



Specifica Tecnica

Informazioni sul documento

Versione	3.0.0
Redazione	Andrea Giacomo Baldan
Verifica	Marco Boseggia
Approvazione	Davide Trevisan
Uso	Esterno
Lista di Distribuzione	ScalateKids Prof. Tullio Vardanega Prof. Riccardo Cardin



Diario delle modifiche

Versione	Autore	Ruolo	Data	Descrizione
3.0.0	Davide Trevisan	Responsabile	2016-06-15	Approvazione documento
2.1.0	Marco Boseggia	Verificatore	2016-06-03	Verifica Incremento
2.0.2	Andrea Giacomo Baldan	Progettista	2016-05-30	Incremento sezione 4.25 4.26
2.0.1	Andrea Giacomo Baldan	Progettista	2016-05-30	Incremento sezione 4.22 4.23 4.24
2.0.0	Francesco Agostini	Responsabile	2016-05-16	Approvazione documento
1.4.0	Alberto De Agostini	Verificatore	2016-05-03	Verifica sezione 4.25 e relative sotto-sezioni
1.3.1	Marco Boseggia	Progettista	2016-05-02	Incremento sezione 4.25 e relative sotto-sezioni
1.3.0	Davide Trevisan	Verificatore	2016-04-30	Verifica sezione 4.24 e relative sotto-sezioni
1.2.1	Andrea Giacomo Baldan	Progettista	2016-04-29	Incremento sezione 4.24 e relative sotto-sezioni
1.2.0	Michael Munaro	Verificatore	2016-04-28	Verifica sezione 4.23 e relative sotto-sezioni
1.1.0	Davide Trevisan	Verificatore	2016-04-28	Verifica sezione 4.22 e sottosezione 4.22.4
1.0.2	Giacomo Vanin	Progettista	2016-04-28	Incremento sezione 4.23 e relative sotto-sezioni
1.0.1	Marco Boseggia	Progettista	2016-04-27	Incremento sezione Driver 4.22 e sotto-sezione 4.22.4
1.0.0	Marco Boseggia	Responsabile	2016-04-08	Approvazione documento
0.8.0	Andrea Giacomo Baldan	Verificatore	2016-04-03	Verifica appendice
0.7.1	Giacomo Vanin	Progettista	2016-04-03	Stesura appendice documento
0.7.0	Alberto De Agostini	Verificatore	2016-04-02	Verifica tracciamenti (sezione 9)
0.6.1	Francesco Agostini	Progettista	2016-04-02	Stesura sezione 9 comprensivo di tutto il tracciamento
0.6.0	Andrea Giacomo Baldan	Verificatore	2016-04-01	Verifica sezioni 7 e 8
0.5.2	Alberto De Agostini	Progettista	2016-03-31	Stesura sezione 8
0.5.1	Alberto De Agostini	Progettista	2016-03-30	Stesura sezione 7 e sottosezioni
0.5.0	Andrea Giacomo Baldan	Verificatore	2016-03-28	Verifica sezione 6 e sottosezioni
0.4.1	Davide Trevisan	Progettista	2016-03-27	Stesura sezione 6 e sottosezioni
0.4.0	Michael Munaro	Verificatore	2016-03-25	Verifica sezione 5
0.3.1	Giacomo Vanin	Progettista	2016-03-24	Stesura sezione 5
0.3.0	Alberto De Agostini	Verificatore	2016-03-22	Verifica sezioni da 4.6 a 4.13



