Università degli Studi di Salerno
Corso di Ingegneria del Software

HairBeautyNow
ODD: Object Design Document
Versione 1.0

Data: 27/11/2024

Progetto: HairBeautyNow	Versione: 1.0
Documento: Object Design Document	Data: 16/12/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Aniello Cirillo	0512117775
Christian Pio Lanziero	0512117931

Scritto da:	

Revision History

Data	Versione	Descrizione	Autore
27/11/2024	0.1	Creazione del documento	Tutto il team
06/12/2024	0.2	Scopo del sistema, Linee guida e definizione dei packages	Tutto il team
12/12/2024	0.3	Packaging del sistema, dei layer e dei singoli sottoinsiemi	Tutto il team
16/12/2024	1.0	Package data, interfaccia delle classi e prima release del documento	Tutto il team

Indice

1. Introduzione

Il presente **Object Design Document** ha l'obiettivo di approfondire in modo dettagliato gli aspetti relativi all'implementazione del sistema **HairBeautyNow**, integrando e ampliando quanto definito nei documenti precedenti, che si sono focalizzati principalmente sull'architettura e sulla progettazione generale del sistema. Questo documento offre una descrizione tecnica e completa delle decisioni progettuali maturate durante le fasi di analisi e design, evidenziando i principali compromessi valutati, le linee guida per la documentazione delle interfacce e l'adozione dei **Design Pattern** più adeguati.

Nel corso del documento verranno inoltre definiti i **packages**, le interfacce delle classi e i diagrammi UML che rappresentano le principali decisioni implementative del sistema. Sarà dedicata particolare attenzione alla descrizione dettagliata delle operazioni, dei parametri e delle firme delle varie interfacce e classi, garantendo

coerenza con i sottosistemi definiti nel **System Design Document (SDD_HairBeautyNow)** e con i requisiti funzionali e non funzionali indicati nel **Requirements Analysis Document (RAD_HairBeautyNow)**.

Lo scopo principale dell'**Object Design Document** è fornire un modello chiaro e dettagliato per implementare tutte le funzionalità identificate, favorendo una transizione senza ostacoli verso la fase di sviluppo e assicurando che il sistema mantenga elevati standard di manutenibilità ed espandibilità.

1.1 Scopo del sistema

Lo scopo del sistema è facilitare la gestione centralizzata e la prenotazione di appuntamenti per una catena di barbieri e saloni di bellezza, offrendo una piattaforma digitale intuitiva e funzionale. In un contesto in cui la cura personale è sempre più importante, il sistema consente agli utenti di prenotare comodamente i loro appuntamenti, visualizzare la disponibilità in tempo reale, e scegliere il servizio o il professionista desiderato presso una qualsiasi delle sedi della catena. I gestori della sede possono amministrare facilmente le prenotazioni a livello di singola sede, monitorare l'occupazione delle postazioni, gestire i servizi offerti, e migliorare la comunicazione con i clienti, ottimizzando così l'efficienza operativa e l'esperienza complessiva del brand.

Il sistema è una web-app pensata sia per i clienti che vogliono prenotare servizi presso una delle sedi della catena di barbieri e saloni di bellezza, sia per i gestori delle singole sedi e per la gestione centrale della catena. Gli utenti possono creare profili personali, visualizzare la disponibilità dei professionisti in tempo reale, e prenotare appuntamenti per diversi servizi come taglio, colore, manicure, e altri trattamenti estetici presso la sede preferita.

I gestori delle singole sedi possono utilizzare la piattaforma per amministrare le prenotazioni, gestire le postazioni, offrire promozioni, e monitorare l'andamento delle attività giornaliere nella propria sede. Inoltre, la gestione centrale del franchise può monitorare e gestire le operazioni di tutte le sedi in modo unificato, coordinando risorse e strategie promozionali per migliorare la performance complessiva del brand. Il sistema permette di personalizzare i servizi offerti, consentendo agli utenti di scegliere specifici professionisti e servizi, favorendo così un'esperienza ottimizzata e uniforme per i clienti di tutta la catena.

1.2 Design goals

Gli obiettivi di progettazione del sistema HairBeautyNow sono strettamente correlati ai requisiti non funzionali specificati nel *Requirements Analysis Document* (RAD). In particolare, si pone un'attenzione particolare su aspetti come usabilità, sicurezza, manutenibilità, robustezza e performance. Di seguito vengono descritti i principali obiettivi progettuali e i compromessi (trade-off) considerati.

1.2.1 Usabilità

- RNFU1 Navigazione Intuitiva: La struttura dell'interfaccia sarà progettata per permettere un'esperienza fluida e priva di ostacoli per l'utente, indipendentemente dal dispositivo utilizzato.
- RNFU2 Navigazione Intuitiva per Utenti Comuni: L'interfaccia sarà chiara e lineare, rendendo semplice l'accesso alle funzionalità principali anche per utenti con basse competenze tecniche.
- RNFU3 Accesso a Strumenti Avanzati per Utenti Gestori: Gli utenti gestori disporranno di un'interfaccia altrettanto intuitiva, ma con accesso a strumenti avanzati e funzionalità di gestione per una configurazione e monitoraggio del sistema efficiente.

1.2.2 Sicurezza

- RNFS1 Crittografia Dati Sensibili: Tutti i dati sensibili, incluse password e informazioni personali, saranno memorizzati in modo sicuro utilizzando metodi di crittografia avanzati all'interno del database.
- RNFS2 Protezione dall'Accesso Non Autorizzato: Saranno implementate misure di sicurezza per proteggere i dati da accessi non autorizzati, garantendo la riservatezza e l'integrità delle informazioni.

1.2.3 Manutenibilità

- RNFM1 Architettura Modulare: Il sistema sarà progettato con un'architettura modulare per facilitare la manutenzione e supportare facilmente l'espansione e l'integrazione di nuove funzionalità.
- RNFM2 Facilità di Aggiornamento: Sarà garantita la possibilità di effettuare aggiornamenti e modifiche senza interrompere il servizio, minimizzando tempi di inattività.

1.2.4 Robustezza

- RNFR1 Integrità dei Dati: Il sistema assicurerà l'integrità dei dati durante tutte le operazioni, prevenendo la perdita o la corruzione delle informazioni.

1.2.5 Performance

- RNFP1 Tempo di Risposta Ottimale: Il sistema dovrà rispondere rapidamente alle richieste degli utenti, mantenendo un tempo di risposta minimo anche in caso di richieste complesse.

1.2.6 Trade-off considerati

1. **Usabilità per Utenti Comuni e Accesso a Strumenti Avanzati**

Creare un'interfaccia intuitiva per gli utenti comuni potrebbe entrare in conflitto con la necessità di fornire strumenti avanzati e dettagliati ai gestori. Si potrebbe optare per un'interfaccia modulare, che presenti un layout semplificato per gli utenti comuni e permetta l'accesso a strumenti avanzati tramite sezioni o pannelli separati per i gestori. Questo aumenta la complessità dello sviluppo ma migliora l'esperienza per entrambi i tipi di utenti.

RNF considerati:

- **RNFU2:** L'interfaccia sarà chiara e lineare, rendendo semplice l'accesso alle funzionalità principali anche per utenti con basse competenze tecniche
- **RNFU3**: Gli utenti gestori disporranno di un'interfaccia altrettanto intuitiva, ma con accesso a strumenti avanzati e funzionalità di gestione per una configurazione e monitoraggio del sistema efficiente.

2. **Sicurezza e Performance**

Ottimizzare il tempo di risposta del sistema potrebbe richiedere di ridurre la frequenza di operazioni di crittografia/decrittografia. Si potrebbe bilanciare il livello di sicurezza in base alla sensibilità dei dati: per operazioni meno critiche si potrebbero usare protocolli meno intensivi, mantenendo quelli più robusti per dati sensibili.
RNF considerati:

- **RNFP1**: Il sistema dovrà rispondere rapidamente alle richieste degli utenti, mantenendo un tempo di risposta minimo anche in caso di richieste complesse.
- **RNFS1**: Tutti i dati sensibili, incluse password e informazioni personali, saranno memorizzati in modo sicuro utilizzando metodi di crittografia avanzati all'interno del database.

