O.D.D. - Object Design Document

Riferimento	C09-ODD-1.0.0
Versione	1.0.0
Data	21/12/2023
Destinatario	Prof. ssa. Ferrucci Filomena
Redatto da	Antonio D'Amato, Ludovica D'Amato, Domenico D'Urso, Gabriel Matteo Balasa, Raffaele Vietri, Giuseppe Raiola Paduano, Raffaele Curcio, Francesco Saporito.
Approvato da	Alfonso Cannavale, Domenico Antonio Gioia, Antonio Scognamiglio

Revision History

Data	Versione	Descrizione	Autori
11/12/2023	0.1.0	Prima redazione	Giuseppe Raiola Paduano
13/12/2023	0.2.0	Stesura Object Design e Trade off	Gabriel Matteo Balasa, Domenico D'Urso, Giuseppe Raiola Paduano, Raffaele Vietri
21/12/2023	1.0.0	Revisionato	Tutto il team

Team Members

Nome	Ruolo del Progetto	Acronimo	Informazioni di contatto
Alfonso Cannavale	Project Manager	AC	a.cannavale7
Antonio Scognamiglio	Project Manager	AS	a.scognamiglio32
Domenico Antonio Gioia	Project Manager	DAG	d.gioia7
Antonio D'Amato	Team Member	AD	a.damato73
Ludovica D'Amato	Team Member	LD	l.damato17
Gabriel Matteo Balasa	Team Member	GMB	g.balasa
Giuseppe Raiola Paduano	Team Member	GRP	g.raiolapaduano
Domenico D'Urso	Team Member	DD	d.durso9
Raffaele Vietri	Team Member	RF	r.vietri22
Raffaele Curcio	Team Member	RC	r.curcio16
Francesco Saporito	Team Member	FS	f.saporito7

Indice

System Design Document del progetto SUSTAINER

1. Introduzione

Sustainer nasce con il principale scopo di offrire l'addestramento di modelli di machine learning, progettata per offrire agli utenti un'esperienza intuitiva e personalizzata. L'applicazione consente agli utenti di selezionare e ottimizzare i parametri di apprendimento in base al proprio contesto, seguito da un monitoraggio dettagliato del progresso durante il processo di addestramento.
In questa parte del documento, esploreremo l'Object Design, fornendo una panoramica degli object design goal, delle linee guida per l'implementazione e una visione dettagliata dei package e delle classi del sistema.

1.1 Object Design e Trade off

Sulla base dei trade-off selezionati considerando i più necessari per l'implementazione, gli object design goal presi in considerazione sono:

- **Dependability vs Tempi di risposta** - Per resistere a potenziali input non validi si aggiungono strati di controllo nel codice prolungando l'attesa dell'utente.
  - **Robustezza:** Capacità del sistema di reagire in maniera coerente e funzionale anche in circostanze non previste o non specificate. L'implementazione avverrà tramite la gestione delle eccezioni nel caso di input invalidi.
- **Maintenance vs Performance** - Per aumentare la manutenibilità del codice, le performance degradano data la complessità delle strutture dati che si adottano.
  - **Manutenibilità:** Capacità di apportare modifiche, correzioni e aggiornamenti del sistema in modo efficiente, senza compromettere la stabilità e affidabilità, ma gravando sulle performance. Un primo approccio è attraverso l'uso di design pattern.
- **Comprensibilità del Codice vs Tempi di implementazione** - Per favorire una migliore leggibilità, il tempo necessario per l'implementazione, rispettando convenzioni e regole, aumenta.

- **Leggibilità:** Facilità di lettura e comprensione del codice sorgente del sistema a svantaggio di un'implementazione temporalmente più costosa, tutto ciò attraverso convenzioni, best practice e uso di applicativi esterni per il controllo della code-quality come ESLint e Prettier.

1.2 Interface Documentation guidelines

In questa sezione verranno descritte le regole di stile per la scrittura del codice in modo da renderlo leggibile e favorire una consistenza tra tutti i file del progetto. Lo standard utilizzato per lo stile di scrittura del codice sono le regole di ESLint utilizzate da Airbnb.
Per la documentazione dovrà essere seguito lo stile dei commenti definito da TypeDoc definito al link seguente:
https://typedoc.org/guides/doccomments/

1.3 Definizioni, acronimi e abbreviazioni

Di seguito un elenco di alcuni termini presenti nel documento:

- **Package:** insieme di classi, interfacce e file correlati.
- **Design Pattern:** soluzioni a problemi ricorrenti impiegati per ottenere riuso, flessibilità e manutenibilità.
- **Interfaccia:** insieme delle firme delle operazioni, o metodi, offerti dalla classe;
- **Service:** classi che implementano la logica di business e forniscono un'interfaccia per la comunicazione con il resto del sistema.

1.4 Riferimenti

Di seguito una lista di documenti progettuali utili per la lettura:

- **RAD:** Requirements Analysis Document
- **SDD:** System Design Document
- **TP:** Test Plan Document
- **TCS:** Test Case Specification Document

