SWEeneyThreads

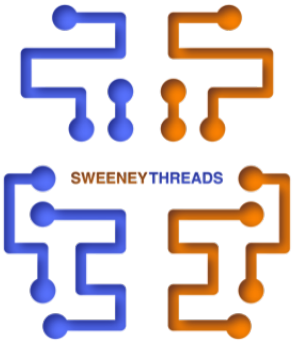
ActorBase

a NoSQL DB based on the Actor model

Specifica Tecnica

Redattori:
Bonato Paolo
Bortolazzo Matteo
Biggeri Mattia
Maino Elia
Nicoletti Luca
Padovan Tommaso
Tommasin Davide

Approvazione:
Maino Elia
Verifica:
Bortolazzo Matteo
Padovan Tommaso



Versione 1.0.0

11 aprile 2016

Indice

Diario delle modifiche

Versione	Data	Autore	Descrizione
1.0.0	2016-04-11	*Responsabile* Maino Elia	Documento approvato
0.3.0	2016-04-11	*Verificatore* Padovan Tommaso	Verificato il documento
0.2.1	2016-04-11	*Progettista* Nicoletti Luca	Riviste sezione da 4.8 a 4.17 e modificata sintassi delle classi correlate ad ogni classe. Sostituite vecchie immagini con immagini aggiornate (inserite aggregazioni tra le classi)
0.2.0	2016-04-11	*Verificatore* Bortolazzo Matteo	Verificato le tabelle inserite in v0.1.2 e il diario delle modifiche
0.1.3	2016-04-10	*Progettista* Nicoletti Luca	Separata tabella del diario della modifiche in file esterno.
0.1.2	2016-04-10	*Progettista* Biggeri Mattia	inserita tabella requisito-componente
0.1.1	2016-04-10	*Progettista* Nicoletti Luca	Sistemazione errori rilevati dalla verifica effettuata in versione 0.0.12
0.1.0	2016-04-10	*Verificatore* Bortolazzo Matteo	Verificato l'intero documento, segnalati vari errori in tutte le sezioni
0.0.11	2016-04-10	*Progettista* Maino Elia	Completata la descrizione di classe e interfacce del componente server
0.0.10	2016-04-10	*Progettista* Nicoletti Luca	Modificate e inserite le nuove immagini dei packages e delle classi per la sezione Server (4.2). Modificate le immagini nei diagrammi di sequenza (Sezione 6) e modificate le descrizioni di tutti in modo da uniformarle. Cambiate le immagini dei Design Pattern (Sezione 9.1)
0.0.9	2016-04-10	*Progettista* Padovan Tommaso	Incremento sezione Componenti e Classi: allineata sezione Driver allo standard, stesura sezione Client.
0.0.8	2016-04-09	*Progettisti* Bonato Paolo Biggeri Mattia	Stesura sezione tracciamento componenti-requisiti.
0.0.7	2016-04-09	*Progettista* Nicoletti Luca	Inseriti tutti i diagrammi di sequenza riguardanti l'implementazione di richieste lato server, sostituita immagine dei Packages generale. Inserita una breve spiegazione di ogni diagramma di sequenza
0.0.6	2016-04-09	*Progettisti* Biggeri Mattia Padovan Tommaso Bonato Paolo	Stesura sezioni Driver, Diagrammi attività, Stime di fattibilità e bisogno di risorse; aggiunta immagine nella sezione 3.2.3, aggiunto Singleton nelle descrizioni dei Design Pattern, aggiornata leggenda.
0.0.5	2016-04-08	*Progettisti* Maino Elia Nicoletti Luca Bortolazzo Matteo	Stesura sezione riguardante le componenti dell'architettura lato server: diagrammi dei package e delle classi e descrizioni testuali
0.0.4	2016-04-06	*Progettisti* Biggeri Mattia Tommasin Davide	Aggiunta sezione in appendice suoi Design Pattern, contiene al momento la descrizione di: MVC, Event-driven, Command
0.0.3	2016-04-03	*Progettista* Bonato Paolo	Accorpate le sezioni "Componenti", "Package" e "Classi" in "Componenti e classi". Riadattata la sezione "Metodo e formalismo di specifica" alla nuova struttura. Inserite le immagini 1 e 2. Apportate le correzioni indicate.

Versione	Data	Autore	Descrizione
0.0.2	2016-03-26	*Progettisti* Bonato Paolo Biggeri Mattia Padovan Tommaso Tommasin Davide Bortolazzo Matteo	Prima stesura di Architettura generale (sezione 3) e componenti (sezione 4)
0.0.1	2016-03-24	*Analisti* Bonato Paolo Biggeri Mattia	Creazione scheletro documento, stesura introduzione, definizione di metodo e formalismo di specifica.

Tabella 1: Diario delle modifiche

1 Introduzione

1.1 Scopo del documento

Il documento definisce la progettazione ad alto livello del progetto Actorbase. Verrà presentata l'architettura generale, le componenti, le classi e i design pattern$_G$ utilizzati per realizzare il prodotto.

1.2 Scopo del prodotto

Il progetto consiste nella realizzazione di un Database NoSQL key-value basato sul modello ad Attori con l'obiettivo di fornire una tecnologia adatta allo sviluppo di moderne applicazioni che richiedono brevissimi tempi di risposta e che elaborano enormi quantità di dati. Lo sviluppo porterà al rilascio del software sotto licenza MIT.

1.3 Glossario

Al fine di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario v2.0.0*. In esso saranno definiti, in modo chiaro e conciso i termini che possono causare ambiguità o incomprensione del testo.

1.4 Riferimenti

- **Slide dell'insegnamento Ingegneria del software$_G$ mod.A:**
  `http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/E02.pdf`
- **Scala:**
  `http://www.scala-lang.org/`
- **Java:**
  `http://www.java.com/`
- **Akka:**
  `http://akka.io/`
- **IntelliJ:**
  `http://www.jetbrains.com/idea/`

1.4.1 Normativi

- **Norme di progetto:** *Norme di progetto v2.0.0*
- **Capitolato d'appalto Actorbase (C1):**
  `http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C1p.pdf`

2 Tecnologie utilizzate

2.1 Scala

Le possibili scelte dettate dal capitolato sono Java e Scala$_G$. Si è scelto di utilizzare Scala perché offre i seguenti vantaggi:

- **Completamente Object-Oriented:** A differenza di Java, Scala$_G$ è completamente orientato agli oggetti. Non c'è distinzione del tipo: oggetto - tipo primitivo, ogni valore è semplicemente un oggetto.
- **Staticamente tipato:** È un linguaggio tipato staticamente, questo permette di effettuare più facilmente i test. Inoltre Scala$_G$ è in grado di stabilire il tipo di un oggetto per inferenza.
- **Può eseguire codice Java:** Scala$_G$ può eseguire codice scritto in Java. È dunque possibile utilizzare classi e librerie scritte in Java all'interno di programmi scritti in Scala.
- **Concorrenza e distribuzione:** Ottimo supporto alla programmazione multi-threaded e distribuita, essenziale per la realizzazione di un prodotto responsive e scalabile.
- **Supporto alla definizione di DSL:** Scala$_G$ supporta nativamente la definizione di DSL.
- **Supporto di Akka$_G$:** Il linguaggio supporta la libreria Akka che è richiesta dal capitolato.

Inoltre il Committente ha espresso esplicitamente la sua preferenza sull'utilizzo di Scala$_G$.

Figura 1: Scala - logo

2.2 Akka

L'utilizzo della libreria Akka$_G$ oltre ad essere reso obbligatorio dal committente, fornisce un'eccellente base su cui sviluppare un sistema basato sul modello ad attori. Akka$_G$ permette di costruire facilmente applicazioni message-driven che siano estremamente concorrenti, distribuite e resilienti. La natura distribuita e asincrona degli attori messi a disposizione da Akka$_G$ soddisfa pienamente i bisogni del sistema da implementare.

Figura 2: Akka - logo

3 Descrizione dell'architettura

3.1 Metodo e formalismo di specifica

Nell'esposizione dell'architettura del prodotto si procederà con un approccio di tipo top-down, ovvero dal generale al particolare.
Inizialmente si descriveranno le tre componenti fondamentali: Client, Server e Driver; poi le componenti più piccole al loro interno, specificando i package e le classi che li compongono.

Per ogni package saranno descritti brevemente il tipo, l'obiettivo e la funzione e saranno specificati eventuali figli, classi ed interazioni con altri package. Ogni classe sarà dotata di una breve descrizione e ne saranno specificate le responsabilità, le classi ereditate, le sottoclassi e le relazioni con altre classi.
Successivamente saranno mostrati e descritti i diagrammi delle attività che coinvolgono l'utente.
Infine si illustreranno degli esempi di utilizzo dei design pattern$_G$ nell'architettura del sistema.

3.2 Architettura generale

L'architettura generale del sistema è di tipo client-server.

Il server ha un'architettura di tipo event-driven basata sul modello ad attori ed espone delle API$_G$ tramite socket TCP.

L'architettura del Client segue il design pattern$_G$ Model-View-Controller con interfaccia da linea di comando e comunica con il server grazie ad un driver$_G$ tramite connessione TCP.

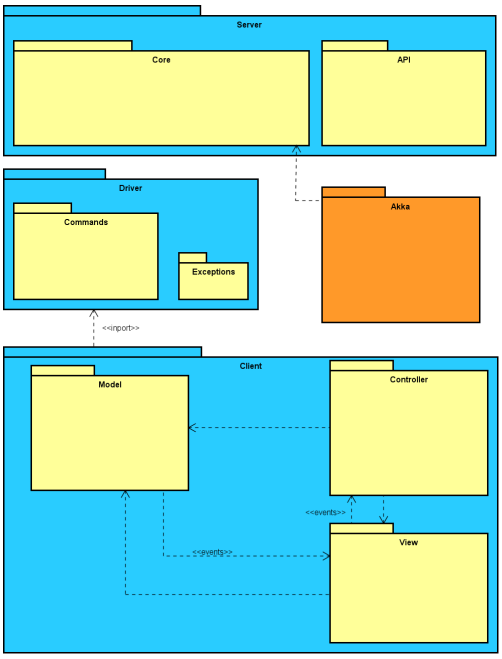


Figura 3: Architettura generale, vista Package

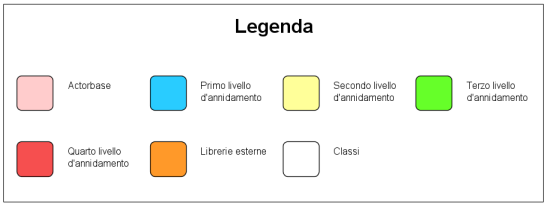


Figura 4: Legenda

3.2.1 Server

Il server di *Actorbase* è composto da due package principali: il package **Core** e il package **API**.
Il package **Core** è a sua volta composto dal package **Actors**, contenente le classi che definiscono gli attori del sistema, e dal package **messages**, contenente i messaggi che gli attori possono inviarsi tra loro.
Il package **API** contiene le classi che forniscono una comunicazione con i client esterni.

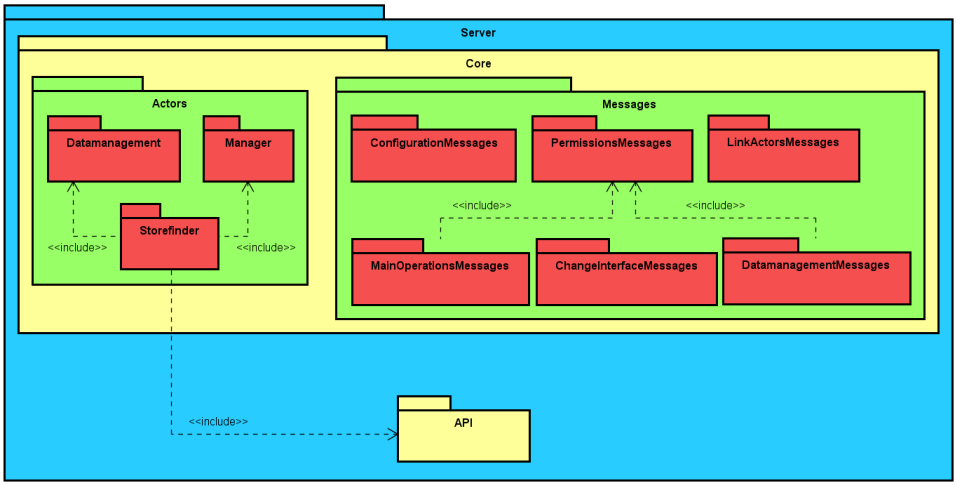


Figura 5: Server, vista Package

3.2.2 Client

L'architettura del Client seguirà il design pattern$_G$ MVC:

- **Model:** Il Model è la componente che si occupa di comunicare con il server usando i metodi del driver$_G$ e di notificare la View quando avviene un cambiamento nel suo stato.
- **View:** La View è la componente che interagisce con l'utente mediante interfaccia a linea di comando. L'utente può usare il DSL per interrogare il Model. La View esegue delle *state query* sul model per avere le informazioni aggiornate.
- **Controller:** Il Controller è la componente che esegue il parsing dei comandi del DSL inseriti nella View e li notifica al Model.

3.2.3 Driver

Il Driver$_G$ è una libreria, invocando i metodi della quale è possibile effettuare richieste TCP verso le API$_G$ esposte dal Server.

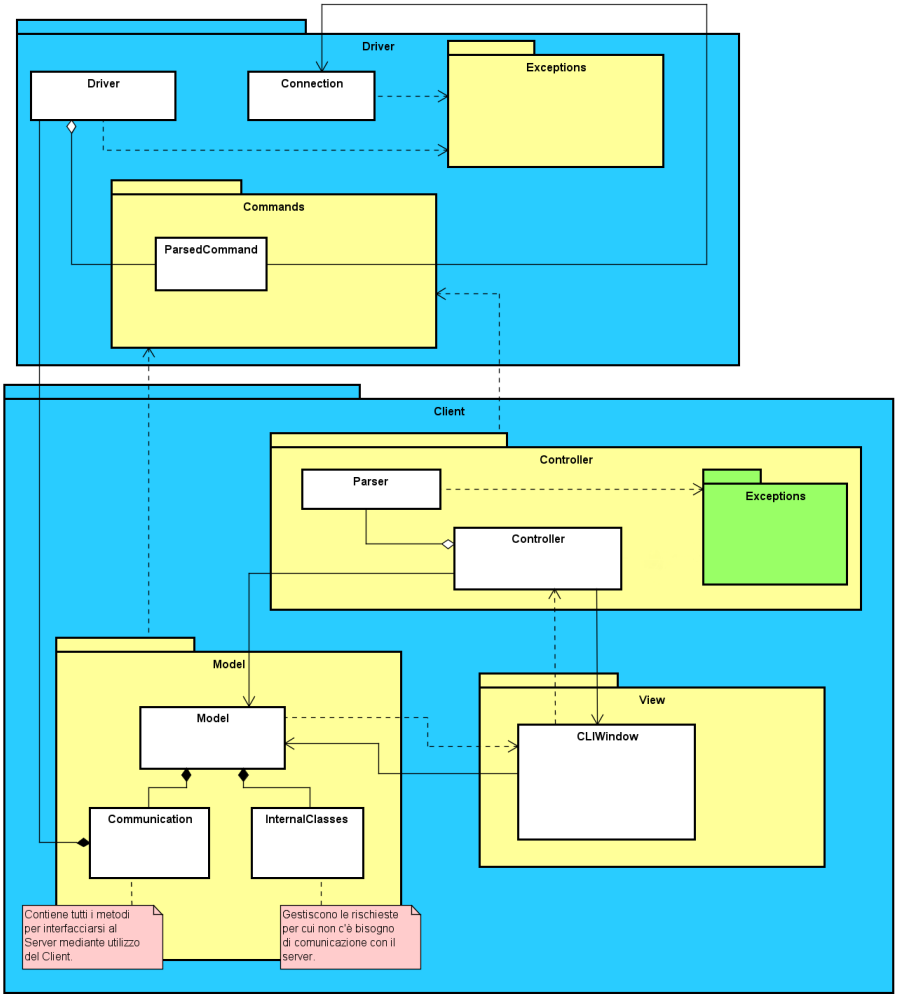


Figura 6: Architettura generale Client e Driver

4 Componenti e Classi

4.1 Actorbase

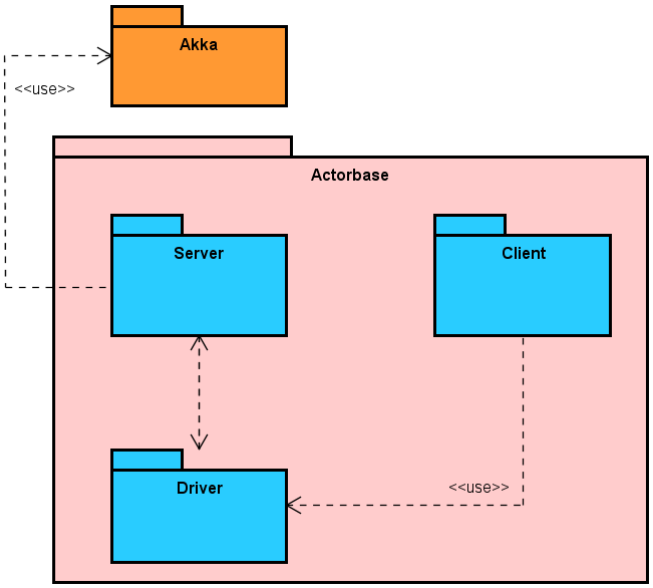


Figura 7: Componente Actorbase

4.1.1 Descrizione

È il package principale del sistema. L'interazione tra i package **Server** e **Driver** definiscono una comunicazione su rete di tipo client-server.
Le classi definite nel package **Server** utilizzano ed estendono le classi della libreria Akka$_G$.

4.1.2 Package Figli

- Actorbase.Server
- Actorbase.Client
- Actorbase.Driver
- Actorbase.Akka

4.2 Actorbase.Server

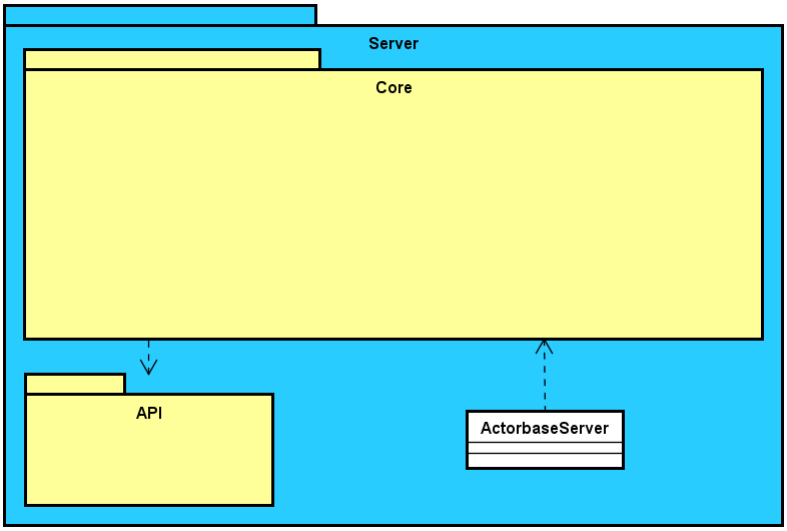


Figura 8: Componente Actorbase.Server

4.2.1 Descrizione

Package per la componente lato server del sistema. È composto dai packages **Core** ed **API** e dalla classe *ActorbaseServer*.

4.2.2 Package Figli

- Actorbase.Server.Core
- Actorbase.Server.API

4.2.3 Classi

- Actorbase.Server.ActorbaseServer

4.3 Actorbase.Server.API

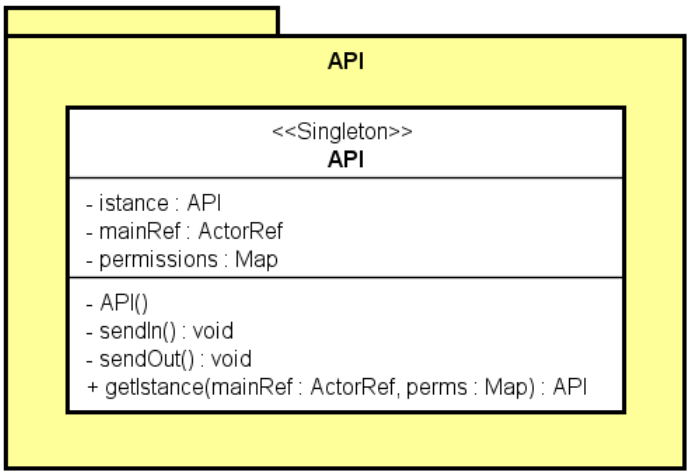


Figura 9: Componente Actorbase.Server.API

4.3.1 Descrizione

Package contenenti le classi che definiscono le API_G attraverso cui i client possono interfacciarsi all'istanza di un server del sistema.

4.3.2 Classi

- Actorbase.Server.API.API

4.4 Actorbase.Server.API.API

4.4.1 Descrizione

Classe *singleton* che definisce l'insieme di API_G esposte dal server e il loro utilizzo.

4.4.2 Utilizzo

Viene utilizzata per gestire l'insieme di connessioni del server dall'esterno, e per accettarne di nuove. Inoltra le richieste provenienti dall'esterno all'actor_G Main_G e invia le risposte ai client.

4.4.3 Relazione con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Main:** relazione entrante, creazione e aggregazione.