-

3. Modularità e Robustezza

Progettare un sistema modulare facilita l'espansione futura, ma potrebbe introdurre vulnerabilità dovute a interfacce di comunicazione più complesse tra i moduli. Si potrebbe garantire robustezza implementando test rigorosi per le interazioni tra moduli, accettando un aumento dei tempi e dei costi di sviluppo.
RNF considerati:

- **RNFM1**: Il sistema sarà progettato con un'architettura modulare per facilitare la manutenzione e supportare facilmente l'espansione e l'integrazione di nuove funzionalità.
- **RNFR1:** Il sistema assicurerà l'integrità dei dati durante tutte le operazioni, prevenendo la perdita o la corruzione delle informazioni.

4. Efficienza e Portabilità

Per garantire prestazioni elevate, il sistema è stato progettato per funzionare al meglio su dispositivi e browser di ultima generazione, riducendo il supporto per versioni meno recenti. Questo approccio assicura tempi di risposta rapidi e una gestione semplificata, ma potrebbe limitare l'accesso al sistema HairBeautyNow per gli utenti che utilizzano dispositivi più datati o software non aggiornati.

RNF considerati:

- **RNFP1:** Il sistema dovrà rispondere rapidamente alle richieste degli utenti, mantenendo un tempo di risposta minimo anche in caso di richieste complesse.

5. Robustezza e Tempi di Aggiornamento

Implementare aggiornamenti senza interrompere il servizio potrebbe aumentare il rischio di errori temporanei o perdita di dati in determinate situazioni. Si potrebbe adottare un modello di aggiornamento a finestre programmate, garantendo una robustezza maggiore accettando brevi periodi di inattività pianificata.

RNF considerati:

- **RNFR1**: Il sistema assicurerà l'integrità dei dati durante tutte le operazioni.
- **RNFM2**: Sarà garantita la possibilità di effettuare aggiornamenti e modifiche senza interrompere il servizio.

1.3 Linee guida per la documentazione dell'interfaccia

Nomi dei Package

I nomi dei package nel sistema **HairBeautyNow** devono essere scritti in **minuscolo** e non devono contenere spazi o caratteri speciali. In caso di nomi composti da più parole, è necessario utilizzare il formato **snake_case**.
Esempio: gestione_servizi, gestione_sede

Nomi delle Classi

I nomi delle classi devono seguire il formato **PascalCase**, iniziando con una lettera maiuscola. Devono essere descrittivi e riflettere chiaramente l'entità o la funzionalità implementata.
Esempio: Utente, Prenotazione, Promozione

Classi DAO

Le classi DAO devono terminare con il suffisso **DAO** per indicare il loro ruolo di accesso ai dati.
Esempio: UtenteDAO, PromozioneDAO

Classi Bean

Le classi Bean devono terminare con il suffisso **Bean**, utilizzato per rappresentare entità del dominio.
Esempio: UtenteBean, ServizioBean

Classi di Gestione

Le classi di gestione devono indicare chiaramente la funzionalità principale svolta.
Esempio: GestionePrenotazioniControl, GestioneSediControl

Nomi delle Servlet

I nomi delle Servlet non utilizzate per la logica di business devono seguire il formato **PascalCase** e terminare con il suffisso **Servlet**.
Esempio: PromozioniServlet

Nomi dei Metodi

I metodi devono avere nomi descrittivi che riflettano chiaramente l'operazione svolta. I nomi devono iniziare con una lettera minuscola e seguire il formato **camelCase**.
Esempio: aggiungiPrenotazione(), calcolaTotale()

Nomi delle Variabili

I nomi delle variabili devono essere descrittivi e seguire il formato **camelCase**, iniziando con una lettera minuscola. Per i nomi composti, ogni parola interna deve iniziare con una lettera maiuscola.
Esempio: numeroClienti, nomeServizio
È possibile utilizzare variabili temporanee abbreviate (ad esempio, **i**), ma solo in contesti limitati come cicli for o altri ambiti ristretti.

Nomi delle Costanti

I nomi delle costanti devono essere scritti in **maiuscolo**, con le parole separate da underscore (_).
Esempio: NUMERO_CLIENTI=50

Nomi delle Eccezioni

Le eccezioni devono avere nomi descrittivi che indichino chiaramente il problema segnalato e terminare con il suffisso **Exception**.
Esempio: ServizioNonDisponibileException

Nomi delle Interfacce

Le interfacce devono seguire il formato **PascalCase**, come le classi, e terminare con il suffisso **Interface**.
Esempio: PrenotazioneInterface

Nomi delle JSP

I file JSP devono seguire il formato **camelCase** e riflettere chiaramente il contenuto o la funzionalità della pagina.
Esempio: prenotazioniView.jsp

Nomi delle Classi che Implementano Strategy

Le classi che implementano il **Pattern Strategy** devono seguire il formato **PascalCase** e terminare con il suffisso **Validator**.
Esempio: EmailValidator

Organizzazione delle Componenti:

● **Classi e Package:** Tutte le classi che realizzano un sottosistema devono essere racchiuse nello stesso package Java, organizzate in modo logico in base alla loro funzione in modo da mantenere un sistema coeso.

● **Risorse Statiche:** Fogli di stile, script e immagini devono essere organizzati nella directory resources, con sottocartelle per ciascun tipo di file (es. resources/css, resources/js, resources/images).

1.4 Design Pattern

Per implementare le funzionalità del sistema **HairBeautyNow**, sono stati adottati due design pattern principali: **Singleton Pattern** e **Strategy Pattern**. Di seguito vengono illustrate le motivazioni che hanno portato all'utilizzo di tali pattern nel contesto dell'applicazione.

Singleton Pattern

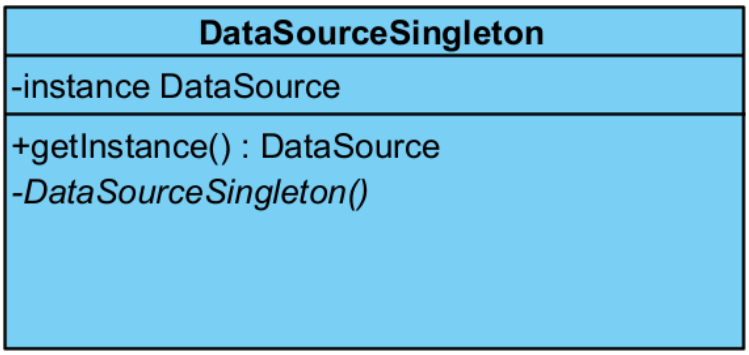
L'applicazione richiede un controllo centralizzato per l'accesso alla risorsa di connessione al database, essendo quest'ultima una risorsa condivisa. L'adozione del **Singleton Pattern** si rivela quindi utile per limitare e gestire l'accesso concorrente alla risorsa **DataSource**.

L'implementazione del **Singleton Pattern** garantisce che esista una sola istanza di **DataSource** durante l'intero ciclo di vita dell'applicazione. In questo modo:

- Si previene la creazione simultanea di connessioni ridondanti.
- Si ottimizza il ciclo di vita della connessione condivisa.
- Si semplifica il coordinamento e la gestione dell'accesso ai dati.

Nel contesto del sistema **HairBeautyNow**, la classe **DatabaseSourceSingleton** rappresenta una specializzazione del pattern Singleton. Questa classe include un attributo statico, di tipo **DataSource**, denominato `instance`.

L'accesso al **DataSource** avviene tramite il metodo **getConnection()**, che restituisce l'istanza unica della connessione. La gestione di tale connessione è integrata nel contesto dell'applicazione web tramite la classe **MainContext**, che centralizza l'uso della risorsa condivisa.



