Università degli Studi di Salerno
Corso di Ingegneria del Software

Ship Management
Object Design Document
Versione 1.1

Data: 20/12/2020

Coordinatore del progetto:

Nome	Matricola
Andrea De Lucia	0512100000
Fabiano Pecorelli	//

Partecipanti:

Nome	Matricola
Gennaro Pio Rimoli	0512105894
Andrea Selice	0512105702
Chiara Santoro	0512105786

Scritto da:	Gennaro Pio Rimoli Chiara Santoro Andrea Selice

Revision History

Data	Versione	Descrizione	Autore
20/12/2020	1.0	Creazione del documento	Andrea Selice
23/12/2020	1.0	Inserimento Introduzione	Chiara Santoro
24/12/2020	1.0	Inserimento Packages, linee guida e convenzioni	Andrea Selice Chiara Santoro
26/12/2020	1.0	Inserimento Class Diagram Interface	Andrea Selice
27/12/2020	1.0	Inserimento Design Pattern	Gennaro Pio Rimoli
30/12/2020	1.1	Revisione Documento e correzioni errori	Gennaro Pio Rimoli Andrea Selice Chiara Santoro

Indice

1- INTRODUZIONE

Fin ora abbiamo definito in maniera dettagliata gli obiettivi del nostro sistema. In questa fase, approfondiremo gli aspetti implementativi, descrivendo con accuratezza tutte le funzionalità chiarite in precedenza, per arrivare alla realizzazione della nostra piattaforma "Ship Management".
In questo documento descriviamo i trade-off, le linee guida per le interfacce dei sottosistemi, la decomposizione dei sottosistemi in package e classi, le classi di interfacce e design pattern.

Iniziamo considerando i seguenti trade-off:

-Interfaccia vs Usabilità
Non conoscendo l'esatto target di età a cui la piattaforma è rivolto è stato pensato di favorire l'usabilità ai requisiti di interfaccia in modo da permettere all'utente di muoversi in maniera intuitiva all'interno del sito. L'utente ha la possibilità di prendere visione di tutte le funzionalità messe a disposizione in base al tipo di ruolo, direttamente dalla sua area riservata, ed è sempre al corrente di tutte le modifiche alle sue mediazioni/imbarcazioni/richieste grazie alla lista delle notifiche. Tutte le ricerche e le visualizzazioni di più elementi sono organizzate in tabelle che possono essere filtrare in qualsiasi momento in base ai parametri specifici di ogni tabella, inoltre, sono presenti le singole operazioni che si possono effettuare con i singoli elementi della tabella.

-Prestazioni, Costi e Robustezza
Per assicurare un costo modico, il sistema utilizza dei componenti off-the-shelf, come ad esempio Lombok per gestire la corretta chiusura dei componenti, Maven per poter aggiornare e gestire le librerie in maniera facile e veloce, MailTrap per gestire l'invio di e-mail in fase di attivazione del profilo e recupero password. Questo permette di concentrarci di più su altri fattori, come le prestazioni della piattaforma, al fine di offrire un prodotto migliore.

-Sicurezza ed Efficienza
Il sistema garantisce la completa protezione dei dati, implementando meccanismi di crittografia dei dati sensibili (credenziali di accesso), principalmente nella fase di autenticazione utilizzando SHA1 in modo tale da rendere più complicato il recupero delle informazioni in caso di compromissione del database. Il sistema offre controlli e supervisione in qualsiasi area del sistema, in modo tale da guadagnarsi la fiducia del cliente. Qualsiasi tipo di input proveniente dall'utente viene filtrato in modo tale da evitare attacchi di tipo XSS o SQL Injection. Il sistema effettua back up giornalieri ed elimina i dati superflui per conservare una copia dei dati in caso di manomissioni al sistema. Nonostante tutte queste misure preventive impattino sulle prestazioni del sistema, sono necessarie per rispettare le norme minime del GDPR.

-Spazio di memoria e tempi di risposta
Per evitare di utilizzare più memoria del necessario, è stato deciso di memorizzare il concetto di firma/terminazione non come un attributo all'interno di mediazione/imbarcazione/richiesta ma come una tabella supplementare i cui record vengono eliminati non appena vi si presenta un cambio di stato nella mediazione.

1.2 INTERFACE DOCUMENTATION E GUIDELINES

Per evitare fraintendimenti e omissioni che potrebbero causare fault e delay e/o creare ambiguità, ci avvaliamo dell'utilizzo di convenzioni che aiutano anche a rendere più chiaro e comprensibile il nostro lavoro.
Le annotazioni descritte permettono di avere un'accurata comunicazione, grazie ad una semantica ben definita e adattabile per rappresentare i vari aspetti del sistema.
La documentazione delle linee guida e le convenzioni saranno un punto di riferimento per tutti i membri del team che andranno a sviluppare le varie parti del sistema, e sono considerate il fattore più importante per il migliorare la comunicazione tra gli sviluppatori.
Di seguito elenchiamo una lista di regole da seguire:

Class Conventions

I nomi delle classi sono costituiti da un singolo nome, più precisamente un sostantivo, al singolare, che descrive in modo specifico cosa rappresenta quella determinata classe. Le classi devono iniziare con la lettera maiuscola e contengono solo lettere.

Methods conventions

I nomi dei metodi sono costituiti da un verbo (es getName) che descrive nella maniera più appropriata le azioni che svolge quel metodo. Essi devono seguire lo stile Camel case in modo tale da essere più leggibili.

Parameters conventions

I nomi dei parametri sono costituiti da un sostantivo che caratterizza quel determinato parametro. Il nome deve iniziare con una lettera minuscola e deve contenere solo lettere. Nel caso in cui si necessita dell'aggiunta di un ulteriore nome, si utilizza lo stile Camel case.
Le stesse convenzioni vengono applicate alle variabili.

Error conventions

Lo stato degli errori deve essere ritornato come un'eccezione, non come un valore di ritorno.

JSP, HTML, Coding conventions

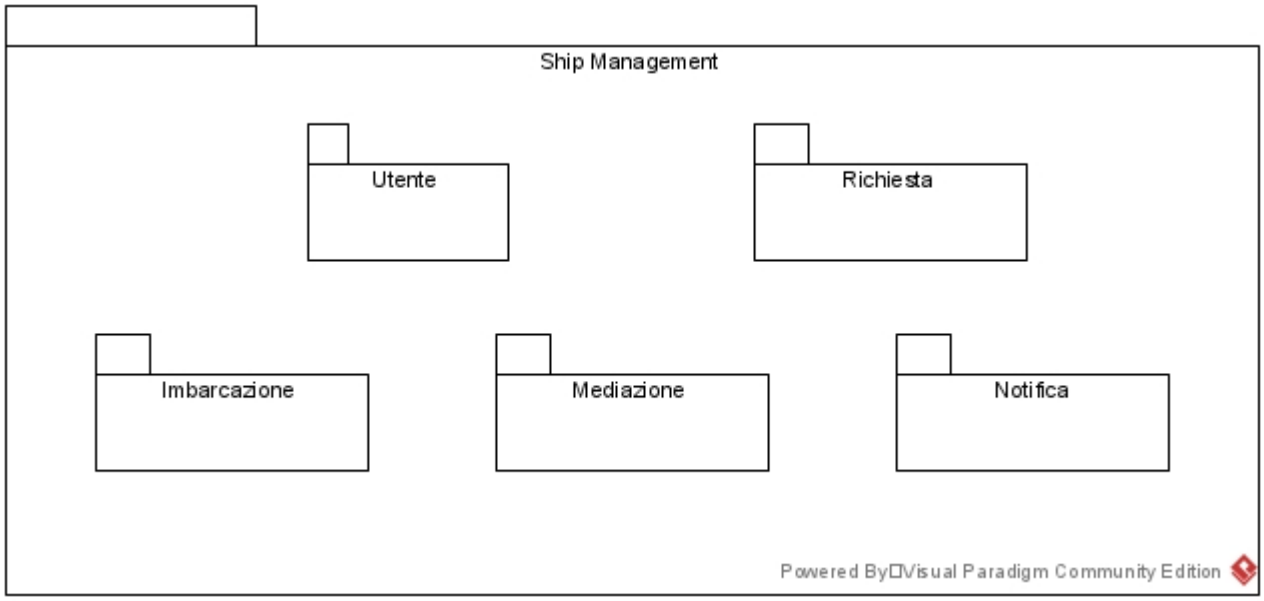
Le pagine JSP possono contenere codice Java, Javascript, HTML e devono seguire le buone norme di indentazione. Ogni azione racchiusa in un gruppo di tag, deve essere come un unico blocco, allo stesso livello di indentazione. Per quanto riguarda le operazioni più interne, bisogna usare i Tab per mostrare la gerarchia del codice tramite l'indentazione.