4.5 Actorbase.Server.Core

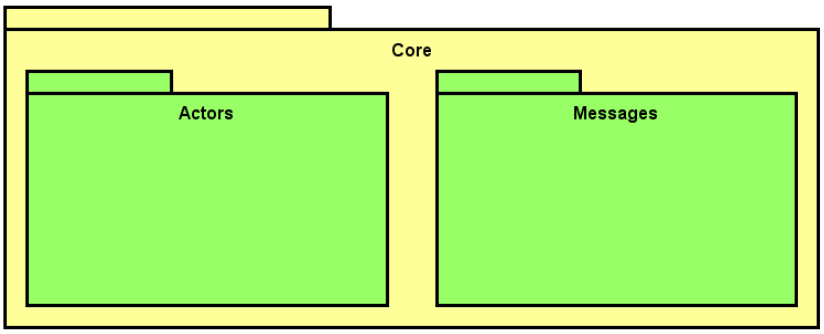


Figura 10: Componente Actorbase.Server.Core

4.5.1 Descrizione

Il package contiene le componenti che costituiscono il nucleo del sistema logico lato server. È composto da due package: **Actors** e **Messages**

4.5.2 Package figli

- Actorbase.Server.Core.Actors
- Actorbase.Server.Core.Messages

4.6 Actorbase.Server.Core.Actors

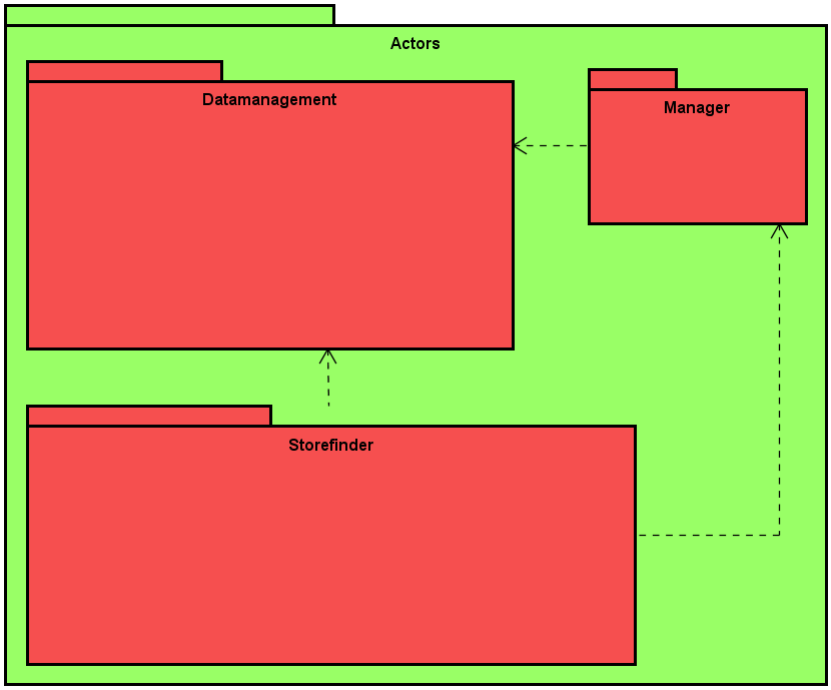


Figura 11: Componente Actorbase.Server.Core.Actors

4.6.1 Descrizione

Il package contiene le componenti che costituiscono i diversi attori definiti nel sistema. Essi definiscono le diverse categorie di attori.

4.6.2 Package figli

- Actorbase.Server.Core.Actors.Datamanagement
- Actorbase.Server.Core.Actors.Manager
- Actorbase.Server.Core.Actors.Storefinder

4.7 Actorbase.Server.Core.Actors.Datamanagement

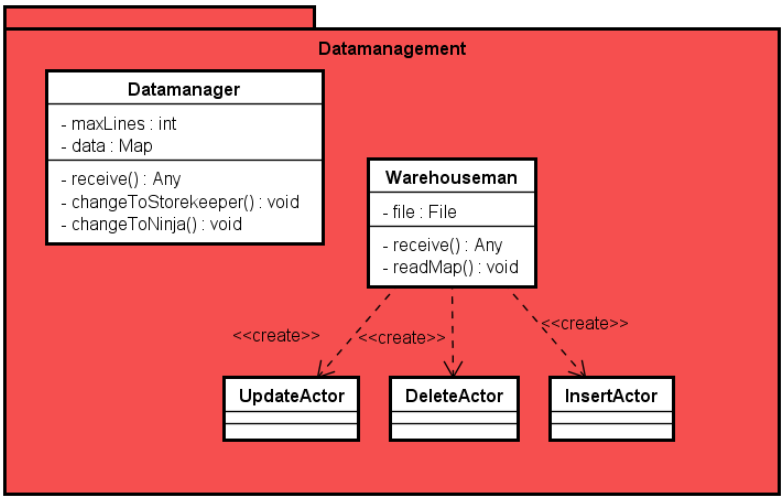


Figura 12: Componente Actorbase.Server.Core.Actors.Datamanagement

4.7.1 Descrizione

All'interno di questo package sono definite le classi che rappresentano gli attori che si occupano direttamente della gestione dei dati.

4.7.2 Classi

- Actorbase.Server.Core.Actors.Datamanagement.Datamanager
- Actorbase.Server.Core.Actors.Datamanagement.Warehouseman
- Actorbase.Server.Core.Actors.Datamanagement.UpdateActor
- Actorbase.Server.Core.Actors.Datamanagement.DeleteActor
- Actorbase.Server.Core.Actors.Datamanagement.InsertActor

4.8 Actorbase.Server.Core.Actors.Datamanagement.Datamanager

4.8.1 Descrizione

Classe che definisce un actor$_G$ di tipo Datamanager.

4.8.2 Utilizzo

Si utilizza per mantenere in RAM una porzione di una mappa di un Database. Un oggetto Datamanager possiede due comportamenti diversi: uno nel caso funzioni da copia principale dei dati (Storekeeper$_G$) e l'altro nel caso funzioni da copia di backup (Ninja$_G$).

4.8.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Storefinder:** relazione entrante, creazione e aggregazione;
- **Actorbase.Server.Core.Actors.Datamanagement.Manager:** relazione entrante, aggregazione.

4.9 Actorbase.Server.Core.Actors.Datamanagement.Warehouseman

4.9.1 Descrizione

Classe che definisce un actor$_G$ di tipo Warehouseman$_G$.

4.9.2 Utilizzo

Si utilizza per mantenere persistenza su disco di una porzione di una mappa di un Database.

4.9.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Storefinder:** relazione entrante, creazione e aggregazione;
- **Actorbase.Server.Core.Actors.Datamanagement.UpdateActor:** relazione uscente, creazione;
- **Actorbase.Server.Core.Actors.Datamanagement.DeleteActor:** relazione uscente, creazione;
- **Actorbase.Server.Core.Actors.Datamanagement.InsertActor:** relazione uscente, creazione.

4.10 Actorbase.Server.Core.Actors.Datamanagement.UpdateActor

4.10.1 Descrizione

Classe che definisce un actor_G di tipo UpdateActor.

4.10.2 Utilizzo

Si utilizza per aggiornare un item su un file su disco.

4.10.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Datamanagement.Warehouseman:** relazione entrante, creazione.

4.11 Actorbase.Server.Core.Actors.Datamanagement.DeleteActor

4.11.1 Descrizione

Classe che definisce un actor_G di tipo DeleteActor.

4.11.2 Utilizzo

Si utilizza per rimuovere un item su un file su disco.

4.11.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Datamanagement.Warehouseman:** relazione entrante, creazione.

4.12 Actorbase.Server.Core.Actors.Datamanagement.InsertActor

4.12.1 Descrizione

Classe che definisce un actor_G di tipo InsertActor.

4.12.2 Utilizzo

Si utilizza per inserire un item su un file su disco.

4.12.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Datamanagement.Warehouseman:** relazione entrante, creazione.

4.13 Actorbase.Server.Core.Actors.Manager

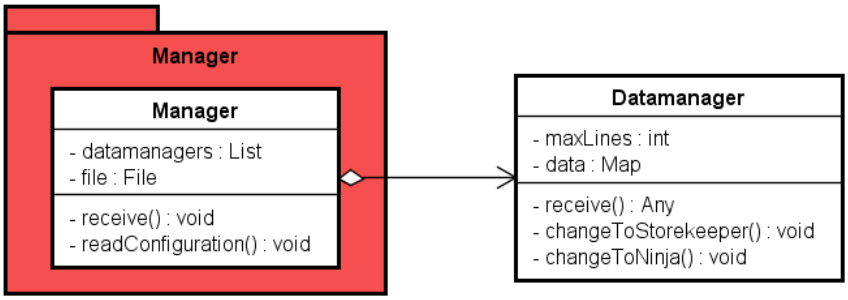


Figura 13: Componente Actorbase.Server.Core.Actors.Manager

4.13.1 Descrizione

All'interno di questo package sono definite le classi che rappresentano gli attori che si occupano della gestione di altri attori e dei vincoli presenti su di essi.

4.13.2 Classi

- Actorbase.Server.Core.Actors.Manager.Manager

4.14 Actorbase.Server.Core.Actors.Manager.Manager

4.14.1 Descrizione

Classe che definisce un actor$_G$ di tipo Manager$_G$.

4.14.2 Utilizzo

Un oggetto di tipo Manager$_G$ è utilizzato per gestire vincoli su altri attori.

4.14.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Datamanagement.Datamanager:** relazione uscente, aggregazione;
- **Actorbase.Server.Core.Actors.Storefinder.Storefinder:** relazione entrante, aggregazione e creazione.

4.15 Actorbase.Server.Core.Actors.Storefinder

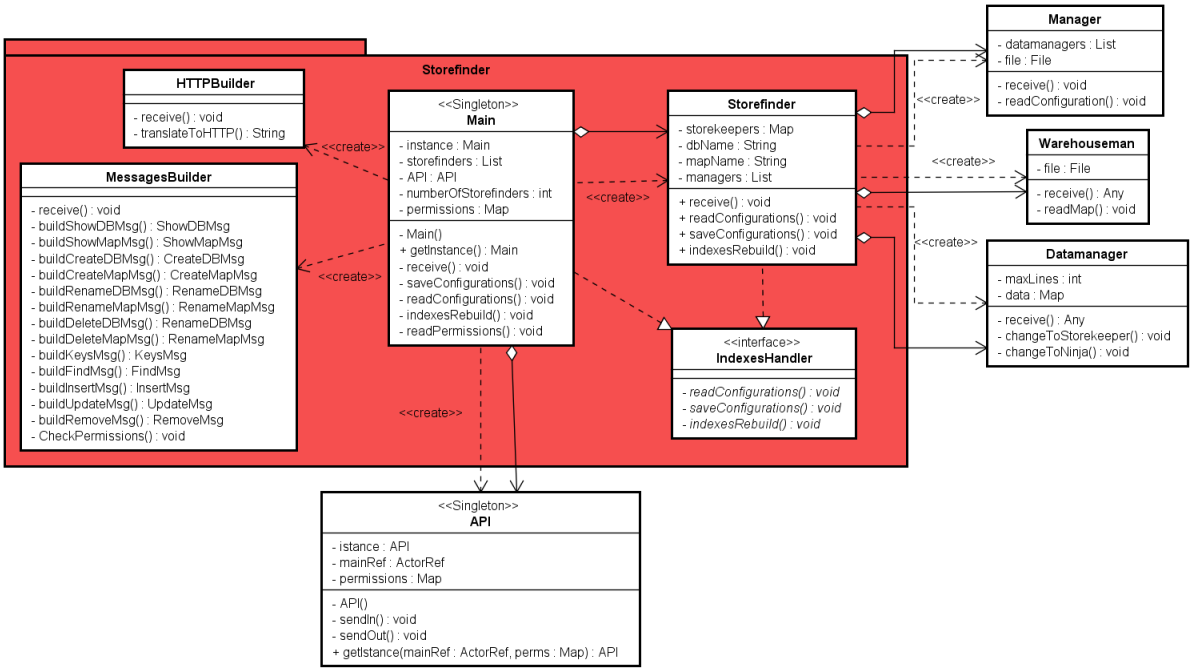


Figura 14: Componente Actorbase.Server.Core.Actors.Storefinder

4.15.1 Descrizione

All'interno di questo package sono definite le classi che rappresentano gli attori che si occupano dell'indicizzazione degli altri attori presenti e del corretto instradamento, creazione e gestione dei messaggi.

4.15.2 Classi

- Actorbase.Server.Core.Actors.Storefinder.Storefinder
- Actorbase.Server.Core.Actors.Storefinder.Main
- Actorbase.Server.Core.Actors.Storefinder.HTTPBuilder
- Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder

4.15.3 Interfacce

- Actorbase.Server.Core.Actors.Storefinder.IndexesHandler

4.16 Actorbase.Server.Core.Actors.Storefinder.IndexesHandler

4.16.1 Descrizione

Interfaccia per gli attori che si occupano di indicizzare altri attori e di inviare messaggi e ricevere risposte da essi. Definisce i contratti di metodi che permettono di leggere, salvare e ricostruire un insieme di indici.

4.16.2 Classi figlie

- Actorbase.Server.Core.Actors.Storefinder.Main
- Actorbase.Server.Core.Actors.Storefinder.Storefinder

4.17 Actorbase.Server.Core.Actors.Storefinder.Storefinder

4.17.1 Descrizione

Classe che definisce un actor$_G$ di tipo Storefinder$_G$. Implementa l'interfaccia IndexesHandler.

4.17.2 Utilizzo

Uno Storefinder$_G$ indicizza un insieme di Datamanager rappresentanti diverse porzioni di una stessa mappa. Per ogni Datamanager indicizzato, inoltre, tiene traccia dei Datamanager Ninja$_G$ e dei Warehouseman$_G$ ad esso associati. Si occupa di inviare gli opportuni messaggi a tali tipologie di attori, di ricevere e instradare le risposte ricevute.
Uno Storefinder$_G$ inoltre si occupa della creazione e distruzione di attori di tipo Manager$_G$, Datamanager, Warehouseman$_G$.

4.17.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Main:** relazione entrante, creazione e aggregazione;
- **Actorbase.Server.Core.Actors.Manager.Manager:** relazione uscente, creazione e aggregazione;
- **Actorbase.Server.Core.Actors.Datamanagement.Datamanager:** relazione uscente, creazione e aggregazione;
- **Actorbase.Server.Core.Actors.Datamanagement.Warehouseman:** relazione uscente, creazione e aggregazione.

4.17.4 Interfacce implementate

- **Actorbase.Server.Core.Actors.Storefinder.IndexesHandler**

4.18 Actorbase.Server.Core.Actors.Storefinder.Main

4.18.1 Descrizione

Classe *singleton* che definisce un actor$_G$ di tipo Main$_G$. Implementa l'interfaccia IndexesHandler.

4.18.2 Utilizzo

Il Main$_G$ indicizza un insieme di Storefinder$_G$ e con essi scambia messaggi, inoltre si occupa di inviare e ricevere messaggi HTTP da API$_G$. Crea attori di tipo Storefinder, MessagesBuilder, HTTPBuilder.

4.18.3 Relazioni con altre classi

- **Actorbase.Server.ActorbaseServer:** relazione entrante, creazione;
- **Actorbase.Server.Core.Actors.Storefinder.Storefinder:** relazione uscente, creazione e aggregazione;
- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione uscente, creazione;
- **Actorbase.Server.Core.Actors.Storefinder.HTTPBuilder:** relazione uscente, creazione;
- **Actorbase.Server.Core.Actors.API.API:** relazione uscente, creazione.

4.18.4 Interfacce implementate

- **Actorbase.Server.Core.Actors.Storefinder.IndexesHandler**

4.19 Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder

4.19.1 Descrizione

Classe che definisce un actor_G di tipo MessagesBuilder.

4.19.2 Utilizzo

Costruisce uno dei messaggi definiti in **Actorbase.Server.Core.Messages** a partire da un messaggio HTTP, basandosi sui permessi del client che ha inviato il messaggio HTTP.

4.19.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Main:** relazione entrante, creazione.

4.20 Actorbase.Server.Core.Actors.Storefinder.HTTPBuilder

4.20.1 Descrizione

Classe che definisce un actor_G di tipo HTTPBuilder.

4.20.2 Utilizzo

Costruisce un messaggio HTTP che le API_G possono inviare a un client a partire da un messaggio definito in **Actorbase.Server.Core.Messages**.

4.20.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Main:** relazione entrante, creazione.

4.21 Actorbase.Server.Core.Messages

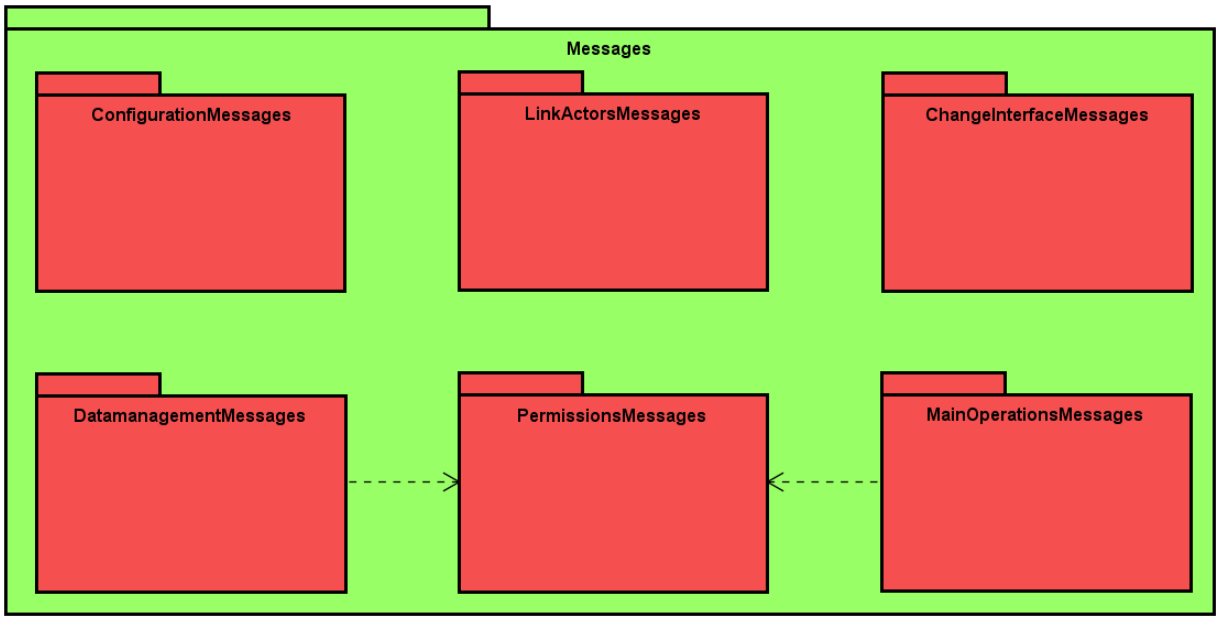


Figura 15: Componente Actorbase.Server.Core.Messages

4.21.1 Descrizione

All'interno di questo package sono definite le componenti che rappresentano i messaggi che i diversi attori del sistema possono inviarsi tra loro.

4.21.2 Package Figli

- Actorbase.Server.Core.Messages.ConfigurationMessages
- Actorbase.Server.Core.Messages.PermissionMessages
- Actorbase.Server.Core.Messages.LinkActorsMessages
- Actorbase.Server.Core.Messages.MainOperationMessages
- Actorbase.Server.Core.Messages.DatamanagementMessages
- Actorbase.Server.Core.Messages.ChangeInterfaceMessages

4.22 Actorbase.Server.Core.Messages.ConfigurationMessages

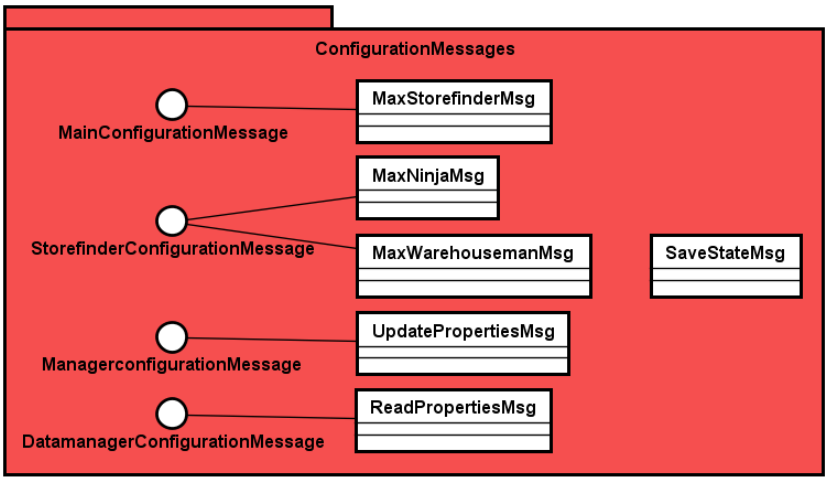


Figura 16: Componente Actorbase.Server.Core.Messages.ConfigurationMessages

4.22.1 Descrizione

All'interno di questo package sono definite le classi e le interfacce che rappresentano i messaggi relativi ad avvisi di configurazione$_G$ delle impostazioni del sistema.

4.22.2 Classi

- Actorbase.Server.Core.Messages.ConfigurationMessages.MaxStorefinderMsg
- Actorbase.Server.Core.Messages.ConfigurationMessages.MaxNinjaMsg
- Actorbase.Server.Core.Messages.ConfigurationMessages.MaxWarehousemanMsg
- Actorbase.Server.Core.Messages.ConfigurationMessages.UpdatePropertiesMsg
- Actorbase.Server.Core.Messages.ConfigurationMessages.ReadPropertiesMsg
- Actorbase.Server.Core.Messages.ConfigurationMessages.SaveStateMsg

4.22.3 Interfacce

- Actorbase.Server.Core.Messages.ConfigurationMessages.MainConfigurationMessage
- Actorbase.Server.Core.Messages.ConfigurationMessages.StorefinderConfigurationMessage
- Actorbase.Server.Core.Messages.ConfigurationMessages.ManagerConfigurationMessage
- Actorbase.Server.Core.Messages.ConfigurationMessages.DatamanagerConfigurationMessage

4.23 Actorbase.Server.Core.Messages.ConfigurationMessages. MainConfigurationMessage

4.23.1 Descrizione

Interfaccia per messaggi di configurazione$_G$ per actor$_G$ di tipo Main$_G$. Le classi che implementano questa interfaccia definiscono messaggi di configurazione$_G$ che possono essere gestiti da actor$_G$ di tipo Main$_G$.

