>null
Post-condizione	**context:** QuizPreliminareService::getByPaziente(string paziente): Promise<QuizPreliminare[]> **post:** paziente<>null

Nome Metodo	+get(number id):Promise<QuizPreliminare>
Descrizione	**Permette di recuperare i dati di un determinato quiz preliminare dato il suo id**
Pre-condizione	**context:** QuizPreliminareService::get(number id): Promise<QuizPreliminare[]> **pre:**QuizPreliminare<>null
Post-condizione	**context:** QuizPreliminareService::get(number id): Promise<QuizPreliminare[]> **post:** quizPreliminare<>null

Nome Metodo	+save(QuizPreliminare quizPreliminare): Promise<number>
Descrizione	**Permette di aggiungere un quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService::saveQuizPreliminare(QuizPreliminare quizPreliminare): Promise<void> **pre:** quizPreliminare<>null
Post-condizione	**N/A**

Nome Metodo	+update(QuizPreliminare quizPreliminare): void
Descrizione	**Permette di modificare un quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService::updateQuizPreliminare(QuizPreliminare quizPreliminare): void **pre:** quizPreliminare<>null
Post-condizione	**N/A**

Nome Metodo	+getAllDomande(): Promise<DomandaQuizPreliminare[]>
Descrizione	**Permette di recuperare tutte le domande**
Pre-condizione	**N/A**
Post-condizione	**context:** QuizPreliminareService::getAllDomande(): Promise<DomandaQuizPreliminare[]> **post:** DomandaQuizPreliminare[]<>null

Nome Metodo	+getDomanda(number id): Promise<DomandaQuizPreliminare>
Descrizione	**Permette di recuperare un quiz preliminare dato da un id**
Pre-condizione	**context:** QuizPreliminareService::getDomanda(number id): Promise <DomandaQuizPreliminare> **pre:** id<>null
Post-condizione	**context:** QuizPreliminareService::get(number id): Promise<QuizPreliminare> **post:** QuizPreliminare<>null

Nome Metodo	+saveDomanda(DomandaQuizPreliminare domanda): Promise<number>
Descrizione	**Permette di salvare una domanda al quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService::saveDomanda(DomandaQuizPreliminare domanda): Promise<number> **pre:** domandaQuizPreliminare<>null
Post-condizione	**N/A**

Nome Metodo	+updateDomanda(DomandaQuizPreliminare domanda): void
Descrizione	**Permette di modificare una domanda di quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService::updateDomanda(DomandaQuizPreliminare Domanda): void **pre:** DomandaquizPreliminare<>null
Post-condizione	**N/A**

Nome Metodo	+getByQuizPreliminare (number quizPreliminare): Promise<DomandaQuizPreliminare[]>
Descrizione	**Permette di recuperare tutte le domande in base ad un quiz preliminare**
Pre-condizione	**context**: QuizPreliminareService:: getByMedico(number medico): Promise<QuizPreliminare[]> **pre:** medico <> null
Post-condizione	**context**: QuizPreliminareService:: getByQuizPreliminare (number quizPreliminare): Promise<DomandaQuizPreliminare[]> **post**: quizPreliminare <> null

Nome Metodo	+ getRispostaByPaziente(string paziente, number id): Promise< RispostaQuizPreliminare>
Descrizione	**Permette di recuperare la risposta di una specifica domanda del quiz preliminare di un paziente**
Pre-condizione	**context**: QuizPreliminareService:: getRispostaByPaziente(string paziente, number id): Promise< RispostaQuizPreliminare> **pre**: paziente<> null and id<> null

Post-condizione	**context**: QuizPreliminareService:: getRispostaByPaziente(string paziente, number id): Promise< RispostaQuizPreliminare> **post**: RispostaQuizPreliminare<> null

Nome Metodo	**+getRisposta(number id): Promise<RispostaQuizPreliminare>**
Descrizione	**Permette di recuperare una risposta in base al suo id**
Pre-condizione	**context:** QuizPreliminareService::getRisposta(number id): Promise<RispostaQuizPreliminare> **pre:** id<>null
Post-condizione	**context**: QuizPreliminareService:: get(number id): Promise<RispostaQuizPreliminare> **post**: risposta<> null

Nome Metodo	**+updateDomanda(DomandaQuizPreliminare domanda): void**
Descrizione	**Permette di modificare una domanda di quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService::updateDomanda(DomandaQuizPreliminare Domanda): void **pre:** DomandaquizPreliminare<>null
Post-condizione	**N/A**