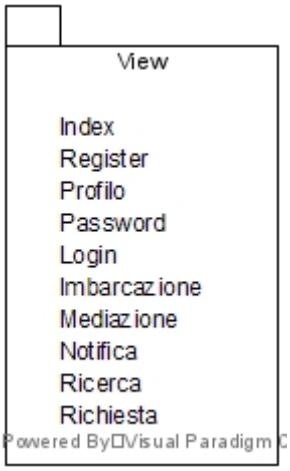
1.3 DEFINIZIONI

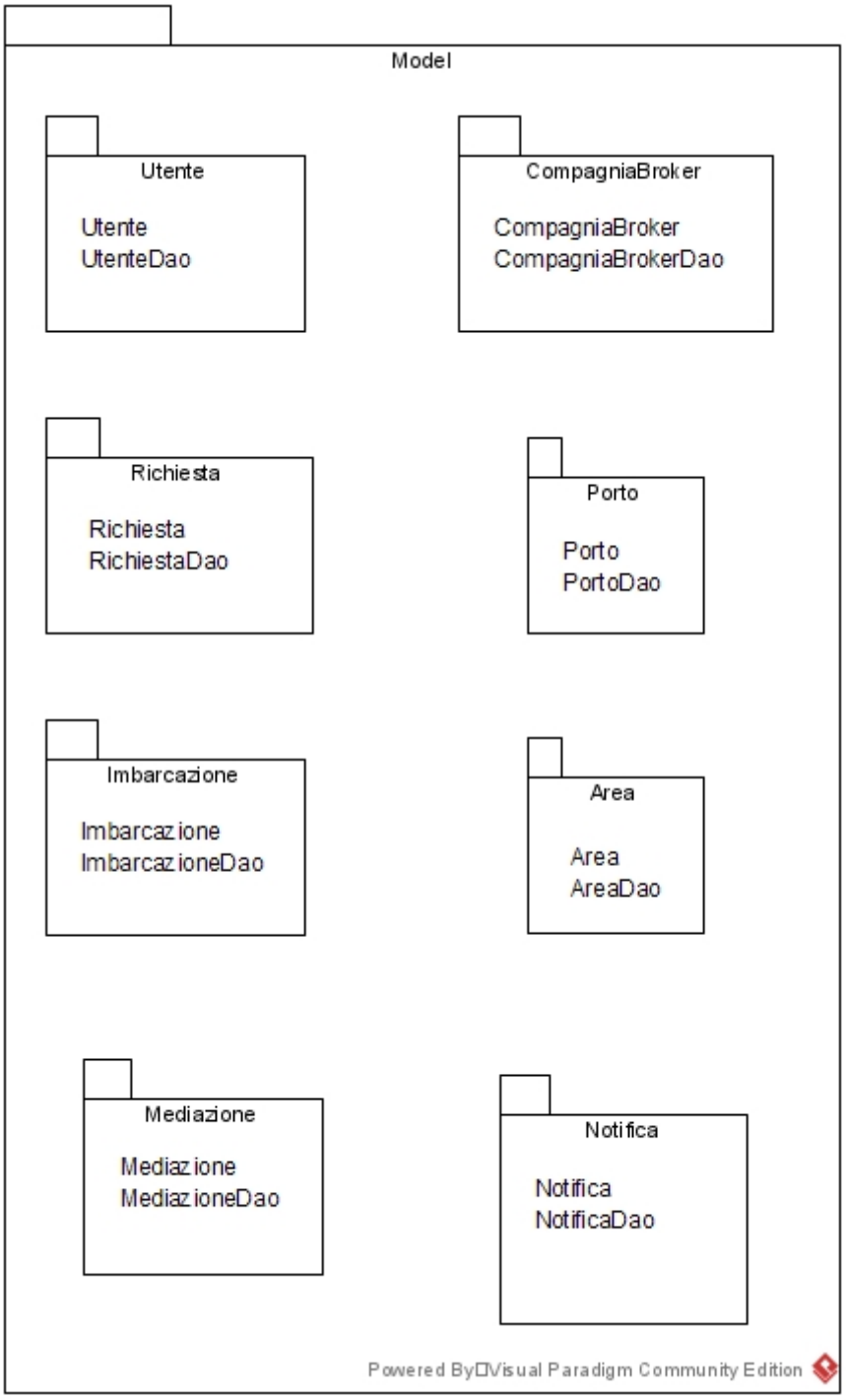
Design pattern: sono soluzioni di template che gli sviluppatori definiscono nel tempo per risolvere una gamma di problemi ricorrenti. Un design pattern ha quattro elementi:

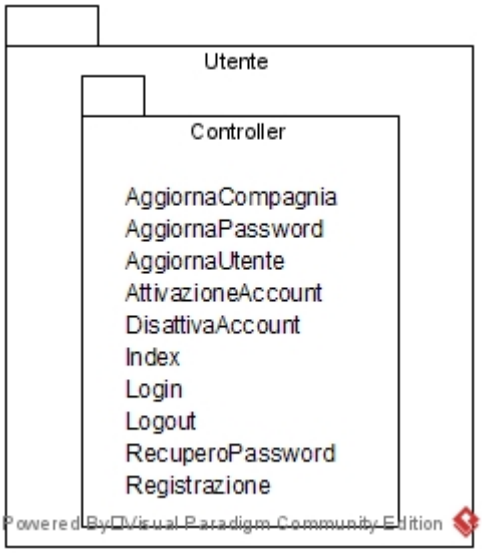
- -Un **nome** che identifica univocamente un pattern da altri pattern;
- -Una **descrizione** di un problema che descrive la situazione dove viene usato il pattern;
- -Una **soluzione** presentata come insieme di classi e interfacce;
- - Un insieme di **conseguenze** che descrive i trade-off e le alternative che devono essere considerate rispetto ai design goal fissati;

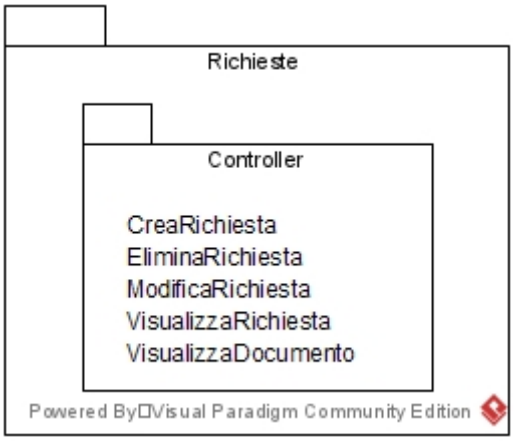
2 – PACKAGES

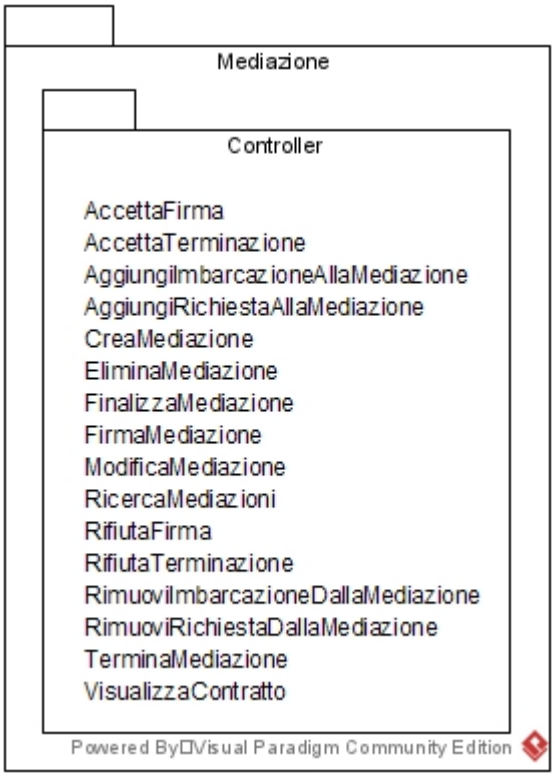

Ship Management
Utente
Richiesta
Imbarcazione
Mediazione
Notifica
Powered By Visual Paradigm Community Edition

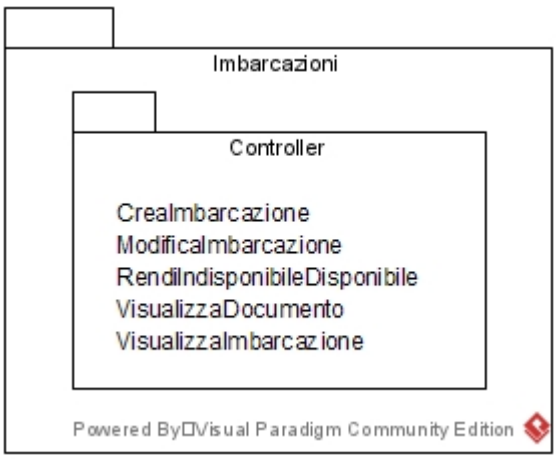

View
Index
Register
Profilo
Password
Login
Imbarcazione
Mediazione
Notifica
Ricerca
Richiesta

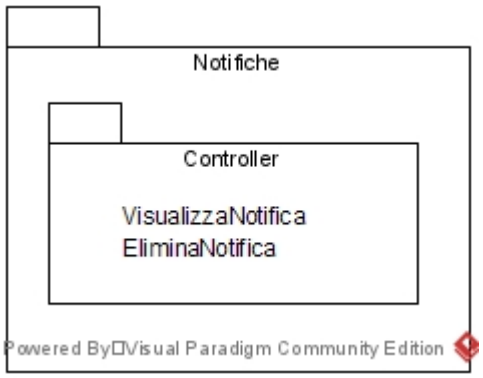

Model
Utente
Utente
UtenteDao
CompagniaBroker
CompagniaBroker
CompagniaBrokerDao
Richiesta
Richiesta
RichiestaDao
Porto
Porto
PortoDao
Imbarcazione
Imbarcazione
ImbarcazioneDao
Area
Area
AreaDao
Mediazione
Mediazione
MediazioneDao
Notifica
Notifica
NotificaDao
Powered By Visual Paradigm Community Edition