4.23.2 Classi figlie

- Actorbase.Server.Core.Messages.ConfigurationMessages.MaxStorefinderMsg

4.24 Actorbase.Server.Core.Messages.ConfigurationMessages.MaxStorefinderMsg

4.24.1 Descrizione

Classe che definisce un messaggio MaxStorefinder.

4.24.2 Utilizzo

Un messaggio MaxStorefinder informa chi lo riceve di aggiornare l'impostazione relativa al numero di actor$_G$ di tipo Storefinder$_G$ creabili per mappa.

4.24.3 Relazioni con altre classi

- **Actorbase.Server.ActorbaseServer:** relazione entrante, creazione

4.24.4 Interfacce implementate

- Actorbase.Server.Core.Messages.ConfigurationMessages.MainConfigurationMessage

4.25 Actorbase.Server.Core.Messages.ConfigurationMessages. StorefinderConfigurationMessage

4.25.1 Descrizione

Interfaccia per messaggi di configurazione$_G$ per actor$_G$ di tipo Storefinder$_G$. Le classi che implementano questa interfaccia definiscono messaggi di configurazione$_G$ che possono essere gestiti da actor$_G$ di tipo Storefinder$_G$.

4.25.2 Classi figlie

- Actorbase.Server.Core.Messages.ConfigurationMessages.MaxNinjaMsg
- Actorbase.Server.Core.Messages.ConfigurationMessages.MaxWarehousemanMsg

4.26 Actorbase.Server.Core.Messages.ConfigurationMessages.MaxNinjaMsg

4.26.1 Descrizione

Classe che definisce un messaggio MaxNinjaMsg.

4.26.2 Utilizzo

Un messaggio MaxNinjaMsg informa chi lo riceve di aggiornare l'impostazione relativa al numero di actor$_G$ di tipo Ninja$_G$ creabili per Datamanager.

4.26.3 Relazioni con altre classi

- **Actorbase.Server.ActorbaseServer:** relazione entrante, creazione

4.26.4 Interfacce implementate

- Actorbase.Server.Core.Messages.ConfigurationMessages.StorefinderConfigurationMessage

4.27 Actorbase.Server.Core.Messages.ConfigurationMessages.MaxWarehousemanMsg

4.27.1 Descrizione

Classe che definisce un messaggio MaxWarehousemanMsg.

4.27.2 Utilizzo

Un messaggio MaxWarehousemanMsg informa chi lo riceve di aggiornare l'impostazione relativa al numero di $actor_G$ di tipo $Warehouseman_G$ creabili per Datamanager.

4.27.3 Relazioni con altre classi

- **Actorbase.Server.ActorbaseServer:** relazione entrante, creazione

4.27.4 Interfacce implementate

- Actorbase.Server.Core.Messages.ConfigurationMessages.StorefinderConfigurationMessage

4.28 Actorbase.Server.Core.Messages.ConfigurationMessages. ManagerConfigurationMessage

4.28.1 Descrizione

Interfaccia per messaggi di $configurazione_G$ per $actor_G$ di tipo $Manager_G$. Le classi che implementano questa interfaccia definiscono messaggi di $configurazione_G$ che possono essere gestiti da $actor_G$ di tipo $Manager_G$.

4.28.2 Classi figlie

- Actorbase.Server.Core.Messages.ConfigurationMessages.UpdatePropertiesMsg

4.29 Actorbase.Server.Core.Messages.ConfigurationMessages.UpdatePropertiesMsg

4.29.1 Descrizione

Classe che definisce un messaggio UpdatePropertiesMsg.

4.29.2 Utilizzo

Un messaggio UpdatePropertiesMsg informa chi lo riceve di aggiornare le impostazioni per $actor_G$ $Manager_G$.

4.29.3 Relazioni con altre classi

- **Actorbase.Server.ActorbaseServer:** relazione entrante, creazione

4.29.4 Interfacce implementate

- Actorbase.Server.Core.Messages.ConfigurationMessages.ManagerConfigurationMessage

4.30 Actorbase.Server.Core.Messages.ConfigurationMessages. DatamanagerConfigurationMessage

4.30.1 Descrizione

Interfaccia per messaggi di $configurazione_G$ per $actor_G$ di tipo Datamanager. Le classi che implementano questa interfaccia definiscono messaggi di $configurazione_G$ che possono essere gestiti da $actor_G$ di tipo Datamanager.

4.30.2 Classi figlie

- Actorbase.Server.Core.Messages.ConfigurationMessages.UpdatePropertiesMsg

4.31 Actorbase.Server.Core.Messages.ConfigurationMessages.ReadPropertiesMsg

4.31.1 Descrizione

Classe che definisce un messaggio ReadPropertiesMsg.

4.31.2 Utilizzo

Un messaggio ReadPropertiesMsg informa chi lo riceve di aggiornare le impostazioni per actor$_G$ Datamanager.

4.31.3 Relazioni con altre classi

- **Actorbase.Server.ActorbaseServer:** relazione entrante, creazione

4.31.4 Interfacce implementate

- Actorbase.Server.Core.Messages.ConfigurationMessages.DatamanagerConfigurationMessage

4.32 Actorbase.Server.Core.Messages.PermissionMessages

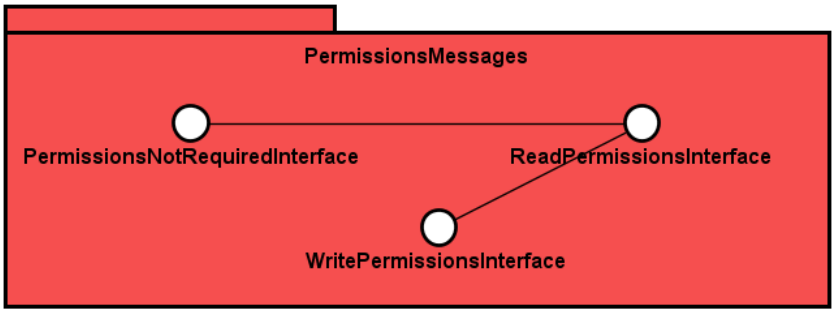


Figura 17: Componente Actorbase.Server.Core.Messages.PermissionMessages

4.32.1 Descrizione

All'interno di questo package sono definite le interfacce che rappresentano i diversi gradi di permesso che un'operazione richiede. Un'operazione può infatti richiedere i permessi di lettura, scrittura o nessun permesso. Ogni messaggio relativo ad un'operazione richiesta da un client estende una di queste interfacce.

4.32.2 Interfacce

- Actorbase.Server.Core.Messages.PermissionMessages.PermissionsNotRequiredInterface
- Actorbase.Server.Core.Messages.PermissionMessages.ReadPermissionsInterface
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionsInterface

4.33 Actorbase.Server.Core.Messages.PermissionMessages. PermissionsNotRequiredInterface

4.33.1 Descrizione

Interfaccia per messaggi che definiscono operazioni per le quali non sono richiesti specifici permessi.

4.33.2 Classi figlie

- Actorbase.Server.Core.Messages.MainOperationMessage.ShowDBMsg
- Actorbase.Server.Core.Messages.MainOperationMessage.ShowMapMsg
- Actorbase.Server.Core.Messages.MainOperationMessage.CreateDBMsg

4.33.3 Interfacce figlie

- Actorbase.Server.Core.Messages.PermissionMessages.ReadPermissionsInterface

4.34 Actorbase.Server.Core.Messages.PermissionMessages.ReadPermissionsInterface

4.34.1 Descrizione

Interfaccia per messaggi che definiscono operazioni per le quali sono richiesti permessi di lettura.

4.34.2 Classi figlie

- Actorbase.Server.Core.Messages.DatamanagementMessage.KeysMsg
- Actorbase.Server.Core.Messages.DatamanagementMessage.FindMsg

4.34.3 Interfacce figlie

- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionsInterface

4.34.4 Interfacce estese

- Actorbase.Server.Core.Messages.PermissionMessages.PermissionsNotRequiredInterface

4.35 Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionsInterface

4.35.1 Descrizione

Interfaccia per messaggi che definiscono operazioni per le quali sono richiesti permessi di scrittura.

4.35.2 Classi figlie

- Actorbase.Server.Core.Messages.MainOperationMessage.CreateMapMsg
- Actorbase.Server.Core.Messages.MainOperationMessage.RenameDBMsg
- Actorbase.Server.Core.Messages.MainOperationMessage.RenameMapMsg
- Actorbase.Server.Core.Messages.MainOperationMessage.DeleteDBMsg
- Actorbase.Server.Core.Messages.MainOperationMessage.DeleteMapMsg
- Actorbase.Server.Core.Messages.DatamanagementMessage.InsertMsg
- Actorbase.Server.Core.Messages.DatamanagementMessage.UpdateMsg
- Actorbase.Server.Core.Messages.DatamanagementMessage.RemoveMsg

4.35.3 Interfacce estese

- Actorbase.Server.Core.Messages.PermissionMessages.ReadPermissionsInterface

4.36 Actorbase.Server.Core.Messages.LinkActorsMessages

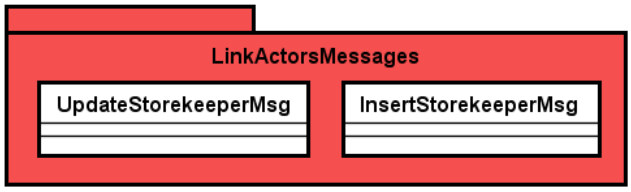


Figura 18: Componente Actorbase.Server.Core.Messages.LinkActorsMessages

4.36.1 Descrizione

All'interno di questo package sono definite le classi che rappresentano i messaggi relativi alla gestione dei collegamenti tra diversi attori.

4.36.2 Classi

- Actorbase.Server.Core.Messages.LinkActorsMessages.UpdateStorekeeperMsg
- Actorbase.Server.Core.Messages.LinkActorsMessages.InsertStorekeeperMsg

4.37 Actorbase.Server.Core.Messages.LinkActorsMessages.UpdateStorekeeperMsg

4.37.1 Descrizione

Classe che definisce un messaggio UpdateStorekeeperMsg.

4.37.2 Utilizzo

Un messaggio UpdateStorekeeperMsg informa il Manager$_G$ che lo riceve di aggiornare il riferimento ad uno degli actor$_G$ Datamanager Storekeeper$_G$ della sua lista.

4.37.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Storefinder:** relazione entrante, creazione

4.38 Actorbase.Server.Core.Messages.LinkActorsMessages.InsertStorekeeperMsg

4.38.1 Descrizione

Classe che definisce un messaggio InsertStorekeeperMsg.

4.38.2 Utilizzo

Un messaggio InsertStorekeeperMsg informa il Manager$_G$ che lo riceve di aggiungere il riferimento ad un actor$_G$ Datamenager Storekeeper$_G$ alla sua lista.

4.38.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.Storefinder:** relazione entrante, creazione

4.39 Actorbase.Server.Core.Messages.MainOperationMessages

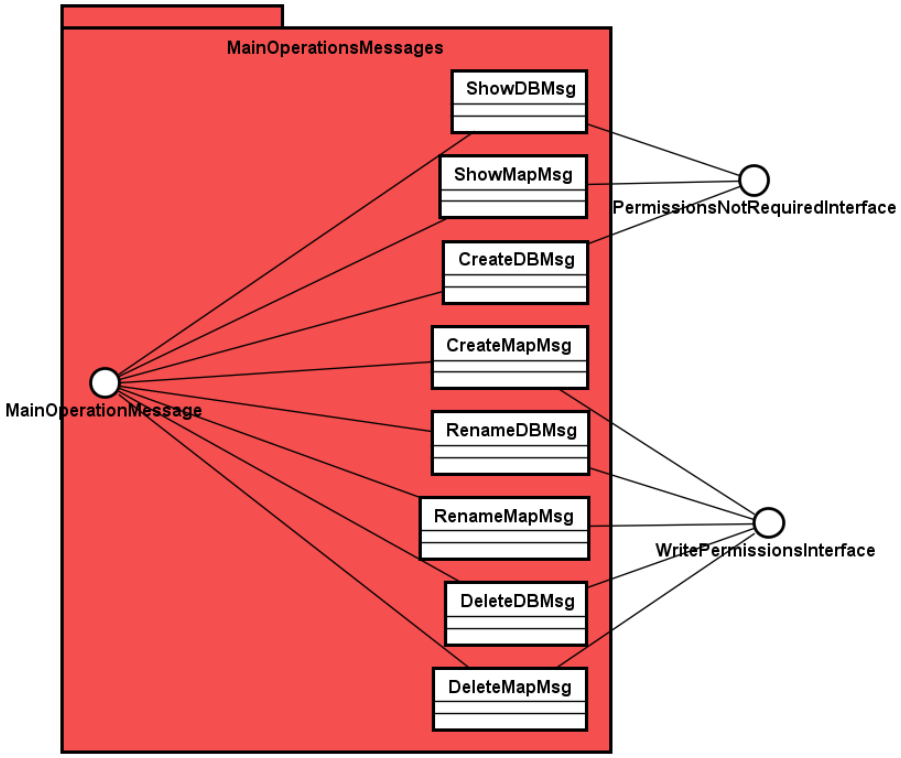


Figura 19: Componente Actorbase.Server.Core.Messages.MainOperationMessages

4.39.1 Descrizione

All'interno di questo package sono definite le classi e le interfacce che rappresentano i messaggi relativi ad operazioni che non richiedono l'inoltro del messaggio stesso.

4.39.2 Classi

- Actorbase.Server.Core.Messages.MainOperationMessages.ShowDBMsg
- Actorbase.Server.Core.Messages.MainOperationMessages.ShowMapMsg
- Actorbase.Server.Core.Messages.MainOperationMessages.CreateDBMsg
- Actorbase.Server.Core.Messages.MainOperationMessages.CreateMapMsg
- Actorbase.Server.Core.Messages.MainOperationMessages.RenameDBMsg
- Actorbase.Server.Core.Messages.MainOperationMessages.RenameMapMsg
- Actorbase.Server.Core.Messages.MainOperationMessages.DeleteDBMsg
- Actorbase.Server.Core.Messages.MainOperationMessages.DeleteMapMsg

4.39.3 Interfacce

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage

4.40 Actorbase.Server.Core.Messages.MainOperationMessages.ShowDBMsg

4.40.1 Descrizione

Classe che definisce un messaggio ShowDBMsg.

4.40.2 Utilizzo

Un messaggio ShowDBMsg richiede la lista dei Database presenti sul server.

4.40.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.40.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.PermissionsNotRequiredInterface

4.41 Actorbase.Server.Core.Messages.MainOperationMessages.ShowMapMsg

4.41.1 Descrizione

Classe che definisce un messaggio ShowMapMsg.

4.41.2 Utilizzo

Un messaggio ShowMapMsg richiede la lista delle mappe del Database selezionato.

4.41.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.41.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.PermissionsNotRequiredInterface

4.42 Actorbase.Server.Core.Messages.MainOperationMessages.CreateDBMsg

4.42.1 Descrizione

Classe che definisce un messaggio CreateDBMsg.

4.42.2 Utilizzo

Un messaggio CreateDBMsg richiede la creazione di un nuovo Database.

4.42.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.42.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.PermissionsNotRequiredInterface

4.43 Actorbase.Server.Core.Messages.MainOperationMessages.CreateMapMsg

4.43.1 Descrizione

Classe che definisce un messaggio CreateMapMsg.

4.43.2 Utilizzo

Un messaggio CreateMapMsg richiede la creazione di una nuova mappa nel Database selezionato.

4.43.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.43.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.44 Actorbase.Server.Core.Messages.MainOperationMessages.RenameDBMsg

4.44.1 Descrizione

Classe che definisce un messaggio RenameDBMsg.

4.44.2 Utilizzo

Un messaggio RenameDBMsg richiede la rinomina di un Database presente sul server.

4.44.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.44.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.45 Actorbase.Server.Core.Messages.MainOperationMessages.RenameMapMsg

4.45.1 Descrizione

Classe che definisce un messaggio RenameMapMsg.

4.45.2 Utilizzo

Un messaggio RenameMapMsg richiede la rinomina di una mappa del Database selezionato.

4.45.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.45.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.46 Actorbase.Server.Core.Messages.MainOperationMessages.DeleteDBMsg

4.46.1 Descrizione

Classe che definisce un messaggio DeleteDBMsg.

4.46.2 Utilizzo

Un messaggio DeleteDBMsg richiede l'eliminazione di un Database dal server.

4.46.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.46.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.47 Actorbase.Server.Core.Messages.MainOperationMessages.DeleteMapMsg

4.47.1 Descrizione

Classe che definisce un messaggio DeleteMapMsg.

4.47.2 Utilizzo

Un messaggio DeleteMapMsg richiede l'eliminazione di una mappa del Database selezionato.

4.47.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.47.4 Interfacce estese

- Actorbase.Server.Core.Messages.MainOperationMessages.MainOperationMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.48 Actorbase.Server.Core.Messages.DatamanagementMessages

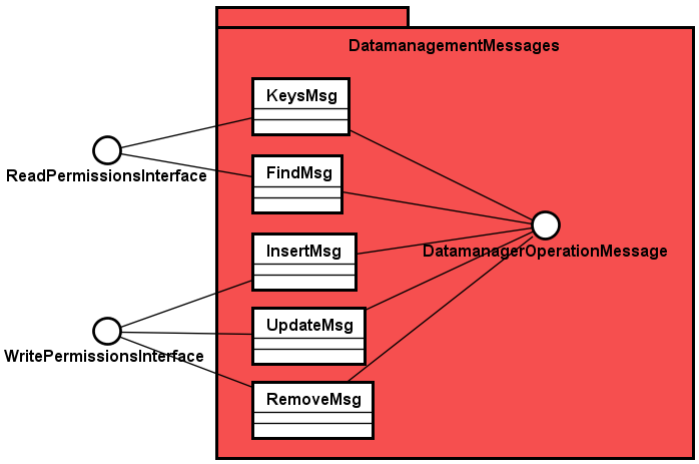


Figura 20: Componente Actorbase.Server.Core.Messages.DatamanagementOperationMessages

4.48.1 Descrizione

All'interno di questo package sono definite le classi e le interfacce che rappresentano messaggi relativi ad operazioni sui dati.

4.48.2 Classi

- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.KeysMsg
- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.FindMsg
- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.InsertMsg
- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.UpdateMsg
- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.RemoveMsg

4.48.3 Interfacce

- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.DatamanagementMessage

4.49 Actorbase.Server.Core.Messages.DatamanagementOperationMessages.KeysMsg

4.49.1 Descrizione

Classe che definisce un messaggio KeysMsg.

4.49.2 Utilizzo

Un messaggio KeysMsg richiede la lista di chiavi della mappa selezionata.

4.49.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.49.4 Interfacce estese

- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.DatamanagementMessage
- Actorbase.Server.Core.Messages.PermissionMessages.ReadPermissionInterface

4.50 Actorbase.Server.Core.Messages.DatamanagementOperationMessages.FindMsg

4.50.1 Descrizione

Classe che definisce un messaggio FindMsg.

4.50.2 Utilizzo

Un messaggio FindMsg richiede il valore dell'item con la chiave contenuta nel messaggio.

4.50.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.50.4 Interfacce estese

- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.DatamanagementMessage
- Actorbase.Server.Core.Messages.PermissionMessages.ReadPermissionInterface

4.51 Actorbase.Server.Core.Messages.DatamanagementOperationMessages.InsertMsg

4.51.1 Descrizione

Classe che definisce un messaggio InsertMsg.

4.51.2 Utilizzo

Un messaggio InsertMsg richiede l'inserimento di un item con chiave e valore contenuti nel messaggio.

4.51.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.51.4 Interfacce estese

- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.DatamanagementMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.52 Actorbase.Server.Core.Messages.DatamanagementOperationMessages.UpdateMsg

4.52.1 Descrizione

Classe che definisce un messaggio UpdateMsg.

4.52.2 Utilizzo

Un messaggio UpdateMsg richiede l'aggiornamento di un item con chiave e valore contenuti nel messaggio.

4.52.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.52.4 Interfacce estese

- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.DatamanagementMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.53 Actorbase.Server.Core.Messages.DatamanagementOperationMessages.DeleteMsg

4.53.1 Descrizione

Classe che definisce un messaggio DeleteMsg.

4.53.2 Utilizzo

Un messaggio DeleteMsg richiede l'eliminazione dell'item con chiave contenuta nel messaggio.

4.53.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.53.4 Interfacce estese

- Actorbase.Server.Core.Messages.DatamanagementOperationMessages.DatamanagementMessage
- Actorbase.Server.Core.Messages.PermissionMessages.WritePermissionInterface

4.54 Actorbase.Server.Core.Messages.ChangeInterfaceMessages

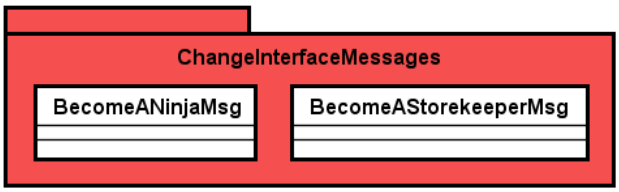


Figura 21: Componente Actorbase.Server.Core.Messages.ChangeInterfaceMessages

4.54.1 Descrizione

All'interno di questo package sono definite le classi e le interfacce che rappresentano i messaggi inviabili per effettuare operazioni di cambio interfaccia per gli attori che supportano tale funzionalità.