Nome Metodo	**+saveDomanda(DomandaQuizPreliminare domanda): void**
Descrizione	**Permette di salvare una risposta al quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService::saveDomanda(RispostaQuizPreliminare risposta): void **pre:** risposta<>null
Post-condizione	**N/A**

Nome Metodo	+getDomandeByQuizPreliminare (number quizPreliminare): Promise<DomandaQuizPreliminare[]>
Descrizione	**Permette di recuperare tutte le domande di un determinato quiz preliminare**
Pre-condizione	**context**: QuizPreliminareService:: getDomandeByQuizPreliminare (number quizPreliminare): Promise<DomandaQuizPreliminare[]> **pre:** quizPreliminare<> null
Post-condizione	**context**: QuizPreliminareService:: getDomandeByQuizPreliminare (number quizPreliminare): Promise<DomandaQuizPreliminare[]> **post**: DomandaQuizPreliminare[] <> null

Nome Metodo	+saveQuizPreliminare(QuizPreliminare quizPreliminare, Map<DomandaQuizPreliminare, RispostaQuizPreliminare> domandeRisposte):void
Descrizione	**Permette di salvare un determinato quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService:: saveQuizPreliminare(QuizPreliminare quizPreliminare, Map<DomandaQuizPreliminare, RispostaQuizPreliminare> domandeRisposte): void **pre:** quizPreliminare<> null and domandeRisposte <> null
Post-condizione	**N/A**

Nome Metodo	+getDomandeRisposte(number quizPreliminare, string paziente): Promise<{ [key:string]: DomandeRisposte }>
Descrizione	**Permette di ricavare le domande e le risposte di un determinato quiz preliminare**
Pre-condizione	**context:** QuizPreliminareService:: getDomandeRisposte(number quizPreliminare, string paziente): Promise<{ [key:string]: DomandeRisposte }> **pre:** quizPreliminare<> null and paziente <> null
Post-condizione	**context:** QuizPreliminareService:: getDomandeRisposte(number quizPreliminare, string paziente): Promise<{ [key:string]: DomandeRisposte }> **post:** DomandeRisposte <> null

3.7 Package gestione_filastrocca

Nome Classe	FilastroccaService
Descrizione	**Permette di gestire le operazioni relative ad una filastrocca**
Metodi	+get(number id): Promise <Filastrocca> +save(Filastrocca filastrocca): void +update(Filastrocca filastrocca): void +getByCaregiverFamiliare(number caregiverFamiliare): Promise<Filastrocca[]>
Invariante di classe	N/A

Nome Metodo	+get(number id): Promise <Filastrocca>
Descrizione	**Permette di recuperare di una filastrocca dato il suo id**
Pre-condizione	**context**: FilastroccaService:: get(number id): Promise<Filastrocca> **pre**: id <> null
Post-condizione	**context**: FilastroccaService::get(number id): Promise<Filastrocca> **post**: filastrocca <> null

Nome Metodo	+save(Filastrocca filastrocca): void
Descrizione	**Permette di creare una nuova filastrocca**
Pre-condizione	**context**: FilastroccaService:: save(Filastrocca filastrocca): Promise<Filastrocca> **pre**: filastrocca <> null
Post-condizione	N/A

Nome Metodo	+update(Filastrocca filastrocca): void
Descrizione	**Permette di modificare una filastrocca**
Pre-condizione	**context**: FilastroccaService:: update(Filastrocca filastrocca): void **pre**: filastrocca <> null
Post-condizione	N/A

Nome Metodo	+getByCaregiverFamiliare(number caregiverFamiliare): Promise<Filastrocca[]>
Descrizione	**Permette di recuperare tutte le filastrocche associate ad un determinato caregiverFamiliare**
Pre-condizione	**context**: FilastroccaService:: getByCaregiverFamiliare(number caregiverFamiliare): Promise<Filastrocca[]> **pre**: caregiverFamiliare <> null
Post-condizione	**context**: FilastroccaService:: getByCaregiverFamiliare(number caregiverFamiliare): Promise<Filastrocca[]> **post**: filastroccha[] <> null

3.8 Package storie

Nome Classe	StoriaService
Descrizione	**Permette di gestire le operazioni relative ad una storia**
Metodi	+getAll(): Promise<Storia[]> +get(number id): Promise <Storia> +save(Storia storia): Promise<number> +update(Storia storia): void +getByCaregiverFamiliare(number caregiverFamiliare): Promise<Storia[]> +getAllMedia(): Promise<Media[]> +getMedia(number id): Promise <Media> +saveMedia(Media media): void +updateMedia(Media media): void +getMediaByStoria(number storia): Promise<Media[]>
Invariante di classe	N/A