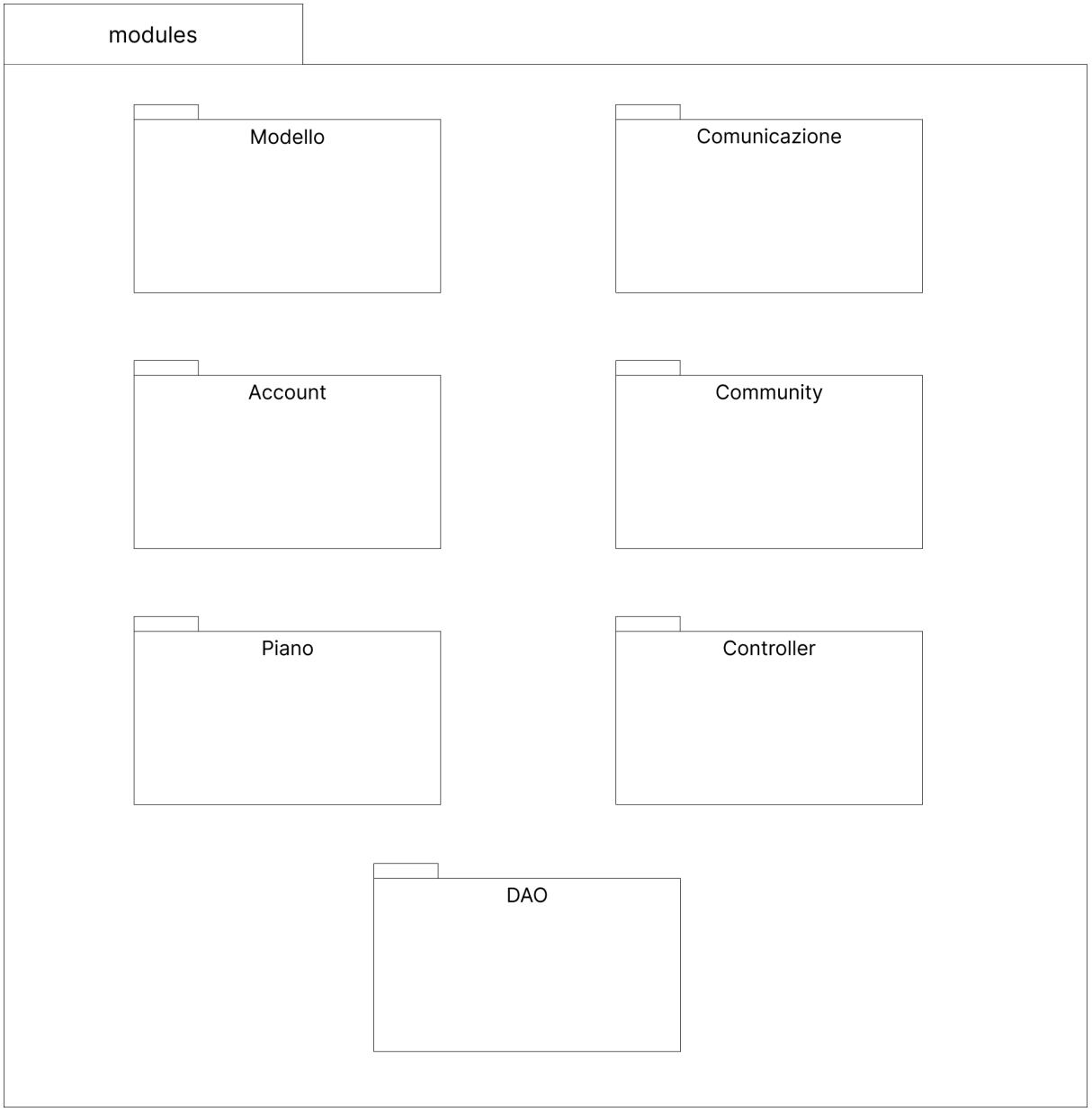
2. Packages

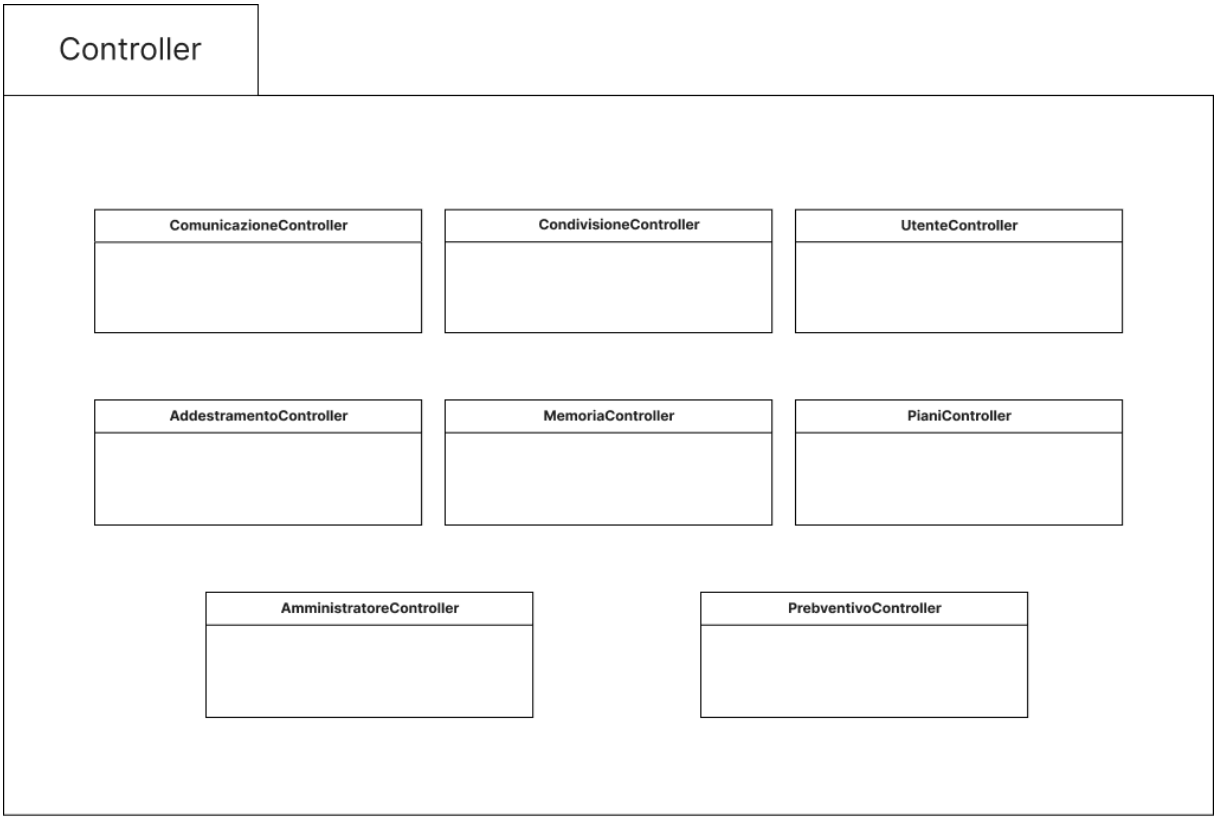
In questa parte del documento, viene presentata la divisione del sistema in package, conforme a quanto specificato nel System Design. La ragione di questa suddivisione è derivata dalle decisioni architetturali adottate e dalla struttura di directory standard per un progetto sviluppato in Node.js e React.

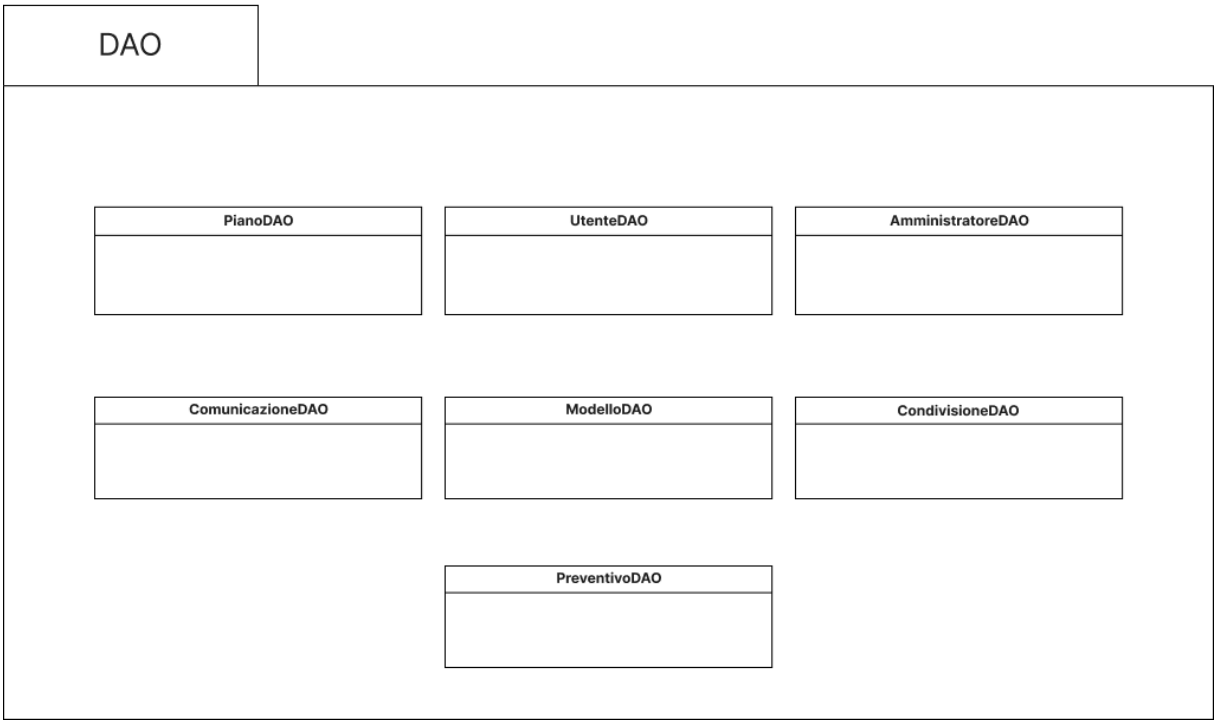
- **node_modules,** contiene le dependency necessarie al progetto
- **public,** contiene risorse pubbliche e statiche accessibili dall'HTML
- **src,** contiene tutti i file sorgente
    - **.eslintrc.cjs,** contiene regole e configurazioni ESLint per definire gli standard del codice
    - **.gitignore,** contiene file e cartelle che dovranno essere ignorati da git
    - **index.html,** contiene la struttura di una pagina HTML
    - **package-lock.json,** contiene informazioni sulla versione specifica di ogni modulo Node.js e le sue dipendenze
    - **package.json,** contiene
        - **scripts,** contiene scripts preimpostati e personalizzati
        - **dependencies,** contiene le dependencies necessarie al funzionamento
        - **devDependencies,** contiene dependencies necessarie durante lo sviluppo
    - **README.md,** contiene informazioni essenziali su come utilizzare, installare e contribuire al progetto.
    - **vite.config.js,** contiene file di configurazione di Vite