4.54.2 Classi

- Actorbase.Server.Core.Messages.ChangeInterfaceMessages.BecomeANinjaMsg
- Actorbase.Server.Core.Messages.ChangeInterfaceMessages.BecomeAStoreKeeperMsg

4.55 Actorbase.Server.Core.Messages.ChangeInterfaceMessages.BecomeANinjaMsg

4.55.1 Descrizione

Classe che definisce un messaggio BecomeANinjaMsg.

4.55.2 Utilizzo

Un messaggio BecomeANinjaMsg richiede ad un actor$_G$ Datamanager di cambiare il suo comportamento in Ninja$_G$.

4.55.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.56 Actorbase.Server.Core.Messages.ChangeInterfaceMessages. BecomeAStorekeeperMsg

4.56.1 Descrizione

Classe che definisce un messaggio BecomeAStorekeeperMsg.

4.56.2 Utilizzo

Un messaggio BecomeAStorekeeperMsg richiede ad un actor$_G$ Datamanager di cambiare il suo comportamento in Storekeeper$_G$.

4.56.3 Relazioni con altre classi

- **Actorbase.Server.Core.Actors.Storefinder.MessagesBuilder:** relazione entrante, creazione

4.57 Actorbase.Driver

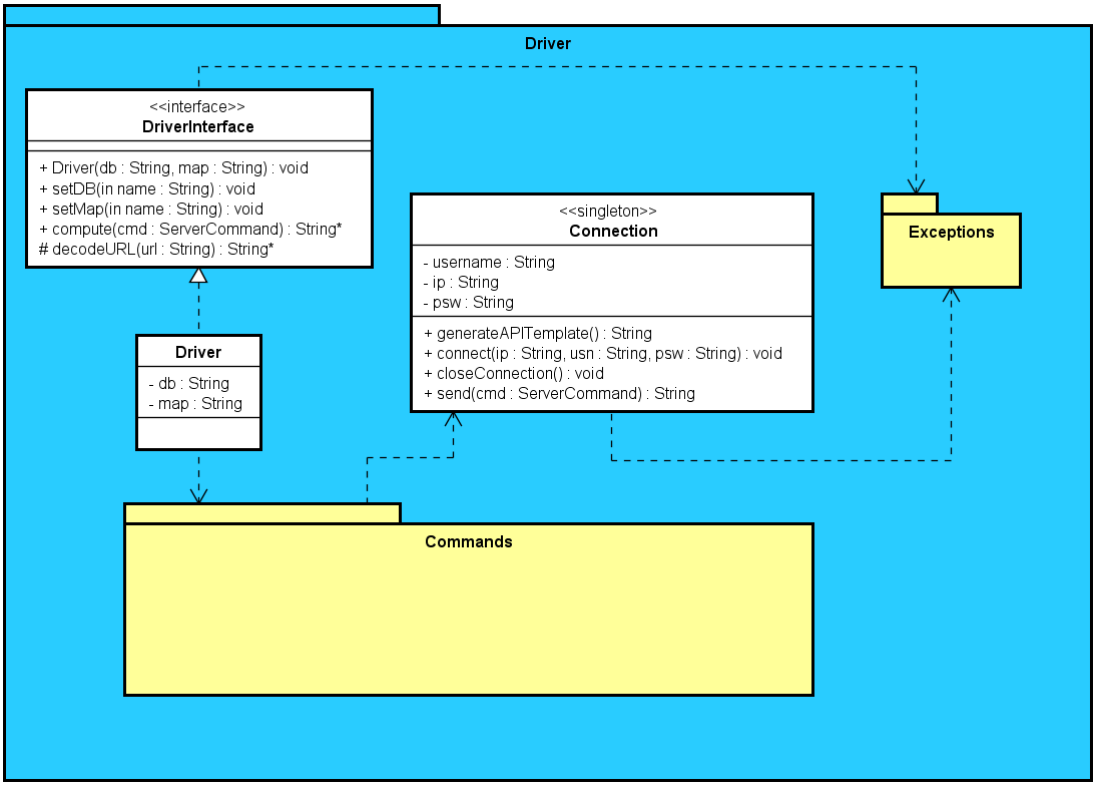


Figura 22: Componente Actorbase.Driver

4.57.1 Descrizione

Package per la componente Driver$_G$ del sistema. Per la gestione dei comandi implementa il Design Pattern$_G$ Command.

4.57.2 Package Figli

- Actorbase.Driver.Components
- Actorbase.Driver.Exceptions

4.57.3 Classi

- Actorbase.Driver.Driver
- Actorbase.Driver.Connection

4.57.4 Interfacce

- Actorbase.Driver.DriverInterface

4.58 Actorbase.Driver.Connection

4.58.1 Descrizione

Classe *singleton* che gestisce la comunicazione del Driver$_G$ con il Server. Svolge il ruolo di *Receiver*.

4.58.2 Utilizzo

Viene usata per aprire e chiudere la connessione con il Server, inviare i messaggi e per generare la prima parte del comando per le API$_G$.

4.58.3 Relazione con altre classi

- **Actorbase.Driver.Commands.ServerCommand:** Relazione entrante, invio comando al Server.
- **Actorbase.Driver.Exception:** Relazione uscente, creazione di eccezioni.

4.59 Actorbase.Driver.Driver

4.59.1 Descrizione

Classe che gestisce la comunicazione mediante API$_G$. Svolge il ruolo di *Invoker*.

4.59.2 Utilizzo

Espone i metodi per codifica e decodifica dei comandi delle API$_G$. Inoltre mantiene le informazioni relative ad eventuali Database o Mappe selezionate.

4.59.3 Interfacce Estese

Actorbase.Driver.DriverInterface

4.59.4 Relazioni con altre classi

- **Actorbase.Driver.Commands:** Relazione uscente, utilizzo del metodo Execute.
- **Actorbase.Driver.Exceptions:** Relazione uscente, creazione di eccezioni.

4.60 Actorbase.Driver.Commands

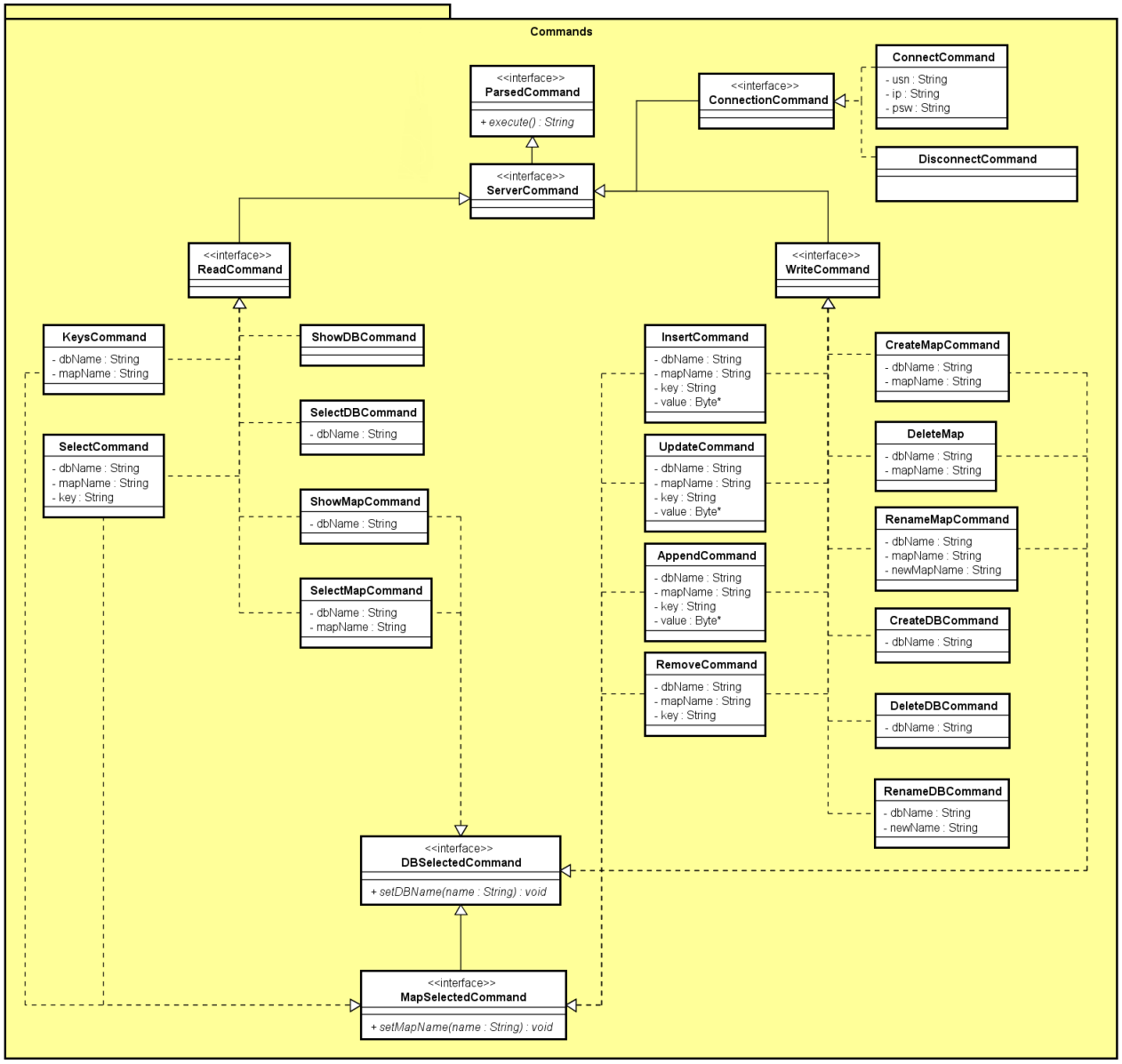


Figura 23: Package Actorbase.Driver.Commands

4.60.1 Descrizione

Package che contiene la gerarchia di comandi dell'interfaccia *Command*. Ognuno di essi rappresenta astrattamente una operazione che deve essere svolta a livello del Server. Sono presenti due gerarchie di interfacce, una per la tipologia di comandi (lato client, lato server, lettura, scrittura) e l'altra per la tipologia di "pre-selezioni" necessarie per eseguire il comando (Database selezionato, Mappa selezionata).

4.60.2 Interfacce

- Actorbase.Driver.Commands.ServerCommand
- Actorbase.Driver.Commands.ReadCommand
- Actorbase.Driver.Commands.WriteCommand
- Actorbase.Driver.Commands.ConnectionCommand

- Actorbase.Driver.Commands.DBSelectedCommand
- Actorbase.Driver.Commands.MapSelectedCommand

4.60.3 Classi

- Actorbase.Driver.Commands.KeysCommand
- Actorbase.Driver.Commands.SelectCommand
- Actorbase.Driver.Commands.ShowDBCommand
- Actorbase.Driver.Commands.SelectDBCommand
- Actorbase.Driver.Commands.ShowMapCommand
- Actorbase.Driver.Commands.SelectMapCommand
- Actorbase.Driver.Commands.InsertCommand
- Actorbase.Driver.Commands.UpdateCommand
- Actorbase.Driver.Commands.AppendCommand
- Actorbase.Driver.Commands.RemoveCommand
- Actorbase.Driver.Commands.CreateMapCommand
- Actorbase.Driver.Commands.DeleteMapCommand
- Actorbase.Driver.Commands.RenameMapCommand
- Actorbase.Driver.Commands.CreateDBCommand
- Actorbase.Driver.Commands.DeleteDBCommand
- Actorbase.Driver.Commands.RenameDBCommand
- Actorbase.Driver.Commands.ConnectCommand
- Actorbase.Driver.Commands.DisconnectCommand

4.60.4 Relazioni con altre componenti

- **Actorbase.Client.Model:** Relazione entrante, inclusione.
- **Actorbase.Client.Controller:** Relazione entrante, inclusione.

4.61 Actorbase.Driver.Commands.ParsedCommand

4.61.1 Descrizione

Interfaccia base che rappresenta un generico comando che il Driver sa trattare.

4.61.2 Interfacce Figle

Actorbase.Driver.Commands.ServerCommand

4.62 Actorbase.Driver.Commands.ServerCommand

4.62.1 Descrizione

Interfaccia base che rappresenta un comando lato Server.

4.62.2 Interfacce Estese

Actorbase.Driver.Commands.ParsedCommand

4.62.3 Interfacce Figlie

- Actorbase.Driver.Commands.ReadCommand
- Actorbase.Driver.Commands.WriteCommand
- Actorbase.Driver.Commands.ConnectionCommand

4.63 Actorbase.Driver.Commands.ReadCommand

4.63.1 Descrizione

Interfaccia base che rappresenta un comando per cui sono necessari i permessi di lettura sulla tabella selezionata

4.63.2 Interfacce Estese

Actorbase.Driver.Commands.ServerCommand

4.63.3 Classi Figlie

- **Actorbase.Driver.Commands.KeysCommand:** Classe concreta che rappresenta il comando per mostrare la lista delle chiavi nella mappa selezionata.
- **Actorbase.Driver.Commands.SelectCommand:** Classe concreta che rappresenta il comando per selezionare il valore di una chiave nella mappa selezionata.
- **Actorbase.Driver.Commands.ShowBDCommand:** Classe concreta che rappresenta il comando per mostrare l'elenco dei Database di cui si hanno permessi di lettura.
- **Actorbase.Driver.Commands.SelectDBCommand:** Classe concreta che rappresenta il comando per selezionare un determinato Database.
- **Actorbase.Driver.Commands.ShowMapCommand:** Classe concreta che rappresenta il comando per mostrare la lista delle mappe nel Database selezionato.
- **Actorbase.Driver.Commands.SelectMapCommand:** Classe concreta che rappresenta il comando per selezionare una determinata mappa.

4.64 Actorbase.Driver.Commands.WriteCommand

4.64.1 Descrizione

Interfaccia base che rappresenta un comando per cui sono necessari i permessi di scrittura sulla tabella selezionata

4.64.2 Interfacce Estese

Actorbase.Driver.Commands.ServerCommand

4.64.3 Classi Figlie

- **Actorbase.Driver.Commands.InsertCommand:** Classe concreta che rappresenta il comando per inserire una coppia chiave-valore nella mappa selezionata.
- **Actorbase.Driver.Commands.UpdateCommand:** Classe concreta che rappresenta il comando per aggiornare il valore di una chiave nella mappa selezionata.
- **Actorbase.Driver.Commands.AppendCommand:** Classe concreta che rappresenta il comando per appendere in coda al valore di una chiave nella mappa selezionata.
- **Actorbase.Driver.Commands.RemoveCommand:** Classe concreta che rappresenta il comando per rimuovere una coppia chiave-valore dalla mappa selezionata.

- **Actorbase.Driver.Commands.CreateMapCommand:** Classe concreta che rappresenta il comando per creare una nuova mappa nel Database selezionato.
- **Actorbase.Driver.Commands.DeleteMapCommand:** Classe concreta che rappresenta il comando per eliminare una mappa nel Database selezionato.
- **Actorbase.Driver.Commands.RenameMapCommand:** Classe concreta che rappresenta il comando per rinominare una mappa nel Database selezionato.
- **Actorbase.Driver.Commands.CreateDBCommand:** Classe concreta che rappresenta il comando per creare un nuovo Database.
- **Actorbase.Driver.Commands.DeleteDBCommand:** Classe concreta che rappresenta il comando per rimuovere un Database.
- **Actorbase.Driver.Commands.RenameDBCommand:** Classe concreta che rappresenta il comando per rinominare un Database.

4.65 Actorbase.Driver.Commands.ConnectionCommand

4.65.1 Descrizione

Interfaccia base che rappresenta un comando di connessione o disconnessione dal Server.

4.65.2 Interfacce Estese

Actorbase.Driver.Commands.ServerCommand

4.65.3 Classi Figlie

- **Actorbase.Driver.Commands.ConnectCommand:** Classe concreta che rappresenta il comando per la connessione al server.
- **Actorbase.Driver.Commands.DisconnectCommand:** Classe concreta che rappresenta il comando per la disconnessione dal server.

4.66 Actorbase.Driver.Commands.DBSelectedCommand

4.66.1 Descrizione

Interfaccia base che rappresenta un generico comando per il quale è necessario avere un Database selezionato.

4.66.2 Classi Figlie

- Actorbase.Driver.Commands.ShowMapCommand
- Actorbase.Driver.Commands.SelectMapCommand
- Actorbase.Driver.Commands.CreateCommand
- Actorbase.Driver.Commands.DeleteCommand
- Actorbase.Driver.Commands.RenameCommand

4.67 Actorbase.Driver.Commands.MapSelectedCommand

4.67.1 Descrizione

Interfaccia base che rappresenta un generico comando per il quale è necessario avere una mappa selezionata.

4.67.2 Interfacce Estese

Actorbase.Driver.Commands.DBSelectedCommand

4.67.3 Classi Figlie

- Actorbase.Driver.Commands.KeysCommand
- Actorbase.Driver.Commands.SelectCommand
- Actorbase.Driver.Commands.InsertCommand
- Actorbase.Driver.Commands.UpdateCommand
- Actorbase.Driver.Commands.AppendCommand
- Actorbase.Driver.Commands.RemoveCommand

4.68 Actorbase.Client

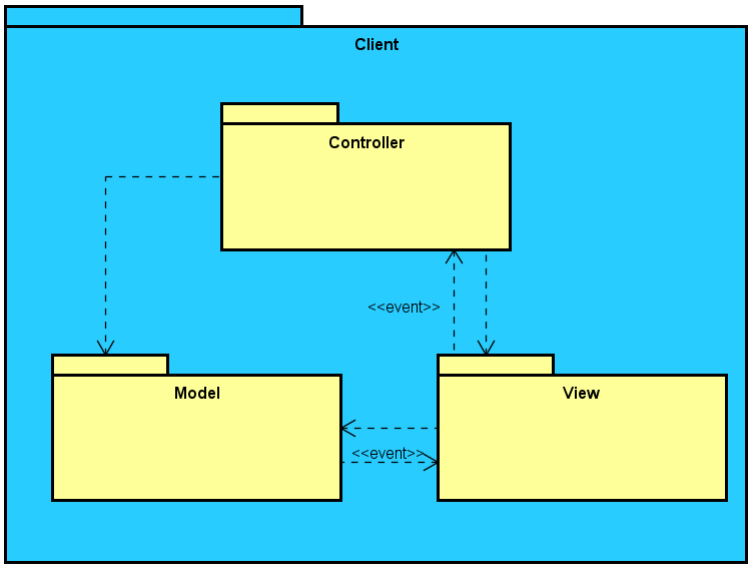


Figura 24: Package Actorbase.Client

4.68.1 Descrizione

Package per la componente Client del sistema. Implementa il Design Pattern$_G$ Model View Controller.

4.68.2 Package Figli

- Actorbase.Client.Model
- Actorbase.Client.View
- Actorbase.Client.Controller

4.69 Actorbase.Client.Model

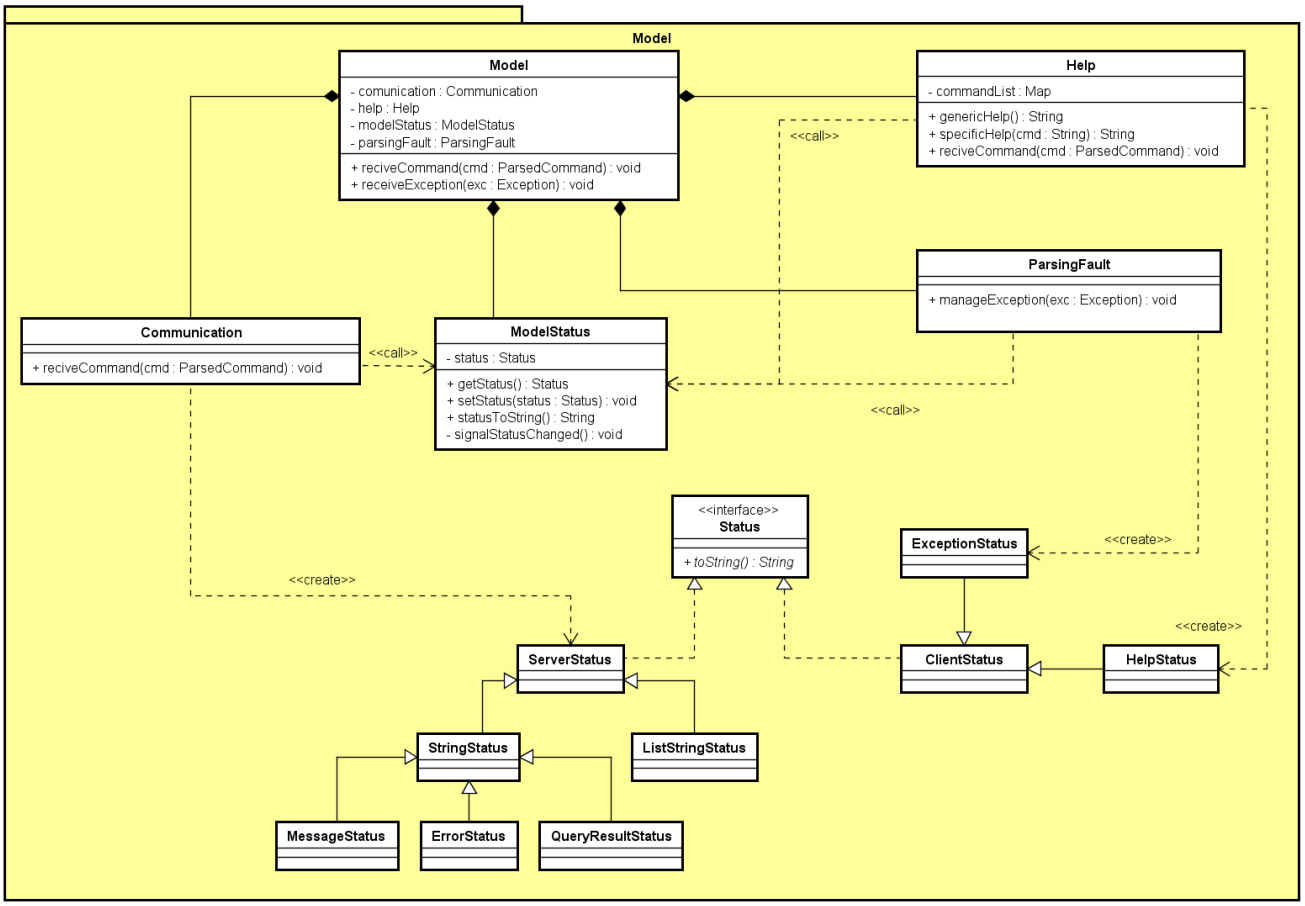


Figura 25: Package Actorbase.Client.Model

4.69.1 Descrizione

Package per la componente Model del pattern MVC della componente Client.