Nome Metodo	+getAll(): Promise<Storia[]>
Descrizione	**Permette di recuperare tutte le storie**
Pre-condizione	N/A
Post-condizione	**context**: StoriaService:: getAll(): Promise<Storia[]> **post**: storia[] <> null

Nome Metodo	+get(number id): Promise<Storia>
Descrizione	**Permette di recuperare una storia in base al suo id**
Pre-condizione	**context**: StoriaService:: get(number id): Promise<Storia> **pre**: id <> null
Post-condizione	**context**: StoriaService:: get(number id): Promise<Storia> **post**: storia <> null

Nome Metodo	+save(Storia storia): Promise<number>
Descrizione	**Permette di creare una nuova storia**
Pre-condizione	**context**: StoriaService:: save(Storia storia): Promise<number> **pre**: storia <> null
Post-condizione	N/A

Nome Metodo	+update(Storia storia): void
Descrizione	**Permette di modificare una storia**
Pre-condizione	**context**: StoriaService:: update(Storia storia): void **pre**: storia <> null
Post-condizione	N/A

Nome Metodo	+getByCaregiverFamiliare(number caregiverFamiliare): Promise<Storia[]>
Descrizione	**Permette di recuperare tutte le storie associate ad un determinato caregiverFamiliare**
Pre-condizione	**context**: StoriaService:: getByCaregiver(number caregiverFamiliare): Promise<Storia[]> **pre**: caregiverFamiliare <> null
Post-condizione	**context**: StoriaService:: getByCaregiver(number caregiverFamiliare): Promise<Storia[]>

	post: storie <> null

Nome Metodo	+getAllMedia(): Promise<Media[]>
Descrizione	**Permette di recuperare tutti media**
Pre-condizione	N/A
Post-condizione	**context**: StoriaService:: getAllMedia(): Promise<Media[]> **post**: media <> null

Nome Metodo	+getMedia(number id): Promise<Media>
Descrizione	**Permette di recuperare una storia in base al suo id**
Pre-condizione	**context**: StoriaService:: getMedia(number id): Promise<Media> **pre**: id <> null
Post-condizione	**context**: StoriaService:: getMedia(number id): Promise<Media> **post**: media <> null

Nome Metodo	+saveMedia(Media media): void
Descrizione	**Permette di creare una nuova storia**
Pre-condizione	**context**: StoriaService:: save(Media media): void **pre**: media <> null
Post-condizione	N/A

Nome Metodo	+updateMedia(Media media): void
Descrizione	**Permette di modificare una storia**
Pre-condizione	**context**: StoriaService:: updateMedia(Media media): void **pre**: media <> null
Post-condizione	N/A

Nome Metodo	+getMediaByStoria(number storia): Promise<Media[]>
Descrizione	**Permette di recuperare tutti i media relativi ad una determinata storia**
Pre-condizione	**context**: StoriaService:: getMediaByStoria(number storia): Promise<Media[]> **pre**: storia <> null
Post-condizione	**context:** StoriaService:: getMediaByStoria(number storia): Promise<Media[]> **post**: media[]<> null

3.9 Package gestione tac

Nome Classe	TacService
Descrizione	**Permette di gestire le operazioni relative alla TAC**
Metodi	+getAll(): Promise<Tac[]> +get(number id): Promise <Tac> +getByMed(number medico): Promise <Tac[]> +getByPaziente(string paz): Promise <Tac[]> +save(Tac tac): void +update(Tac tac): void
Invariante di classe	N/A

Nome Metodo	+getAll(): Promise<Tac[]>
Descrizione	**Permette di recuperare tutte le TAC**
Pre-condizione	N/A
Post-condizione	**context**: TacService:: getAll(): Promise<Tac[]> **post**: Tac[] <> null

Nome Metodo	+get(number id): Promise<Tac>
Descrizione	**Permette di recuperare una specifica TAC**
Pre-condizione	**context**: TacService:: get(number id): Promise<Tac> **pre**: id <> null

Post-condizione	**context**: TacService:: get(number id): Promise<Tac> **post**: Tac <> null
Nome Metodo	+getByMed(number med): Promise <Tac[]>
Descrizione	**Permette di recuperare tutte le TAC di un determinato Medico**
Pre-condizione	**Context:** TacService:: getByMed(number medico): Promise <Tac[]> **pre:** med <> null
Post-condizione	**Context:** TacService: getByMed(number medico): Promise <Tac[]> **post:** Tac[]<>null