Package modules

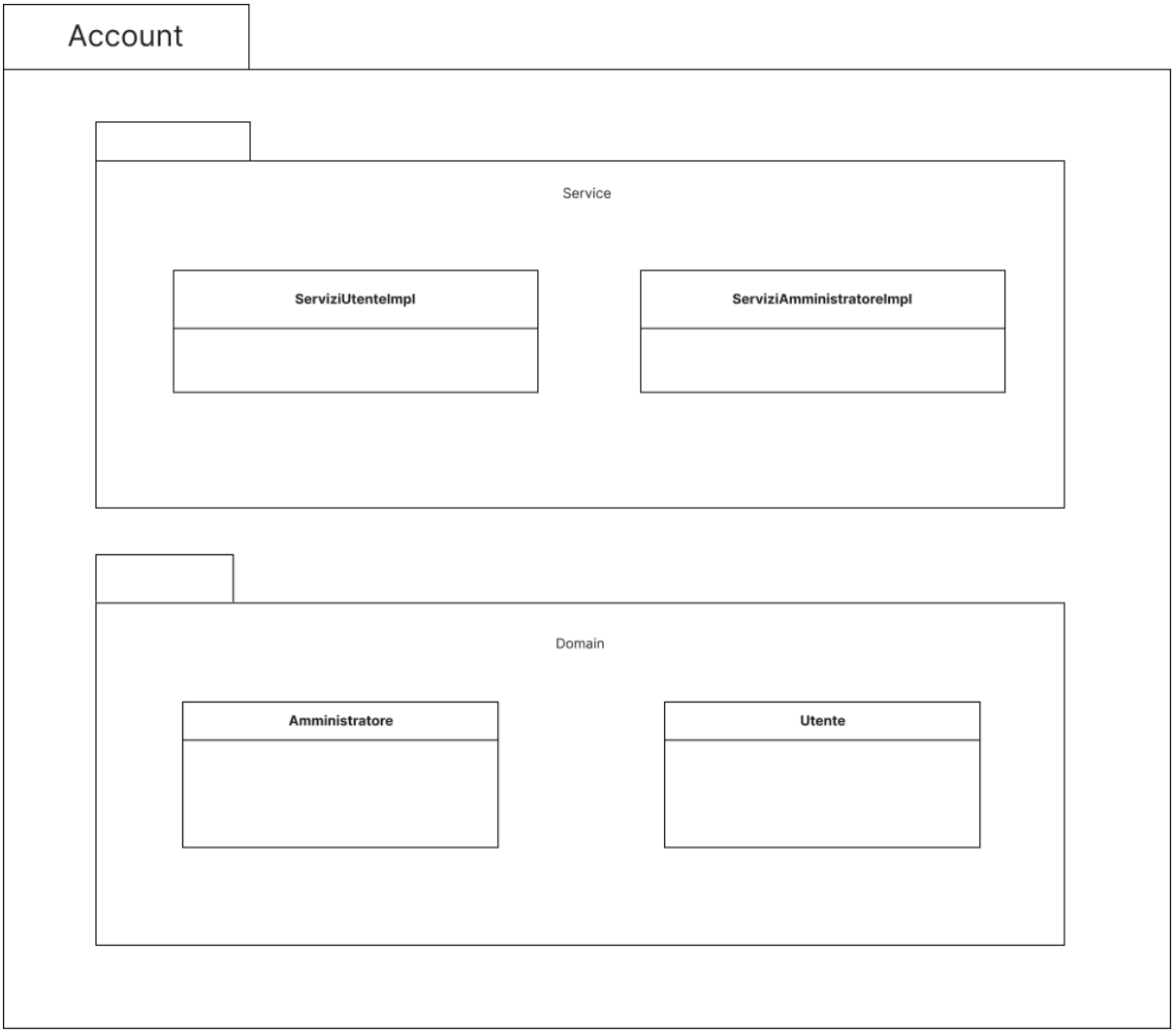
Di seguito è riportata la struttura del package principale di Sustainer. Per la suddivisione è stato usato come criterio la suddivisione in package per ogni sottosistema, contenente le classi service ed eventuali classi utili al contesto del sottosistema.


modules
Modello
Comunicazione
Account
Community
Piano
Controller
DAO

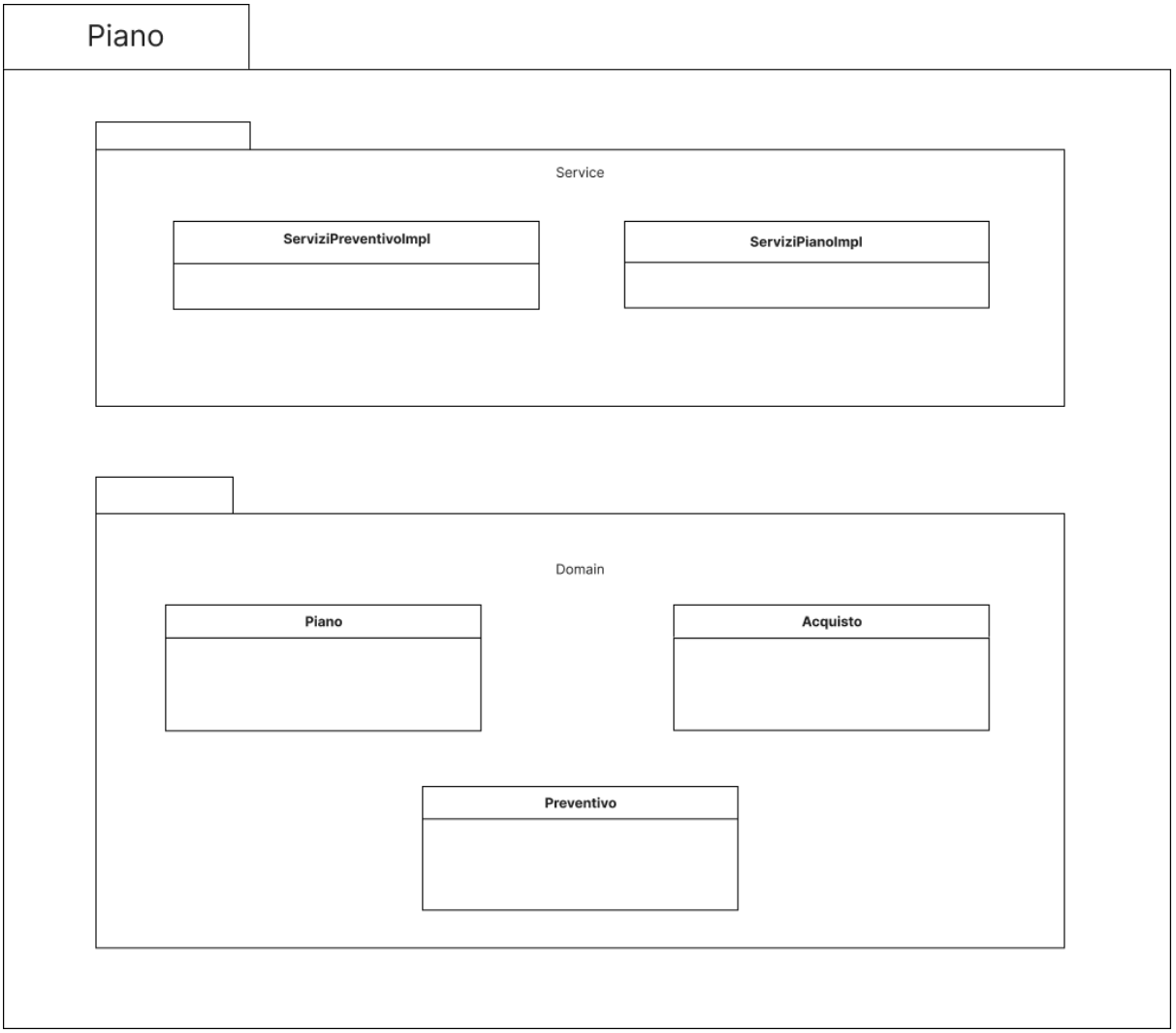

Controller
ComunicazioneController
CondivisioneController
UtenteController
AddestramentoController
MemoriaController
PianiController
AmministratoreController
PrebventivoController