4.69.2 Interfacce

- Actorbase.Client.Model.Status

4.69.3 Classi

- Actorbase.Client.Model.Model
- Actorbase.Client.Model.Help
- Actorbase.Client.Model.ParsingFault
- Actorbase.Client.Model.Comunication
- Actorbase.Client.Model.ModelStatus
- Actorbase.Client.Model.ServerStatus
- Actorbase.Client.Model.StringStatus
- Actorbase.Client.Model.ListStringStatus
- Actorbase.Client.Model.MessageErrorStatus

- Actorbase.Client.Model.QueryResultStatus
- Actorbase.Client.Model.ExceptionStatus
- Actorbase.Client.Model.HelpStatus

4.69.4 Relazioni con altre componenti

- **Actorbase.Driver.Commands:** Relazione uscente, inclusione.

4.70 Actorbase.Client.Model.Status

4.70.1 Descrizione

Interfaccia base della gerarchia degli stati del Model.

4.70.2 Utilizzo

Rappresenta un generico stato che il Model può assumere, è la rappresentazione astratta della frazione di stato dell'intero sistema Actorbase che è richiesta dall'utente. Le sue classi figlie espongono tutte un metodo per poter restituire in forma di stringa le informazioni alla View.

4.70.3 Classi Figlie

- Actorbase.Client.Model.ServerStatus
- Actorbase.Client.Model.ClientStatus

4.71 Actorbase.Client.Model.ServerStatus

4.71.1 Descrizione

Classe base per un generico stato restituito dal Server.

4.71.2 Utilizzo

Rappresenta un generico stato del Server che il Model può assumere.

4.71.3 Interfacce Estese

Actorbase.Client.Model.Status

4.71.4 Relazioni con altre classi

- **Actorbase.Client.Model.Communication:** Relazione entrante, creazione.

4.71.5 Classi Figlie

- **Actorbase.Client.Model.ListStringStatus:** Stato che rappresenta un generico stato di tipo lista di stringhe.
- **Actorbase.Client.Model.StringStatus:** Stato che rappresenta un generico stato di tipo stringa.
  - **Actorbase.Client.Model.MessageStatus:** Stato che rappresenta un messaggio del server.
  - **Actorbase.Client.Model.ErrorStatus:** Stato che rappresenta un errore a livello del server.
  - **Actorbase.Client.Model.QueryResultStatus:** Stato che rappresenta il risultato di una query ritornato in forma di stringa.

4.71.6 Actorbase.Client.Model.ClientStatus

4.71.7 Descrizione

Classe base per un generico stato restituito dal Client stesso.

4.71.8 Utilizzo

Rappresenta un generico stato del Client che il Model può assumere.

4.71.9 Interfacce Estese

Actorbase.Client.Model.Status

4.71.10 Relazioni con altre classi

- **Actorbase.Client.Model.Help:** Relazione entrante, creazione.
- **Actorbase.Client.Model.ParsingFault:** Relazione entrante, creazione.

4.71.11 Classi Figlie

- **Actorbase.Client.Model.ExceptionStatus:** Stato che rappresenta una eccezione.
- **Actorbase.Client.Model.HelpStatus:** Stato che rappresenta una richiesta di help da parte dell'utente.

4.72 Actorbase.Client.Model.Model

4.72.1 Descrizione

Classe base per la componente Model del Pattern MVC della componente Client.

4.72.2 Utilizzo

Fornisce un accesso unico al sottosistema delle classi ad esso composte.

4.72.3 Relazioni con altre classi

- **Actorbase.Client.Model.Communication:** Relazione uscente, composizione.
- **Actorbase.Client.Model.ModelStatus:** Relazione uscente, composizione.
- **Actorbase.Client.Model.Help:** Relazione uscente, composizione.
- **Actorbase.Client.Model.ParsingFault:** Relazione uscente, composizione.
- **Actorbase.Client.Controller:** Relazione entrante, chiamata metodo per la ricezione del comando.

4.73 Actorbase.Client.Model.Comunication

4.73.1 Descrizione

Classe che gestisce la comunicazione con il server tramite Driver_G.

4.73.2 Utilizzo

Gestisce la comunicazione con il server tramite Driver_G, crea gli ServerStatus e chiama il metodo per modificare l'attributo status del Model.

4.73.3 Relazioni con altre classi

- **Actorbase.Client.Model.Model:** Relazione entrante, composizione.
- **Actorbase.Client.Model.ServerStatus:** Relazione uscente, creazione.
- **Actorbase.Client.Model.ModelStatus:** Relazione uscente, impostazione dello status.
- **Actorbase.Driver.Driver:** Relazione uscente, composizione.

4.74 Actorbase.Client.Model.ModelStatus

4.74.1 Descrizione

Classe che gestisce l'attributo stato di Model.

4.74.2 Relazioni con altre classi

- **Actorbase.Client.Model.Model:** Relazione entrante, composizione.
- **Actorbase.Client.Model.Communication:** Relazione entrante, impostazione dello status.
- **Actorbase.Client.Model.Help:** Relazione entrante, impostazione dello status.
- **Actorbase.Client.Model.ParsingFault:** Relazione entrante, impostazione dello status.

4.75 Actorbase.Client.Model.Help

4.75.1 Descrizione

Classe che gestisce le richieste di aiuto.

4.75.2 Utilizzo

Gestisce le richieste di aiuto, crea gli HelpStatus e chiama il metodo per modificare l'attributo status del Model.

4.75.3 Relazioni con altre classi

- **Actorbase.Client.Model.Model:** Relazione entrante, composizione.
- **Actorbase.Client.Model.HelpStatus:** Relazione uscente, creazione.
- **Actorbase.Client.Model.ModelStatus:** Relazione uscente, impostazione dello status.

4.76 Actorbase.Client.Model.ParsingFault

4.76.1 Descrizione

Classe che gestisce le eccezioni.

4.76.2 Utilizzo

Gestisce le eccezioni, crea gli ExceptionStatus e chiama il metodo per modificare l'attributo status del Model.

4.76.3 Relazioni con altre classi

- **Actorbase.Client.Model.Model:** Relazione entrante, composizione.
- **Actorbase.Client.Model.ExceptionStatus:** Relazione uscente, creazione.
- **Actorbase.Client.Model.ModelStatus:** Relazione uscente, impostazione dello status.

4.77 Actorbase.Client.View

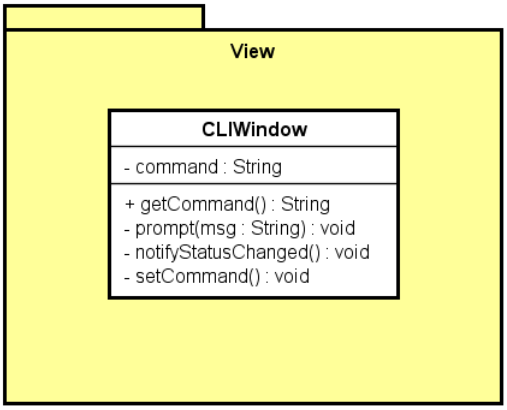


Figura 26: Package Actorbase.Client.View

4.77.1 Descrizione

Package per la componente View del pattern MVC della componente Client.

4.77.2 Classi

Actorbase.Client.View.View

4.78 Actorbase.Client.View.View

4.78.1 Descrizione

Classe che gestisce l'interfaccia utente.

4.78.2 Utilizzo

Gestisce l'interfaccia utente, permette di inserire comandi e stamparne l'output a video.

4.78.3 Relazioni con altre classi

- **Actorbase.Client.Controller.Controller:** Relazione entrante, richiesta comando inserito.
- **Actorbase.Client.Controller.Controller:** Relazione uscente, segnalazione di cambio di stato.
- **Actorbase.Client.Model.Model:** Relazione entrante, ricezione segnale di cambio di stato.
- **Actorbase.Client.Model.Model:** Relazione uscente, query sullo stato del Model.

4.79 Actorbase.Client.Controller

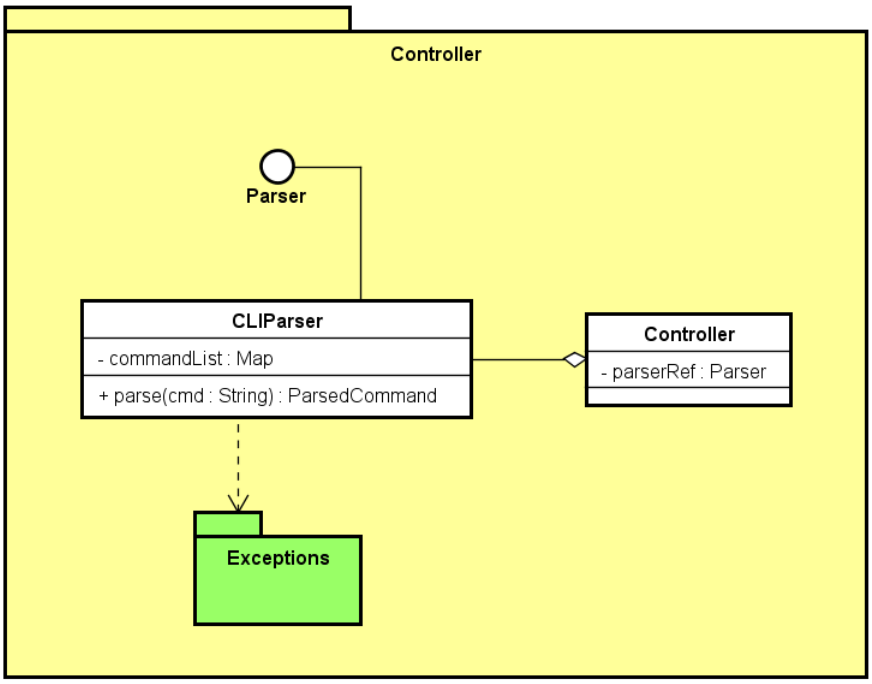


Figura 27: Package Actorbase.Client.Controller

4.79.1 Descrizione

Package per la componente Controller del pattern MVC della componente Client.

4.79.2 Interfacce

Actorbase.Client.Controller.Parser

4.79.3 Classi

- Actorbase.Client.Controller.CLIParser
- Actorbase.Client.Controller.Controller

4.79.4 Package Figli

Actorbase.Client.Controller.Exceptions

4.79.5 Relazioni con altre componenti

- **Actorbase.Driver.Commands:** Relazione uscente, inclusione.

4.80 Actorbase.Client.Controller.Parser

4.80.1 Descrizione

Interfaccia base per la gerarchia dei parser.

4.80.2 Utilizzo

Fornisce una interfaccia comune a qualsiasi parser.

4.80.3 Relazioni con altre classi

- **Actorbase.Client.Controller.ClIParser:** Relazione entrante, realizzazione di interfaccia.

4.81 Actorbase.Client.Controller.CLIParser

4.81.1 Descrizione

Classe concreta che rappresenta un parser della DSL di Actorbase.

4.81.2 Utilizzo

Mantiene una mappa dei comandi e delle relative norme, è in grado di trasformare un comando DSL inserito dall'utente nel rispettivo comando della gerarchia Actorbase.Driver.Commands.

4.81.3 Relazioni con altre classi

- **Actorbase.Client.Controller.Parser:** Relazione entrante, realizzazione di interfaccia.
- **Actorbase.Client.Controller.Controller:** Relazione entrante, aggregazione.
- **Actorbase.Client.Controller.Exception:** Relazione uscente, importazione, crea le classi presenti nel package.

4.82 Actorbase.Client.Controller.Controller

4.82.1 Descrizione

Classe base per la componente Controller del Pattern MVC della componente Client.

4.82.2 Utilizzo

Viene notificata dalla View quando il suo stato cambia, si occupa di leggere il nuovo stato della View e modificare lo status del Model.

4.82.3 Relazioni con altre classi

- **Actorbase.Client.Controller.ClIParser:** Relazione uscente, aggregazione.
- **Actorbase.Client.View.View:** Relazione uscente, query sullo stato.
- **Actorbase.Client.View.View:** Relazione entrate, segnalazione di cambio stato.
- **Actorbase.Client.Model.Model:** Relazione uscente, cambio stato.

4.83 Actorbase.Client.Controller.Exception

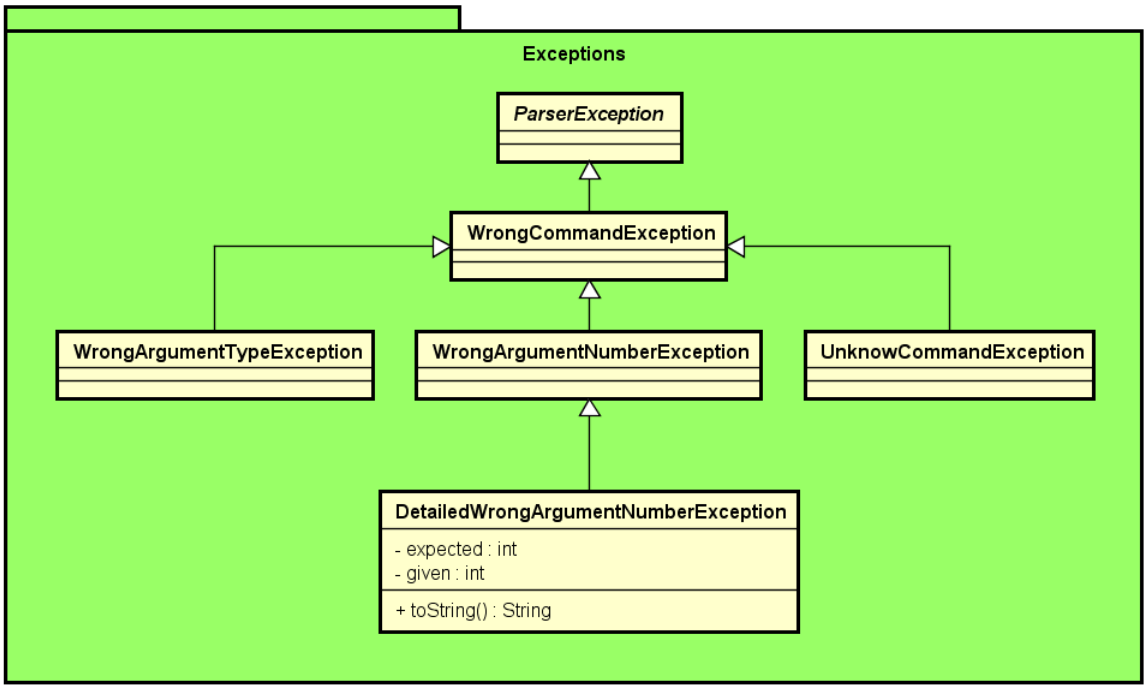


Figura 28: Package Actorbase.Client.Exceptions

4.83.1 Descrizione

Package che contiene le eccezioni che il Parser può lanciare.

4.83.2 Classi

- **Actorbase.Client.Controller.Exceptions.ParserException:** Generica eccezione che può essere lanciata dal Parser.
  - **Actorbase.Client.Controller.Exceptions.WrongCommandException:** Generica eccezione riguardante l'errata sintassi del comando.
    - **Actorbase.Client.Controller.Exceptions.WrongArgumentTypeException:** Eccezione lanciata nel caso in cui il tipo dell'argomento sia errato.
    - **Actorbase.Client.Controller.Exceptions.UnknownCommandException:** Eccezione lanciata nel caso in cui il comando inserito sia inesistente.
    - **Actorbase.Client.Controller.Exceptions.WrongArgumentNumberException:** Eccezione lanciata nel caso in cui ci sia un numero errato di parametri per il comando inserito.
      - **Actorbase.Client.Controller.Exceptions.DetailedWrongArgumentNumberException:** Eccezione lanciata nel caso in cui ci sia un numero errato di parametri per il comando inserito; fornisce anche il numero di parametri atteso e quello dei parametri inseriti dall'utente.

5 Diagrammi delle attività

Segue la descrizione dei diagrammi delle attività che mostrano le possibili interazioni dell'utente con Actorbase. Il diagramma iniziale illustrerà le attività possibili che saranno successivamente mostrate in sotto diagrammi specifici, queste attività sono segnate nel diagramma principale con un fork.

5.0.1 Diagramma attività principale

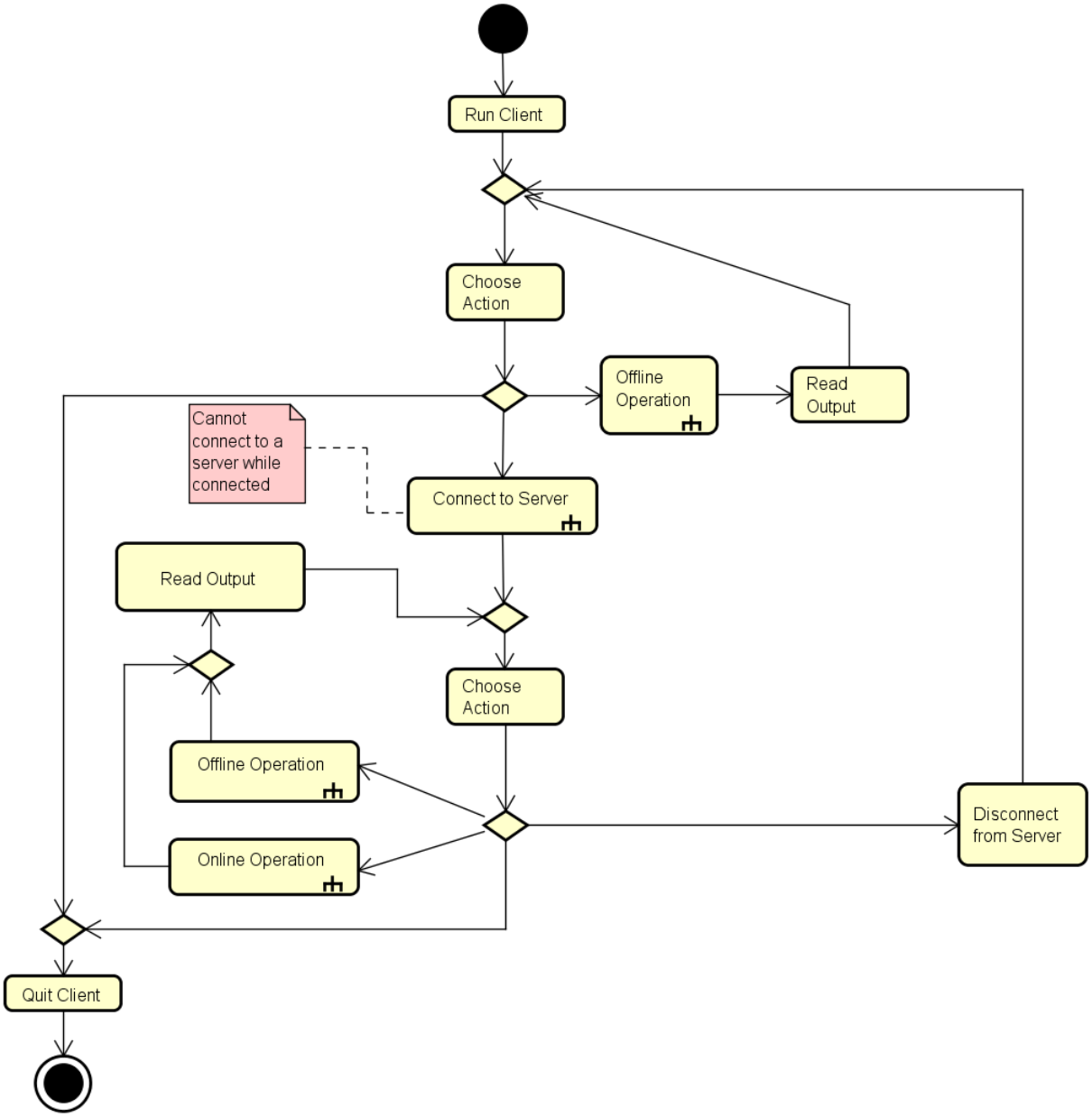


Figura 29: Diagramma attività principale

Dopo aver avviato il client l'utente può svolgere tre tipi di operazione: connettersi ad un server, chiudere l'applicativo o svolgere un'operazione offline. Se sceglie di connettersi può sempre chiudere l'applicativo e svolgere operazioni che non necessitano di essere connessi al server, può in più disconnettersi o svolgere operazioni sul server, non potrà però più connettersi ad un server finché non effettua la disconnessione dal server corrente.