0.2.2	Marco Boseggia	Progettista	2016-03-22	Stesura sezioni 4.10 4.11 4.12 4.13
0.2.1	Marco Boseggia	Progettista	2016-03-21	Stesura sezioni 4.6 4.7 4.8 4.9
0.2.0	Michael Munaro	Verificatore	2016-03-21	Verifica sottosezioni 3.1 3.2 4.1 4.2 4.3 4.4 4.5
0.1.2	Andrea Giacomo Baldan	Progettista	2016-03-17	Stesura sottosezioni 4.1 4.2 4.3 4.4 4.5
0.1.1	Andrea Giacomo Baldan	Progettista	2016-03-17	Stesura sottosezioni 3.1 3.2
0.1.0	Michael Munaro	Verificatore	2016-03-16	Verifica sezioni 1 e 2
0.0.2	Marco Boseggia	Progettista	2016-02-26	Stesura sezioni 1 e 2
0.0.1	Andrea Giacomo Baldan	Amministratore	2016-02-26	Creazione scheletro del documento

Indice

1 Sommario	1
1.1 Scopo del documento	1
1.2 Scopo del Prodotto	1
1.3 Glossario	1
1.4 Riferimenti	1
1.4.1 Normativi	1
1.4.2 Informativi	1
2 Tecnologie utilizzate	3
2.1 Akka	3
2.2 Scala	3
2.3 sbt	4
3 Descrizione architettura	5
3.1 Metodo e formalismo di specifica	5
3.2 Architettura generale	5
3.3 Protocollo di comunicazione Client-Server	5
4 Componenti	6
4.1 actorbase	6
4.1.1 Descrizione	6
4.1.2 Package contenuti	7
4.2 actorbase::actorsystem	7
4.2.1 Descrizione	7
4.2.2 Package contenuti	7
4.2.3 Interazioni con altre componenti	8
4.3 actorbase::actorsystem::utils	9
4.3.1 Descrizione	9
4.3.2 Interazioni con altre componenti	10
4.3.3 Classi	10
4.3.3.1 CryptoUtils	10
4.3.3.1.1 Descrizione	10
4.3.3.1.2 Utilizzo	10
4.3.3.1.3 Interazioni con altre classi	10
4.3.3.2 ActorbaseCollection	10
4.3.3.2.1 Descrizione	10
4.3.3.2.2 Utilizzo	11
4.3.3.2.3 Realizza	11
4.3.3.2.4 Interazioni con altre classi	11
4.4 actorbase::actorsystem::actors	12
4.4.1 Descrizione	12
4.4.2 Package contenuti	12
4.5 actorbase::actorsystem::actors::clientactor	13
4.5.1 Descrizione	13



4.5.2	Interazioni con altre componenti	14
4.5.3	Classi	14
4.5.3.1	actorbase::actorsystem::actors::clientactor::Authenticator	14
4.5.3.1.1	Descrizione	14
4.5.3.1.2	Utilizzo	14
4.5.3.1.3	Interazioni con altre classi	14
4.5.3.2	actorbase::actorsystem::actors::clientactor::CollectionApi	14
4.5.3.2.1	Descrizione	14
4.5.3.2.2	Utilizzo	14
4.5.3.2.3	Interazioni con altre classi	15
4.5.3.3	actorbase::actorsystem::actors::clientactor::UserApi	15
4.5.3.3.1	Descrizione	15
4.5.3.3.2	Utilizzo	15
4.5.3.3.3	Interazioni con altre classi	15
4.5.3.4	actorbase::actorsystem::actors::clientactor::ClientActor	15
4.5.3.4.1	Descrizione	15
4.5.3.4.2	Utilizzo	15
4.5.3.4.3	Classi ereditate	15
4.5.3.4.4	Interazioni con altre classi	16
4.6	actorbase::actorsystem::actors::httpserver	16
4.6.1	Descrizione	16
4.6.2	Interazioni con altre componenti	17
4.6.3	Classi	17
4.6.3.1	actorbase::actorsystem::actors::httpserver::HTTPServer	17
4.6.3.1.1	Descrizione	17
4.6.3.1.2	Utilizzo	17
4.6.3.1.3	Classi ereditate	17
4.6.4	Interazioni con altre componenti	17
4.6.4.1	actorbase::actorsystem::actors::httpserver::SslConfiguration	17
4.6.4.1.1	Descrizione	17
4.6.4.1.2	Utilizzo	17
4.7	actorbase::actorsystem::actors::warehouseman	19
4.7.1	Descrizione	20
4.7.2	Interazioni con altre componenti	20
4.7.3	Classi	20
4.7.3.1	actorbase::actorsystem::actors::warehouseman::Warehouseman	20
4.7.3.1.1	Descrizione	20
4.7.3.1.2	Utilizzo	20
4.7.3.1.3	Classi ereditate	20
4.7.3.1.4	Interazioni con altre classi	20
4.8	actorbase::actorsystem::actors::warehouseman::messages	21
4.8.1	Descrizione	21
4.8.2	Interazioni con altre componenti	21
4.9	actorbase::actorsystem::actors::main	22
4.9.1	Descrizione	22