Strategy Pattern

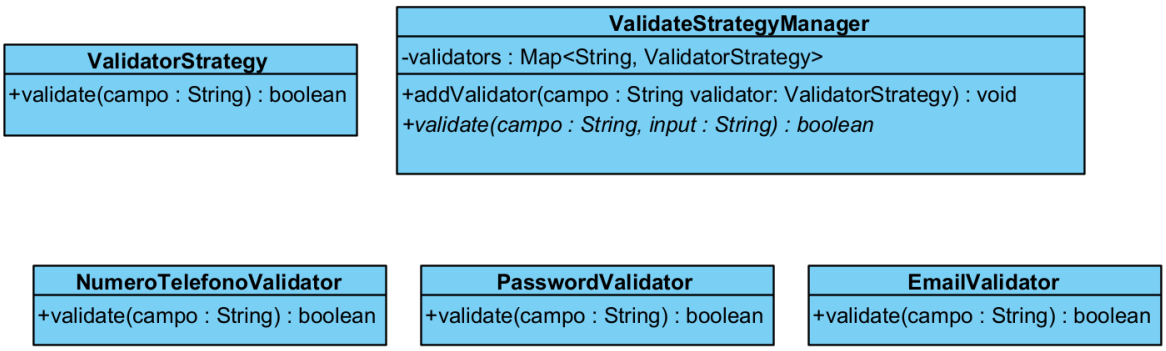
Nel sistema **HairBeautyNow**, i requisiti di validazione per i campi di input, come email, password o altri dati specifici, possono variare frequentemente, richiedendo modifiche o estensioni regolari. Per affrontare questa esigenza, è stato adottato il **Strategy Pattern**, che permette di definire un'interfaccia comune per implementare differenti strategie di validazione, facilmente intercambiabili.

Grazie a questo approccio, è possibile:

- Gestire con maggiore flessibilità e modularità l'aggiunta di nuove logiche di validazione senza alterare l'architettura esistente.
- Riutilizzare logiche di validazione comuni tra diverse istanze di classi che ne necessitano.
- Semplificare la manutenibilità e il testing del sistema.

Per il sistema **HairBeautyNow**, sono stati definiti i seguenti componenti:

- **ValidatorStrategy**: Un'interfaccia che espone il metodo per la validazione dei campi.
- **ValidationManager**: Una classe che gestisce i diversi Validator che implementano l'interfaccia e ne coordina l'utilizzo.
- **Classi specifiche per i Validator**: Implementazioni concrete dell'interfaccia, progettate per gestire tipi di dati differenti (es. email, password, numeri di telefono).



1.5 Definizione, acronimi e abbreviazioni

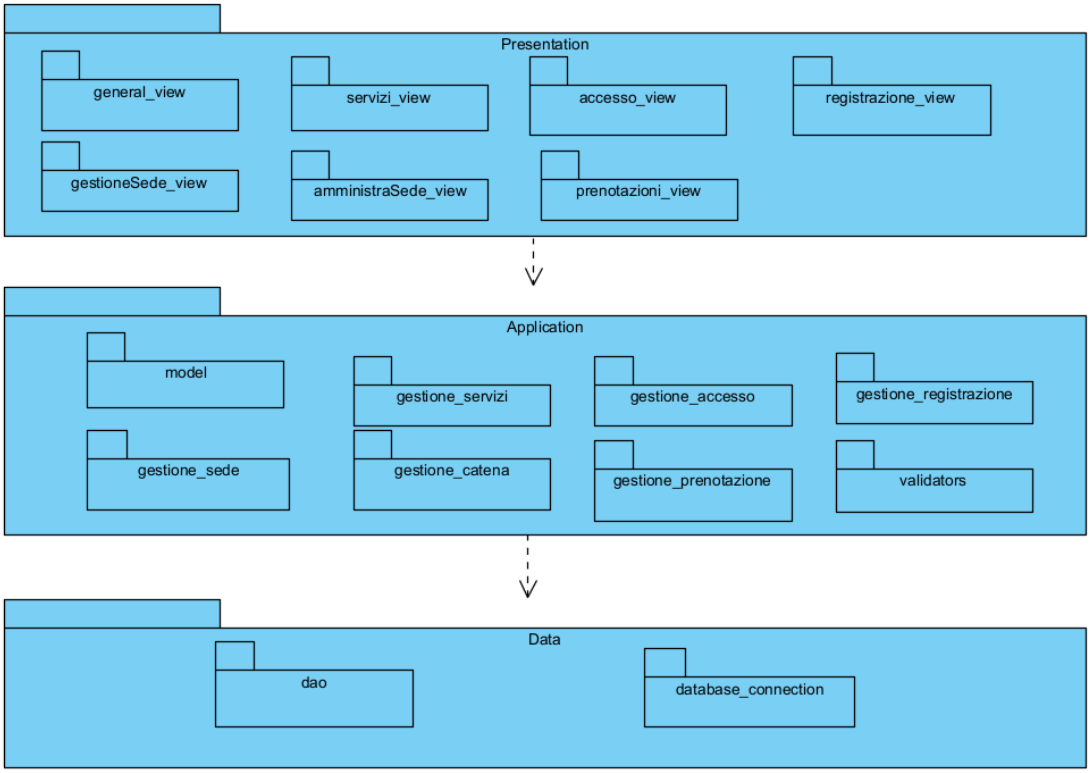
- **HairBeautyNow:** Nome della piattaforma che verrà sviluppata per la gestione delle prenotazioni e delle operazioni nei saloni di bellezza e dei barbieri del franchising, con funzionalità per la gestione centralizzata delle sedi.
- **Dashboard:** Un'interfaccia grafica dove l'utente può visualizzare e/o gestire i dati in base ai suoi permessi.
- **Utente Generale:** Utente che può visitare il sito ed esplorare i servizi offerti dal franchising. Tuttavia, per prenotare un servizio, l'Utente Generale deve registrarsi.
- **Utente Acquirente:** Un utente registrato su HairBeautyNow che può prenotare servizi, selezionare la sede, il professionista e l'orario preferito, oltre a visualizzare in tempo reale le offerte promozionali disponibili.
- **Utente Gestore Sede:** Un utente registrato su HairBeautyNow che ha il ruolo di gestire una singola sede del franchising. L'Utente Gestore Sede può amministrare le prenotazioni, gestire le postazioni e i professionisti della sede, monitorare la disponibilità delle risorse e promuovere offerte e sconti per la sede specifica. L'Utente Gestore Sede è unico per ogni sede, ed è necessario affinché la singola sede possa essere operativa.
- **Utente Gestore Catena:** Un utente registrato su HairBeautyNow che ha il compito di gestire più sedi di una catena di saloni. L'Utente Gestore Catena può monitorare e amministrare centralmente le attività di tutte le sedi, visualizzare le performance complessive della catena e gestire le promozioni a livello multi-sede.