DAO
PianoDAO
UtenteDAO
AmministratoreDAO
ComunicazioneDAO
ModelloDAO
CondivisioneDAO
PreventivoDAO

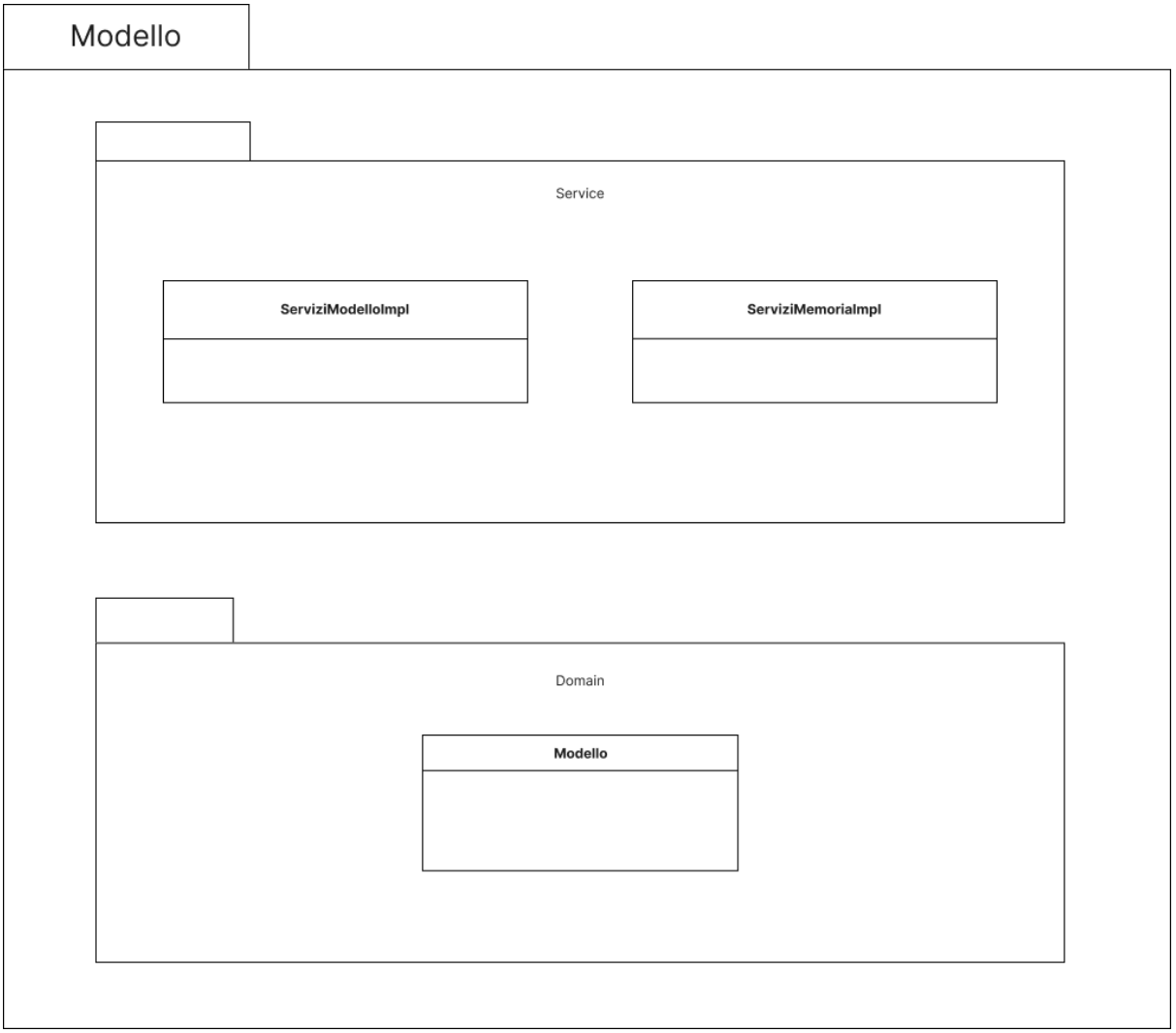
2.1 Package Account



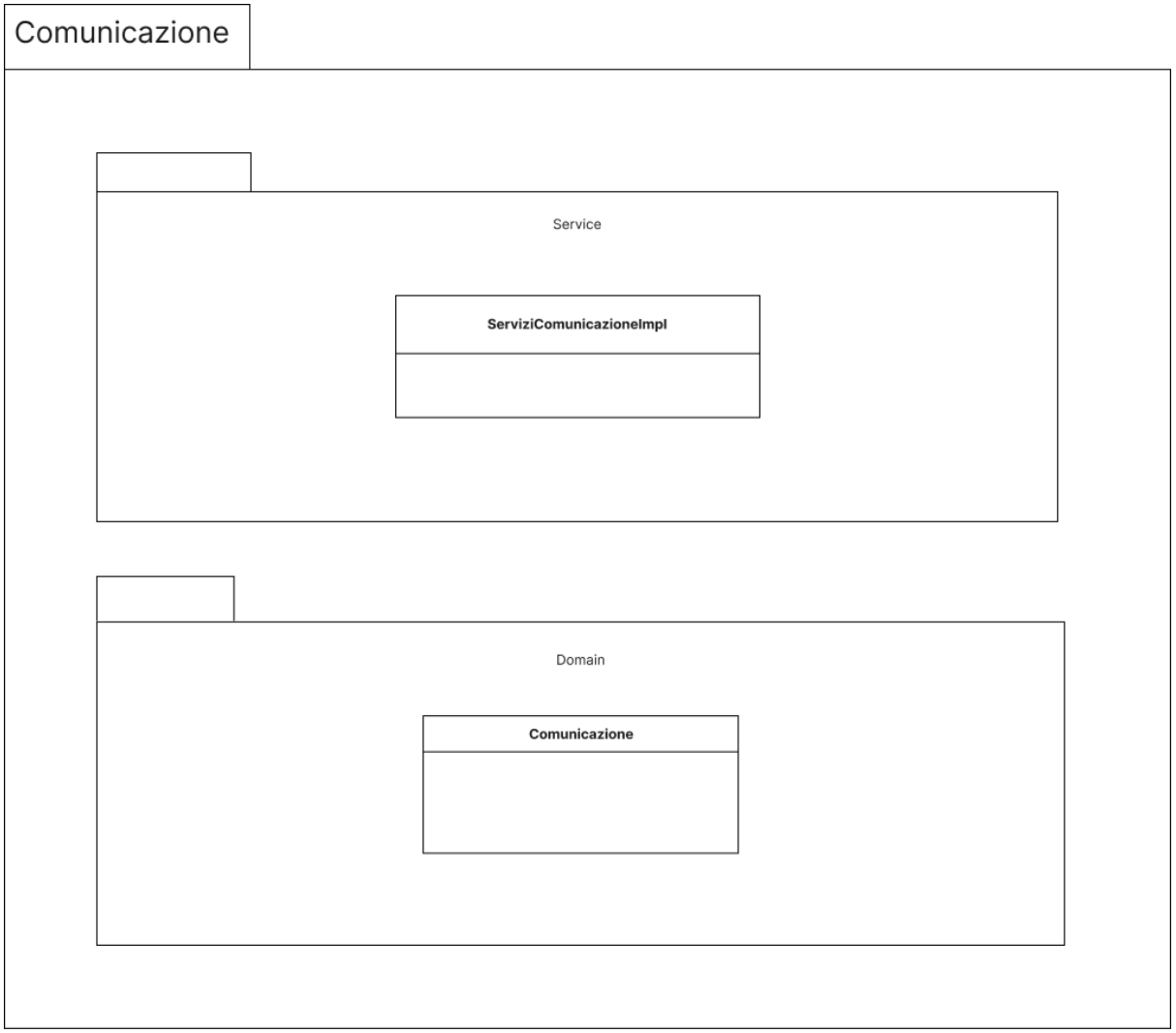
2.2 Package Piano



2.3 Package Modello



2.4 Package Comunicazione



2.5 Package Community

3. Class interfaces

3.1 Package Account

Nome classe	**ServiziUtenteImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni di registrazione, autenticazione e gestione account utente.
Metodi	+register(String nome, String cognome,String email, String password): Utente +login(String email, String password): Utente +getIdUtente(String email): int +getUtenteById(int idUtente) : Utente +recuperoPasswordMail(String email): String +modificaPassword(String email, String password): String +verificaUtenteEsistente(String mail): Boolean
Invariante di classe	/