4.9.2 Interazioni con altre componenti	22
4.9.3 Classi	23
4.9.3.1 actorbase::actorsystem::actors::main::Main	23
4.9.3.1.1 Descrizione	23
4.9.3.1.2 Utilizzo	23
4.9.3.1.3 Classi ereditate	23
4.9.3.1.4 Interazioni con altre classi	23
4.10 actorbase::actorsystem::actors::authactor	25
4.10.1 Descrizione	26
4.10.2 Classi	26
4.10.2.1 actorbase::actorsystem::actors::authactor::Authactor	26
4.10.2.1.1 Descrizione	26
4.10.2.1.2 Utilizzo	26
4.10.2.1.3 Classi ereditate	26
4.10.2.1.4 Interazioni con altre classi	26
4.10.2.2 actorbase::actorsystem::actors::authactor::Profile	26
4.10.2.2.1 Descrizione	26
4.10.2.2.2 Utilizzo	26
4.10.2.2.3 Interazioni con altre classi	26
4.11 actorbase::actorsystem::actors::storefinder	27
4.11.1 Descrizione	27
4.11.2 Interazioni con altre componenti	28
4.11.3 Classi	28
4.11.3.1 actorbase::actorsystem::actors::storefinder::Storefinder	28
4.11.3.1.1 Descrizione	28
4.11.3.1.2 Utilizzo	28
4.11.3.1.3 Classi ereditate	28
4.11.3.1.4 Interazioni con altre classi	28
4.12 actorbase::actorsystem::actors::storekeeper	29
4.12.1 Descrizione	29
4.12.2 Interazioni con altre componenti	29
4.12.3 Classi	30
4.12.3.1 actorbase::actorsystem::actors::storekeeper::Storekeeper	30
4.12.3.1.1 Descrizione	30
4.12.3.1.2 Utilizzo	30
4.12.3.1.3 Classi ereditate	30
4.12.3.1.4 Interazioni con altre classi	30
4.13 actorbase::actorsystem::actors::manager	31
4.13.1 Descrizione	32
4.13.2 Interazioni con altre componenti	32
4.13.3 Classi	32
4.13.3.1 actorbase::actorsystem::actors::manager::Manager	32
4.13.3.1.1 Descrizione	32
4.13.3.1.2 Utilizzo	32
4.13.3.1.3 Classi ereditate	32