2.Packages

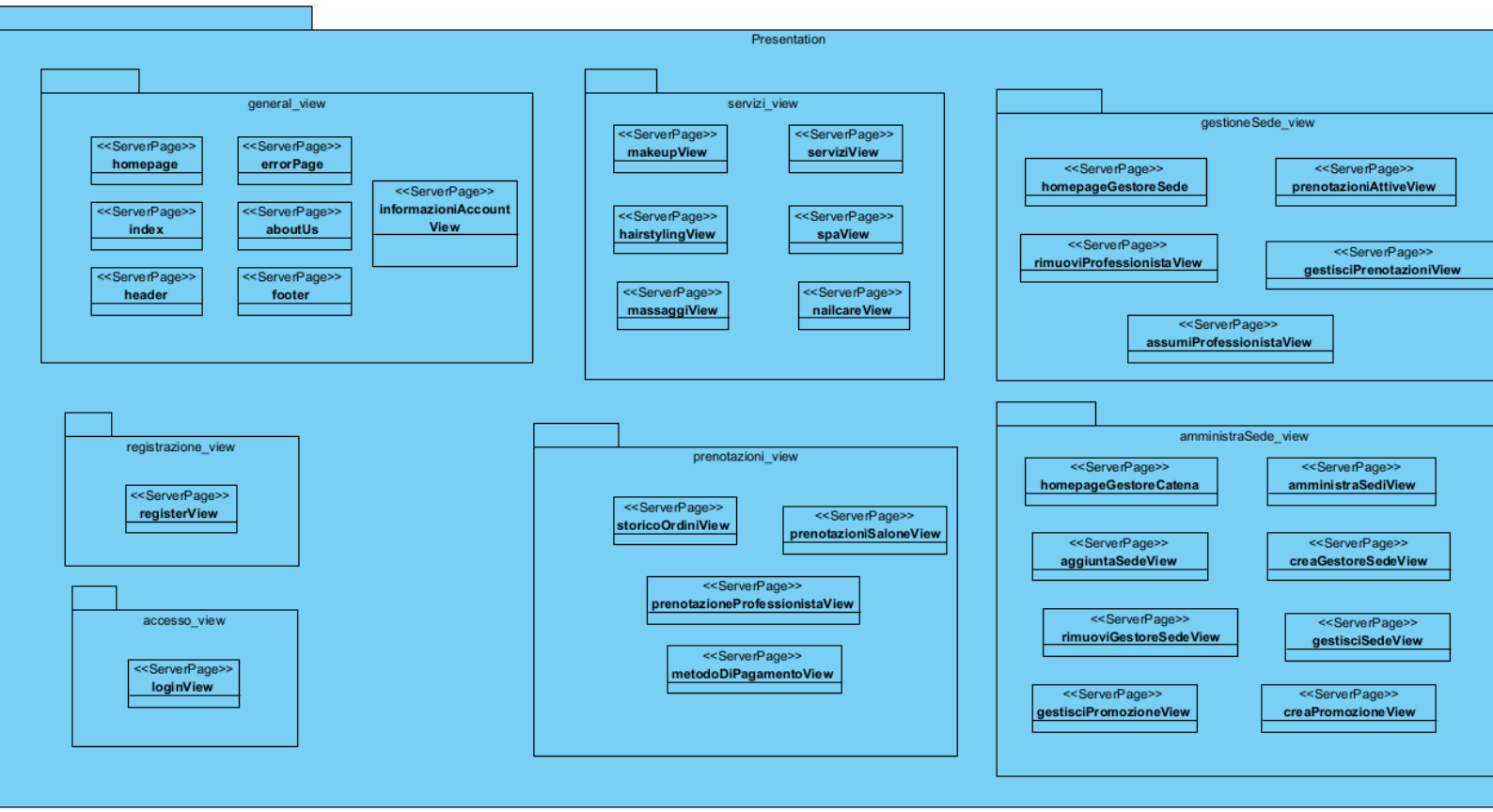
2.1 Panoramica

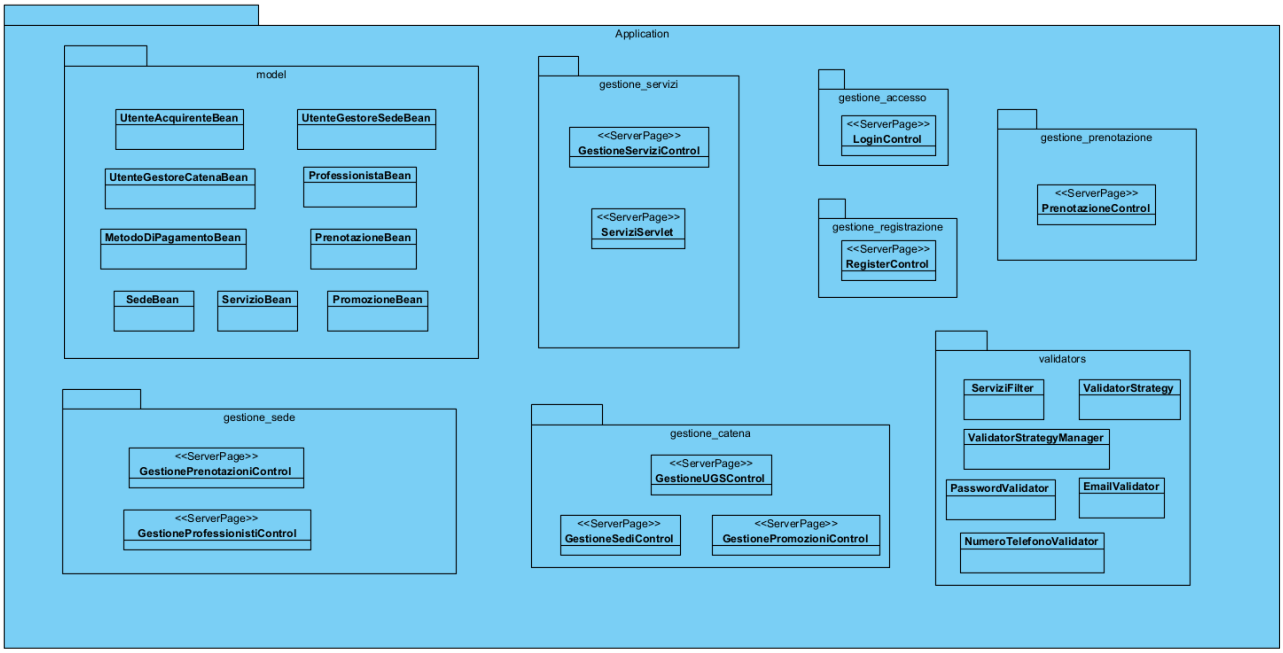
n questa sezione verrà fornita una panoramica generale della struttura completa dei package del sistema **HairBeautyNow**. Successivamente, nella sezione 2.3, si approfondirà il contenuto di ciascun package, analizzandone nel dettaglio la composizione e le funzionalità.