5.0.2 Offline Operation

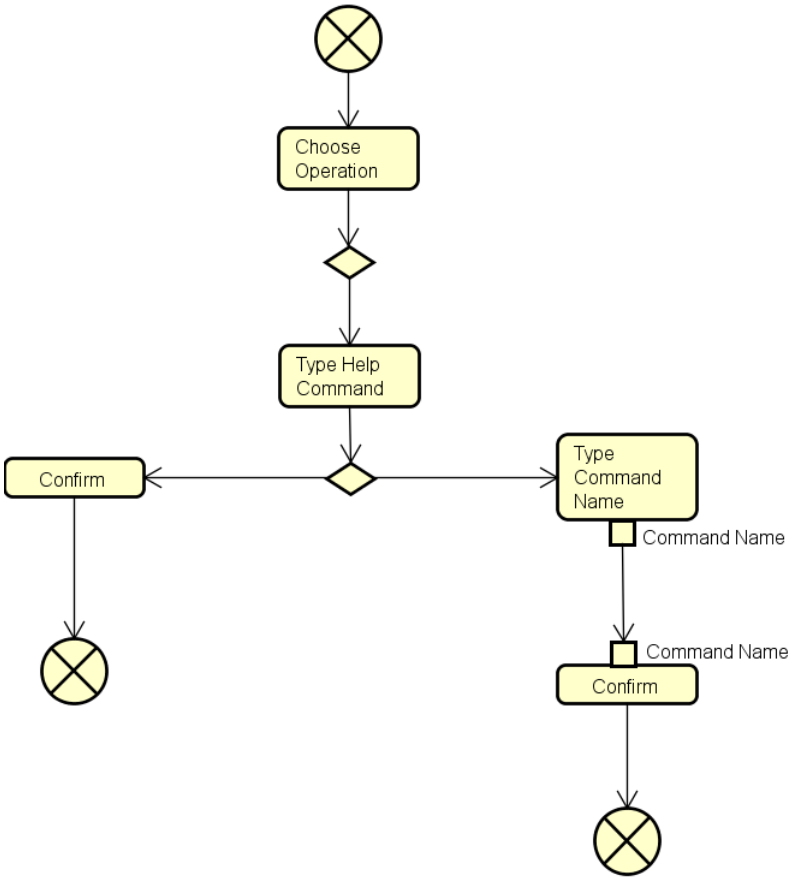


Figura 30: Diagramma attività operazioni offline

L'utente sceglie se chiedere un aiuto generale, o riguardante un comando specifico, nel primo caso è sufficiente che digiti il comando di aiuto e lo confermi, altrimenti dovrà successivamente scrivere il comando del quale vuole avere chiarimenti e poi confermare.

5.0.3 Connect to Server

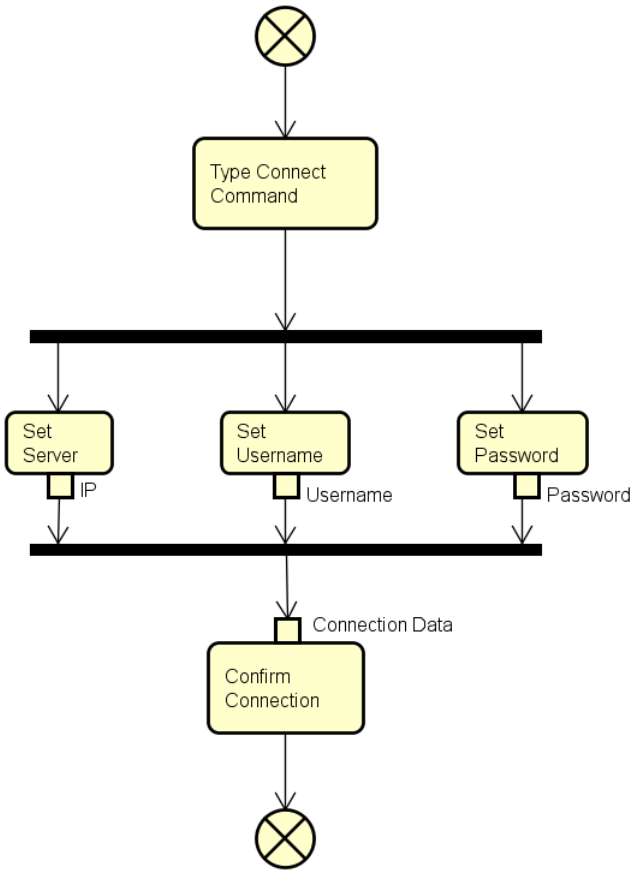


Figura 31: Diagramma attività connessione ad un server

Per effettuare una connessione l'utente dovrà digitare il commando di connessione e successivamente fornire i dati per la connessione, nello specifico: l'indirizzo IP al quale ci si vuole connettere, il nome utente e la password. Dopo di che è sufficiente che confermi il comando.

5.0.4 Online Operation

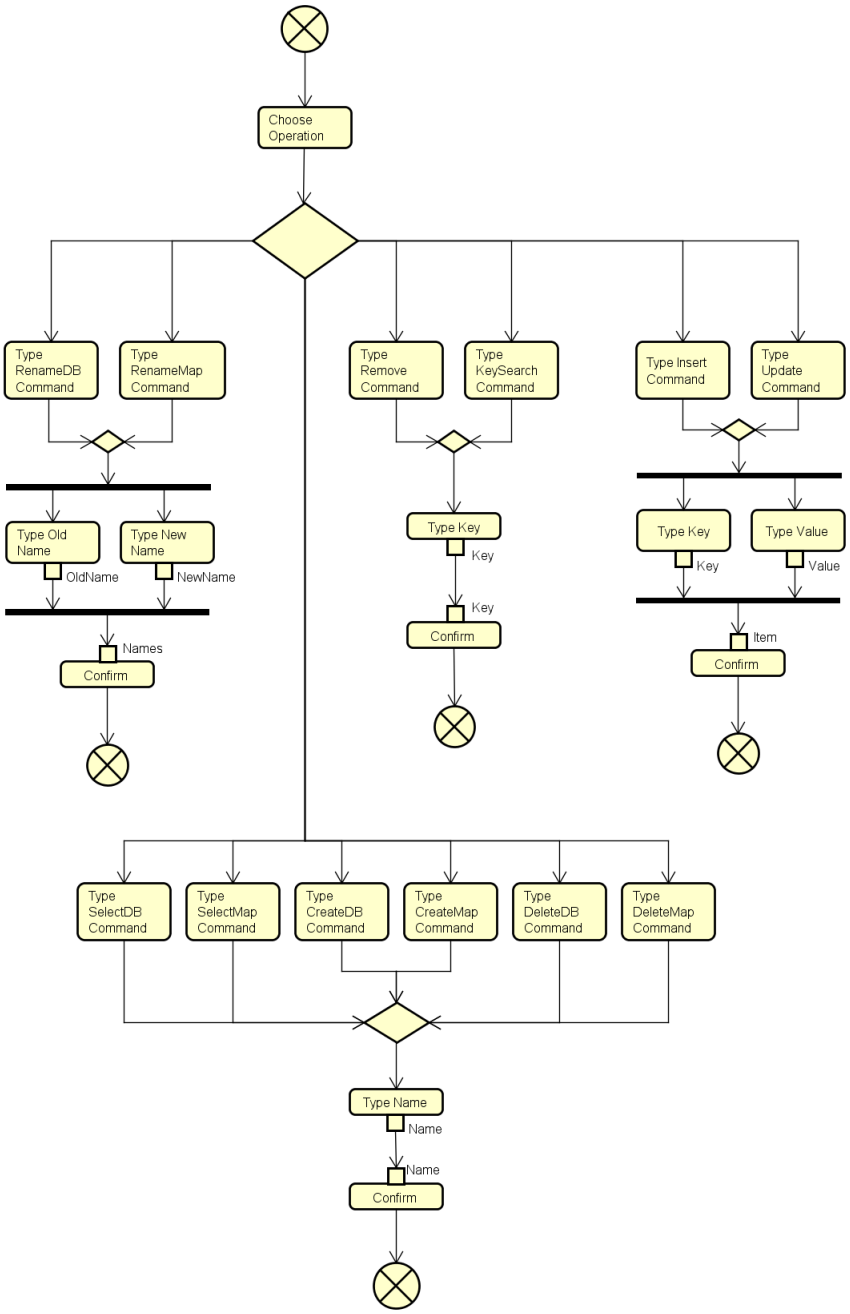


Figura 32: Diagramma attività operazioni online

L'utente sceglie che tipo di operazione vuole effettuare, se vuole effettuare un' operazione di rinomina su un Database o una mappa è sufficiente che fornisca il comando di rinomina, il vecchio e il nuovo nome e che poi confermi l'operazione; se vuole effettuare un' operazione di selezione o di creazione o di cancellazione sia di un Database che di una mappa, deve digitare il comando corretto e successivamente il nome del Database o della mappa sulla quale vuole effettuare l'operazione dopo di che è sufficiente che confermi il comando; se vuole effettuare un' operazione di rimozione o selezione di un dato è necessario che digiti il comando desiderato, digiti la chiave completa e che confermi l'operazione; se vuole effettuare un inserimento di un dato o un aggiornamento di un dato deve inserire il comando corretto, digitare la chiave completa e confermare l'operazione.

6 Diagrammi di sequenza

In questa sezione verranno illustrati e descritti i principali diagrammi di sequenza realizzati. Questi ultimi illustrano ad alto livello le azioni compiute dal server e dal client per gestire tutte le richieste di un utente.

Per esplicitare l'invio di un messaggio tra due attori, nei diagrammi viene chiamato il metodo *receive()* dell'attore$_G$ che riceve il messaggio da parte di chi lo invia. Questo è dovuto al fatto che tutte le classi ereditano dall'interfaccia *Actor* messa a disposizione da *Akka* senza effettuare l'override del metodo per inviare i messaggi; mentre sovrascrivono il metodo di ricezione per gestire i messaggi in entrata.

Trattandosi di diagrammi di sequenza non molto specifici, alcuni attori effettuano operazioni senza chiamare dei metodi specifici già dichiarati. Un esempio si può vedere nel diagramma riguardante la chiusura, in esso infatti, tutti gli attori effettuano una chiamata ad un metodo *chiusura()*.

6.1 Avvio

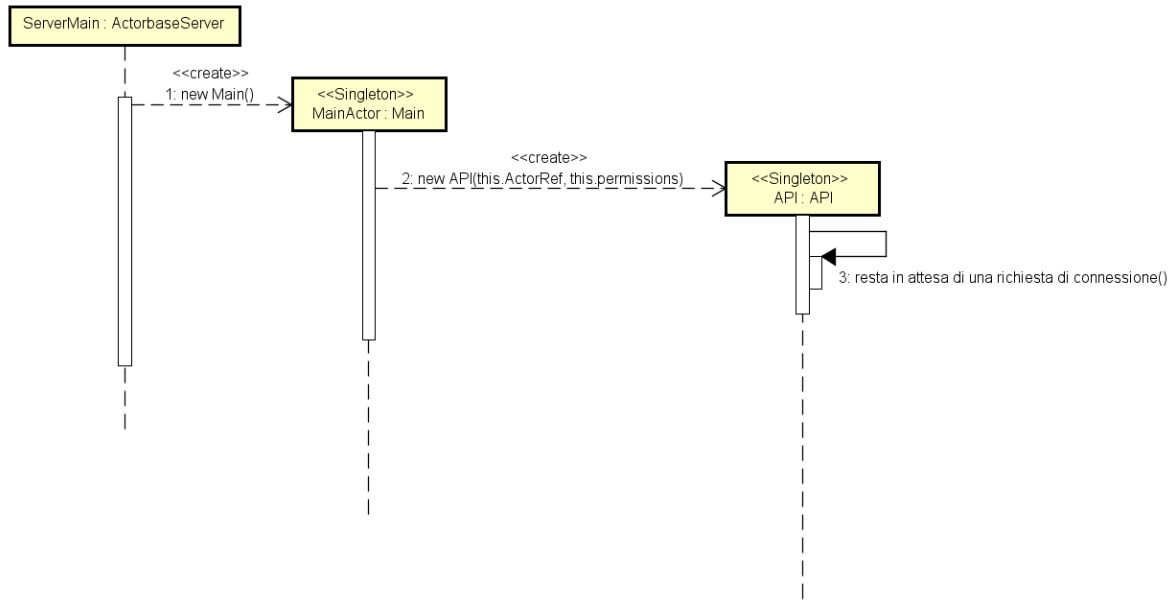


Figura 33: Diagramma di sequenza - avvio del server principale.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite per avviare il server. Esse partono dall'apertura della shell, ed effettuano le prime operazioni necessarie avviando gli attori Main$_G$ e API$_G$.

6.2 Chiusura

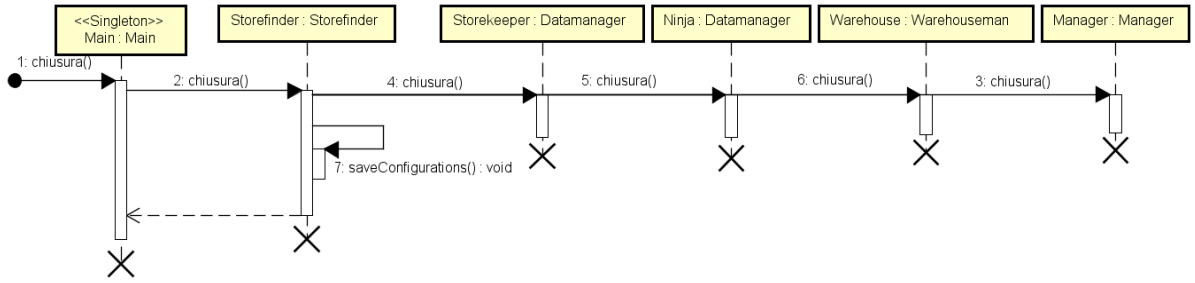


Figura 34: Diagramma di sequenza - chiusura del server.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite alla chiusura del server. La classe ActorbaseServer quando viene inserito il comando di chiusura del server dalla shell di comando, chiude l'attore$_G$ Main$_G$, che a sua volta, chiude tutti i suoi attori figli tramite il metodo di chiusura messo a disposizione dalla libreria *Akka*. Una volta chiusi tutti i suoi figli (nel diagramma è espresso con l'invio di una richiesta di *chiusura* ai suoi figli), l'attore$_G$ Main$_G$ termina, effettuando i salvataggi delle configurazioni necessarie.

6.3 Richiesta esterna

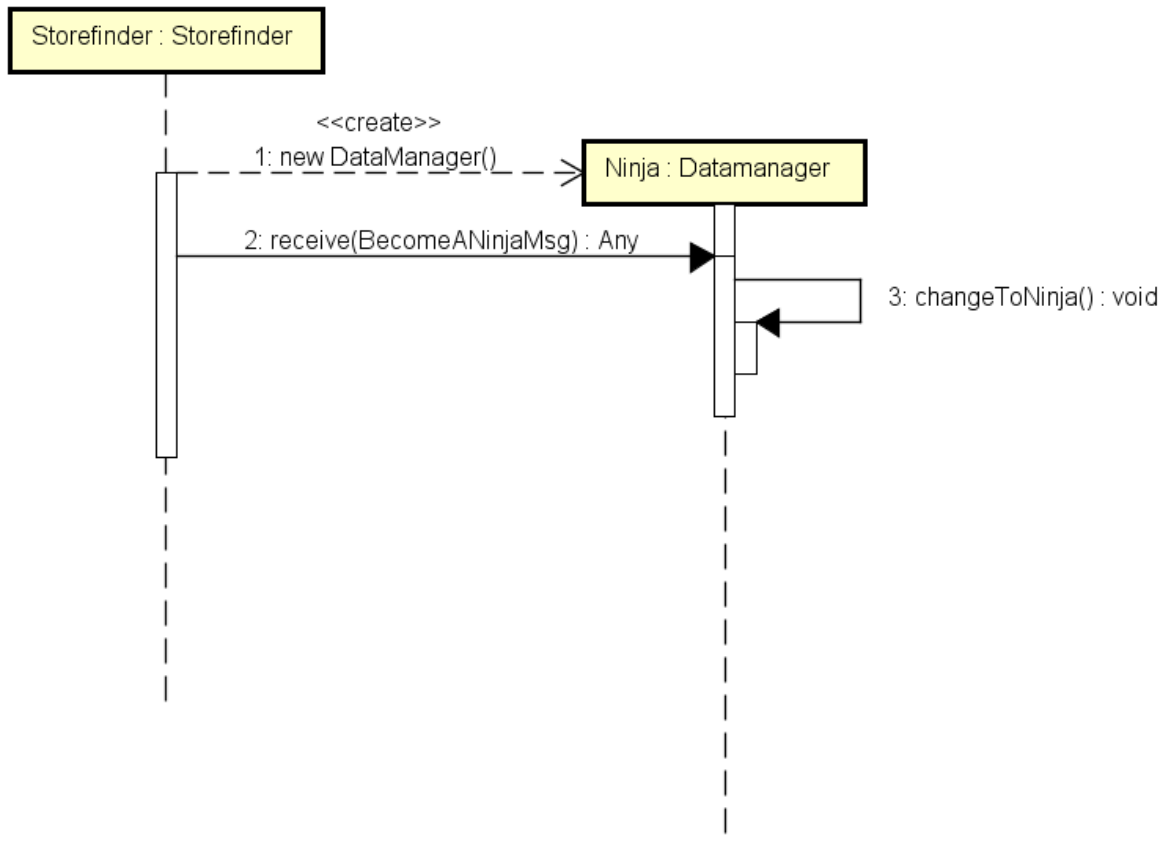


Figura 35: Diagramma di sequenza - gestione di richiesta esterna ad alto livello.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite da quando le API$_G$ ricevono una richiesta da un client esterno. Le API$_G$ inviano la richiesta HTTP all'attore$_G$ Main$_G$, che crea un attore MessagesBuilder che interpreta il messaggio, genera un messaggio tra quelli definiti nel package **Actorbase.Server.Core.Messages** e lo restituisce tramite una future al Main$_G$. Una volta ricevuta la risposta, l'attore$_G$ Main gestisce il messaggio (nei diagrammi seguenti vengono specificate più nel dettaglio le operazioni effettuate in base al tipo di messaggio ricevuto). Una volta terminata la gestione del messaggio, crea un HTTPBuilder a cui manda la risposta da trasformare in HTTP, una volta trasformata la rimanda alle API$_G$, che la restituiscono al Client che ha effettuato la richiesta.

6.4 Richiesta di creazione DB

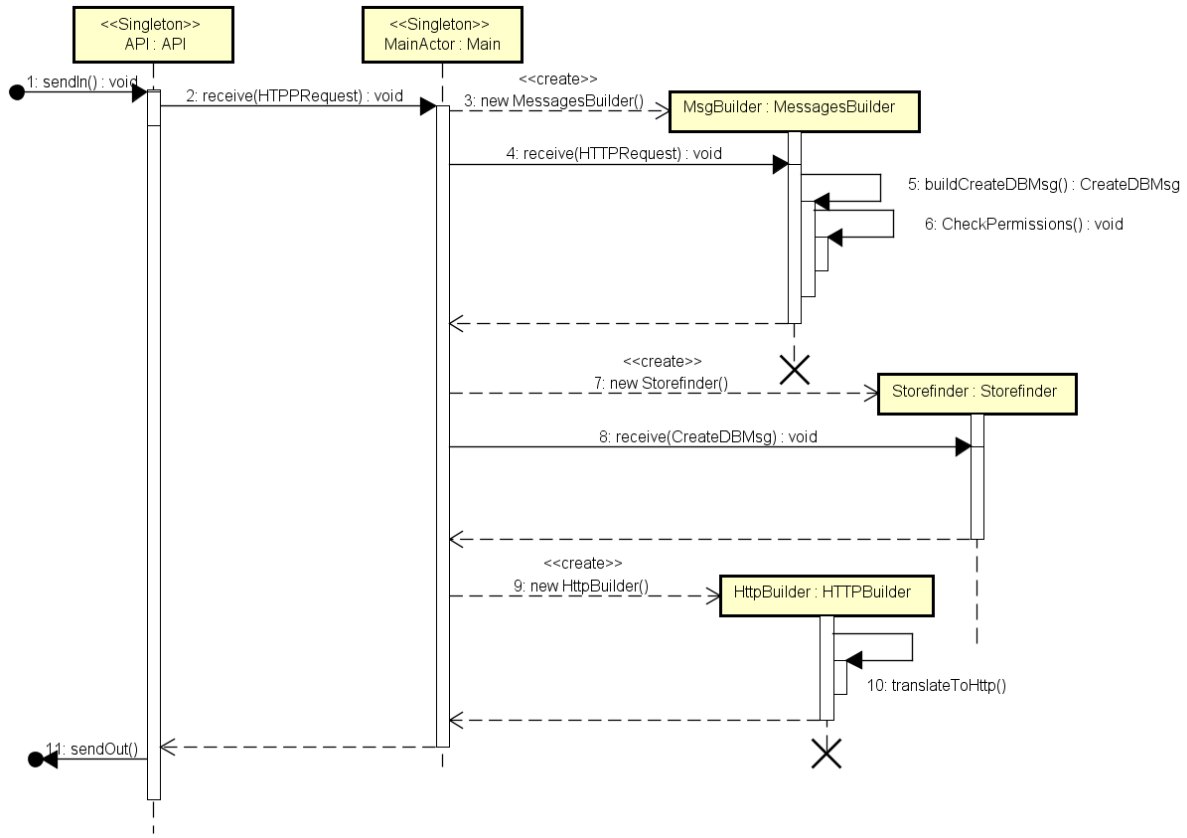


Figura 36: Diagramma di sequenza - gestione di richiesta di creazione DB.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite quando il Main_G riceve una richiesta di creazione di un DB. Come illustrato nella sezione precedente, viene trasformata la richiesta HTTP in un messaggio tramite un attore_G MessagesBuilder. Una volta ricevuto questo messaggio, l'attore_G Main_G crea un attore Storefinder_G a cui assegna quel DB. Genera la risposta da restituire al Client tramite HTTPBuilder e la spedisce all'esterno tramite API_G.