Utente
Controller
AggiornaCompagnia
AggiornaPassword
AggiornaUtente
AttivazioneAccount
DisattivaAccount
Index
Login
Logout
RecuperoPassword
Registrazione
Powered By Visual Paradigm Community Edition


Richieste
Controller
CreaRichiesta
EliminaRichiesta
ModificaRichiesta
VisualizzaRichiesta
VisualizzaDocumento
Powered By Visual Paradigm Community Edition


Mediazione
Controller
AccettaFirma
AccettaTerminazione
AggiungiImbarcazioneAllaMediazione
AggiungiRichiestaAllaMediazione
CreaMediazione
EliminaMediazione
FinalizzaMediazione
FirmaMediazione
ModificaMediazione
RicercaMediazioni
RifiutaFirma
RifiutaTerminazione
RimuoviImbarcazioneDallaMediazione
RimuoviRichiestaDallaMediazione
TerminaMediazione
VisualizzaContratto
Powered By Visual Paradigm Community Edition


Imbarcazioni
Controller
CreaImbarcazione
ModificaImbarcazione
RendiIndisponibileDisponibile
VisualizzaDocumento
VisualizzaImbarcazione
Powered By Visual Paradigm Community Edition



CONTROLLER	
Registrazione	Questa Servlet viene chiamata nel momento in cui si effettua la registrazione. Si occupa di prelevare i dati dell'utente, verificarne la correttezza, creare un nuovo Utente che sarà poi inviato all'UtenteDao per il salvataggio nel database e inviare la mail per l'attivazione del profilo. In caso di registrazione di un Broker vengono inviati dei dati anche a CompagniaBrokerDao.
Login	Questa Servlet viene chiamata nel momento in cui si effettua il login, verifica la correttezza dei dati inseriti dall'utente per poi passarli all'UtenteDao e ricevere da esso il profilo o eventuali errori
Logout	Questa Servlet viene chiamata nel momento in cui si effettua il logout. L'utente viene disconnesso e il suo profilo eliminato dalla sessione
RecuperoPassword	Questa Servlet viene chiamata nel momento in cui l'utente deve recuperare la password per accedere al proprio account. Viene chiamato UtenteDao per poter effettuare il recupero
AttivazioneAccount	Questa Servlet viene chiamata nel momento in cui si deve effettuare l'attivazione dell'account, viene chiamato UtenteDao per poter effettuare tale operazione
AggiornaUtente	Questa Servlet viene chiamata nel momento in cui un utente vuole modificare i dati del proprio profilo. La servlet si occupa di verificare la correttezza dei dati, modificarli in sessione ed inviarli all'UtenteDao per la modifica nel database
AggiornaCompagnia	Questa Servlet viene chiamata nel momento in cui un Broker deve aggiornare i dati della propria compagnia di appartenenza. Questi ultimi vengono poi mandati a

	CompagniaBrokerDao per effettuare la modifica nel database
AggiornaPassword	Questa Servlet viene chiamata nel momento in cui un utente vuole modificare la propria password. Viene chiamato UtenteDao per effettuare la modifica
DisattivaAccount	Questa Servlet viene chiamata nel momento in cui un utente vuole disattivare il proprio profilo. La servlet si occupa di eliminare il profilo dalla sessione e di passarlo all'UtenteDao per la disattivazione nel database
Index	Questa Servlet viene chiamata per effettuare il filtraggio delle informazioni che devono essere viste dall'utente, in base al proprio ruolo, nella pagina di Index
CreaRichiesta	Questa Servlet viene chiamata nel momento in cui un Cliente vuole inserire una richiesta nel sistema. La servlet si occupa di verificare la correttezza dei dati, e di inviarli alla RichiestaDao per poterli salvare nel database
ModificaRichiesta	Questa Servlet viene chiamata nel momento in cui un Cliente vuole modificare una propria richiesta presente nel sistema. La servlet si occupa di verificare la correttezza dei dati, e di inviarli alla RichiestaDao per poterli aggiornare nel database. In caso di presenza della richiesta in una mediazione, la servlet si occupa anche di chiamare NotificaDao per inviare una notifica ai Broker che gestiscono le mediazioni ove compare la richiesta modificata
EliminaRichiesta	Questa Servlet viene chiamata nel momento in cui un Cliente vuole eliminare una propria richiesta presente nel sistema. La servlet si occupa di chiamare RichiestaDao per poter eliminare dal database la Richiesta. In caso di presenza della richiesta in una mediazione, la servlet si occupa anche di chiamare NotificaDao per inviare una notifica ai Broker che gestiscono le mediazioni ove compare la richiesta eliminata
VisualizzaRichiesta	Questa Servlet viene chiamata nel momento in cui un Cliente vuole visualizzare i dati di una propria richiesta presente nel sistema o un Broker vuole visualizzare i dati di una determinata richiesta. La servlet si occupa di inviare la domanda di visualizzazione alla RichiestaDao la quale, insieme a PortoDao, può restituire la Richiesta che deve essere visualizzata
VisualizzaDocumento	Questa Servlet viene chiamata nel momento in cui si vuole visualizzare il documento presente all'interno di una richiesta. La servlet si occupa di inviare la domanda di visualizzazione alla RichiestaDao la quale restituisce il documento che deve essere visualizzato
CreaImbarcazione	Questa Servlet viene chiamata nel momento in cui un Armatore vuole inserire un'imbarcazione nel sistema. La

	servlet si occupa di verificare la correttezza dei dati, e di inviarli ad ImbarcazioneDao per poterli salvare nel database
ModificaImbarcazione	Questa Servlet viene chiamata nel momento in cui un Armatore vuole modificare una propria imbarcazione presente nel sistema. La servlet si occupa di verificare la correttezza dei dati, e di inviarli ad ImbarcazioneDao per poterli aggiornare nel database. In caso di presenza dell'imbarcazione in una mediazione, la servlet si occupa anche di chiamare NotificaDao per inviare una notifica ai Broker che gestiscono le mediazioni ove compare l'imbarcazione modificata
RendiIndisponibileDisponibile	Questa Servlet viene chiamata nel momento in cui un Armatore vuole rendere disponibile o indisponibile una propria imbarcazione presente nel sistema. La servlet si occupa di chiamare ImbarcazioneDao per poter aggiornare lo stato dell'imbarcazione nel database. In caso di presenza dell'imbarcazione in una mediazione, la servlet si occupa anche di chiamare NotificaDao per inviare una notifica ai Broker che gestiscono le mediazioni ove compare l'imbarcazione che è diventata indisponibile.
VisualizzaImbarcazione	Questa Servlet viene chiamata nel momento in cui un Armatore vuole visualizzare i dati di una propria imbarcazione presente nel sistema o un Broker vuole visualizzare i dati di una determinata imbarcazione. La servlet si occupa di inviare la domanda di visualizzazione ad ImbarcazioneDao la quale può restituire l'imbarcazione che deve essere visualizzata
VisualizzaDocumento	Questa Servlet viene chiamata nel momento in cui si vuole visualizzare il documento presente all'interno di un'imbarcazione. La servlet si occupa di inviare la domanda di visualizzazione ad ImbarcazioneDao la quale restituisce il documento che deve essere visualizzato
CreaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker vuole creare una mediazione. La servlet si occupa di verificare la correttezza dei dati, e di inviarli a MediazioneDao per poterli inserire nel database
EliminaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker vuole eliminare una propria mediazione presente nel sistema. La servlet si occupa di chiamare ImbarcazioneDao per aggiornare lo stato delle imbarcazioni presenti nella mediazione; RichistaDao per poter aggiornare lo stato delle Richieste presenti nella mediazione; NotificaDao per poter inviare una notifica ai partecipanti in merito all'eliminazione della mediazione; MediazioneDao per poter eliminare i dati della

	mediazione dal database
VisualizzaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker, un Armatore o un Cliente vuole visualizzare le informazioni relative ad una mediazione. La servlet si occupa di inviare la domanda di visualizzazione alla MediazioneDao, ImbarcazioneDao, RichiestaDao i quali restituiscono le informazioni della Mediazione che deve essere visualizzata
FinalizzaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker finalizza la mediazione una volta inseriti tutti i dati necessari. La servlet si occupa di verificare la correttezza dei dati, e di inviarli a MediazioneDao per poter aggiornarli nel database e per poter aggiornare lo stato della mediazione. Viene chiamato anche NotificaDao per poter inviare una notifica a tutti i Clienti e Armatori facenti parte della mediazione
AggiungiImbarcazioneAllaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker aggiunge un'imbarcazione ad una mediazione. La servlet si occupa di chiamare MediazioneDao per poter memorizzare l'aggiunta alla mediazione
AggiungiRichiestaAllaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker aggiunge una richiesta alla mediazione. La servlet si occupa di chiamare MediazioneDao per poter memorizzare l'aggiunta alla mediazione.
ModificaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker vuole modificare i dati di una propria mediazione presente nel sistema. La servlet si occupa di verificare la correttezza dei dati inseriti, e di inviarli a MediazioneDao per poter aggiornare i dati nel database.
RimuoviImbarcazioneDallaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker rimuove dalla mediazione un'imbarcazione. La servlet si occupa di chiamare MediazioneDao per poter eliminare il riferimento nel database.
RimuoviRichiestaDallaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker rimuove dalla mediazione una richiesta. La servlet si occupa di chiamare MediazioneDao per poter eliminare il riferimento nel database.
VisualizzaContratto	Questa Servlet viene chiamata nel momento in cui si vuole visualizzare il contratto presente all'interno di una mediazione. La servlet si occupa di inviare la domanda di visualizzazione a MediazioneDao la quale restituisce il documento che deve essere visualizzato
AccettaFirma	Questa Servlet viene chiamata nel momento in cui un Cliente o un Armatore decide di firmare una mediazione. La servlet si occupa di chiamare MediazioneDao per salvare la decisione presa. Nel caso in cui l'utente sia l'ultimo ad aver firmato, viene chiamato anche