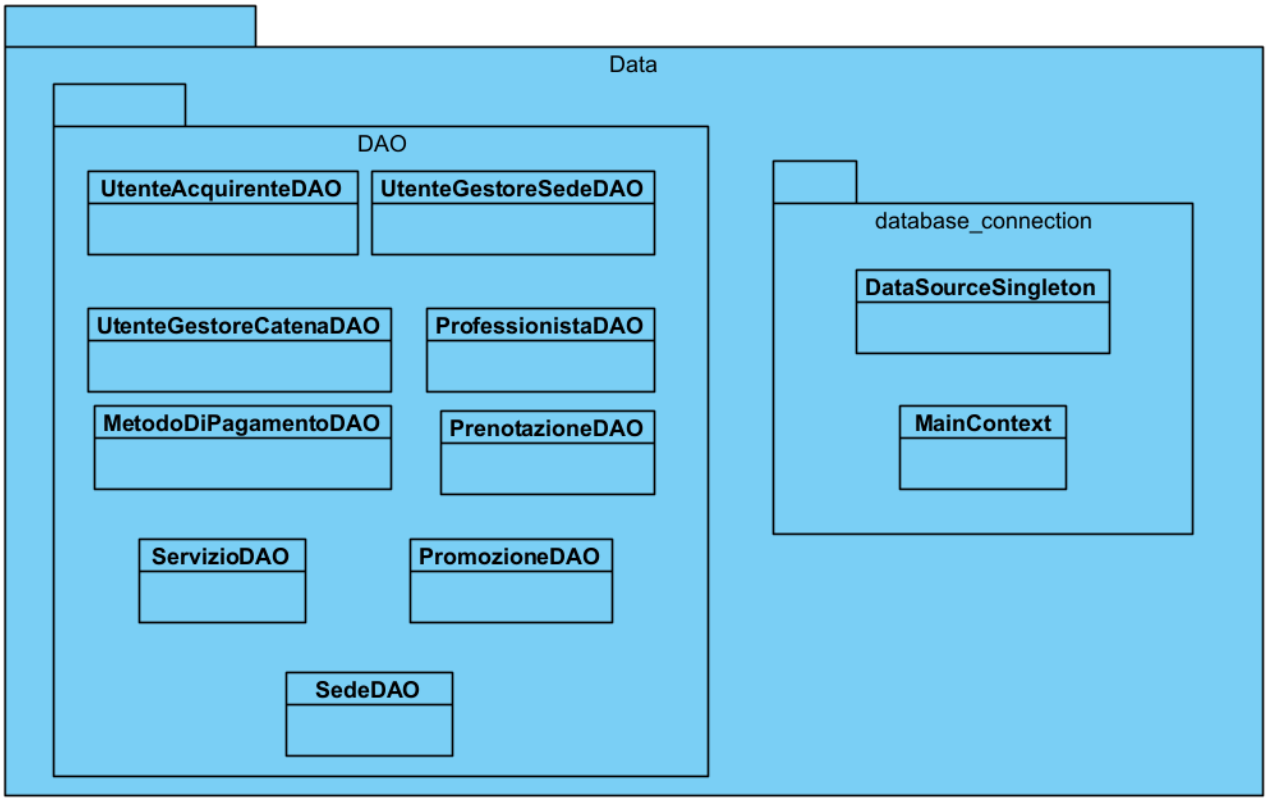
2.2 Packaging del sistema



2.3 Packaging dei layer



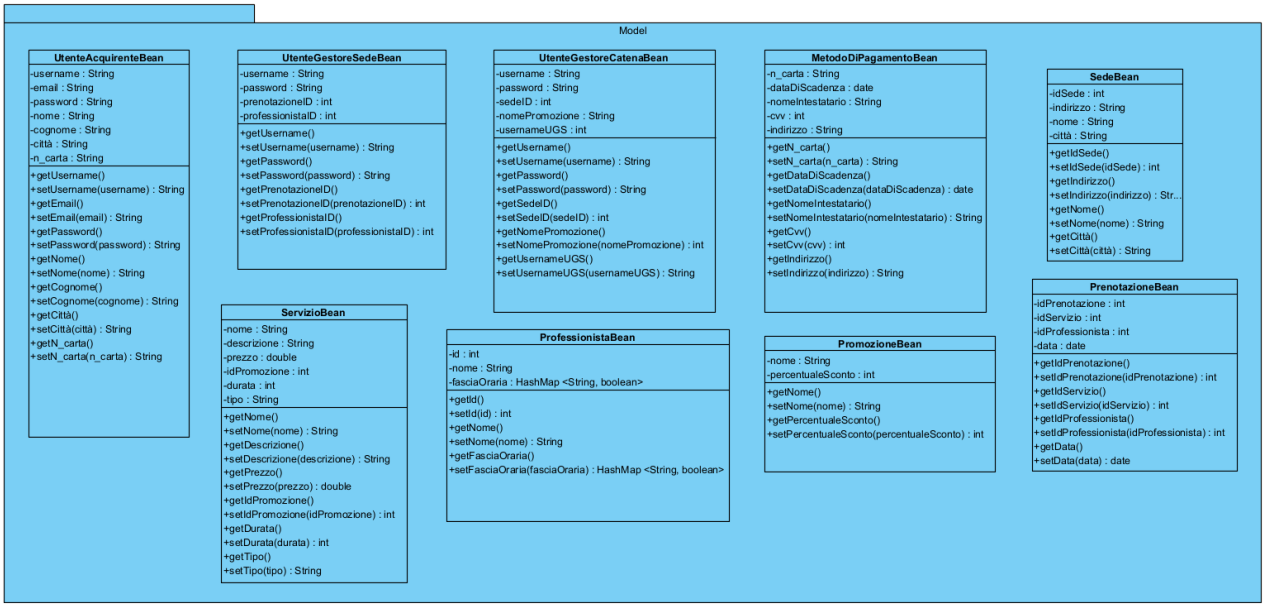

Application
model
UtenteAcquirenteBean
UtenteGestoreSedeBean
UtenteGestoreCatenaBean
ProfessionistaBean
MetodoDiPagamentoBean
PrenotazioneBean
SedeBean
ServizioBean
PromozioneBean
gestione_servizi
<<ServerPage>>
GestioneServiziControl
<<ServerPage>>
ServiziServlet
gestione_accesso
<<ServerPage>>
LoginControl
gestione_registrazione
<<ServerPage>>
RegisterControl
gestione_prenotazione
<<ServerPage>>
PrenotazioneControl
validators
ServiziFilter
ValidatorStrategy
ValidatorStrategyManager
PasswordValidator
EmailValidator
NumeroTelefonoValidator
gestione_sede
<<ServerPage>>
GestionePrenotazioniControl
<<ServerPage>>
GestioneProfessionistiControl
gestione_catena
<<ServerPage>>
GestioneUGSControl
<<ServerPage>>
GestioneSediControl
<<ServerPage>>
GestionePromozioniControl


Data
DAO
UtenteAcquirenteDAO
UtenteGestoreSedeDAO
UtenteGestoreCatenaDAO
ProfessionistaDAO
MetodoDiPagamentoDAO
PrenotazioneDAO
ServizioDAO
PromozioneDAO
SedeDAO
database_connection
DataSourceSingleton
MainContext