Metodo	**+register(String nome, String cognome, String email, String password): Utente**
Descrizione	Questa funzionalità consente la registrazione di un nuovo utente.
Pre-condizione	**Context:** ServiziUtenteImpl::register(String nome, String cognome, String email, String password) **Pre:** verificaUtenteEsistente(mail)==false
Post-condizione	**Context:** ServiziUtenteImpl::register(String nome, String cognome, String email, String password) **Post:** verificaUtenteEsistente(mail)==true
Metodo	**+login(String email, String password): Utente**
Descrizione	Questa funzionalità consente l'accesso ad un utente registrato.
Pre-condizione	**Context:** ServiziUtenteImpl::login(String email, String password)

	Pre: verificaUtenteEsistente(mail)==true
Post-condizione	**Context:** ServiziUtenteImpl::login(String email, String password) **Post:**/
Metodo	**+getIdUtente(String email): int**
Descrizione	Questa funzionalità consente di ottenere l'id utente in base all'email
Pre-Condizione	**Context:** ServiziUtenteImpl::getIdUtente(String email) **Pre:** verificaIdUtenteEsistente(mail)==true
Post-Condizione	**Context:** ServiziUtenteImpl::getIdUtente(String email) **Post:**/
Metodo	**+getUtenteById(int idUtente): Utente**
Descrizione	Questa funzionalità consente di ottenere un utente in base all'id utente
Pre-condizione	**Context:** ServiziUtenteImpl::getUtenteById(int idUtente) **Pre:** verificaUtenteEsistente(mail)==true
Post-condizione	**Context:** ServiziUtenteImpl::getUtenteById(int idUtente) **Post:**/
Metodo	**+recuperoPasswordMail(String email): String**
Descrizione	Questa funzionalità consente all'utente di recuperare la password.
Pre-condizione	**Context:** ServiziUtente::recuperoPasswordMail(String email) **Pre:**verificaUtenteEsistente(mail)==true
Post-condizione	**Context:** ServiziUtente::recuperoPasswordMail(String email) **Post:**/
Metodo	**+modificaPassword(String email, String**

	password): Boolean
Descrizione	Questa funzionalità consente la modifica della password ad un utente registrato.
Pre-condizione	**Context:**ServiziUtente::modificaPassword(String email, String password) **Pre:**/
Post-condizione	**Context:**ServiziUtente::modificaPassword(String email, String password) **Post:**/
Metodo	**+verificaUtenteEsistente(String mail): Boolean**
Descrizione	Questa funzionalità verifica la presenza di un utente.
Pre-condizione	**Context:** ServiziUtente::verificaUtenteEsistente(String mail) **Pre:**/
Post-condizione	**Context:** ServiziUtente::verificaUtenteEsistente(String mail) **Post:**/

Nome classe	**ServiziAmministratoreImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni di autenticazione e gestione account all'amministratore.
Metodi	+loginIMP(String email, String password): Amministratore +visualizzaUtentiReg(): List<Utente> +modificaInformazioniUtenteIMP(String email, String nome, String cognome): Utente +cancellaUtenteIMP(String email): Utente +verificaAmministratoreEsistente(String email): Boolean
Invariante di classe	/

Metodo	**+loginIMP(String email, String password): Amministratore**
Descrizione	Questa funzionalità consente l'autenticazione di un amministratore.
Pre-condizione	**Context:** ServiziAmministratoreImpl::loginIMP(String email, String password) **Pre:** verificaAmministratoreEsistente(email)==true
Post-condizione	**Context:** ServiziAmministrazioneImpl::loginIMP(String email, String password) **Post:**/
Metodo	**+visualizzaUtentiReg(): List<Utente>**
Descrizione	Questa funzionalità consente all'amministratore di vedere tutti gli utenti registrati.
Pre-condizione	**Context:** ServiziAmministratoreImpl::visualizzaUtentiReg() **Pre:** /
Post-condizione	**Context:** ServiziAmministratoreImpl::visualizzaUtentiReg() **Post:**/
Metodo	**+modificaInformazioniUtenteIMP(String email, String nome, String cognome)**
Descrizione	Questa funzionalità consente all'amministratore la modifica delle informazioni di un utente registrato.
Pre-condizione	**Context:** ServiziAmministratoreImpl::modificaInformazioniUtenteIMP(String email, String nome, String cognome) **Pre:** verificaUtenteEsistente(email)==true
Post-condizione	**Context:** ServiziAmministratoreImpl::modificaInformazioniUtenteIMP(String email, String nome, String cognome) **Post:**/
Metodo	**+cancellaUtenteIMP(String email)**

Descrizione	Questa funzionalità consente all'amministratore di eliminare un account utente.
Pre-condizione	**Context:** ServiziAmministratoreImpl::cancellaUtenteIMP(String email) **Pre:** verificaUtenteEsistente(email)==true
Post-condizione	**Context:** ServiziAmministratoreImpl::cancellaUtenteIMP(String email) **Post:** verificaUtenteEsistente(email)==false
Metodo	**+verificaAmministratoreEsistente(String email): Boolean**
Descrizione	Questa funzionalità consente di controllare l'esistenza di un amministratore.
Pre-condizione	**Context:** ServiziAmministratoreImpl::verificaAmministratoreEsistente(String email) **Pre:**/
Post-condizione	**Context:** ServiziAmministratoreImpl::verificaAmministratoreEsistente(String email) **Post:**/

3.2 Package Piano

Nome classe	**ServiziPianoImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni di acquisto dei piani di abbonamento.
Metodi	+acquistoPiano(int idUtente, int idPiano): Piano +getUltimoAcquistoUtente(int idUtente): Acquisto +getPianoUtente(int idPiano): Piano +InserimentoPianoEnterprise(int prezzo, int limiteSalvataggi, int limiteAddestramenti): Piano +getAllPiani(): List<Piano> +getTipiPiani(): List<Piano> +acquistoPianoFree(int idUtente) +annullaPiano(int idUtente, int idPiano): Piano
Invariante di classe	/

Metodo	**+acquistoPiano(int idUtente, int idPiano): Piano**
Descrizione	Questa funzionalità consente ad un utente di acquistare un piano di abbonamento
Pre-condizione	**Context:** ServiziPianoImpl::acquistoPiano(int idUtente, int idPiano) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPianoImpl::acquistoPiano(int idUtente, int idPiano) **Post:** getAllPiani().size ==getAllPiani().size + 1
Metodo	**+getUltimoAcquistoUtente(int idUtente): Acquisto**
Descrizione	Questa funzionalità consente di ottenere l'ultimo piano acquistato
Pre-condizione	**Context:**ServiziPianoImpl::getUltimoAcquistoUtente(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:**ServiziPianoImpl::getUltimoAcquistoUtente(int idUtente)