4.14 actorbase::actorsystem::messages	33
4.14.1 Descrizione	33
4.14.2 Package contenuti	33
4.15 actorbase::actorsystem::messages::authactormessages	34
4.15.1 Descrizione	35
4.15.2 Classi	35
4.15.2.1 actorbase::actorsystem::messages::authactormessages::AddCredentials	35
4.15.2.1.1 Descrizione	35
4.15.2.1.2 Utilizzo	35
4.15.2.2 actorbase::actorsystem::messages::authactormessages::UpdateCredentials	35
4.15.2.2.1 Descrizione	35
4.15.2.2.2 Utilizzo	35
4.15.2.3 actorbase::actorsystem::messages::authactormessages::RemoveCredentials	35
4.15.2.3.1 Descrizione	35
4.15.2.3.2 Utilizzo	36
4.15.2.4 actorbase::actorsystem::messages::authactormessages::Authenticate	36
4.15.2.4.1 Descrizione	36
4.15.2.4.2 Utilizzo	36
4.15.2.5 actorbase::actorsystem::messages::authactormessages::AddCollectionTo	36
4.15.2.5.1 Descrizione	36
4.15.2.5.2 Utilizzo	36
4.15.2.6 actorbase::actorsystem::messages::authactormessages::RemoveCollectionFrom	36
4.15.2.6.1 Descrizione	36
4.15.2.6.2 Utilizzo	37
4.15.2.7 actorbase::actorsystem::messages::authactormessages::ListCollectionsOf	37
4.15.2.7.1 Descrizione	37
4.15.2.7.2 Utilizzo	37
4.16 actorbase::actorsystem::messages::clientactormessages	38
4.16.1 Descrizione	38
4.16.2 Classi	38
4.16.2.1 actorbase::actorsystem::messages::clientactormessages::ListResponse	38
4.16.2.1.1 Descrizione	38
4.16.2.1.2 Utilizzo	38
4.16.2.2 actorbase::actorsystem::messages::clientactormessages::MapResponse	39
4.16.2.2.1 Descrizione	39
4.16.2.2.2 Utilizzo	39
4.16.2.3 actorbase::actorsystem::messages::clientactormessages::Response	39
4.16.2.3.1 Descrizione	39
4.16.2.3.2 Utilizzo	39
4.17 actorbase::actorsystem::messages::mainmessages	40
4.17.1 Descrizione	40
4.17.2 Classi	41
4.17.2.1 actorbase::actorsystem::messages::mainmessages::InsertTo	41
4.17.2.1.1 Descrizione	41
4.17.2.1.2 Utilizzo	41



4.17.2.2 actorbase::actorsystem::messages::mainmessages::GetFrom	41
4.17.2.2.1 Descrizione	41
4.17.2.2.2 Utilizzo	41
4.17.2.3 actorbase::actorsystem::messages::mainmessages::CompleteTransaction	41
4.17.2.3.1 Descrizione	41
4.17.2.3.2 Utilizzo	42
4.17.2.4 actorbase::actorsystem::messages::mainmessages::RemoveFrom	42
4.17.2.4.1 Descrizione	42
4.17.2.4.2 Utilizzo	42
4.17.2.5 actorbase::actorsystem::messages::mainmessages::AddContributor	42
4.17.2.5.1 Descrizione	42
4.17.2.5.2 Utilizzo	42
4.17.2.6 actorbase::actorsystem::messages::mainmessages::RemoveContributor	42
4.17.2.6.1 Descrizione	42
4.17.2.6.2 Utilizzo	43
4.17.2.7 actorbase::actorsystem::messages::mainmessages::CreateCollection	43
4.17.2.7.1 Descrizione	43
4.17.2.7.2 Utilizzo	43
4.18 actorbase::actorsystem::messages::storefindermessages	44
4.18.1 Descrizione	44
4.18.2 Classi	45
4.18.2.1 actorbase::actorsystem::messages::storefinder::GetAllItems	45
4.18.2.1.1 Descrizione	45
4.18.2.1.2 Utilizzo	45
4.18.2.2 actorbase::actorsystem::messages::storefinder::UpdateCollectionSize	45
4.18.2.2.1 Descrizione	45
4.18.2.2.2 Utilizzo	45
4.18.2.3 actorbase::actorsystem::messages::storefinder::Get	45
4.18.2.3.1 Descrizione	45
4.18.2.3.2 Utilizzo	45
4.18.2.4 actorbase::actorsystem::messages::storefinder::PartialMapTransaction	46
4.18.2.4.1 Descrizione	46
4.18.2.4.2 Utilizzo	46
4.18.2.5 actorbase::actorsystem::messages::storefinder::Remove	46
4.18.2.5.1 Descrizione	46
4.18.2.5.2 Utilizzo	46
4.18.2.6 actorbase::actorsystem::messages::storefinder::Insert	46
4.18.2.6.1 Descrizione	46
4.18.2.6.2 Utilizzo	46
4.19 actorbase::actorsystem::messages::storekeepermessages	47
4.19.1 Descrizione	47
4.19.2 Classi	47
4.19.2.1 actorbase::actorsystem::messages::storekeeper::Persist	47
4.19.2.1.1 Descrizione	47
4.19.2.1.2 Utilizzo	47