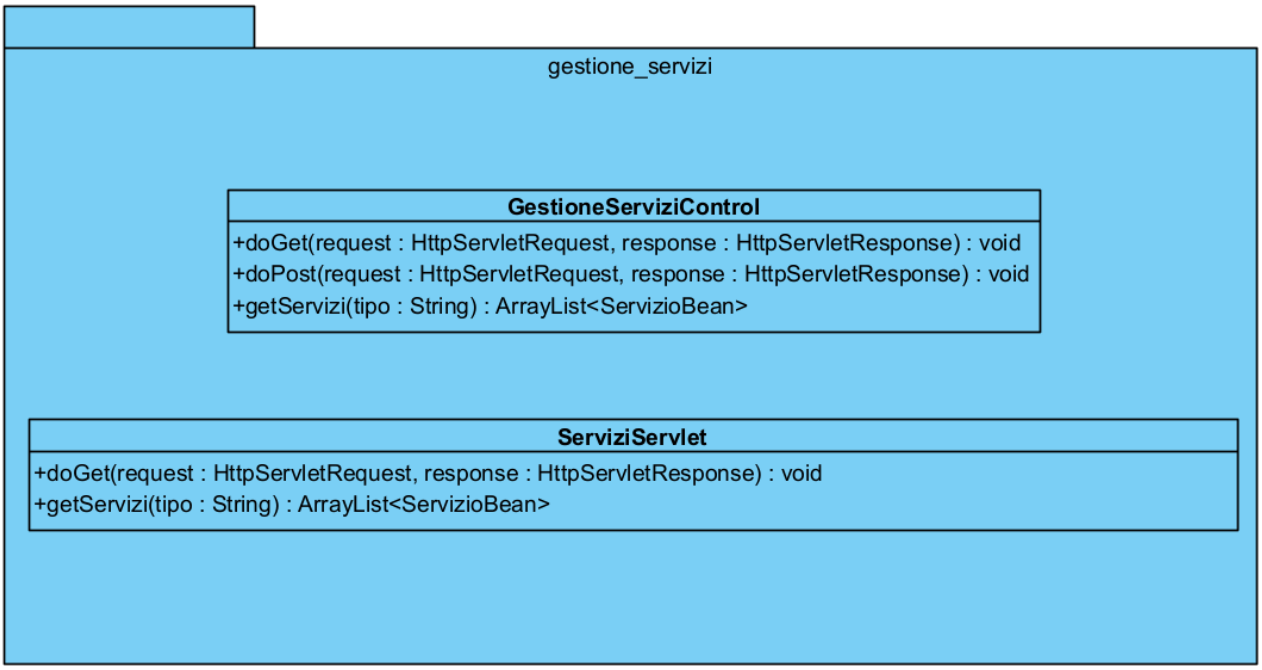
2.4 Dettaglio dei singolo package

2.4.1 Package Application

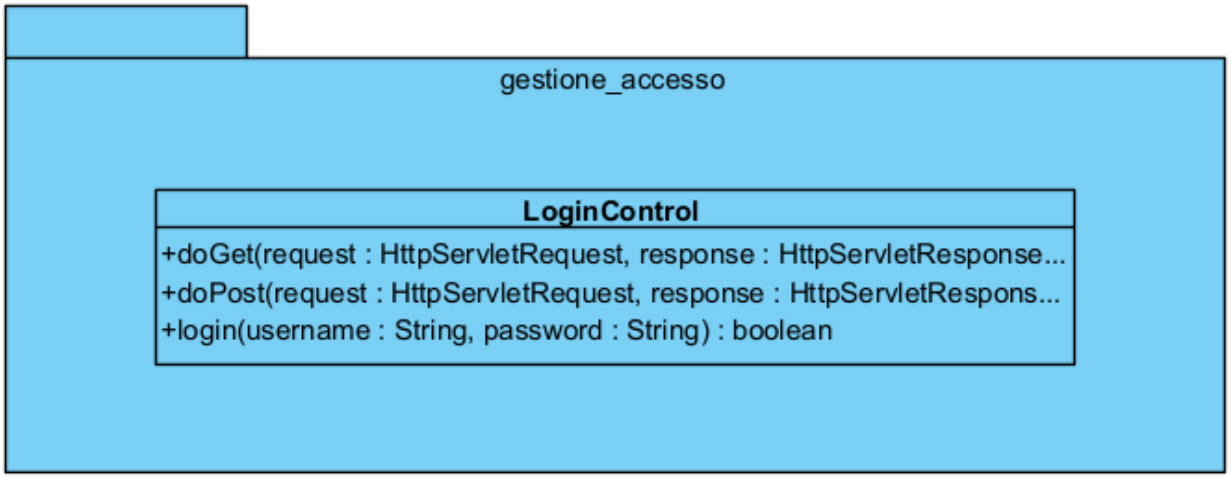
Model



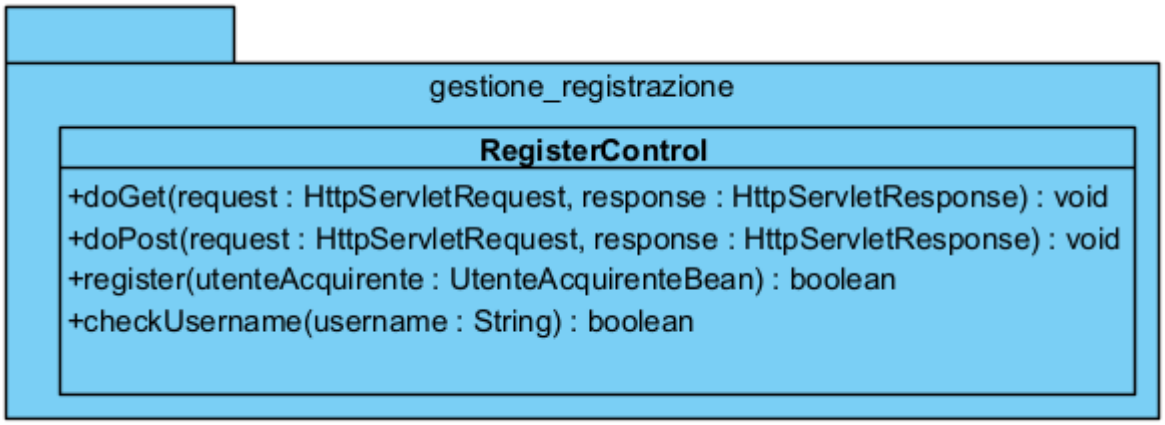
Gestione_servizi



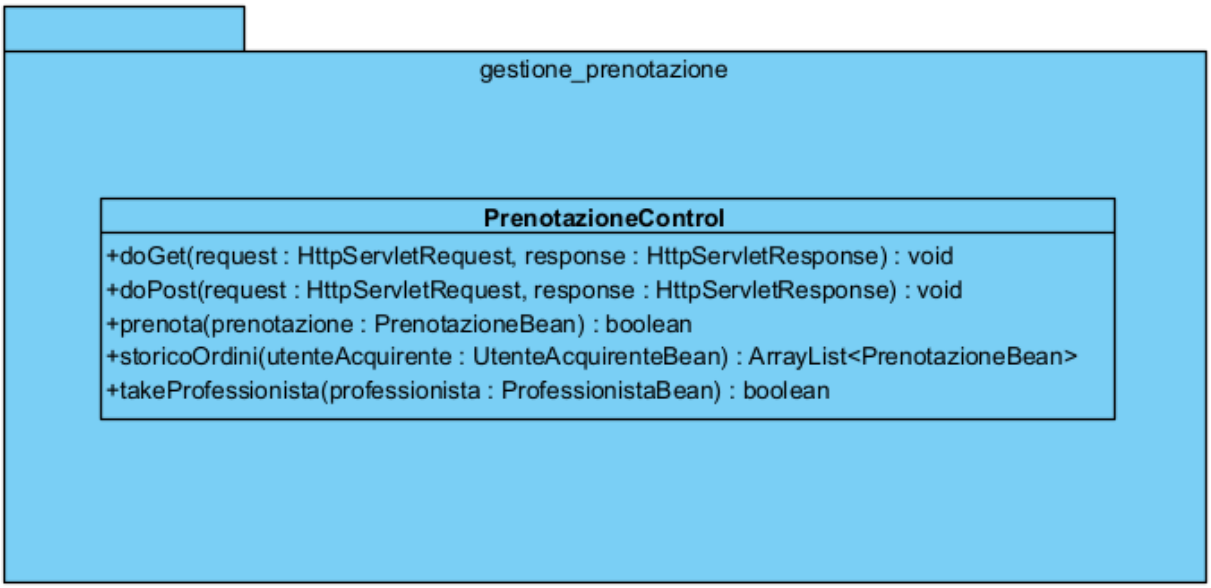
Gestione_accesso



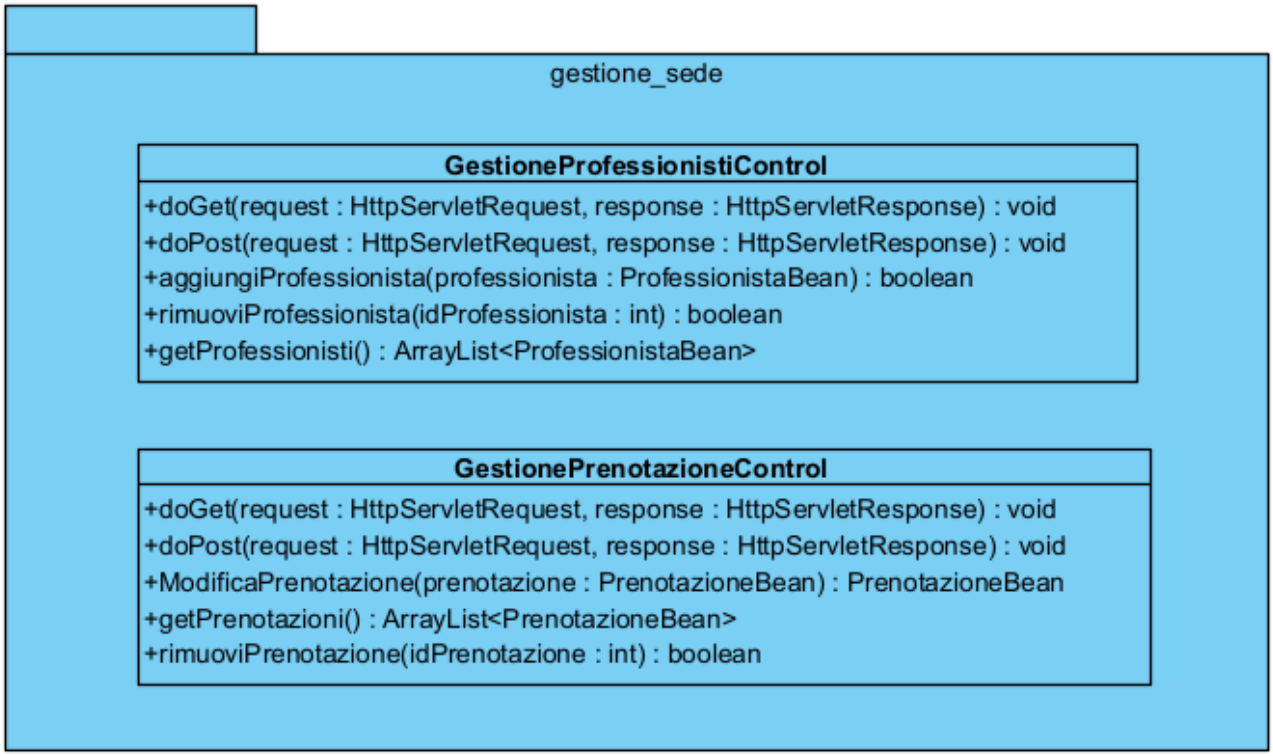
Gestione_registrazione



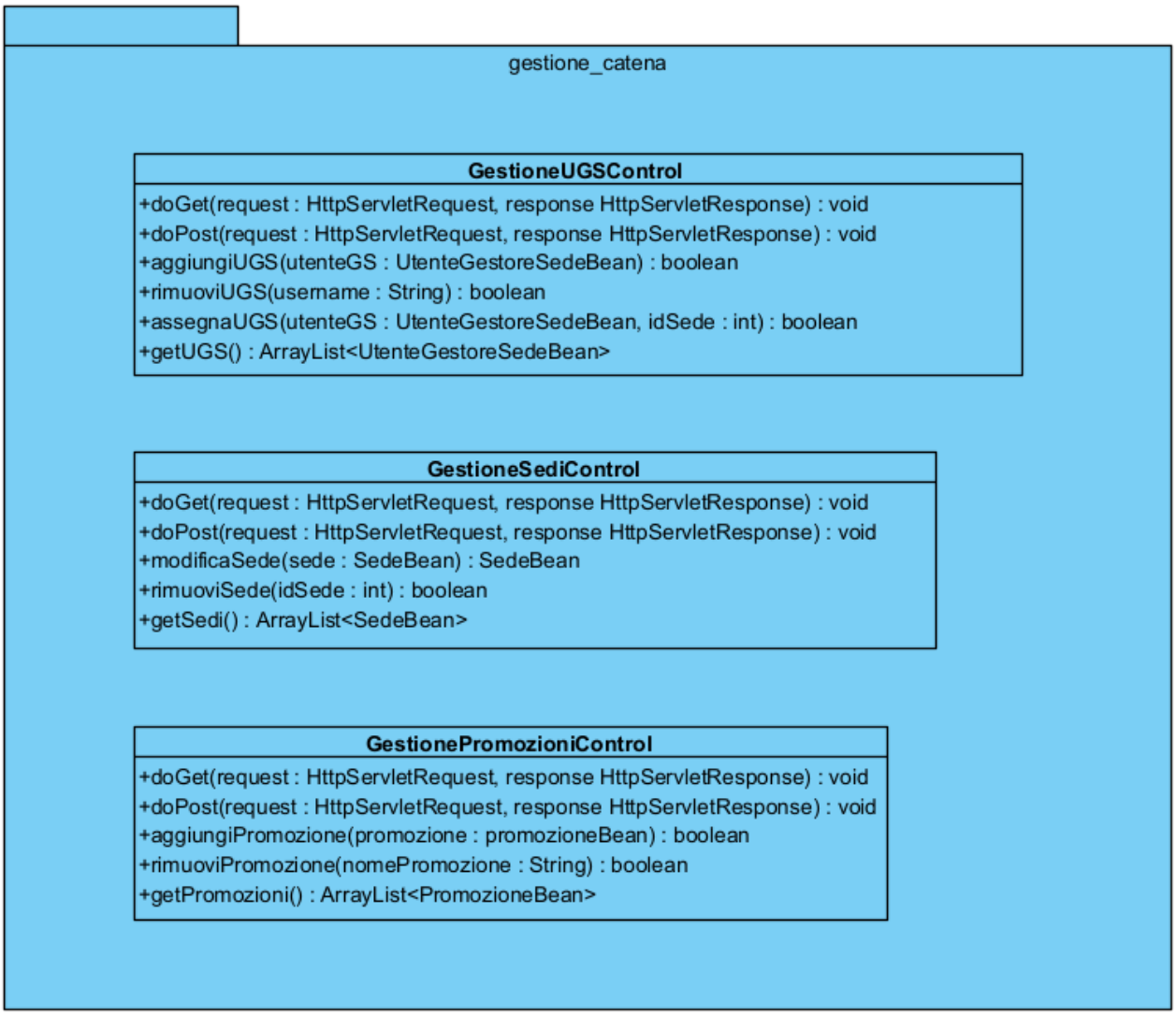
Gestione_prenotazione



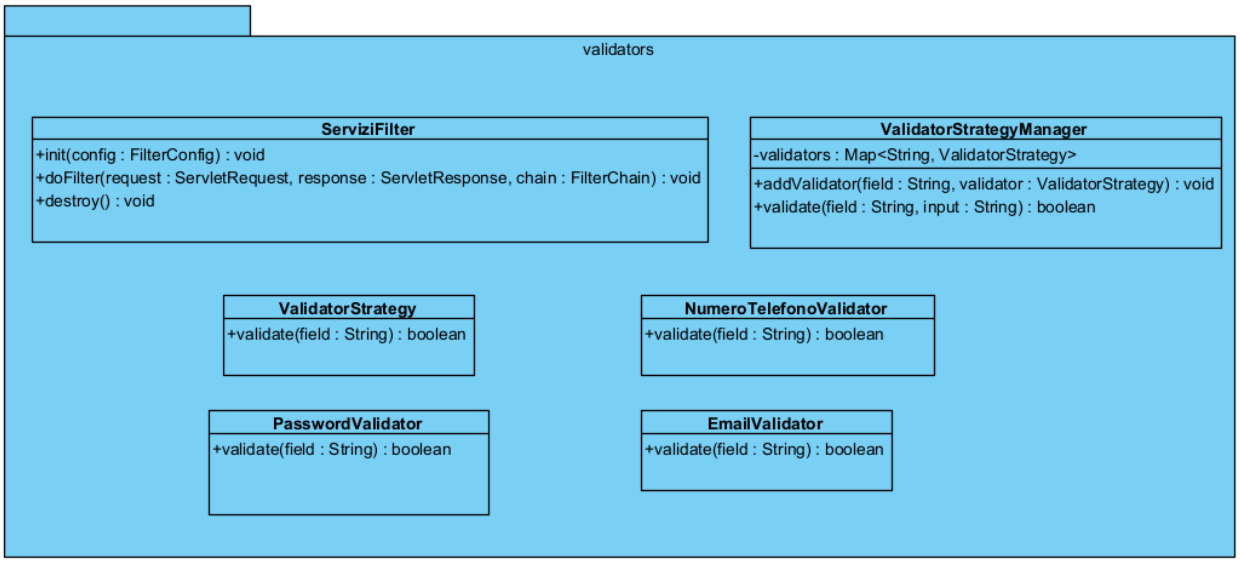
Gestione_sede



Gestione_catena

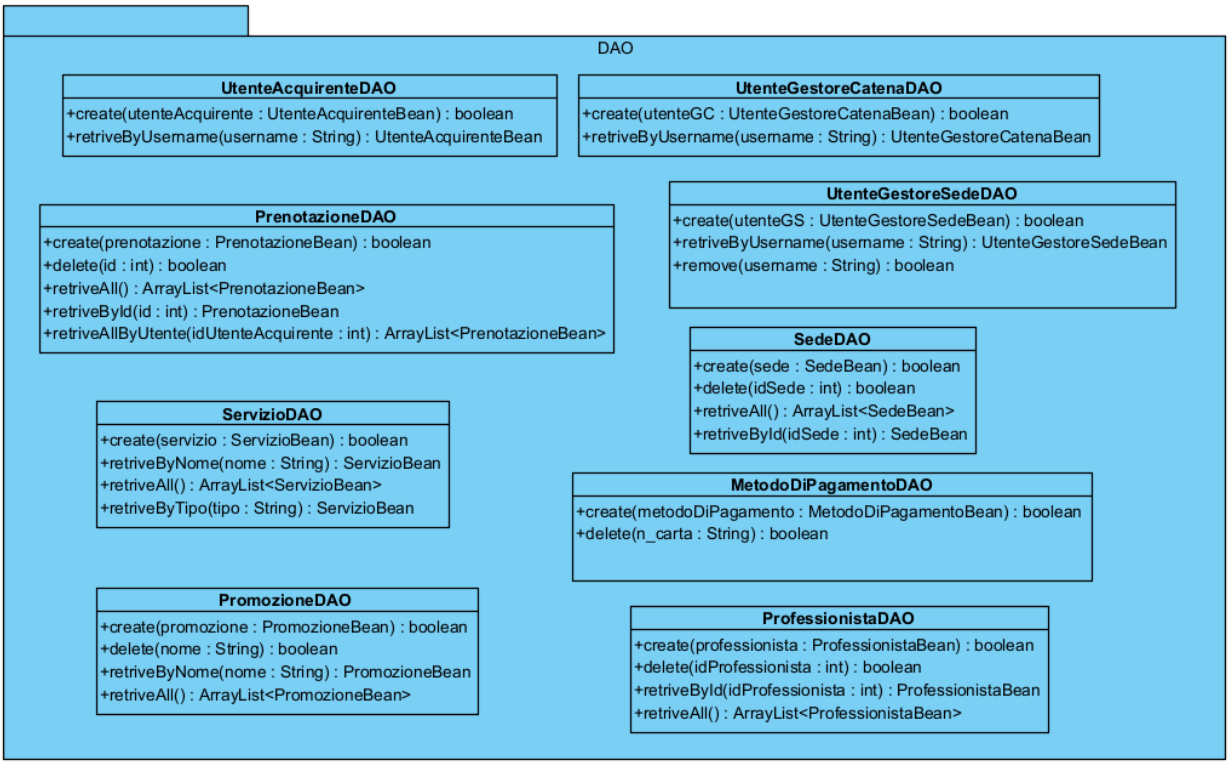


Validators

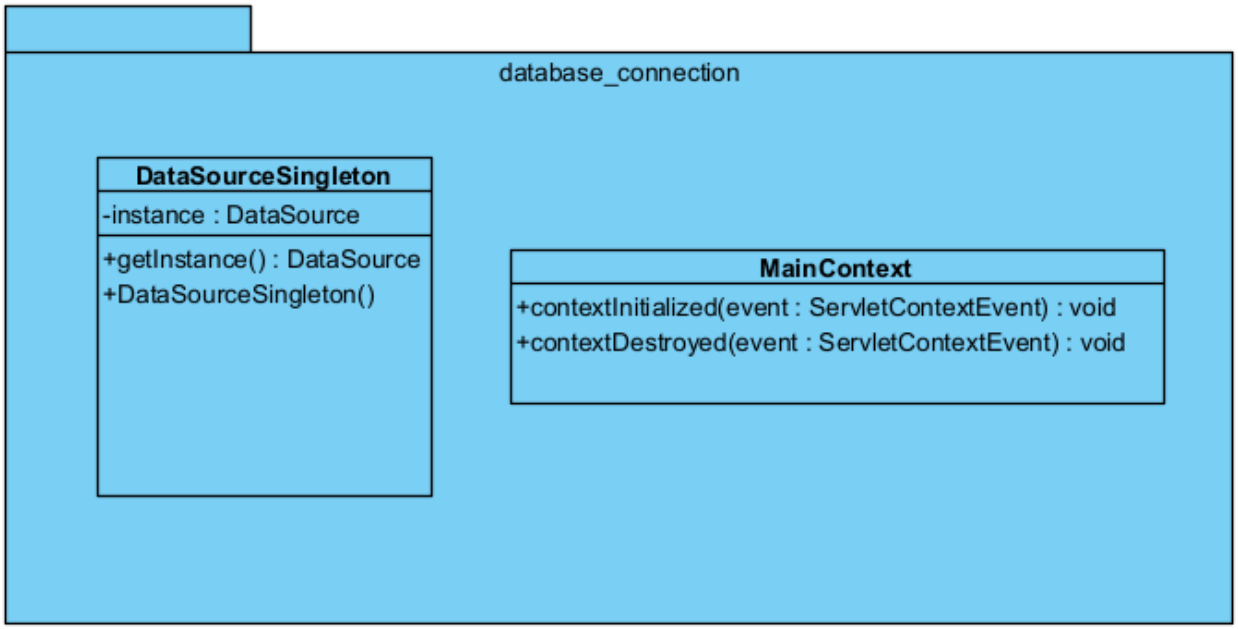

validators
ServiziFilter
+init(config : FilterConfig) : void
+doFilter(request : ServletRequest, response : ServletResponse, chain : FilterChain) : void
+destroy() : void
ValidatorStrategyManager
-validators : Map<String, ValidatorStrategy>
+addValidator(field : String, validator : ValidatorStrategy) : void
+validate(field : String, input : String) : boolean
ValidatorStrategy
+validate(field : String) : boolean
NumeroTelefonoValidator
+validate(field : String) : boolean
PasswordValidator
+validate(field : String) : boolean
EmailValidator
+validate(field : String) : boolean

2.4.1 Package Data

DAO



database_connection



3. Interfaccia delle classi

La terza sezione, **Interfacce delle Classe**, fornisce una descrizione delle classi e delle relative interfacce pubbliche. Questa sezione include una panoramica di ciascuna classe, evidenziando le dipendenze con altre classi e package, gli attributi pubblici, le operazioni disponibili e le eventuali eccezioni che possono essere generate.