	Post:/
Metodo	**+getPianoUtente(int idPiano): Piano**
Descrizione	Questa funzionalità consente ad un utente di vedere il proprio piano attivo.
Pre-condizione	**Context:** ServiziPianoImpl::getPianoUtente(int idPiano) **Pre:** idPiano != null
Post-condizione	**Context:** ServiziPianoImpl::getPianoUtente(int idPiano) **Post:** /
Metodo	**+InserimentoPianoEnterprise(int prezzo, int limiteSalvataggi, int limiteAddestramenti)**
Descrizione	Questa funzionalità consente di inserire un piano enterprise nei piani
Pre-condizione	**Context:** ServiziPianoImpl::InserimentoPianoEnterprise(int prezzo, int limiteSalvataggi, int limiteAddestramenti) **Pre:** prezzo != null AND limiteSalavataggi != null AND limiteAddestramenti != null
Post-condizione	**Context:** ServiziPianoImpl::inserimentoPianoEnterprise(int prezzo, int limiteSalvataggi, int limiteAddestramenti) **Post:** getAllPiani().size ==getAllPiani().size + 1
Metodo	**+getAllPiani(): List<Piano>**
Descrizione	Questa funzionalità restituisce una lista dei piani acquistati.
Pre-condizione	**Context:** ServiziPianoImpl::getAllPiani() **Pre:**/
Post-condizione	**Context:** ServiziPianoImpl::getAllPiani() **Post:**/
Metodo	**+getTipiPiani(): List<Piano>**
Descrizione	Questa funzionalità restituisce tutti i piani non

	enterprise
Pre-condizione	**Context:** ServiziPianoImpl::getTipiPiani() **Pre:**/
Post-condizione	**Context:** ServiziPianoImpl::getTipiPiani() **Post:**/
Metodo	**+acquistoPianoFree(int idUtente)**
Descrizione	Questa funzionalità consente di acquistare il piano free
Pre-condizione	**Context:** ServiziPianoImpl::acquistoPianoFree(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPianoImpl::acquistoPianoFree(int idUtente) **Post:**/
Metodo	**+annullaPiano(int idUtente, int idPiano): Piano**
Descrizione	Questa funzionalità consente di annullare un piano attivo
Pre-condizione	**Context:** ServiziPianoImpl::annullaPiano(int idUtente, int idPiano) **Pre:** getUtenteById(idUtente) != null AND getPianoUtente(int idPiano) != null
Post-condizione	**Context:** ServiziPianoImpl::annullaPiano(int idUtente, int idPiano) **Post:**/

Nome classe	**ServiziPreventivoImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni di richiesta preventivo piano enterprise.
Metodi	+ TuttiPreventivi(): List<Preventivo> + creaPreventivo(int idUtente, int limitiAddestramenti, int limitiSalvataggi): Preventivo

	+ eliminaPreventivo(int idUtente) + getPreventivo(int idUtente): Preventivo + getIdPreventivo(int idUtente): int + getLimitiAddestramenti(int idUtente): int + getLimitiSalvataggi (int idUtente): int + getPrezzo(int idUtente): double + getStato(int idUtente): String + aggiornaStato(int idPreventivo, String stato) + aggiornaPrezzo(int idPreventivo, double prezzo) + eliminaPreventivoById(int idPreventivo) + ModificaPreventivo(String stato, double prezzo, int idPreventivo): Preventivo
Invariante di classe	/

Metodo	**+TuttiPreventivi(): List<Preventivo>**
Descrizione	Restituisce una lista di tutti i preventivi presenti nel sistema.
Pre-condizione	**Context:** ServiziPreventivoImpl::TuttiPreventivi() **Pre:**/
Post-condizione	**Context:** ServiziPreventivoImpl::TuttiPreventivi() **Post:** /
Metodo	**+creaPreventivo(int idUtente, int limitiAddestramenti, int limitiSalvataggi): Preventivo**
Descrizione	Questa funzionalità crea un nuovo preventivo per un utente con i parametri specificati.
Pre-condizione	**Context:**ServiziPreventivoImpl::creaPreventivo(int idUtente, int limitiAddestramenti, int limitiSalvataggi) **Pre:** getUtenteById(idUtente) != null AND (limitiAddestramenti>4 AND limitiAddestramenti<=20) AND (limitiSalvataggi>10 AND limitiSalvataggi<=50)

Post-condizione	**Context:**ServiziPreventivoImpl::creaPreventivo(int idUtente, int limitiAddestramenti, int limitiSalvataggi) **Post:** getPreventivo(idUtente)!=null AND TuttiPreventivi().size== TuttiPreventivi().size+1
Metodo	**+eliminaPreventivo(int idUtente)**
Descrizione	Questa funzionalità elimina il preventivo associato all'utente specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::eliminaPreventivo(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPianoImpl::getPianoUtente(int idPiano) **Post:** TuttiPreventivi().size== TuttiPreventivi().size-1
Metodo	**+getPreventivo(int idUtente): Preventivo**
Descrizione	Questa funzionalità restituisce il preventivo associato all'utente specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::getPreventivo(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPreventivoImpl::getPreventivo(int idUtente) **Post:** getPreventivo(idUtente)!=null
Metodo	**+getIdPreventivo(int idUtente): int**
Descrizione	Questa funzionalità restituisce l'ID del preventivo associato all'utente specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::getIdPreventivo(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPreventivoImpl::getIdPreventivo(int idUtente) **Post:** getIdPreventivo(int idUtente)!=null

Metodo	**+getLimitiAddestramenti(int idUtente): int**
Descrizione	Questa funzionalità restituisce il numero limite di addestramenti del preventivo associato all'utente specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::getLimitiAddestramenti(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPreventivoImpl::getLimitiAddestramenti(int idUtente) **Post:** getLimitiAddestramenti()!=null
Metodo	**+getLimitiSalvataggi (int idUtente): int**
Descrizione	Questa funzionalità restituisce il numero limite di salvataggi del preventivo associato all'utente specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::getLimitiSalvataggi (int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPreventivoImpl::getLimitiSalvataggi (int idUtente) **Post:** getLimitiSalvataggi()!=null
Metodo	**+getPrezzo(int idUtente): double**
Descrizione	Questa funzionalità restituisce il prezzo del preventivo associato all'utente specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::getPrezzo(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPreventivoImpl::getPrezzo(int idUtente) **Post:** getPrezzo()!=null
Metodo	**+getStato(int idUtente): String**

Descrizione	Questa funzionalità restituisce lo stato del preventivo associato all'utente specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::getStato(int idUtente) **Pre:** getUtenteById(idUtente) != null
Post-condizione	**Context:** ServiziPreventivoImpl::getStato(int idUtente) **Post:** getStato()!=null
Metodo	**+aggiornaStato(int idPreventivo, String stato)**
Descrizione	Questa funzionalità aggiorna lo stato del preventivo specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::aggiornaStato(int idPreventivo, String stato) **Pre:** getIdPreventivo()!=null AND getStato()!=null
Post-condizione	**Context:** ServiziPreventivoImpl::aggiornaStato(int idPreventivo, String stato) **Post:** /
Metodo	**+aggiornaPrezzo(int idPreventivo, double prezzo)**
Descrizione	Questa funzionalità aggiorna il prezzo del preventivo specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::aggiornaPrezzo(int idPreventivo, double prezzo) **Pre:** getIdPreventivo()!=null AND getPrezzo()>0
Post-condizione	**Context:** ServiziPreventivoImpl::aggiornaPrezzo(int idPreventivo, double prezzo) **Post:**/
Metodo	**+eliminaPreventivoById(int idPreventivo)**
Descrizione	Questa funzionalità elimina il preventivo con l'Id specificato
Pre-condizione	**Context:** ServiziPreventivoImpl::eliminaPreventivoById(int idPreventivo)

	Pre: getIdPreventivo()!=null
Post-condizione	**Context:** ServiziPreventivoImpl::eliminaPreventivoById(int idPreventivo) **Post:** TuttiPreventivi().size==TuttiPreventivi().size-1
Metodo	**+ModificaPreventivo(String stato, double prezzo, int idPreventivo): Preventivo**
Descrizione	Questa funzionalità modifica lo stato e/o il prezzo del preventivo specificato.
Pre-condizione	**Context:** ServiziPreventivoImpl::ModificaPreventivo(String stato, double prezzo, int idPreventivo) **Pre:** getStato()!=null AND getPrezzo()>0 AND getIdPreventivo()!=null
Post-condizione	**Context:** ServiziPreventivoImpl::ModificaPreventivo(String stato, double prezzo, int idPreventivo) **Post:** getPreventivo()!=null

3.3 Package Modello

Nome classe	**ServiziModelloImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni necessarie ad ottenere un modello addestrato.
Metodi	+listaModelli(): List<Modello> +salvaModelloImpl(int idUtente, String gruppoPrivilegato, int recall, int precision, int accuracy, int sustainability, int disparateImpact, String tipoModello)
Invariante di classe	/