4.19.2.2 actorbase::actorsystem::messages::storekeeper::InitMn	48
4.19.2.2.1 Descrizione	48
4.19.2.2.2 Utilizzo	48
4.19.2.3 actorbase::actorsystem::messages::storekeeper::GetAll	48
4.19.2.3.1 Descrizione	48
4.19.2.3.2 Utilizzo	48
4.19.2.4 actorbase::actorsystem::messages::storekeeper::GetItem	48
4.19.2.4.1 Descrizione	48
4.19.2.4.2 Utilizzo	48
4.19.2.5 actorbase::actorsystem::messages::storekeeper::InsertItem	49
4.19.2.5.1 Descrizione	49
4.19.2.5.2 Utilizzo	49
4.19.2.6 actorbase::actorsystem::messages::storekeeper::RemoveItem	49
4.19.2.6.1 Descrizione	49
4.19.2.6.2 Utilizzo	49
4.20 actorbase::actorsystem::messages::warehousemanmessages	50
4.20.1 Descrizione	50
4.20.2 Classi	50
4.20.2.1 actorbase::actorsystem::messages::warehouseman::Clean	50
4.20.2.1.1 Descrizione	50
4.20.2.1.2 Utilizzo	50
4.20.2.2 actorbase::actorsystem::messages::warehouseman::Init	51
4.20.2.2.1 Descrizione	51
4.20.2.2.2 Utilizzo	51
4.20.2.3 actorbase::actorsystem::messages::warehouseman::Save	51
4.20.2.3.1 Descrizione	51
4.20.2.3.2 Utilizzo	51
4.20.2.4 actorbase::actorsystem::messages::warehouseman::Read	51
4.20.2.4.1 Descrizione	51
4.20.2.4.2 Utilizzo	51
4.21 actorbase::actorsystem::messages::managermessages	52
4.21.1 Descrizione	52
4.21.2 Classi	52
4.21.2.1 actorbase::actorsystem::messages::managermessages::OneMore	52
4.21.2.1.1 Descrizione	52
4.21.2.1.2 Utilizzo	52
4.22 actorbase::driver	53
4.22.1 Descrizione	53
4.22.2 Interazioni con altre componenti	53
4.22.3 Package contenuti	53
4.22.4 Classi	54
4.22.4.1 ActorbaseDriver	54
4.22.4.1.1 Descrizione	54
4.22.4.1.2 Utilizzo	54
4.22.4.1.3 Eredità	54