NOME CLASSE	**UtenteAcquirenteDAO**
METODI	+create(utenteAcquirente : UtenteAcquirenteDAO) : boolean +retriveByUsername(username : String) : UtenteAcquirenteDAO
INVARIANTI	

NOME CLASSE	**UtenteGestoreSedeDAO**
METODI	+create(utenteGS : UtenteGestoreSedeDAO) : boolean +retriveByUsername(username : String) : UtenteGestoreSedeDAO +remove(username : String) boolean
INVARIANTI	

NOME CLASSE	**UtenteGestoreCatenaDAO**
METODI	+create(utenteGC : UtenteGestoreCatenaDAO) : boolean +retriveByUsername(username : String) : UtenteGestoreCatenaDAO
INVARIANTI	

NOME CLASSE	**PrenotazioneDAO**
METODI	+create(prenotazione : PrenotazioneBean) : boolean +delete(id : int) : boolean +retriveAll() : ArrayList<PrenotazioneBean) +retriveById(id : int) +retriveAllByUtente(idUtenteAcquirente : int) : ArrayList<PrenotazioneBean>
INVARIANTI	

NOME CLASSE	**MetodoDiPagamentoDAO**
METODI	+create(metodoDiPagamento : MetodoDiPagamentoBean) : boolean +delete (n_carta : int) : boolean
INVARIANTI	

NOME CLASSE	**SedeDAO**
METODI	+create(sede : SedeBean) : boolean +delete(idSede : int) : boolean +retriveAll() : ArrayList<SedeBean> +retriveById(idSede : int) : SedeBean
INVARIANTI	

NOME CLASSE	**ServizioDAO**
METODI	+create(servizio : ServizioBean) : boolean +retriveByNome(nome : String) : ServizioBean +retriveAll() : ArrayList<ServizioBean> +retriveByTipo(tipo : String) : ServizioBean
INVARIANTI	