Metodo	**+listaModelli(): List<Modello>**
Descrizione	Questa funzionalità restituisce una lista dei modelli salvati.
Pre-condizione	**Context:** ServiziModelloImpl:::listaModelli() **Pre:**/
Post-condizione	**Context:** ServiziModelloImpl:::listaModelli() **Post:**/
Metodo	**+salvaModelloImpl(int idUtente, String gruppoPrivilegato, int recall, int precision, int accuracy, int sustainability, int disparateImpact, String tipoModello)**
Descrizione	Questa funzionalità salva un modello addestrato nel database.
Pre-condizione	**Context:** ServiziModelloImpl:::salvaModelloImpl() **Pre:**/
Post-condizione	**Context:** ServiziModelloImpl:::salvaModelloImpl() **Post:**/

Nome classe	**ServiziMemoriaImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni necessarie a scaricare un modello addestrato.
Metodi	+visualizzazioneStorico(Utente utente): List<Modello> +salvataggioMemoria(Utente utente, Modello modello): Boolean +visualizzazioneModelliCloud(Utente utente): List<Modello> +cancellazioneDaMemoria(Utente utente, Modello modello): Boolean +downloadModelloCloud(Modello modello): File +downloadModelloAddestrato(Modello modello): File +listaModelliCloud(): List<Modello> +listaModelliStorico(): List<Modello>
Invariante di classe	/

Metodo	**+visualizzazioneStorico(Utente utente): List<Modello>**
Descrizione	Questa funzionalità permette ad un utente di visualizzare l'elenco di metriche e parametri generati.
Pre-condizione	**Context:** ServiziModelloImpl::visualizzazioneStorico(Utente utente) **Pre:**/
Post-condizione	**Context:** ServiziModelloImpl::visualizzazioneStorico(Utente utente) **Post:**/
Metodo	**+salvataggioMemoria(Utente utente, Modello modello): Boolean**
Descrizione	Questa funzionalità permette ad un utente di salvare

	il modello generato in memoria
Pre-condizione	**Context:** ServiziModelloImpl::salvataggioMemoria(Utente utente, Modello modello) **Pre:**
Post-condizione	**Context:** ServiziModelloImpl::salvataggioMemoria(Utente utente, Modello modello) **Post:**
Metodo	**+visualizzazioneModelliCloud(Utente utente): List<Modello>**
Descrizione	Questa funzionalità permette ad un utente di visualizzare l'elenco di modelli salvati in memoria.
Pre-condizione	**Context:** ServiziModelloImpl::visualizzazioneMemoria(Utente utente) **Pre:** /
Post-condizione	**Context:** ServiziModelloImpl::visualizzazioneMemoria(Utente utente) **Post:** /
Metodo	**+cancellazioneDaMemoria(Utente utente, Modello modello): Boolean**
Descrizione	Questa funzionalità permette ad un utente di cancellare un modello dalla memoria così da liberare spazio in memoria.
Pre-condizione	**Context:** ServiziModelloImpl::cancellazioneDaMemoria(Utente utente, Modello modello) **Pre:**
Post-condizione	**Context:** ServiziModelloImpl::cancellazioneDaMemoria(Utente utente, Modello modello) **Post:**

Metodo	**+downloadModelloAddestrato(Modello modello): File**
Descrizione	Questa funzionalità permette all'utente di scaricare un modello appena addestrato.
Pre-condizione	**Context:** ServiziModelloImpl::downloadModelloAddestrato(Modello modello) **Pre:/**
Post-condizione	**Context:**ServiziModelloImpl::downloadModelloAddestrato(Modello modello) **Post:/**
Metodo	**+downloadModelloCloud(Modello modello): File**
Descrizione	Questa funzionalità permette di scaricare un modello salvato in cloud.
Pre-condizione	**Context:**ServiziModelloImpl::downloadModelloCloud(Modello modello) **Pre:**
Post-condizione	**Context:**ServiziModelloImpl::downloadModelloCloud(Modello modello) **Post:/**
Metodo	**+listaModelliCloud(): List<Modello>**
Descrizione	Questa funzionalità permette di visualizzare la lista dei modelli salvati in cloud.
Pre-condizione	**Context:**ServiziModelloImpl::listaModelliCloud() **Pre:/**
Post-condizione	**Context:**ServiziModelloImpl::listaModelliCloud() **Post:/**
Metodo	**+listaModelliStorico(): List<Modello>**
Descrizione	Questa funzionalità permette di visualizzare la lista dei modelli salvati nello storico.
Pre-condizione	**Context:**ServiziModelloImpl::listaModelliStorico() **Pre:/**

Post-condizione	**Context:**ServiziModelloImpl::listaModelliStorico() **Post:/**

3.4 Package Comunicazione

Nome classe	**ServiziComunicazioneImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni di invio comunicazione, richiesta assistenza e ricezione di notifica.
Metodi	+invioComunicazione(List<Utente> utenti, String messaggio): Boolean +richiestaAssistenza(Utente utente, String messaggio): Boolean +notificaComunicazione(Utente utente, List<Messaggio>) +listaComunicazioni(): List<Comunicazione>
Invariante di classe	/

Metodo	**+invioComunicazione(List<Utente> utenti, String messaggio): Boolean**
Descrizione	Questa funzionalità permette ad un amministratore di inviare una comunicazione.
Pre-condizione	**Context:** ServiziComunicazioneImpl::invioComunicazione(List <Utente> utenti, String messaggio) **Pre:**
Post-condizione	**Context:** ServiziComunicazioneImpl::invioComunicazione(List <Utente> utenti, String messaggio) **Post:**
Metodo	**+richiestaAssistenza(Utente utente, String messaggio): Boolean**
Descrizione	Questa funzionalità permette ad un utente di

	richiedere assistenza.
Pre-condizione	**Context:** ServiziComunicazioneImpl::richiestaAssistenza(Utente utente, String messaggio) **Pre:**
Post-condizione	**Context:** ServiziComunicazioneImpl::richiestaAssistenza(Utente utente, String messaggio) **Post:**
Metodo	**+notificaComunicazione(Utente utente, List<Messaggio> Messaggi)**
Descrizione	Questa funzionalità permette ad un utente di ricevere la notifica di una comunicazione.
Pre-condizione	**Context:** ServiziComunicazioneImpl::notificaComunicazione(Utente utente, List<Messaggio> Messaggi) **Pre:** /
Post-condizione	**Context:** ServiziComunicazioneImpl::notificaComunicazione(Utente utente, List<Messaggio> Messaggi) **Post:** /
Metodo	**+listaComunicazioni(): List<Comunicazione>**
Descrizione	Questa funzionalità permette di visualizzare tutte le comunicazioni.
Pre-condizione	**Context:** ServiziComunicazioneImpl::listaComunicazioni() **Pre:** /
Post-condizione	**Context:** ServiziComunicazioneImpl::listaComunicazioni() **Post:** /

3.5 Package Community

Nome classe	**ServiziCommunityImpl**
Descrizione	Questa classe fornisce i metodi per effettuare le operazioni necessarie a condividere i modelli nella community.
Metodi	+condividiModelloMemoria(Utente utente, Modello modello, String titolo, String descrizione, String hashtag, Boolean download): Condivisione +condividiMetricheParametri(Utente utente, Modello modello, String titolo, String descrizione, String hashtag): Condivisione +modificaCondivisione(Condivisione condivisione, Utente utente): Boolean +listaCondivisioni(): List<Condivisione>
Invariante di classe	/