	NotificaDao per poter inviare una notifica agli utenti della mediazione.
RifiutaFirma	Questa Servlet viene chiamata nel momento in cui un Cliente o un Armatore decide di rifiutare una mediazione. La servlet si occupa di chiamare MediazioneDao per salvare la decisione presa, verifica la correttezza dei dati inseriti e viene chiamato NotificaDao per poter inviare una notifica al Broker della mediazione, inoltre si chiama MediazioneDao per poter aggiornare lo stato della mediazione.
TerminaMediazione	Questa Servlet viene chiamata nel momento in cui un Broker termina la mediazione. La servlet si occupa di chiamare MediazioneDao per poter aggiornare lo stato della mediazione nel database. Viene chiamato anche NotificaDao per poter inviare una notifica a tutti i Clienti e Armatori facenti parte della mediazione
AccettaTerminazione	Questa Servlet viene chiamata nel momento in cui un Cliente o un Armatore decide di accettare la terminazione di una mediazione. La servlet si occupa di salvare la decisione presa, Nel caso in cui l'utente sia l'ultimo ad aver accettato, viene chiamato anche RichiestaDao ed ImbarcazioneDao e MediazioneDao per poter aggiornare gli stati di questi ultimi.
RifiutaTerminazione	Questa Servlet viene chiamata nel momento in cui un Cliente o un Armatore decide di rifiutare la terminazione di una mediazione. La servlet si occupa di salvare la decisione presa, verifica la correttezza dei dati inseriti e viene chiamato NotificaDao per poter inviare una notifica al Broker della mediazione, inoltre si chiama MediazioneDao per poter aggiornare lo stato della mediazione.
RicercaMediazioni	Questa Servlet viene chiamata quando un Broker sta effettuando ricerche su Imbarcazioni e Richieste, si occupa di chiamare ImbarcazioneDao, RichiesteDao, AreaDao e PortoDao
VisualizzaNotifica	Questa Servlet viene chiamata nel momento in cui un Broker, un Armatore o un Cliente vuole visualizzare le informazioni relative ad una Notifica. La servlet si occupa di inviare la domanda di visualizzazione a NotificaDao la quale può restituire la Notifica che deve essere visualizzata
EliminaNotifica	Questa Servlet viene chiamata nel momento in cui un Broker, un Armatore o un Cliente vuole eliminare una propria Notifica presente nel Sistema. La servlet si occupa di chiamare NotificaDao per poter eliminare dal database la Notifica
RicercaUtenti	Questa Servlet viene chiamata nel momento in cui un

	Broker ricerca un profilo di un Utente. La servlet si occupa di verificare la correttezza dei dati e di inviarli a UtenteDao per poter ottenere i dati relativi alla ricerca

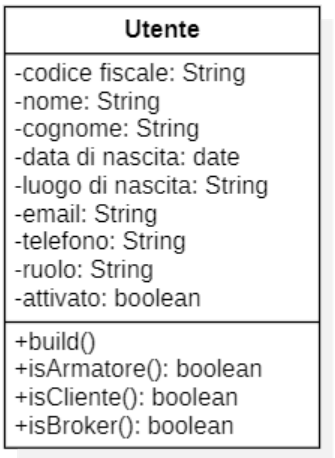
Model	
Utente	Questa classe contiene il builder di Utente e i metodi isArmatore, isCliente, isBroker
UtenteDao	Questa classe contiene i metodi per poter effettuare inserimento, modifica, login, ricerca ed eliminazione di un Utente
CompagniaBroker	Questa classe contiene il builder di CompagniaBroker
CompagniaBrokerDao	Questa classe contiene i metodi per poter effettuare inserimento, modifica, eliminazione di una compagnia di Broker
Richiesta	Questa classe contiene il builder di Richiesta e il metodo getDocumento
RichiestaDao	Questa classe contiene i metodi per poter effettuare inserimento, modifica, ricerca ed eliminazione di una Richiesta
Porto	Questa classe contiene il builder di Porto
PortoDao	Questa classe contiene i metodi per effettuare inserimento, eliminazione, modifica, ricerca di un Porto
Imbarcazione	Questa classe contiene il builder di Imbarcazione e il metodo getDocumento
ImbarcazioneDao	Questa classe contiene i metodi per poter effettuare inserimento, modifica, ricerca ed eliminazione di un'Imbarcazione
Area	Questa classe contiene il builder di Area
AreaDao	Questa classe contiene i metodi per effettuare inserimento, eliminazione, modifica, ricerca di un'Area
Mediazione	Questa classe contiene il builder di Mediazione e il metodo getDocumento
MediazioneDao	Questa classe contiene i metodi per poter effettuare creazione, finalizzazione, operazioni di modifica, ricerca, ed operazioni di eliminazione di una Mediazione
Notifica	Questa classe contiene il builder di Notifica
NotificaDao	Questa classe contiene i metodi per poter effettuare inserimento, ricerca, ed eliminazione di un'Imbarcazione

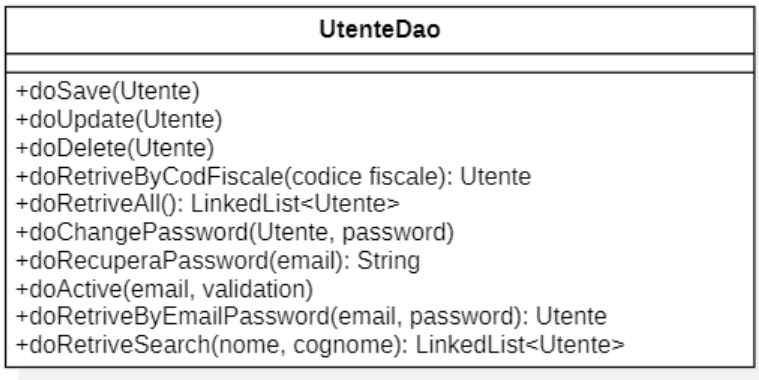
View	
Register	Pagina che permette all'utente non registrato di poter effettuare la registrazione
Login	Pagina che permette all'utente di effettuare il login o di andare alla pagina per recuperare la password
Password	Pagina che permette il recupero della password
Index	Pagina principale dell'utente loggato, permette di visualizzare le notifiche, la lista delle mediazioni e di effettuare il logout. Per il Cliente vi è anche la possibilità di aggiungere una richiesta e di poter andare alla pagina per visualizzarla. Per l'Armatore vi è anche la possibilità di aggiungere un'imbarcazione, e di poter andare alla pagina per visualizzarla. Per il Broker vi è anche la possibilità di poter creare una mediazione, di poter andare alla pagina che permette ricercare un profilo o pagina che permette di ricercare un'imbarcazione/richiesta Per l'utente non loggato vi è un redirect verso la pagina di Login
Profilo	Pagina che permette di modificare i dati del proprio profilo e di disattivare il profilo
Richiesta	Pagina che permette al Cliente di poter visualizzare o modificare una propria Richiesta all'interno del Sistema. Per il Broker vi è la possibilità di poter visualizzare i dati della richiesta e di poterla aggiungere ad una mediazione da lui creata e di poter andare alla pagina per poter visualizzare il profilo del proprietario della richiesta
Imbarcazione	Pagina che permette all'Armatore di poter visualizzare o modificare una propria Imbarcazione all'interno del Sistema. Per il Broker vi è la possibilità di poter visualizzare i dati dell'imbarcazione e di poterla aggiungere ad una mediazione da lui creata e di poter andare alla pagina per poter visualizzare il profilo del proprietario dell'imbarcazione
Mediazione	Pagina che permette al Broker di poter visualizzare, modificare, finalizzare o terminare una propria mediazione all'interno del Sistema. Da qui può inoltre andare alla pagina per poter visualizzare un'imbarcazione o una richiesta. Per Cliente e Armatore vi è la possibilità di poter visualizzare i dati della mediazione e di poter

	accettare/rifiutare la mediazione e la terminazione di quest'ultima.
Notifica	Pagina che permette di visualizzare le informazioni di una notifica
Ricerca	Pagina che permette a Broker di ricercare Imbarcazioni e Richieste e permette di andare alla pagina per visualizzare l'Imbarcazione o la Richiesta scelta