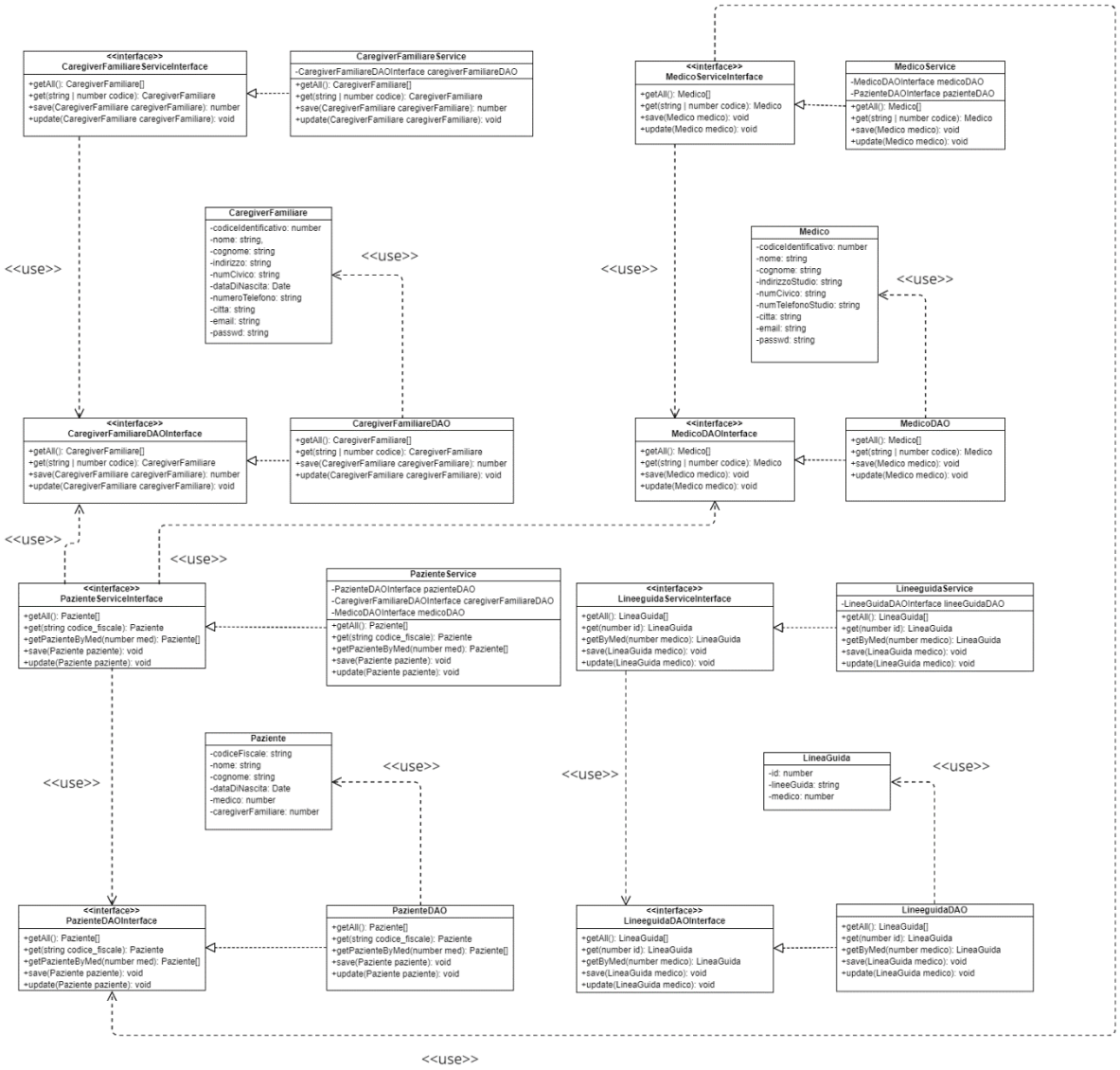
Nome Metodo	**+getByPaziente(number paz): Promise <Tac[]>**
Descrizione	**Permette di recuperare tutte le TAC di un determinato Caregiver/Familiare**
Pre-condizione	**Context:** TacService:: getByPaziente(string paz): Promise <Tac[]> **pre:** paz<> null
Post-condizione	**Context:** TacService:: getByPaziente(string paz): Promise <Tac[]> **post:** Tac[]<>null

Nome Metodo	**+save(Tac tac): void**
Descrizione	**Permette di creare una nuova TAC**
Pre-condizione	**context**: TacService:: save(Tac tac): void **pre**: tac <> null
Post-condizione	N/A