6.5 Richiesta di find

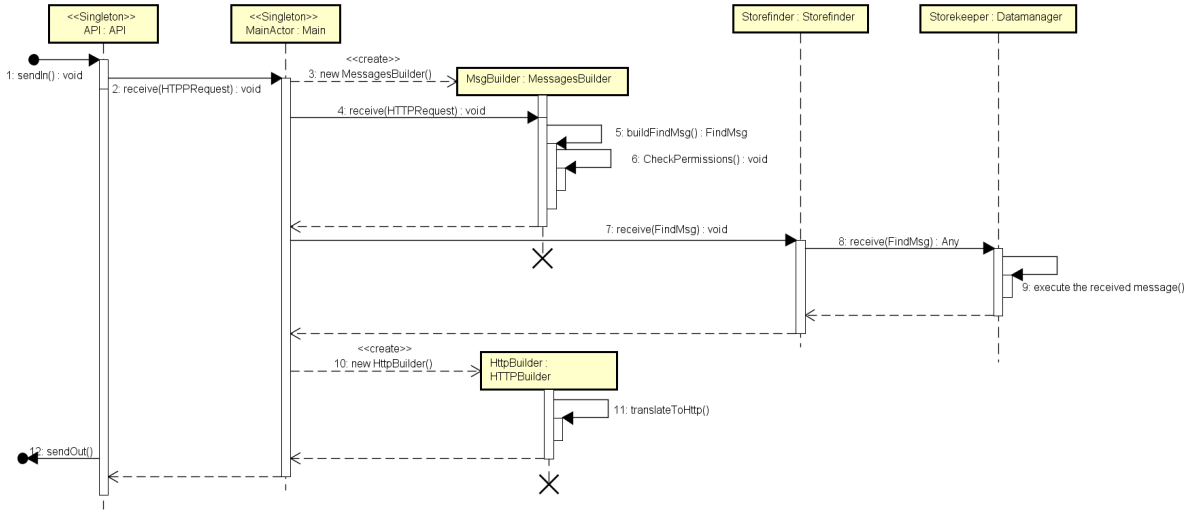


Figura 37: Diagramma di sequenza - gestione di richiesta di Find.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite per gestire una richiesta di find. L'attore$_G$ Main$_G$ instrada il messaggio all'attore$_G$ Storefinder$_G$ corrispondente, il quale inoltra la richiesta all'attore$_G$ Storekeeper$_G$ di competenza, che interpreta la sua mappa e restituisce la risposta. L'attore$_G$ Main$_G$ risponde alle API$_G$ come specificato nei diagrammi precedenti.

6.6 Richiesta di aggiornamento item

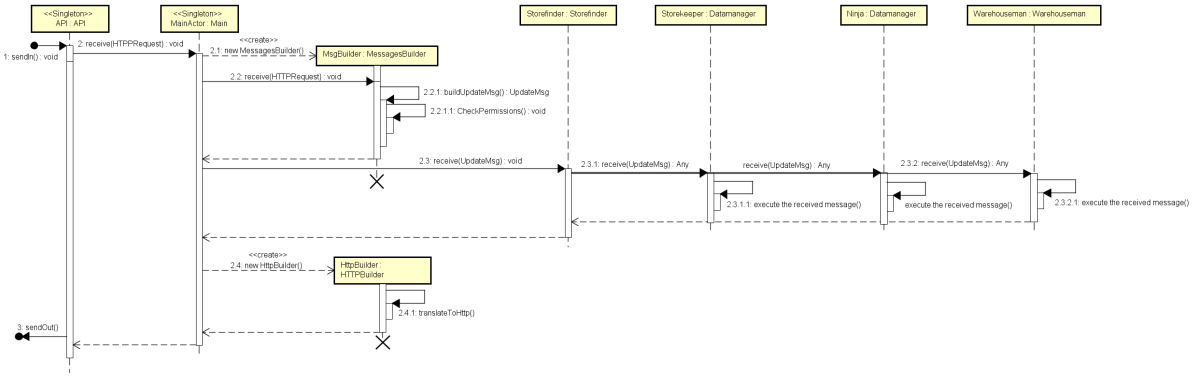


Figura 38: Diagramma di sequenza - gestione di richiesta di aggiornamento item.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite per gestire una richiesta di aggiornamento di un item all'interno di una mappa. La richiesta, arrivata allo Storekeeper$_G$ come nel caso precedentemente visto, viene gestita in locale (la mappa viene aggiornata). Inoltre, in questo caso, l'attore$_G$ Storefinder$_G$ inoltra in messaggio di aggiornamento anche agli attori Ninja$_G$ e Warehouseman$_G$ corrispondendi allo Storekeeper$_G$ corretto. L'attore$_G$ Ninja$_G$ aggiorna come lo Storekeeper$_G$ la sua mappa interna, mentre l'attore$_G$ Warehouseman$_G$ si occupa di aggiornare l'item su disco, in modo da garantire persistenza dei dati. La risposta torna allo Storefinder$_G$, poi all'attore$_G$ Main$_G$ che come nei casi precedenti lo fa trasformare in HTTP e lo inoltra all'esterno tramite API$_G$.

6.7 Creazione Ninja

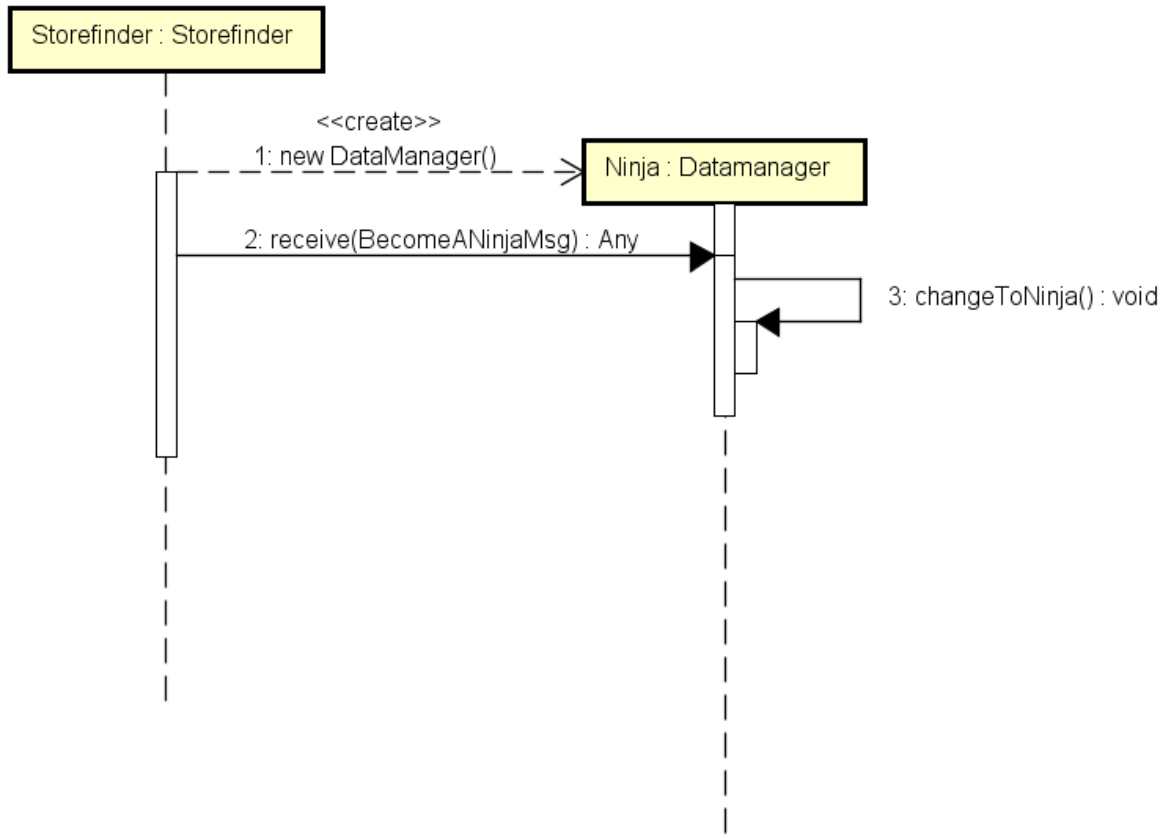


Figura 39: Diagramma di sequenza - creazione di un attore Ninja.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite per la creazione di un attore$_G$ di tipo Ninja$_G$. L'attore$_G$ Storefinder$_G$ crea un attore di tipo Datamanger, e immediatamente dopo gli invia un messaggio dedito alla sua trasformazione. L'attore$_G$ Datamanager non fa altro che chiamare un suo metodo, che tramite il metodo *become()* fornito dalla libreria *Akka* permette di cambiare il comportamento di un attore$_G$ alla ricezione di un messaggio.

6.8 Creazione Storekeeper

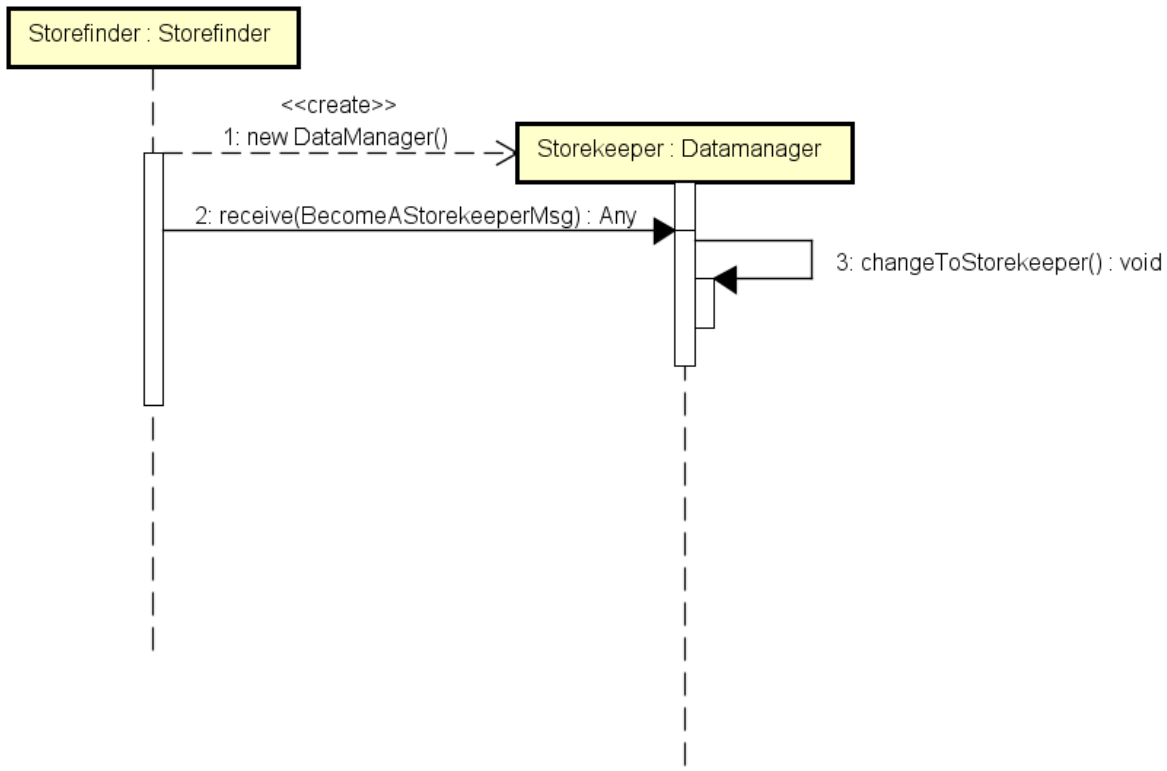


Figura 40: Diagramma di sequenza - creazione di un attore Storekeeper.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite per la creazione di un attore$_G$ di tipo Storekeeper$_G$. L'attore$_G$ Storefinder$_G$ crea un attore di tipo Datamanger, e immadiatamente dopo gli invia un messaggio dedito alla sua trasformazione. L'attore$_G$ Datamanager, come per il caso precedente, tramite il metodo *become()* cambia il suo comportamento.

6.9 Sostituzione di uno Storekeeper

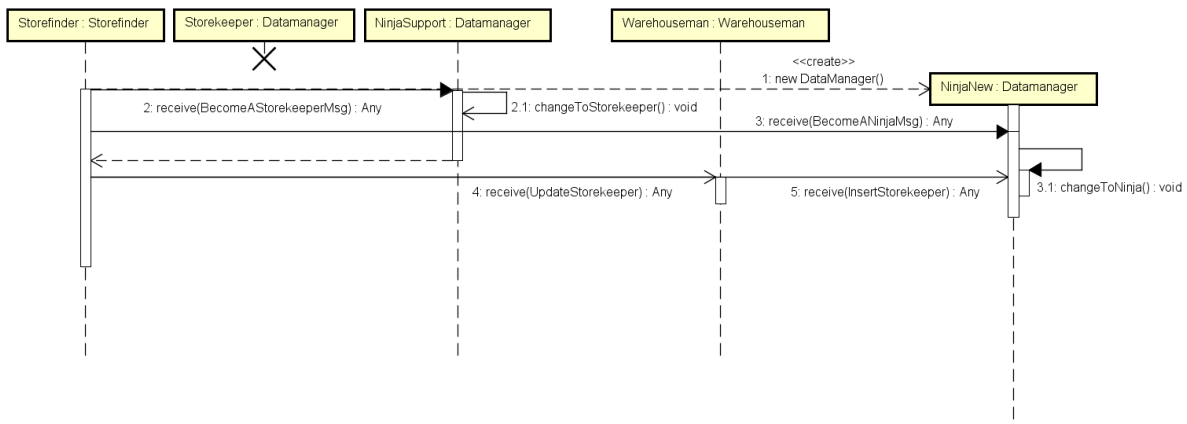


Figura 41: Diagramma di sequenza - sostituzione di un attore Storekeeper.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite quando è necessario sostiuire uno Storekeeper$_G$ che si è arrestato in modo anomalo. L'attore$_G$ Storefinder$_G$ manda un messaggio al Ninja$_G$ di supporto dello Storekeeper$_G$ morto in modo da fargli prendere il suo posto. Successivamente crea un nuovo attore$_G$ Ninja$_G$ (come spiegato nel diagramma di sequenza visto prima) e notifica gli attori già esistenti del cambio avvenuto.

6.10 Sequenza di un comando in client e driver

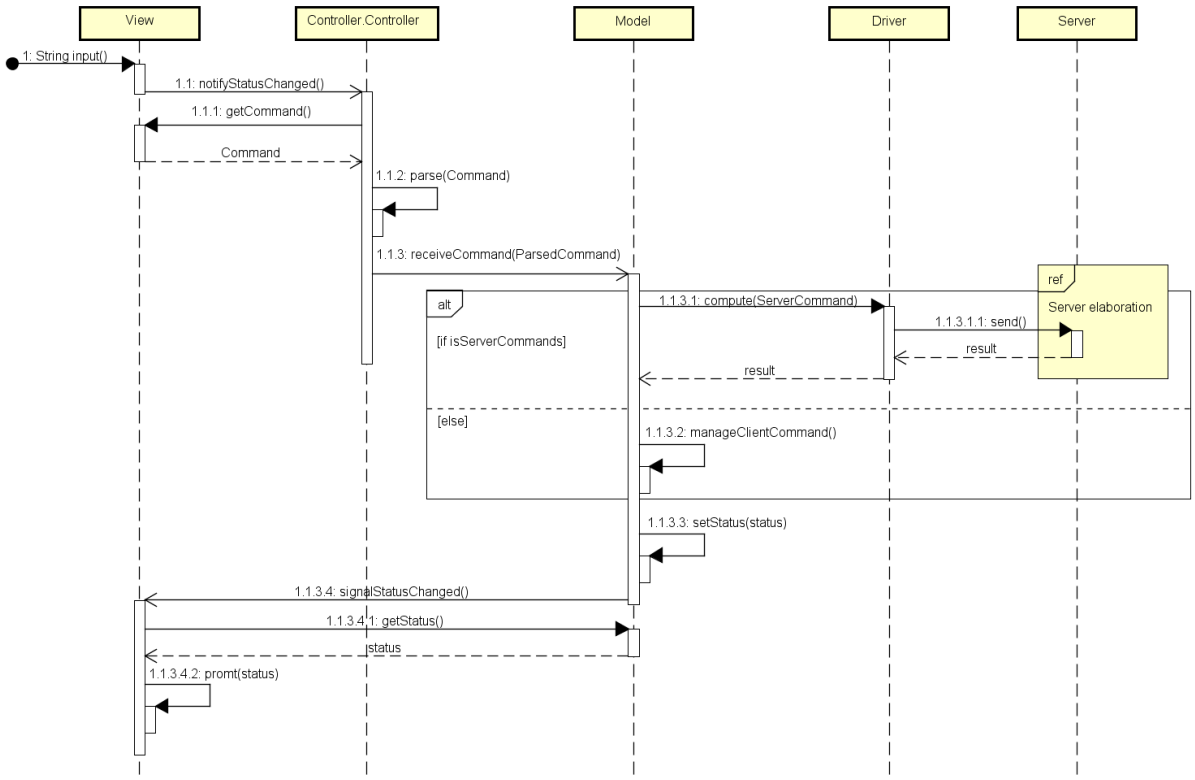


Figura 42: Diagramma di sequenza - sequenza di un comando in client e driver.

Nel diagramma precedente è possibile visualizzare quali sono le operazioni che vengono eseguite quando viene inserito un comando nella shell del client e viene premuto invio.
La view notifica che il suo stato è cambiato e il controller prende la stringa inserita dall'utente per tradurla in un comando. Successivamente manda il comando al model.
A questo punto la sequenza prevede due diversi flussi: se il comando riguarda il server viene inviato al driver$_G$ che a sua volta lo invia al server per l'elaborazione. Quando il driver$_G$ riceve una risposta, la ritorna al model del client che era rimasto in attesa. Se invece il comando non riguarda il server, come ad esempio il comando `HELP`, il model lo elabora al suo interno. In entrambi i casi alla fine viene generato ed impostato un nuovo stato del model e notificato il cambiamento. La view quindi prende il nuovo stato e aggiorna il proprio di conseguenza, rendendolo visibile sulla shell per l'utente.

7 Stime di fattibilità e di bisogno di risorse

L'architettura definita fino a questo punto è sufficiente per fornire una stima della fattibilità del prodotto e delle risorse richieste per la realizzazione.
Il gruppo inizialmente non aveva conoscenze sufficienti per stimare in modo appropriato la complessità dell'implementazione di un Database basato sulla logica ad attori. Grazie al livello di dettaglio raggiunto sono stati fugati molti dei dubbi e delle incertezze a riguardo, confermando le previsioni sull'esito positivo del progetto.
Sono state inoltre individuate con chiarezza le risorse tecnologiche che verranno utilizzate:

- Akka$_G$: libreria per modello ad attori.
- IntelliJ: framework per la stesura del codice.
- JVM: piattaforma per il funzionamento di Scala$_G$.

Il gruppo in contemporanea si è dedicato allo studio delle nuove tecnologie raggiungendo un buon livello di conoscenza. L'insieme di queste risorse potrà garantire la realizzazione di tutte le componenti dell'architettura.