3 - CLASS INTERFACE GLOSSARY

3.1 UTENTE





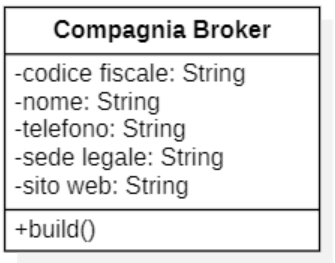
Utente		
Metodo	**Precondizione**	**Post Condizione**
Utente.builder().codFiscale(codFiscale) .nome(nome).cognome(cognome) .dataNascita(data di nascita).luogoNascita(luogoDiNascita) .email(email) .telefono(telefono).attivato(false) .ruolo(ruolo).build()	codice.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$) nome.matches(^[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9 _]{2,50}$) cognome.matches(^[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9 _]{2,50}$) DataAttuale – data >=18 luogo.matches(^[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9 _]{2,50}$) email.matches(^\w+([\.-]?\w+)*@\w+([\.-	L'utente è stato creato

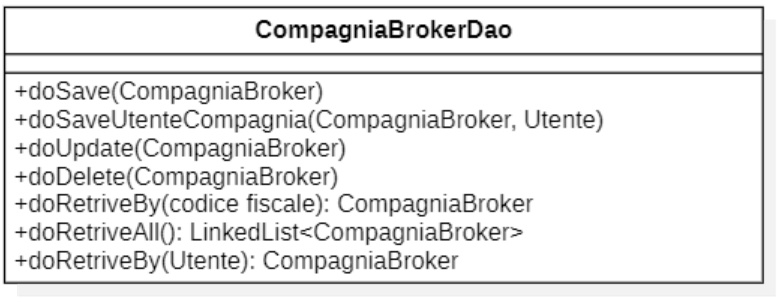
	]?\w+)*(\.\w+)+$) telefono.matches(^[0-9]{10}$) ruolo.equals("Armatore") || ruolo.equals("Broker) || ruolo.equals("Cliente")	

UtenteDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(Utente)	Si rimanda a: Precondizioni Utente	L'Utente è stato salvato nel database
doUpdate(Utente)	Si rimanda a: Precondizioni Utente	L'Utente è stato aggiornato nel database
doDelete(Utente)	Si rimanda a: Precondizioni Utente	L'Utente è stato disattivato
doRetriveByCodFiscale(codice fiscale)	codicefiscale.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$)	Viene restituito l'utente al quale corrisponde il codice fiscale
doRetriveAll()		Viene restituita la lista di tutti gli utenti presenti nel database
doChangePassword(Utente, password)	Per Utente si rimanda a: Precondizioni Utente password.matches(^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{8,}$)	La Password dell'utente viene modificata nel database
doRecuperaPassword(email)	email.matches(^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w+)+$)	Viene cambiata la password dell'utente al quale corrisponde la mail inserita, e viene restituita la password modificata
doActive(email,validation)	email.matches(^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w+)+$)	Il Profilo viene attivato
doRetriveByEmailPassword(email,password)	email.matches(^\w+([\.-	Viene restituito

	]?\w+)*@\w+([\.-]?\w+)*(\.\w+)+$) password.matches(^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{8,}$)	l'utente a cui corrisponde e-mail e password
doRetriveSearch(nome,congome)	nome.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9_]{2,50}$) cognome.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9_]{2,50}$)	Viene restituita la lista di utenti che hanno nome e cognome corrispondente ai parametri inseriti

3.2 COMPAGNIA BROKER



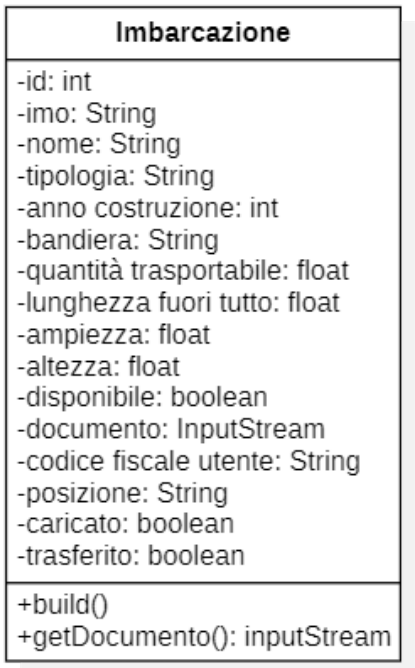


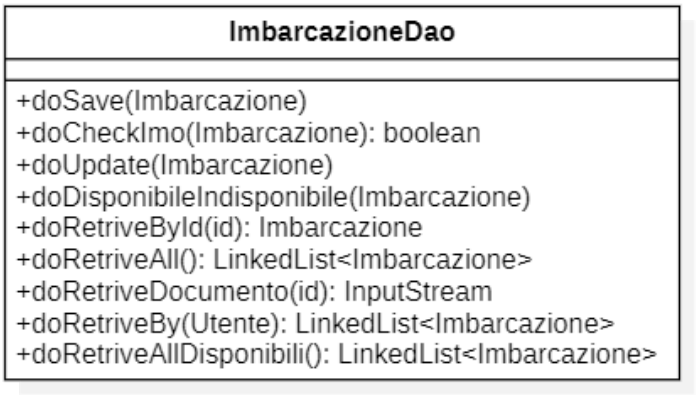
Compagnia Broker		
Metodo	**Precondizione**	**Post Condizione**
CompagniaBroker. Builder.nome(nome). Codicefiscale(codice).telefono(telefono). SedeLegale(sede).sitoWeb(sito).build()	codicefiscale.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$) nome.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9_]{2,50}$) telefono.matches(^[0-9]{10}$) sede.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9_]{2,50}$) sito.matches(^(http:\/\/www\.|https:\/\/www\.|http:\/\/|https:\/\/)?[a-z0-9]+([\-\.]	La Compagnia Broker è stata creata

	{1}[a-z0-9]+)*\.[a-z]{2,5}(:[0-9]{1,5})?(\/.*)?$))	

CompagniaBrokerDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(CompagniaBroker)	Si rimanda a: Precondizioni Compagnia Broker	La Compagnia Broker è stata salvata nel database
doUpdate(CompagniaBroker)	Si rimanda a: Precondizioni Compagnia Broker	La Compagnia Broker è stata aggiornata nel database
doDelete(CompagniaBroker)	Si rimanda a: Precondizioni Compagnia Broker	La Compagnia Broker è stata eliminata dal database
doRetriveBy(codice fiscale)	codicefiscale.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$)	Viene restituita la Compagnia Broker alla quale corrisponde il codice fiscale inserito
doRetriveAll()		Viene restituita la lista di tutte le Compagnia Broker presenti nel database
doRetriveBy (Utente)	Si rimanda a: Precondizioni Utente	Viene restituita la Compagnia Broker associata all'utente inserito

3.3 IMBARCAZIONE





Imbarcazione		
Metodo	**Precondizione**	**Post Condizione**
Imbarcazione.builder().imo(imo). Nome(nome).tipologia(tipologia). annoCostruzione(anno costruzione). Bandiera(bandiera).quantità(quantità).lunghezza(lunghezza). Ampiezza(ampiezza).altezza(altezza).disponibile(disponibile). Documento(documento). codiceFiscaleUtente(codice fiscale utente).build()	imo.matches(^[A-Za-z0-9]{7}$) nome.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9_]{2,50}$) tipologia.equals(“Portacontainer”) || tipologia.equals(“Carboniera”) || tipologia.equals(“Chimichiera”) || tipologia.equals(“Lift-on/lift-off”) || tipologia.equals(“Nave da carico”) || tipologia.equals(“Nave frigorifera”) || tipologia.equals(“Portarinfuse”) || tipologia.equals(“Roll-on/Roll-off”) AnnoAttuale>anno costruzione bandiera.equals(AF) || bandiera.equals(AL) || bandiera.equals(DZ) || bandiera.equals(AD) || bandiera.equals(AO) || bandiera.equals(AI) || bandiera.equals(AQ) ||	L’imbarcazione è stata creata

	bandiera.equals(AG) || bandiera.equals(AN) || bandiera.equals(SA) || bandiera.equals(AR) || bandiera.equals(AM) || bandiera.equals(AW) || bandiera.equals(AU) || bandiera.equals(AT) || bandiera.equals(AZ) || bandiera.equals(BS) || bandiera.equals(BH) || bandiera.equals(BD) || bandiera.equals(BB) || bandiera.equals(BE) || bandiera.equals(BZ) || bandiera.equals(BJ) || bandiera.equals(BM) || bandiera.equals(BY) || bandiera.equals(BT) || bandiera.equals(BO) || bandiera.equals(BA) || bandiera.equals(BW) || bandiera.equals(BR) || bandiera.equals(BN) || bandiera.equals(BG) || bandiera.equals(BF) || bandiera.equals(BI) || bandiera.equals(KH) || bandiera.equals(CM) || bandiera.equals(CA) || bandiera.equals(CV) || bandiera.equals(TD) || bandiera.equals(CL) || bandiera.equals(CN) || bandiera.equals(CY) || bandiera.equals(VA) || bandiera.equals(CO) || bandiera.equals(KM) || bandiera.equals(KP) || bandiera.equals(KR) || bandiera.equals(CR) || bandiera.equals(CI) || bandiera.equals(HR) || bandiera.equals(CU) || bandiera.equals(DK) || bandiera.equals(DM) || bandiera.equals(EC) ||	