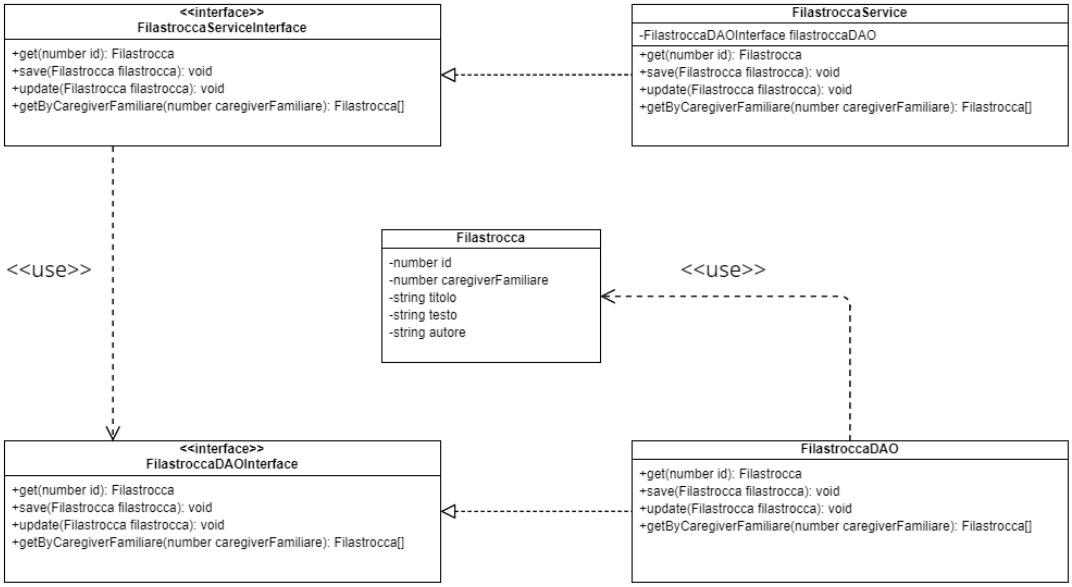
Nome Metodo	**+update(Tac tac): void**
Descrizione	**Permette di modificare una TAC**
Pre-condizione	**context**: TacService:: update(Tac tac): void **pre**: tac <> null
Post-condizione	N/A