Metodo	**+condividiModelloMemoria(Utente utente, Modello modello, String titolo, String descrizione, String hashtag, Boolean download): Condivisione**
Descrizione	Questa funzionalità permette ad un utente di condividere un modello addestrato salvato in memoria.
Pre-condizione	**Context:** ServiziCommunityImpl::condividiModelloMemoria(Utente utente, Modello modello, String titolo, String descrizione, String hashtag, Boolean download) **Pre:**
Post-condizione	**Context:** ServiziCommunityImpl::condividiModelloMemoria(Utente utente, Modello modello, String titolo, String descrizione, String hashtag, Boolean download) **Post:**
Metodo	**+condividiMetricheParametri(Utente utente, Modello modello, String titolo, String descrizione, String hashtag): Condivisione**
Descrizione	Questa funzionalità permette ad un utente di

	condividere metriche e parametri generati dall'addestramento.
Pre-condizione	**Context:** ServiziCommunityImpl::condividiMetricheParametri(Utente utente, Modello modello, String titolo, String descrizione, String hashtag) **Pre:**
Post-condizione	**Context:** ServiziCommunityImpl::condividiMetricheParametri(Utente utente, Modello modello, String titolo, String descrizione, String hashtag) **Post:**
Metodo	**+modificaCondivisione(Condivisione condivisione, Utente utente): Boolean**
Descrizione	Questa funzionalità permette ad un utente di modificare una condivisione effettuata.
Pre-condizione	**Context:** ServiziCommunityImpl::modificaCondivisione(Condivisione condivisione, Utente utente): Boolean **Pre:**
Post-condizione	**Context:** ServiziCommunityImpl::modificaCondivisione(Condivisione condivisione, Utente utente): Boolean **Post:** /
Metodo	**+listaCondivisioni(): List<Condivisione>**
Descrizione	Questa funzionalità permette di visualizzare tutte le condivisioni.
Pre-condizione	**Context:** ServiziCommunityImpl::listaCondivisioni(): List<Condivisione> **Pre:** /
Post-condizione	**Context:** ServiziCommunityImpl::listaCondivisioni(): List<Condivisione> **Post:** /

4. Class Diagram Ristrutturato

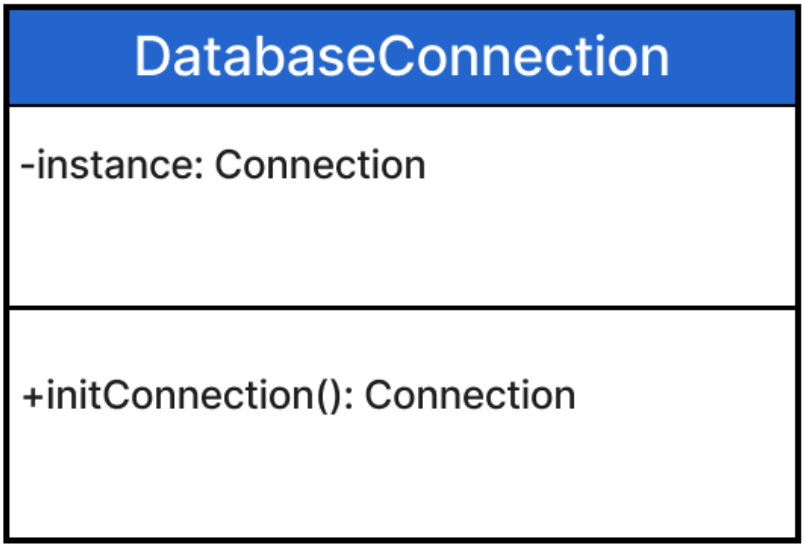
Nella cartella è allegato il file .pdf inerente al Class Diagram.

5. Design Patterns

I Design Pattern a cui si è fatto riferimento sono i seguenti:

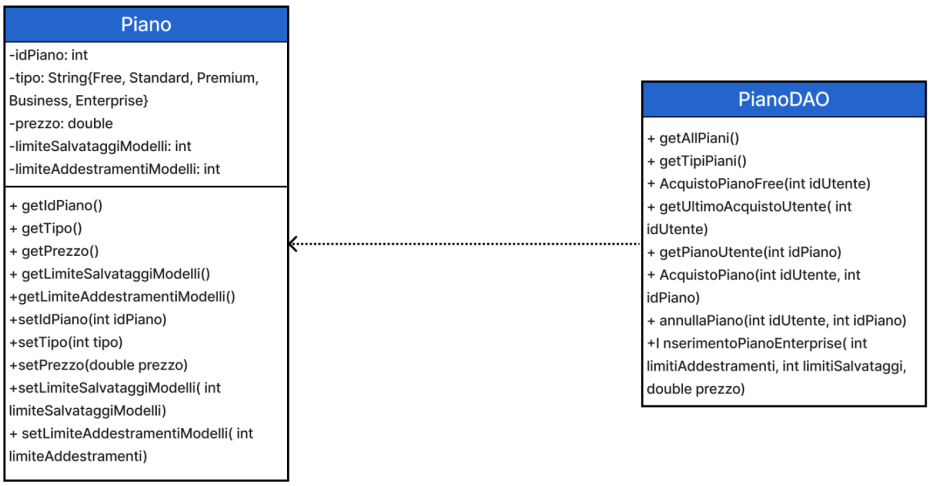
Singleton

Il Singleton Pattern è stato implementato nel codice del database. La classe assicura che esista un'unica istanza di connessione al database durante l'esecuzione dell'applicazione, evitando la creazione di duplicati. Questo contribuisce a ottimizzare l'uso delle risorse e a garantire coerenza nella gestione delle connessioni al database.



DAO

Il Pattern DAO (Data Access Object) è stato implementato nel contesto del codice fornendo un'astrazione per la gestione delle interazioni con i dati associati alle entità presenti nel database. Tale implementazione è finalizzata a isolare la logica di accesso ai dati dalla logica di business dell'applicazione. Questa separazione contribuisce significativamente al miglioramento della modularità e della manutenibilità del sistema, consentendo una chiara distinzione tra le operazioni di accesso ai dati e le altre funzionalità intrinseche dell'applicazione.



6. Glossario

Termine	Definizione
Object design	L'Object Design è il processo di definizione dettagliata di classi, oggetti e delle loro relazioni in un sistema software orientato agli oggetti.
Trade Off	Il trade-off rappresenta la scelta tra vantaggi e svantaggi o compromessi che si verificano quando si opta per una soluzione rispetto a un'altra.

Interfaccia	Contratto che definisce i metodi che una classe deve implementare, facilitando l'interazione tra componenti del software.
Package	Struttura organizzativa che raggruppa e organizza classi e risorse correlate, facilitando la gestione e la modularità del codice.
ESLint	Strumento di linting per JavaScript che aiuta gli sviluppatori a individuare e correggere errori nel codice, mantenendo uno stile coerente e applicando le best practices del linguaggio.
Vite	Framework di sviluppo per applicazioni web che offre un'esperienza di sviluppo veloce e reattiva, ottimizzata.
Context	Si riferisce alle circostanze o situazioni in cui una determinata operazione o funzionalità viene eseguita.
Pre	è una condizione o un insieme di condizioni che devono essere verificate prima dell'esecuzione di un'azione o di una funzione. Assicura che l'ambiente sia adeguato per eseguire l'operazione senza errori.
Post	è una condizione o un insieme di condizioni che devono essere verificate dopo l'esecuzione di un'azione o di una funzione. Le postcondizioni descrivono lo stato atteso o le proprietà del sistema dopo che l'operazione è stata completata con successo.
Design Pattern	Il design pattern è una soluzione ricorrente a un problema comune nell'ambito dello sviluppo del software. Fornisce uno schema progettuale generale che può essere adattato e riutilizzato per risolvere diverse istanze di un problema

	specifico. I design pattern aiutano a migliorare la modularità, la flessibilità e la manutenibilità del codice.
Singleton	Il Singleton è un design pattern creazionale che garantisce che una classe abbia una sola istanza e offre un punto di accesso globale a essa. Questo è utile per controllare attentamente l'istanziazione di una classe e prevenire la creazione di più istanze. Il pattern assicura che l'istanza sia unica e fornisce un modo centralizzato per accedervi in tutto il programma.
DAO	Design pattern che separa l'accesso ai dati dalla logica di business, fornendo un'interfaccia astratta per interagire con il sistema di persistenza, garantendo flessibilità e facilitando eventuali cambiamenti nel sistema di memorizzazione senza impattare il codice principale dell'applicazione.