	bandiera.equals(EG) || bandiera.equals(IE) || bandiera.equals(SV) || bandiera.equals(AE) || bandiera.equals(ER) || bandiera.equals(EE) || bandiera.equals(ET) || bandiera.equals(RU) || bandiera.equals(FJ) || bandiera.equals(PH) || bandiera.equals(FI) || bandiera.equals(FR) || bandiera.equals(GA) || bandiera.equals(GM) || bandiera.equals(GE) || bandiera.equals(DE) || bandiera.equals(GH) || bandiera.equals(JM) || bandiera.equals(JP) || bandiera.equals(GI) || bandiera.equals(DJ) || bandiera.equals(JO) || bandiera.equals(GR) || bandiera.equals(GD) || bandiera.equals(GL) || bandiera.equals(GP) || bandiera.equals(GU) || bandiera.equals(GT) || bandiera.equals(GN) || bandiera.equals(GW) || bandiera.equals(GQ) || bandiera.equals(GY) || bandiera.equals(GF) || bandiera.equals(HT) || bandiera.equals(HN) || bandiera.equals(HK) || bandiera.equals(IN) || bandiera.equals(ID) || bandiera.equals(IR) || bandiera.equals(IQ) || bandiera.equals(BV) || bandiera.equals(CX) || bandiera.equals(HM) || bandiera.equals(KY) || bandiera.equals(CC) || bandiera.equals(CK) || bandiera.equals(FK) ||	

	bandiera.equals(FO) || bandiera.equals(MH) || bandiera.equals(MP) || bandiera.equals(UM) || bandiera.equals(NF) || bandiera.equals(SB) || bandiera.equals(TC) || bandiera.equals(VI) || bandiera.equals(VG) || bandiera.equals(IL) || bandiera.equals(IS) || bandiera.equals(IT) || bandiera.equals(KZ) || bandiera.equals(KE) || bandiera.equals(KG) || bandiera.equals(KI) || bandiera.equals(KW) || bandiera.equals(LA) || bandiera.equals(LV) || bandiera.equals(LS) || bandiera.equals(LB) || bandiera.equals(LR) || bandiera.equals(LY) || bandiera.equals(LI) || bandiera.equals(LT) || bandiera.equals(LU) || bandiera.equals(MO) || bandiera.equals(MK) || bandiera.equals(MG) || bandiera.equals(MW) || bandiera.equals(MV) || bandiera.equals(MY) || bandiera.equals(ML) || bandiera.equals(MT) || bandiera.equals(MA) || bandiera.equals(MQ) || bandiera.equals(MR) || bandiera.equals(MU) || bandiera.equals(YT) || bandiera.equals(MX) || bandiera.equals(MD) || bandiera.equals(MC) || bandiera.equals(MN) || bandiera.equals(MS) || bandiera.equals(MZ) || bandiera.equals(MM) || bandiera.equals(NA) ||	

	bandiera.equals(NR) || bandiera.equals(NP) || bandiera.equals(NI) || bandiera.equals(NE) || bandiera.equals(NG) || bandiera.equals(NU) || bandiera.equals(NO) || bandiera.equals(NC) || bandiera.equals(NZ) || bandiera.equals(OM) || bandiera.equals(NL) || bandiera.equals(PK) || bandiera.equals(PW) || bandiera.equals(PA) || bandiera.equals(PG) || bandiera.equals(PY) || bandiera.equals(PE) || bandiera.equals(PN) || bandiera.equals(PF) || bandiera.equals(PL) || bandiera.equals(PT) || bandiera.equals(PR) || bandiera.equals(QA) || bandiera.equals(GB) || bandiera.equals(CZ) || bandiera.equals(CF) || bandiera.equals(CG) || bandiera.equals(CD) || bandiera.equals(DO) || bandiera.equals(RE) || bandiera.equals(RO) || bandiera.equals(RW) || bandiera.equals(EH) || bandiera.equals(KN) || bandiera.equals(PM) || bandiera.equals(VC) || bandiera.equals(WS) || bandiera.equals(AS) || bandiera.equals(SM) || bandiera.equals(SH) || bandiera.equals(LC) || bandiera.equals(ST) || bandiera.equals(SN) || bandiera.equals(XK) || bandiera.equals(SC) || bandiera.equals(SL) || bandiera.equals(SG) ||	

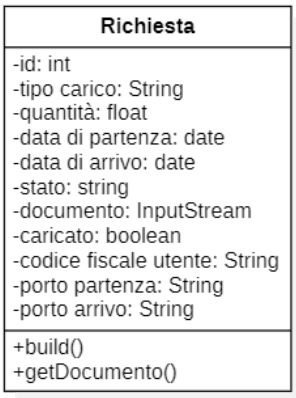
	bandiera.equals(SY) || bandiera.equals(SK) || bandiera.equals(SI) || bandiera.equals(SO) || bandiera.equals(ES) || bandiera.equals(LK) || bandiera.equals(FM) || bandiera.equals(US) || bandiera.equals(ZA) || bandiera.equals(GS) || bandiera.equals(SD) || bandiera.equals(SR) || bandiera.equals(SJ) || bandiera.equals(SE) || bandiera.equals(CH) || bandiera.equals(SZ) || bandiera.equals(TJ) || bandiera.equals(TH) || bandiera.equals(TW) || bandiera.equals(TZ) || bandiera.equals(IO) || bandiera.equals(TF) || bandiera.equals(PS) || bandiera.equals(TL) || bandiera.equals(TG) || bandiera.equals(TK) || bandiera.equals(TO) || bandiera.equals(TT) || bandiera.equals(TN) || bandiera.equals(TR) || bandiera.equals(TM) || bandiera.equals(TV) || bandiera.equals(UA) || bandiera.equals(UG) || bandiera.equals(HU) || bandiera.equals(UY) || bandiera.equals(UZ) || bandiera.equals(VU) || bandiera.equals(VE) || bandiera.equals(VN) || bandiera.equals(WF) || bandiera.equals(YE) || bandiera.equals(ZM) || bandiera.equals(ZW) || bandiera.equals(RS) || bandiera.equals(ME) || bandiera.equals(TP) ||	

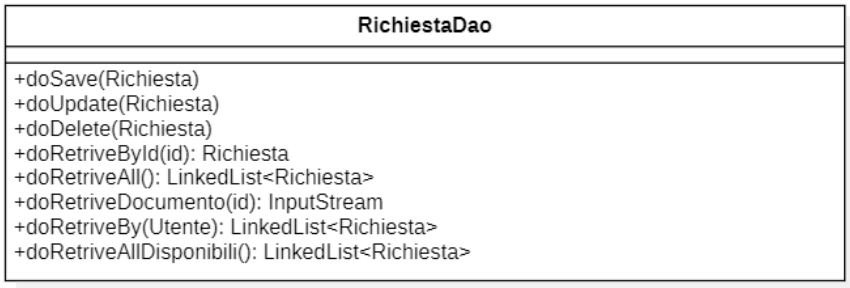
	bandiera.equals(GG) 0<quantità<2147483647 && quantità.matches(^[0-9]+$) 0<lunghezza<2147483647 && lunghezza.matches(^[0-9]+$) 0<ampiezza<2147483647 && ampiezza.matches(^[0-9]+$) disponibile.equals(“true”) \| \| disponibile.equals(“false”) documento.matches(^[A-Za-z]+\.(pdf)$) 0<documento.lenght<4294967295 codicefiscaleUtente.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$)	

ImbarcazioneDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(Imbarcazione)	Si rimanda a: Precondizioni Imbarcazione	L’Imbarcazione è stata salvata nel database
doCheckImo(Imbarcazione)	Si rimanda a: Precondizioni Imbarcazione	Viene restituita l’imbarcazione a cui corrisponde l’imo presente nell’imbarcazione inserita
doUpdate(Imbarcazione)	Si rimanda a: Precondizioni Imbarcazione	L’Imbarcazione è stata aggiornata nel database
doDisponibileIndisponibile(Imbarcazione)	Imbarcazione.disponibile = = true | | Imbarcazione.disponibile = =false	La disponibilità dell’imbarcazione viene aggiornata nel database
doRetriveById(id)	id.matches(^[0-9]+$)	Viene restituita l’imbarcazione a cui corrisponde l’id inserito

doRetriveAll()		Viene restituita la lista di tutte le imbarcazioni presenti nel database
doRetriveDocumento(id)	id.matches(^[0-9]+$)	Viene restituito il documento dell'imbarcazione a cui è associato l'id inserito
doRetrivebyUtente(Utente)	Si rimanda a: Precondizioni Utente	Viene restituito la lista delle imbarcazioni associate all'utente inserito
doRetriveAllDisponibili()		Viene restituito la lista delle imbarcazioni che sono disponibili