4.22.4.1.4 Interazioni con le altre classi	54
4.22.4.2 ActorbaseServices	54
4.22.4.2.1 Descrizione	54
4.22.4.2.2 Utilizzo	54
4.22.4.2.3 Eredita	54
4.22.4.2.4 Interazioni con le altre classi	55
4.22.4.3 ActorbaseAdminServices	55
4.22.4.3.1 Descrizione	55
4.22.4.3.2 Utilizzo	55
4.22.4.3.3 Eredita	55
4.22.4.3.4 Interazioni con le altre classi	55
4.22.4.4 Connection	55
4.22.4.4.1 Descrizione	55
4.22.4.4.2 Utilizzo	55
4.22.4.4.3 Interazioni con le altre classi	55
4.23 actorbase::driver::client	56
4.23.1 Descrizione	56
4.23.2 Interazioni con altre componenti	56
4.23.3 Classi	56
4.23.3.1 Client	56
4.23.3.1.1 Descrizione	56
4.23.3.1.2 Utilizzo	57
4.23.3.1.3 Realizza	57
4.23.3.2 SSLClient (abstract)	57
4.23.3.2.1 Descrizione	57
4.23.3.2.2 Utilizzo	57
4.23.3.2.3 Realizza	57
4.23.3.3 actorbase::driver::client::ActorbaseClient	57
4.23.3.3.1 Descrizione	57
4.23.3.3.2 Utilizzo	57
4.23.3.3.3 Realizza	57
4.23.3.3.4 Eredita	57
4.23.3.4 Serializer (abstract)	58
4.23.3.4.1 Descrizione	58
4.23.3.4.2 Utilizzo	58
4.23.3.4.3 Ereditata da	58
4.23.3.5 actorbase::driver::client::Connector (abstract)	58
4.23.3.5.1 Descrizione	58
4.23.3.5.2 Utilizzo	58
4.23.3.5.3 Eredita	58
4.24 actorbase::driver::client::api	59
4.24.1 Descrizione	59
4.24.2 Interazioni con altre componenti	59
4.24.3 Classi	60
4.24.3.1 actorbase::driver::client::api::RequestMethod (abstract)	60
4.24.3.1.1 Descrizione	60



4.24.3.1.2 Utilizzo	60
4.24.3.1.3 Interazioni con le altre classi	60
4.24.3.2 actorbase::driver::client::api::GET (abstract)	60
4.24.3.2.1 Descrizione	60
4.24.3.2.2 Utilizzo	60
4.24.3.2.3 Interazioni con le altre classi	60
4.24.3.2.4 Eredita	60
4.24.3.3 actorbase::driver::client::api::POST (abstract)	61
4.24.3.3.1 Descrizione	61
4.24.3.3.2 Utilizzo	61
4.24.3.3.3 Interazioni con le altre classi	61
4.24.3.3.4 Eredita	61
4.24.3.4 actorbase::driver::client::api::PUT (abstract)	61
4.24.3.4.1 Descrizione	61
4.24.3.4.2 Utilizzo	61
4.24.3.4.3 Interazioni con le altre classi	61
4.24.3.4.4 Eredita	61
4.24.3.5 actorbase::driver::client::api::DELETE (abstract)	62
4.24.3.5.1 Descrizione	62
4.24.3.5.2 Utilizzo	62
4.24.3.5.3 Interazioni con le altre classi	62
4.24.3.5.4 Eredita	62
4.24.3.6 actorbase::driver::client::api::Request	62
4.24.3.6.1 Descrizione	62
4.24.3.6.2 Utilizzo	62
4.24.3.6.3 Interazioni con le altre classi	62
4.24.3.7 actorbase::driver::client::api::Response	63
4.24.3.7.1 Descrizione	63
4.24.3.7.2 Utilizzo	63
4.24.3.7.3 Interazioni con le altre classi	63
4.24.3.8 Status	63
4.24.3.8.1 Descrizione	63
4.24.3.8.2 Utilizzo	63
4.24.3.8.3 Interazioni con le altre classi	63
4.24.3.9 RequestBuilder	63
4.24.3.9.1 Descrizione	63
4.24.3.9.2 Utilizzo	64
4.24.3.9.3 Interazioni con le altre classi	64
4.25 actorbase::driver::client::api::data	64
4.25.1 Descrizione	64
4.25.2 Interazioni con altre componenti	65
4.25.3 Classi	65
4.25.3.1 ActorbaseObject	65
4.25.3.1.1 Descrizione	65
4.25.3.1.2 Utilizzo	65