NOME CLASSE	**PromozioneDAO**
METODI	+create(promozione : PromozioneBean) : boolean +delete(nome : String) : boolean +retriveByNome(nome : String) : PromozioneBean +retriveAll() : ArrayList<PromozioneBean>
INVARIANTI	

NOME CLASSE	**ProfessionistaDAO**
METODI	+create(professionista : ProfessionistaBean) : boolean +delete(idProfessionista : int) boolean +retriveById(idProfessionista : int) ProfessionistaBean +retriveAll() : ArrayList<ProfessionistaBean>
INVARIANTI	

4. Glossario

Utente Generale	Rappresenta un utente non autenticato nel sistema, sono presenti quindi restrizioni nell'utilizzo di quest'ultimo.
Utente Acquirente	Rappresenta un utente autenticato nel sistema, che quindi puo' sfruttare le funzionalità riguardanti le prenotazioni e lo storico degli ordini.
Utente Gestore Sede	Rappresenta un utente registrato con il compito di gestire una singola sede dell'intera catena, esso può gestire le prenotazioni e i professionisti in salone.
Utente Gestore Catena	Rappresenta un utente registrato, che gestisce l'intera catena HairBeautyNow, ha il compito di amministrare le sedi, gli Utenti Gestori Sede e di creare e attivare Promozioni in tutta la catena.
Prenotazione	Rappresenta l'acquisto di un servizio con relative informazioni riguardo la sede, il

	professionista scelto e il servizio.
Franchise	Un sistema di collaborazione tra un produttore e un distributore, ove il primo cede al secondo la facoltà di distribuire il servizio con alcune definizioni contrattuali.