3.4 RICHIESTA





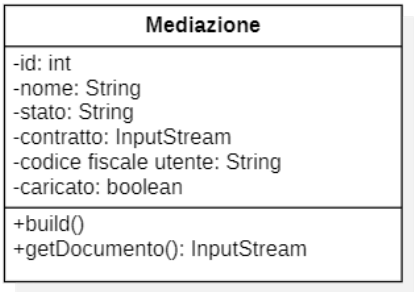
Richiesta		
Metodo	**Precondizione**	**Post Condizione**
Richiesta.builder().tipoCarico(carico). Quantità(quantità).dataPartenza(data partenza).dataArrivo(data arrivo). Stato(stato).documento(documento). Caricato(caricato).codiceFiscaleUtente(codice fiscale utente).portoPartenza(porto partenza).portoArrivo(porto arrivo).build()	carico.equals(container) | | carico.equals(carico alla rinfusa) | | carico.equals(prodotti chimici solidi) | | carico.equals(prodotti chimici liquidi) | | carico.equals(prodotti chimici gassosi) | | carico.equals(prodotti alimentari) | | carico.equals(autoveicoli)	La Richiesta è stata creata

	quantità>1 && quantità.matches(^[0-9]+$) data partenza>data attuale && data arrivo > data partenza stato.equals(Disponibile) || stato.equals(In lavorazione) || stato.equals(Terminata) documento.matches(^[A-Za-z]+\.(pdf)$) 0<documento.lenght<4294967295 Caricato = = true || Caricato = = false codicefiscaleUtente.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$) porto partenza != porto arrivo	

RichiestaDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(Richiesta)	Si rimanda a: Precondizioni Richiesta	La Richiesta è stata salvata nel database
doUpdate(Richiesta)	Si rimanda a: Precondizioni Richiesta	La Richiesta è stata aggiornata nel database
doDelete(Richiesta)	Si rimanda a: Precondizioni Richiesta	La Richiesta è stata eliminata dal database
doRetriveById(id)	id.matches(^[0-9]+$)	Viene restituita la Richiesta alla quale corrisponde l'id inserito
doRetriveAll()		Viene restituita la lista di tutte le Richieste presenti nel database
doRetriveDocumento(id)	id.matches(^[0-9]+$)	Viene restituito il documento della Richiesta alla quale corrisponde l'id inserito

doRetriveBy(Utente)	Si rimanda a: Precondizioni Utente	Viene restituita la lista delle Richieste associate all'utente inserito
doRetriveAllDisponibili()		Viene restituita la lista delle Richieste con stato "Disponibile"

3.5 MEDIAZIONE



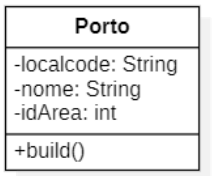
Mediazione		
Metodo	**Precondizione**	**Post Condizione**
Mediazione.builder().nome(nome).Stato(stato).contratto(contratto).codiceFiscaleUtente(codice fiscale utente).caricato(caricato).build()	nome.matches(^[A-Za-z0-9 _]*[A-Za-z0-9][A-Za-z0-9 _]{2,50}$) stato.equals(Default) \|\| stato.equals(In corso) \|\| stato.equals(Richiesta Modifica) \|\| stato.equals(Richiesta Terminazione) \|\| stato.equals(In Attesa di Firma) \|\| stato.equals(Terminata) contratto.matches(^[A-Za-z]+\.(pdf)$) && 0<contratto.lenght<4294967295	La Mediazione è stata creata

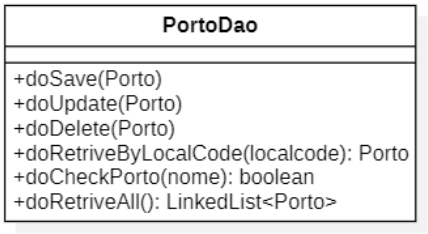
	codicefiscaleUtente.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$) caricato = = true || caricato = = false	

MediazioneDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(Mediazione)	Si rimanda a: Precondizioni Mediazione	La Mediazione è stata salvata nel database
doUpdate(Mediazione)	Si rimanda a: Precondizioni Mediazione	La Mediazione è stata aggiornata nel database
doDelete(Mediazione)	Si rimanda a: Precondizioni Mediazione	La Mediazione è stata eliminata dal database
doRetriveById(id)	id.matches(^[0-9]+$)	Viene restituita la Mediazione alla quale è associato l'id inserito
doRetriveAll()		Viene restituita la lista delle mediazioni presenti nel database
doRetriveDocumento(id)	id.matches(^[0-9]+$)	Viene restituito il documento
doRetriveBy(Utente)	Si rimanda a: Precondizioni Utente	Viene restituita la lista delle mediazioni associata all'utente inserito
doRetriveRichiesteFrom(Mediazione)	Si rimanda a: Precondizioni Mediazione	Viene restituita la lista delle Richieste presenti nella Mediazione inserita
doRetriveImbarcazioniFrom(Mediazione)	Si rimanda a: Precondizioni Mediazione	Viene restituita la lista delle Imbarcazioni presenti nella Mediazione inserita
doSaveImbarcazioneMediazione (idMediazione,idImbarcazione)	idMediazione.matches(^[0-9]+$) idImbarcazione.matches(^[0-9]+$)	Il Riferimento dell'imbarcazione

		presente nella mediazione è salvato nel database
doSaveRichiesteMediazione (idMediazione,idRichieste)	idMediazione.matches(^[0-9]+$) idRichieste.matches(^[0-9]+$)	Il Riferimento della richiesta presente nella mediazione è salvato nel database
doDeleteRichiestaMediazione (idMediazione, idRichiesta)	idMediazione.matches(^[0-9]+$) idRichiesta.matches(^[0-9]+$)	Il Riferimento della Richiesta presente nella mediazione è stato eliminato dal database
doDeleteImbarcazioneMediazione (idMediazione, idImbarcazione)	idMediazione.matches(^[0-9]+$) idImbarcazione.matches(^[0-9]+$)	Il Riferimento dell'Imbarcazione presente nella mediazione è stato eliminato dal database
doRetriveFirme(Mediazione)	Si rimanda a: Precondizioni Mediazione	Viene restituita la lista delle firme presenti nella mediazione inserita
doSaveFirma(Mediazione,codice fiscale)	Si rimanda a: Precondizioni Mediazione codicefiscale.matches(^[A-Z]{6}[A-Z0-9]{2}[A-Z][A-Z0-9]{2}[A-Z][A-Z0-9]{3}[A-Z]$)	Il codice fiscale dell'utente che ha firmato la mediazione viene salvato nel database
doRifiutaFirma(Mediazione)	Si rimanda a: Precondizioni Mediazione	Le firme vengono eliminate dal database e lo stato della mediazione viene aggiornato
doCheck(Imbarcazione)	Si rimanda a: Precondizioni Imbarcazione	Viene restituito un boolean che rappresenta la presenza o meno in una mediazione di stato "Default" dell'imbarcazione inserita

3.6 PORTO

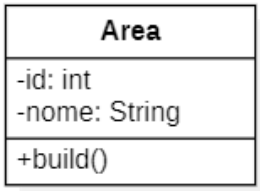


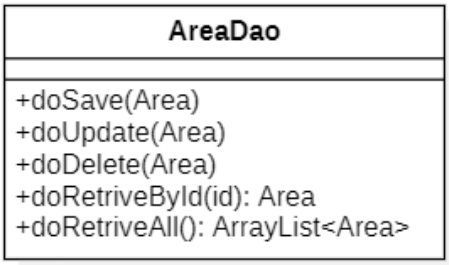


Porto		
Metodo	**Precondizione**	**Post Condizione**
Porto.builder().localcode(localcode).Nome(nome).idArea(idArea).build()	localcode.length<6 && localcode.matches(^[A-Za-z0-9]+$) nome.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9_]{2,50}$) idArea.matches(^[0-9]+$)	Il Porto è stato creato

PortoDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(Porto)	Si rimanda a: Precondizioni Porto	Il Porto è stato salvato nel database
doUpdate(Porto)	Si rimanda a: Precondizioni Porto	Il Porto è stato aggiornato nel database
doDelete(Porto)	Si rimanda a: Precondizioni Porto	Il Porto è stato eliminato dal database
doRetriveByLocalcode(localcode)	localcode.length<6 && localcode.matches(^[A-Za-z0-9]+$)	Viene restituito il porto al quale corrisponde il localcode inserito
doCheckPorto(nome)	nome.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9_]{2,50}$)	Viene restituito un boolean che indica se il nome inserito corrisponde ad un porto presente nel database
doRetriveAll()		Viene restituita la lista dei Porti presenti nel database