4.25.3.1.3 Interazioni con le altre classi	65
4.25.3.2 actorbase::driver::data::ActorbaseCollection	65
4.25.3.2.1 Descrizione	65
4.25.3.2.2 Utilizzo	65
4.25.3.2.3 Eredita	65
4.25.3.2.4 Interazioni con le altre classi	65
4.25.3.3 actorbase::driver::data::ActorbaseCollectionMap	66
4.25.3.3.1 Descrizione	66
4.25.3.3.2 Utilizzo	66
4.25.3.3.3 Eredita	66
4.25.3.3.4 Interazioni con le altre classi	66
4.26 actorbase::driver::exceptions	66
4.26.1 Descrizione	66
4.26.2 Interazione con altre componenti	67
4.26.3 Classi	67
4.26.3.1 actorbase::driver::exceptions::WrongCredentialExc	67
4.26.3.1.1 Descrizione	67
4.26.3.1.2 Utilizzo	67
4.26.3.1.3 Interazioni con altre classi	67
4.26.3.2 actorbase::driver::exceptions::WrongPasswordExc	67
4.26.3.2.1 Descrizione	67
4.26.3.2.2 Utilizzo	67
4.26.3.2.3 Interazioni con altre classi	67
4.26.3.3 actorbase::driver::exceptions::WrongNewPasswordExc	67
4.26.3.3.1 Descrizione	67
4.26.3.3.2 Utilizzo	68
4.26.3.3.3 Interazioni con altre classi	68
4.26.3.4 actorbase::driver::exceptions::CollectionAlreadyExistsExc	68
4.26.3.4.1 Descrizione	68
4.26.3.4.2 Utilizzo	68
4.26.3.4.3 Interazioni con altre classi	68
4.26.3.5 actorbase::driver::exceptions::UndefinedCollectionExc	68
4.26.3.5.1 Descrizione	68
4.26.3.5.2 Utilizzo	68
4.26.3.5.3 Interazioni con altre classi	68
4.26.3.6 actorbase::driver::exceptions::UndefinedUsernameExc	69
4.26.3.6.1 Descrizione	69
4.26.3.6.2 Utilizzo	69
4.26.3.6.3 Interazioni con altre classi	69
4.26.3.7 actorbase::driver::exceptions::UsernameAlreadyExistsExc	69
4.26.3.7.1 Descrizione	69
4.26.3.7.2 Utilizzo	69
4.26.3.7.3 Interazioni con altre classi	69
4.26.3.8 actorbase::driver::exceptions::DuplicateKeyExc	69
4.26.3.8.1 Descrizione	69



4.26.3.8.2 Utilizzo	69
4.26.3.8.3 Interazioni con altre classi	70
4.26.3.9 actorbase::driver::exceptions::UndefinedFileExc	70
4.26.3.9.1 Descrizione	70
4.26.3.9.2 Utilizzo	70
4.26.3.9.3 Interazioni con altre classi	70
4.26.3.10 actorbase::driver::exceptions::MalformedFileExc	70
4.26.3.10.1 Descrizione	70
4.26.3.10.2 Utilizzo	70
4.26.3.10.3 Interazioni con altre classi	70
4.27 actorbase::cli	71
4.27.1 Descrizione	72
4.27.2 Interazioni con altre componenti	72
4.27.3 Package contenuti	72
4.28 actorbase::cli::views	74
4.28.1 Descrizione	75
4.28.2 Interazioni con altre componenti	75
4.28.3 Classi	75
4.28.3.1 actorbase::cli::views::CommandLoop	75
4.28.3.1.1 Descrizione	75
4.28.3.1.2 Utilizzo	75
4.28.3.1.3 Interazioni con altre classi	75
4.28.3.2 actorbase::cli::views::ActorbaseBanner	75
4.28.3.2.1 Descrizione	75
4.28.3.2.2 Utilizzo	75
4.28.3.2.3 Interazioni con altre classi	76
4.28.3.3 actorbase::cli::views::PromptProvider	76
4.28.3.3.1 Descrizione	76
4.28.3.3.2 Utilizzo	76
4.28.3.3.3 Interazioni con altre classi	76
4.28.3.4 actorbase::cli::views::ActorbasePrompt	76
4.28.3.4.1 Descrizione	76
4.28.3.4.2 Utilizzo	76
4.28.3.4.3 Classi ereditate	76
4.28.3.5 actorbase::cli::views::Observer	76
4.28.3.5.1 Descrizione	76
4.28.3.5.2 Utilizzo	77
4.28.3.5.3 Interazioni con altre classi	77
4.28.3.6 actorbase::cli::views::ResultView	77
4.28.3.6.1 Descrizione	77
4.28.3.6.2 Utilizzo	77
4.28.3.6.3 Classi ereditate	77
4.28.3.6.4 Interazioni con altre classi	77
4.29 actorbase::cli::controllers	78
4.29.1 Descrizione	79