4.Class diagram

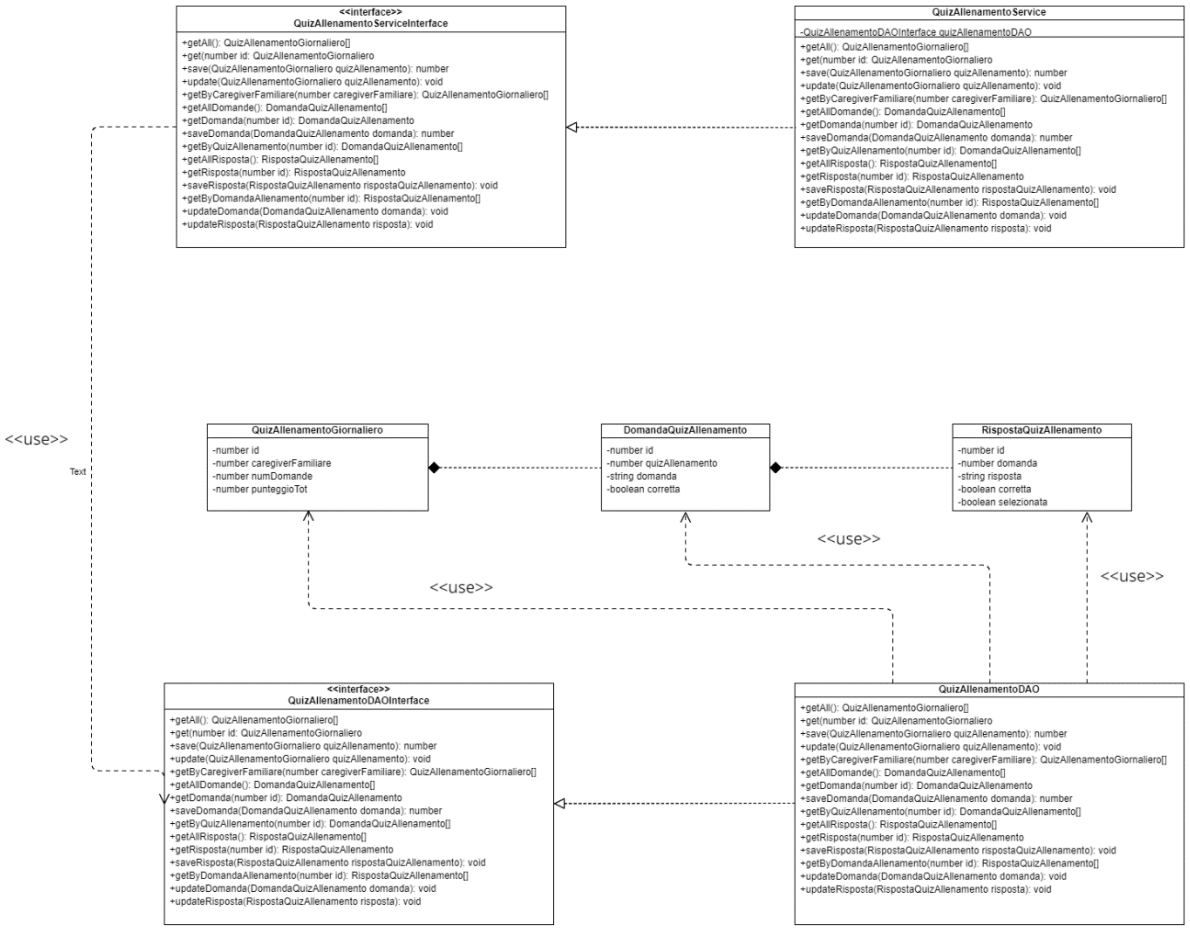
4.1 Package gestione_autenticazione



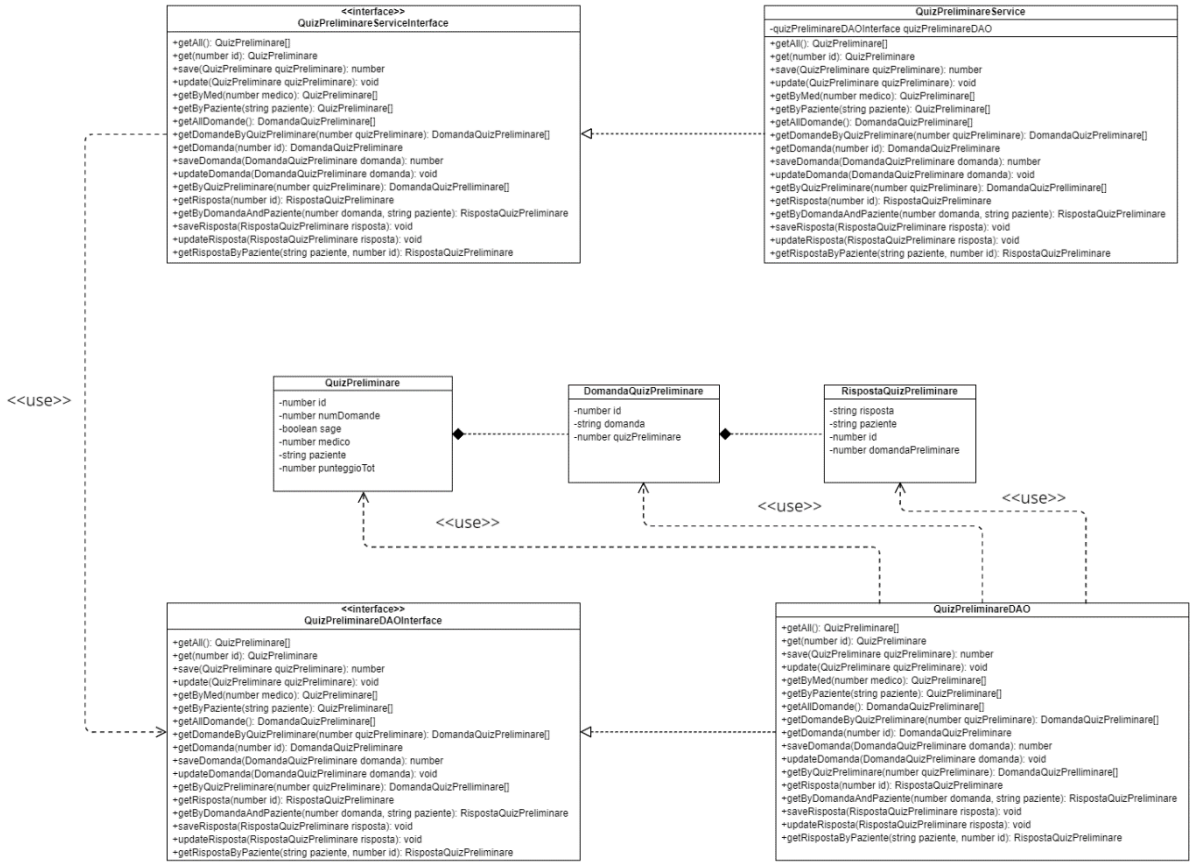
4.2 Package gestione_filastrocca



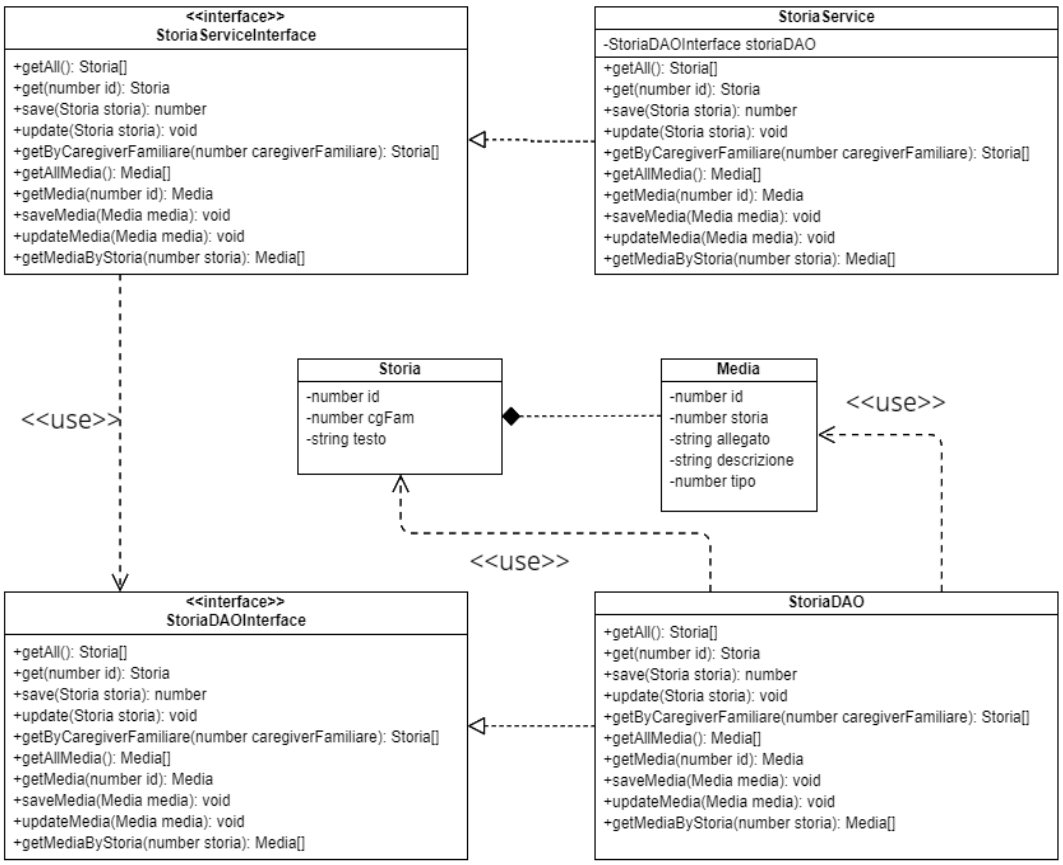
4.3 Package gestione_quiz_allenamento



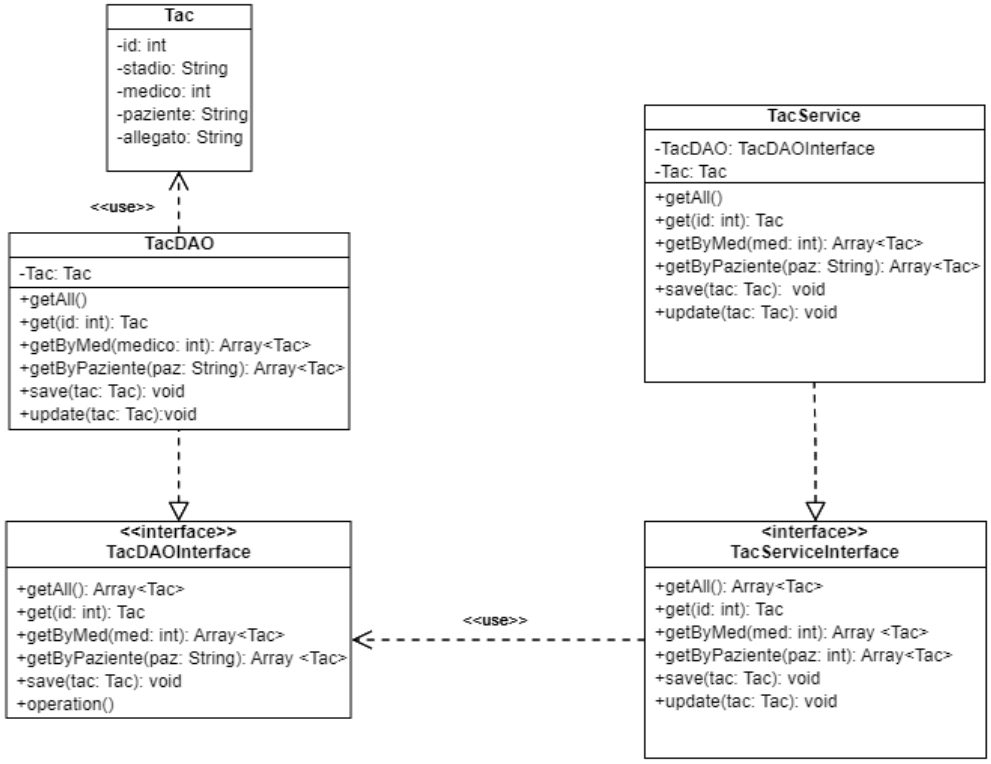
4.4 Package gestione_quiz_preliminare



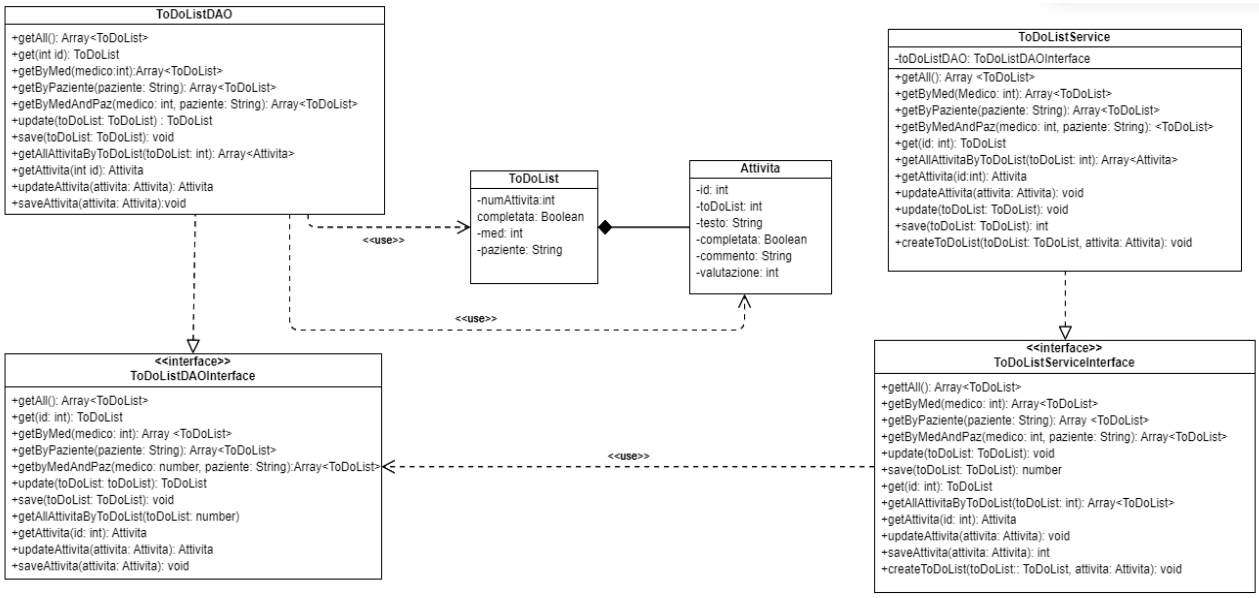
4.5 Package gestione_storia



4.6 Package gestione_tac



4.7 Package gestione_todolist



5.Glossario

Sigla/Termine	Definizione
Package	Raggruppamento di classi ed interfacce
DAO	Data Access Object, implementazione dell'omonimo pattern architetturale che si occupa di fornire l'accesso in modo astratto ai dati persistenti
Control	Classe che si occupa di gestire le richieste effettuate dal client
Service	Classi che implementano la logica di servizio
Routes	Package che raggruppa classi che forniscono endpoint tramite metodi get e post
Facade	Un oggetto che permette, attraverso un'interfaccia più semplici, di accedere a sottosistemi che espongono interfacce diverse e complesse
Adapter	Un pattern Strutturale che può basarsi su classi o oggetti, il cui fine è fornire una soluzione astratta al problema dell'interoperabilità tra interfacce differenti
Singleton	Un pattern creazionale con lo scopo di garantire che di una determinata classe venga strutturata una sola istanza e di fornire a quest'ultima un punto di accesso globale
Interfaccia	Classe non istanziabile che presenta un elenco esaustivo di tutti i metodi che verranno implementati dalle classi che la implementano
Design Patterns	Soluzioni universalmente riconosciute a problemi ricorrenti nello sviluppo di un progetto software