3.7 AREA

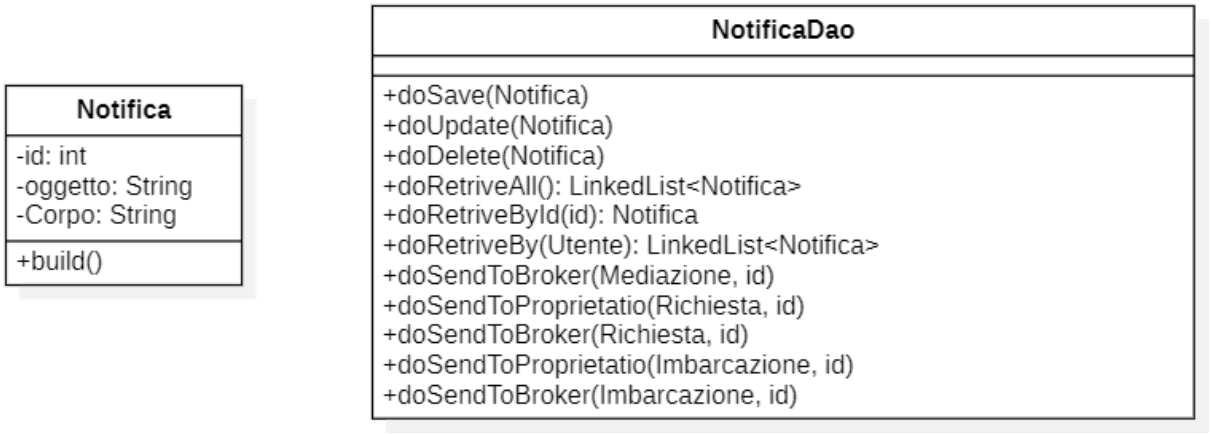




Area		
Metodo	**Precondizione**	**Post Condizione**
Area.builder().id(id).nome(nome).Build()	Id>0 && id.matches(^[0-9]+$) nome.matches(^[A-Za-z0-9_]*[A-Za-z0-9][A-Za-z0-9]{2,10}$)	L'Area è stata creata

AreaDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(Area)	Si rimanda a: Precondizioni Area	L'Area è stata salvata nel database
doUpdate(Area)	Si rimanda a: Precondizioni Area	L'Area è stata aggiornata nel database
doDelete(Area)	Si rimanda a: Precondizioni Area	L'Area è stata eliminata dal database
doRetriveById(id)	Id>0 && id.matches(^[0-9]+$)	Viene restituita l'Area alla quale corrisponde l'id inserito
doRetriveAll()		Viene restituita la lista delle Aree presenti nel database

3.8 NOTIFICA



Notifica		
Metodo	**Precondizione**	**Post Condizione**
Notifica.builder().oggetto(oggetto).Corpo(corpo).build()	oggetto.length<50 corpo.length<5000	La Notifica è stata creata

NotificaDao		
Metodo	**Precondizione**	**Post Condizione**
doSave(Notifica)	Si rimanda a: Precondizioni Notifica	La Notifica è stata salvata nel database
doUpdate(Notifica)	Si rimanda a: Precondizioni Notifica	La Notifica è stata aggiornata nel database
doDelete(Notifica)	Si rimanda a: Precondizioni Notifica	La Notifica è stata eliminata dal database
doRetriveAll()		Viene restituita la lista delle Notifiche presenti nel database
doRetriveById(id)	id.matches(^[0-9]+$)	Viene restituita la Notifica alla quale è associato l'id inserito
doRetriveBy(Utente)	Si rimanda a: Precondizioni Utente	Viene restituita la lista delle Notifiche associate all'utente inserito
doSendToBroker(Mediazione,id)	Si rimanda a: Precondizioni Mediazione id.matches(^[0-9]+$)	Il Riferimento della notifica e dell'utente al quale mandare la

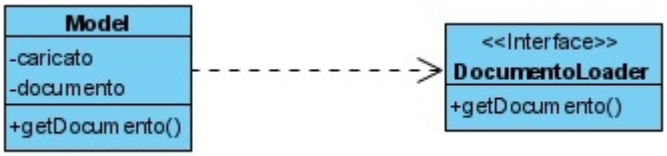
		notifica viene salvato nel database
doSendToProprietario(Richiesta,id)	Si rimanda a: Precondizioni Richiesta id.matches(^[0-9]+$)	Il Riferimento della notifica e dell'utente al quale mandare la notifica viene salvato nel database
doSendToBroker(Richiesta,id)	Si rimanda a: Precondizioni Richiesta id.matches(^[0-9]+$)	Il Riferimento della notifica e dell'utente al quale mandare la notifica viene salvato nel database
doSendToProprietario(Imbarcazione,id)	Si rimanda a: Precondizioni Imbarcazione id.matches(^[0-9]+$)	Il Riferimento della notifica e dell'utente al quale mandare la notifica viene salvato nel database
doSendToBroker(Imbarcazione,id)	Si rimanda a: Precondizioni Imbarcazione id.matches(^[0-9]+$)	Il Riferimento della notifica e dell'utente al quale mandare la notifica viene salvato nel database

4 – DESIGN PATTERN

4.1 PROXY PATTERN

Descrizione del Problema e Soluzione:

All'interno del sistema vengono salvate delle informazioni come, ad esempio, i "Documento" le cui informazioni non sempre sono immediatamente necessarie. Risulta quindi controproducente recuperare immediatamente i file di "grandi" dimensioni, producendo overhead e "rubando" risorse utili ad altre operazioni, se quest'ultimi il più delle volte non vengono consultati. Per risolvere questo problema si è pensato di implementare un Proxy Pattern in particolare il Lazy Loading. Ogni oggetto model avrà un attributo "documento" di tipo InputStream e un attributo "caricato" di tipo boolean, quando si procede a recuperare l'istanza del model dal DBMS, "documento" di default sarà null e "caricato" viene impostato in base alla presenza o meno del documento nel database. Quando sarà necessario l'attributo documento, poiché ogni model che contiene questi due attributi estende l'interfaccia DocumentoLoader, si utilizzerà il metodo getDocumento() che si preoccuperà di caricare in attributo il documento presente nel database ed impostare caricato a true. Qualora fosse necessario l'utilizzo del documento quest'ultimo non verrà richiesto nuovamente al DBMS, ma il model restituirà quello presente in memoria.



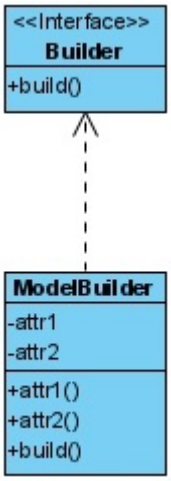
Conseguenze:

- Ottimizzazione delle chiamate al DBMS e della memoria del server.

4.2 BUILDER PATTERN

Descrizione del Problema e Soluzione:

Quando un Model comincia ad avere molte variabili, la creazione di un oggetto tramite il builder inizia a diventare complessa ed è facile dimenticare l'ordine dei parametri con cui costruire l'oggetto. Il design pattern Builder separa la costruzione di un oggetto complesso dalla sua rappresentazione permettendo inoltre di facilitare la validazione della classe centralizzando in un unico metodo i controlli migliorando la manutenzione futura.
Ogni classe Model avrà un suo ModelBuilder, una classe che estende l'interfaccia Builder, che conterrà tutti gli attributi del Model con i relativi metodi per settare gli attributi. Ogni ModelBuilder avrà il metodo build() che oltre a restituire un'istanza del Model con i valori salvati si occuperà anche delle validazioni necessarie alla realizzazione di un oggetto valido. Il Model non conterrà un builder pubblico per evitare di creare istanze di oggetti senza che quest'ultime siano state validate dal Modelbuilder.



Conseguenze:

- Aumenta il codice da scrivere per la realizzazione dei model.
- Migliora la manutenibilità del codice e la possibilità di individuare errori.

5 – PRIORITA' DI SVILUPPO

In questa sezione vengono presentate le priorità per le funzionalità presenti all'interno del sistema.

UTENTE	
FUNZIONALITA'	**PRIORITA'**
Registrazione	High Priority
Login	High Priority
Modifica Profilo	High Priority
Logout	High Priority
Recupero Password	High Priority
Attivazione Profilo	High Priority
Disattiva Profilo	High Priority
Visualizza Profilo	High Priority

IMBARCAZIONE	
FUNZIONALITA'	**PRIORITA'**
Inserimento Imbarcazione	High Priority
Modifica Imbarcazione	Medium Priority
Elimina Imbarcazione	High Priority
Visualizza Imbarcazione	High Priority

RICHIESTA	
FUNZIONALITA'	**PRIORITA'**
Inserimento Richiesta	High Priority
Modifica Richiesta	Medium Priority
Elimina Richiesta	High Priority
Visualizza Richiesta	High Priority

MEDIAZIONE	
FUNZIONALITA'	**PRIORITA'**
Crea Mediazione	High Priority
Modifica Mediazione	Medium Priority
Elimina Mediazione	High Priority
Visualizza Mediazione	High Priority
Finalizza Mediazione	Low Priority
Aggiungere Imbarcazione alla Mediazione	High Priority
Aggiungere Richiesta alla Mediazione	High Priority
Eliminare Imbarcazione dalla Mediazione	Medium Priority
Eliminare Richiesta dalla Mediazione	Medium Priority
Eliminare Contratto dalla Mediazione	Medium Priority
Firmare/Rifiutare Mediazione	Medium Priority
Termina Mediazione	Low Priority

Accetta Terminazione Mediazione	Low Priority
Rifiuta Terminazione Mediazione	Low Priority

RICERCA	
FUNZIONALITA'	**PRIORITA'**
Ricerca Imbarcazione	High Priority
Ricerca Richiesta	High Priority
Ricerca Profilo	Low Priority

NOTIFICA	
FUNZIONALITA'	**PRIORITA'**
Visualizza Notifica	Medium Priority
Elimina Notifica	Low Priority