8 Tracciamento

8.1 Tracciamento componenti-requisiti

Componente	Requisiti
Actorbase.Server	R[1][N][F] e figli
Actorbase.Server.API	R[1.3][N][F] e figli
	R[1.4.1.3][D][F]
	R[1.4.4.3][N][F]
	R[1.4.4.3][N][F]
	R[1.4.5.3][N][F]
	R[1.4.4.4][N][F]
	R[1.4.6.4][D][F]
	R[1.4.6.5][D][F]
	R[1.4.7.3][N][F]
	R[1.4.7.4][N][F]
	R[1.5.1.2][D][F]
	R[1.5.2.3][N][F]
	R[1.5.2.4][N][F]
	R[1.5.3.3][N][F]
	R[1.5.3.4][N][F]
	R[1.5.4.4][O][F]
	R[1.5.4.5][O][F]
	R[1.5.5.3][N][F]
	R[1.5.5.4][N][F]
	R[1.6.1.2][D][F]
	R[1.6.2.3][N][F]
	R[1.6.2.4][N][F]
	R[1.6.3.3][N][F]
	R[1.6.3.4][N][F]
	R[1.6.4.3][N][F]
	R[1.6.4.4][N][F]
	R[1.6.6.3][N][F]
	R[1.6.6.4][N][F]
Actorbase.Server.Core	R[1.4][N][F] e figli
	R[1.5][N][F] e figli
	R[1.6][N][F] e figli
Actorbase.Server.Core.Actors	R[1.4][N][F] e figli
	R[1.5][N][F] e figli
	R[1.6][N][F] e figli

Componente	Requisiti
Actorbase.Server.Core.Actors.DataManagement	R[1.4.4.2.3][N][F]
	R[1.4.4.2.4][N][F]
	R[1.4.4.2.5][N][F]
	R[1.4.5.2.3][N][F]
	R[1.4.5.2.4][N][F]
	R[1.4.5.2.5][N][F]
	R[1.4.6.3.3][N][F]
	R[1.4.6.3.4][N][F]
	R[1.4.6.3.5][N][F]
	R[1.5.2.2.3][N][F]
	R[1.5.2.2.4][N][F]
	R[1.5.2.2.5][O][F]
	R[1.5.3.2.3][N][F]
	R[1.5.3.2.4][N][F]
	R[1.5.3.2.5][O][F]
	R[1.5.4.3.3][N][F]
	R[1.5.4.3.4][N][F]
	R[1.5.4.3.5][O][F]
	R[1.6.2.1][N][F]
	R[1.6.2.2.3][N][F]
	R[1.6.2.2.4][N][F]
	R[1.6.2.3][N][F]
	R[1.6.3.2.3][N][F]
	R[1.6.3.2.4][N][F]
	R[1.6.3.2.5][O][F]
	R[1.6.4.1][N][F]
	R[1.6.3.2.3][N][F]
	R[1.6.3.2.4][N][F]
	R[1.6.3.2.5][O][F]
	R[1.6.6.1][N][F]
	R[1.6.6.2.3][N][F]
	R[1.6.6.2.4][N][F]
	R[1.6.6.2.5][O][F]
Actorbase.Server.Core.Actors.Manager	R[1.4.4.2.6][N][F]
	R[1.4.5.2.6][N][F]
	R[1.4.6.3.6][N][F]
	R[1.5.2.2.6][O][F]
	R[1.5.3.2.6][O][F]
	R[1.5.4.3.6][O][F]
	R[1.6.3.2.6][O][F]
	R[1.6.6.2.6][O][F]

Componente	Requisiti
Actorbase.Server.Core.Actors.StoreFinder	R[1.4.1][D][F] e figli R[1.4.4.1][D][F] R[1.4.4.1.1][D][F] R[1.4.4.2.1][N][F] R[1.4.4.2.2][N][F] R[1.4.5.1][D][F] R[1.4.5.1.1][D][F] R[1.4.5.1.2][N][F] R[1.4.5.2.2][N][F] R[1.4.6.1][D][F] R[1.4.6.1.1][D][F] R[1.4.6.2][D][F] R[1.4.6.2.1][D][F] R[1.4.6.3.1][N][F] R[1.4.6.3.2][N][F] R[1.4.7.1][D][F] R[1.4.7.1.1][D][F] R[1.4.7.2][N][F] R[1.4.7.2.1][N][F] R[1.5.1.1][D][F] R[1.5.2.1][D][F] R[1.5.2.1.1][D][F] R[1.5.2.2.1][N][F] R[1.5.2.2.2][N][F] R[1.5.3.1][D][F] R[1.5.3.1.1][D][F] R[1.5.3.2.1][N][F] R[1.5.3.2.2][N][F] R[1.5.4.1][D][F] R[1.5.4.1.1][D][F] R[1.5.4.2][D][F] R[1.5.4.2.1][D][F] R[1.5.4.3.1][N][F] R[1.5.4.3.2][N][F] R[1.5.5.1][D][F] R[1.5.5.1.1][D][F] R[1.5.5.2][N][F] R[1.5.5.2.1][N][F] R[1.6.1.1][D][F] R[1.6.2.2.1][N][F] R[1.6.2.2.2][N][F] R[1.6.3.2.1][N][F] R[1.6.3.2.2][N][F] R[1.6.4.2.1][N][F] R[1.6.4.2.2][N][F] R[1.6.6.2.1][N][F] R[1.6.6.2.2][N][F]
Actorbase.Server.Core.Messages	R[1.4][N][F] e figli R[1.5][N][F] e figli R[1.6][N][F] e figli
Actorbase.Driver	R[3][N][F] e figli
Actorbase.Client	R[2][N][F] e figli

Componente	Requisiti
Actorbase.Client.Model	R[2.1.3][N][F] R[2.2.3][N][F] R[2.3.4.2][D][F] R[2.4.3][D][F] R[2.7.3][N][F] R[2.8.3][N][F] R[2.9.3][N][F] R[2.10.3][N][F] R[2.11.3][D][F] R[2.12.3][N][F] R[2.13.3][N][F] R[2.14.3][D][F] R[2.15.3][N][F] R[2.18.3][D][F] R[2.19.3][N][F] R[2.20.3][N][F] R[2.21.3][N][F] R[2.23.3][N][F]
Actorbase.Client.View	R[2.1.5][N][F] R[2.1.6][N][F] R[2.2.4][N][F] R[2.3.2.2][D][F] R[2.3.4.3][D][F] R[2.4.4][D][F] R[2.7.4][N][F] R[2.7.5][N][F] R[2.8.4][N][F] R[2.8.5][N][F] e figli R[2.9.4][N][F] R[2.9.5][N][F] e figli R[2.10.4][N][F] R[2.10.5][N][F] e figli R[2.11.4][D][F] R[2.12.4][N][F] R[2.12.5][N][F] e figli R[2.13.4][N][F] R[2.13.5][N][F] R[2.14.4][D][F] R[2.14.5][D][F] e figli R[2.15.4][N][F] R[2.15.5][N][F] con figli R[2.18.4][D][F] R[2.19.4][N][F] R[2.20.4][N][F] R[2.20.5][N][F] e figli R[2.21.4][N][F] R[2.21.5][N][F] R[2.23.4][N][F] R[2.23.5][N][F] e figli

Componente	Requisiti
Actorbase.Client.Controller	R[2.1.1][N][F]
	R[2.1.2][N][F] e figli
	R[2.2.1][N][F]
	R[2.2.2][N][F]
	R[2.3.1][D][F]
	R[2.3.2.1][D][F]
	R[2.3.3][D][F]
	R[2.3.4.1][D][F]
	R[2.4.1][D][F]
	R[2.4.2][D][F]
	R[2.7.1][N][F]
	R[2.7.2][N][F] e figli
	R[2.8.1][N][F]
	R[2.8.2][N][F] e figli
	R[2.9.1][N][F]
	R[2.9.2][N][F] e figli
	R[2.10.1][N][F]
	R[2.10.2][N][F] e figli
	R[2.11.1][D][F]
	R[2.11.2][D][F]
	R[2.12.1][N][F]
	R[2.12.2][N][F] e figli
	R[2.13.1][N][F]
	R[2.13.2][N][F] e figli
	R[2.14.1][D][F]
	R[2.14.2][D][F] e figli
	R[2.15.1][N][F]
	R[2.15.2][N][F] e figli
	R[2.18.1][D][F]
	R[2.18.2][D][F]
	R[2.19.1][N][F]
	R[2.19.2][N][F] e figli
	R[2.20.1][N][F]
	R[2.20.2][N][F] e figli
	R[2.21.1][N][F]
	R[2.21.2][N][F] e figli
	R[2.23.1][N][F]
	R[2.23.2][N][F] e figli

Tabella 2: Componenti-Requisiti

8.2 Tracciamento requisiti-componenti

Requisito	Componenti
R[1][N][F] e figli	Actorbase.Server
R[1.3][N][F] e figli	Actorbase.Server.API
R[1.4][N][F] e figli	Actorbase.Server.Core
	Actorbase.Server.Core.Actors
	Actorbase.Server.Core.Messages
R[1.4.1][D][F] e figli	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.1.3][D][F]	Actorbase.Server.API
R[1.4.4.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.4.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.4.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.4.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.4.2.3][N][F]	Actorbase.Server.Core.Actors.DataManagement

Requisito	Componenti
R[1.4.4.2.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.4.2.5][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.4.2.6][N][F]	Actorbase.Server.Core.Actors.Manager
R[1.4.4.3][N][F]	Actorbase.Server.API
R[1.4.4.4][N][F]	Actorbase.Server.API
R[1.4.5.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.5.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.5.1.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.5.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.5.2.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.5.2.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.5.2.5][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.5.2.6][N][F]	Actorbase.Server.Core.Actors.Manager
R[1.4.5.3][N][F]	Actorbase.Server.API
R[1.4.6.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.6.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.6.2][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.6.2.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.6.3.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.6.3.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.6.3.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.6.3.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.6.3.5][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.4.6.3.6][N][F]	Actorbase.Server.Core.Actors.Manager
R[1.4.6.4][D][F]	Actorbase.Server.API
R[1.4.6.5][D][F]	Actorbase.Server.API
R[1.4.7.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.7.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.7.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.7.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.4.7.3][N][F]	Actorbase.Server.API
R[1.4.7.4][N][F]	Actorbase.Server.API
R[1.5][N][F] e figli	Actorbase.Server.Core Actorbase.Server.Core.Actors Actorbase.Server.Core.Messages
R[1.5.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.1.2][D][F]	Actorbase.Server.API
R[1.5.2.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.2.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.2.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.2.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.2.2.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.2.2.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.2.2.5][O][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.2.2.6][O][F]	Actorbase.Server.Core.Actors.Manager
R[1.5.2.3][N][F]	Actorbase.Server.API
R[1.5.2.4][N][F]	Actorbase.Server.API
R[1.5.3.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.3.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.3.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.3.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.3.2.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.3.2.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.3.2.5][O][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.3.2.6][O][F]	Actorbase.Server.Core.Actors.Manager

Requisito	Componenti
R[1.5.3.3][N][F]	Actorbase.Server.API
R[1.5.3.4][N][F]	Actorbase.Server.API
R[1.5.4.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.4.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.4.2][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.4.2.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.4.3.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.4.3.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.4.3.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.4.3.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.4.3.5][O][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.5.4.3.6][O][F]	Actorbase.Server.Core.Actors.Manager
R[1.5.4.4][O][F]	Actorbase.Server.API
R[1.5.4.5][O][F]	Actorbase.Server.API
R[1.5.5.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.5.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.5.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.5.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.5.5.3][N][F]	Actorbase.Server.API
R[1.5.5.4][N][F]	Actorbase.Server.API
R[1.6][N][F] e figli	Actorbase.Server.Core Actorbase.Server.Core.Actors Actorbase.Server.Core.Messages
R[1.6.1.1][D][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.1.2][D][F]	Actorbase.Server.API
R[1.6.2.1][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.2.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.2.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.2.2.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.2.2.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.2.3][N][F]	Actorbase.Server.API Actorbase.Server.Core.Actors.DataManagement
R[1.6.2.4][N][F]	Actorbase.Server.API
R[1.6.3.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.3.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.3.2.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.3.2.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.3.2.5][O][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.3.2.6][O][F]	Actorbase.Server.Core.Actors.Manager
R[1.6.3.3][N][F]	Actorbase.Server.API
R[1.6.3.4][N][F]	Actorbase.Server.API
R[1.6.4.1][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.4.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.4.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.4.3][N][F]	Actorbase.Server.API
R[1.6.4.4][N][F]	Actorbase.Server.API
R[1.6.6.1][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.6.2.1][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.6.2.2][N][F]	Actorbase.Server.Core.Actors.StoreFinder
R[1.6.6.2.3][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.6.2.4][N][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.6.2.5][O][F]	Actorbase.Server.Core.Actors.DataManagement
R[1.6.6.2.6][O][F]	Actorbase.Server.Core.Actors.Manager
R[1.6.6.3][N][F]	Actorbase.Server.API
R[1.6.6.4][N][F]	Actorbase.Server.API

Requisito	Componenti
R[2][N][F] e figli	Actorbase.Client
R[2.1.1]	
R[2.1.2][N][F] e	
R[2.1.3][N][F]	Actorbase.Client.Model
R[2.1.5][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.1.6][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.2.1]	Actorbase.Client.Controller
R[2.2.2]	Actorbase.Client.Controller
R[2.2.3][N][F]	Actorbase.Client.Model
R[2.2.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.3.1]	Actorbase.Client.Controller
R[2.3.2.1]	Actorbase.Client.Controller
R[2.3.2.2][D][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.3.3]	Actorbase.Client.Controller
R[2.3.4.1]	Actorbase.Client.Controller
R[2.3.4.2][D][F]	Actorbase.Client.Model
R[2.3.4.3][D][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.4.1]	Actorbase.Client.Controller
R[2.4.2]	Actorbase.Client.Controller
R[2.4.3][D][F]	Actorbase.Client.Model
R[2.4.4][D][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.7.1]	Actorbase.Client.Controller
R[2.7.2][N][F] e figli	Actorbase.Client.Controller
R[2.7.3][N][F]	Actorbase.Client.Model
R[2.7.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.7.5][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.8.1]	Actorbase.Client.Controller
R[2.8.2][N][F] e figli	Actorbase.Client.Controller
R[2.8.3][N][F]	Actorbase.Client.Model
R[2.8.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.8.5][N][F] e figli	Actorbase.Client.View Actorbase.Client.Controller
R[2.9.1]	Actorbase.Client.Controller
R[2.9.2][N][F] e figli	Actorbase.Client.Controller
R[2.9.3][N][F]	Actorbase.Client.Model
R[2.9.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.9.5][N][F] e figli	Actorbase.Client.View Actorbase.Client.Controller
R[2.10.1]	Actorbase.Client.Controller
R[2.10.2][N][F] e figli	Actorbase.Client.Controller
R[2.10.3][N][F]	Actorbase.Client.Model
R[2.10.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.10.5][N][F] e figli	Actorbase.Client.View Actorbase.Client.Controller

Requisito	Componenti
R[2.11.1]	Actorbase.Client.Controller
R[2.11.2]	Actorbase.Client.Controller
R[2.11.3][D][F]	Actorbase.Client.Model
R[2.11.4][D][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.12.1]	Actorbase.Client.Controller
R[2.12.2][N][F] e figli	Actorbase.Client.Controller
R[2.12.3][N][F]	Actorbase.Client.Model
R[2.12.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.12.5][N][F] e figli	Actorbase.Client.View Actorbase.Client.Controller
R[2.13.1]	Actorbase.Client.Controller
R[2.13.2][N][F] e figli	Actorbase.Client.Controller
R[2.13.3][N][F]	Actorbase.Client.Model
R[2.13.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.13.5][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.14.1]	Actorbase.Client.Controller
R[2.14.2][D][F] e figli	Actorbase.Client.Controller
R[2.14.3][D][F]	Actorbase.Client.Model
R[2.14.4][D][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.14.5][D][F] e figli	Actorbase.Client.View Actorbase.Client.Controller
R[2.15.1]	Actorbase.Client.Controller
R[2.15.2][N][F] e figli	Actorbase.Client.Controller
R[2.15.3][N][F]	Actorbase.Client.Model
R[2.15.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.15.5][N][F] con figli	Actorbase.Client.View Actorbase.Client.Controller
R[2.18.1]	Actorbase.Client.Controller
R[2.18.2]	Actorbase.Client.Controller
R[2.18.3][D][F]	Actorbase.Client.Model
R[2.18.4][D][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.19.1]	Actorbase.Client.Controller
R[2.19.2][N][F] e figli	Actorbase.Client.Controller
R[2.19.3][N][F]	Actorbase.Client.Model
R[2.19.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.20.1]	Actorbase.Client.Controller
R[2.20.2][N][F] e figli	Actorbase.Client.Controller
R[2.20.3][N][F]	Actorbase.Client.Model
R[2.20.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.20.5][N][F] e figli	Actorbase.Client.View Actorbase.Client.Controller
R[2.21.1]	Actorbase.Client.Controller
R[2.21.2][N][F] e figli	Actorbase.Client.Controller
R[2.21.3][N][F]	Actorbase.Client.Model
R[2.21.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller

Requisito	Componenti
R[2.21.5][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.23.1]	Actorbase.Client.Controller
R[2.23.2][N][F] e figli	Actorbase.Client.Controller
R[2.23.3][N][F]	Actorbase.Client.Model
R[2.23.4][N][F]	Actorbase.Client.View Actorbase.Client.Controller
R[2.23.5][N][F] e figli	Actorbase.Client.View Actorbase.Client.Controller
R[3][N][F] e figli	Actorbase.Driver

Tabella 3: Requisito-componente

9 Appendice

9.1 Descrizione Design Pattern

Segue, per ogni Design Pattern$_G$ utilizzato, la descrizione dello scopo, motivazione e applicabilità.

9.1.1 Event-driven

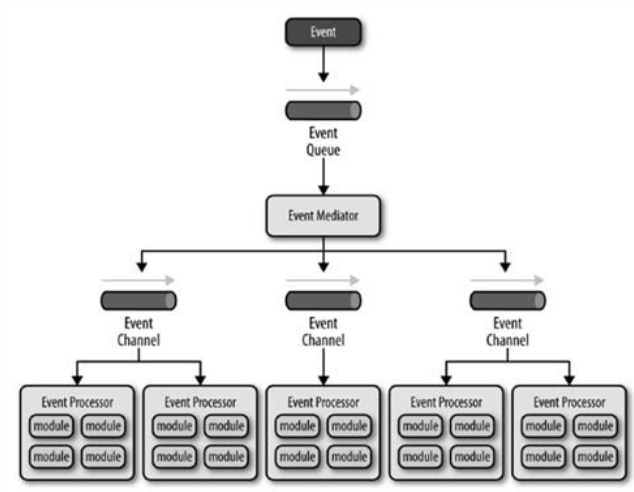


Figura 43: Diagramma del Design Pattern Event-driven

- **Scopo:** Produrre applicazioni molto scalabili e processare eventi asincroni disaccoppiati.
- **Motivazione:** Gestire le richieste che vengono volte all' applicativo tramite eventi processati in modo asincrono.
- **Applicabilità:** Gestione di eventi attraverso l'utilizzo di un mediatore e elaboratori di eventi

9.1.2 MVC

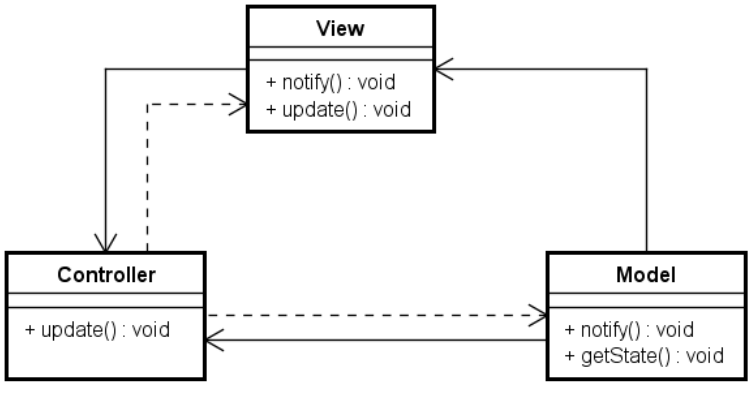


Figura 44: Diagramma del Design Pattern MVC

- **Scopo:** Disaccoppiamento delle seguenti componenti:
  - Model regole di accesso e dati di business
  - View rappresentazione grafica
  - Controller reazioni della UI agli input utente

- **Motivazione:** Lo scopo di molti applicativi è di recuperare dati e mostrarli all'Utente. Si è visto che la migliore soluzione di questo scopo è dividere la modellazione del dominio, la presentazione e le reazioni basate sugli input degli utenti i tre classi separate, esistono vari design pattern$_G$ che svolgono questa separazione, uno di questi è MVC;
- **Applicabilità:**
  - Applicazioni che devono presentare attraverso una UI un insieme di informazioni
  - Le persone responsabili dello sviluppo hanno compentenze differenti

9.1.3 Command

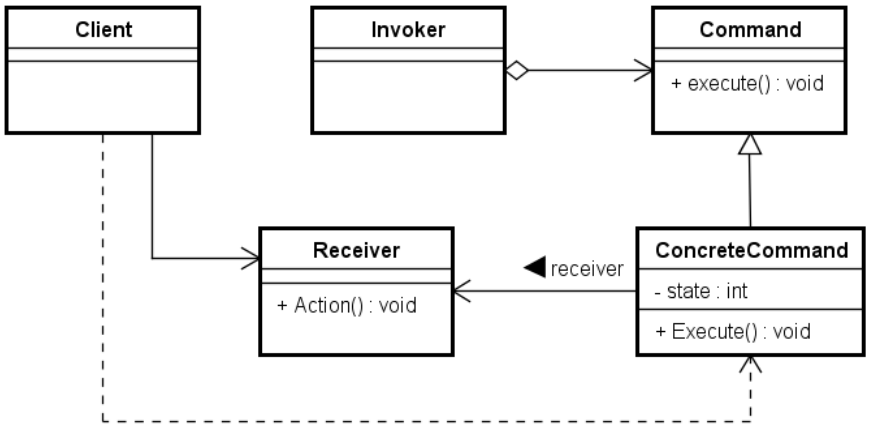


Figura 45: Diagramma del Design Pattern Command

- **Scopo:** Incapsulare una richiesta in un oggetto, cosicché i client siano indipendenti dalle richieste
- **Motivazione:** Risolvere la necessità di gestire richieste di sui non si conoscono i particolari, tramite una classe astratta, Command, che definisce un interfaccia per eseguire la richiesta
- **Applicabilità:**
  - Parametrizzazione di oggetti sull'azione da eseguire
  - Specificare, accordare ed eseguire richieste molteplici volte
  - Supporto ad operazioni di Undo e Redo
  - Supporto a transazione, un comando equivale ad una operazione atomica

9.1.4 Singleton

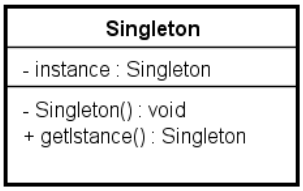


Figura 46: Diagramma del Design Pattern Command

- **Scopo:** Assicurare che una classe abbia una sola istanza con un unico punto di accesso globale.
- **Motivazione:** È necessario assicurare che esista una sola istanza di alcune classi. Una classe Singleton ha la responsabilità sulle proprie istanze, in modo che nessuna altra istanza possa essere creata, e fornisce un punto di accesso unico.

- **Applicabilità:**
  - Deve esistere una ed una sola istanza di una classe in tutta l'applicazione, accessibile dai client in modo noto.
  - L'istanza deve essere estendibile con ereditarietà, consentendo ai client di non modificare il proprio codice.

Elenco delle figure

Elenco delle tabelle