4.29.2	Interazioni con altre componenti	79
4.29.3	Classi	79
4.29.3.1	actorbase::cli::controllers::GrammarParser	79
4.29.3.1.1	Descrizione	79
4.29.3.1.2	Utilizzo	79
4.29.3.1.3	Interazioni con altre classi	79
4.30	actorbase::cli::models	80
4.30.1	Descrizione	81
4.30.2	Classi	81
4.30.2.1	actorbase::cli::models::Observable	81
4.30.2.1.1	Descrizione	81
4.30.2.1.2	Utilizzo	81
4.30.2.1.3	Interazioni con altre classi	81
4.30.2.2	actorbase::cli::models::CommandInvoker	81
4.30.2.2.1	Descrizione	81
4.30.2.2.2	Utilizzo	81
4.30.2.2.3	Classi ereditate	81
4.30.2.2.4	Interazioni con altre classi	82
4.30.2.3	actorbase::cli::models::Command	82
4.30.2.3.1	Descrizione	82
4.30.2.3.2	Utilizzo	82
4.30.2.3.3	Interazioni con altre classi	82
4.30.2.4	actorbase::cli::models::FindCommand	82
4.30.2.4.1	Descrizione	82
4.30.2.4.2	Utilizzo	82
4.30.2.4.3	Classi ereditate	82
4.30.2.5	actorbase::cli::models::LogoutCommand	82
4.30.2.5.1	Descrizione	82
4.30.2.5.2	Utilizzo	83
4.30.2.5.3	Classi ereditate	83
4.30.2.6	actorbase::cli::models::ResetPasswordCommand	83
4.30.2.6.1	Descrizione	83
4.30.2.6.2	Utilizzo	83
4.30.2.6.3	Classi ereditate	83
4.30.2.7	actorbase::cli::models::ChangePasswordCommand	83
4.30.2.7.1	Descrizione	83
4.30.2.7.2	Utilizzo	83
4.30.2.7.3	Classi ereditate	83
4.30.2.8	actorbase::cli::models::ListCommand	84
4.30.2.8.1	Descrizione	84
4.30.2.8.2	Utilizzo	84
4.30.2.8.3	Classi ereditate	84
4.30.2.9	actorbase::cli::models::AddContributorCommand	84
4.30.2.9.1	Descrizione	84
4.30.2.9.2	Utilizzo	84



4.30.2.9.3 Classi ereditate	84
4.30.2.10 actorbase::cli::models::LoginCommand	84
4.30.2.10.1 Descrizione	84
4.30.2.10.2 Utilizzo	84
4.30.2.10.3 Classi ereditate	85
4.30.2.11 actorbase::cli::models::RemoveCollectionCommand	85
4.30.2.11.1 Descrizione	85
4.30.2.11.2 Utilizzo	85
4.30.2.11.3 Classi ereditate	85
4.30.2.12 actorbase::cli::models::RemoveContributorCommand	85
4.30.2.12.1 Descrizione	85
4.30.2.12.2 Utilizzo	85
4.30.2.12.3 Classi ereditate	85
4.30.2.13 actorbase::cli::models::RenameCollectionCommand	85
4.30.2.13.1 Descrizione	85
4.30.2.13.2 Utilizzo	86
4.30.2.13.3 Classi ereditate	86
4.30.2.14 actorbase::cli::models::AddUserCommand	86
4.30.2.14.1 Descrizione	86
4.30.2.14.2 Utilizzo	86
4.30.2.14.3 Classi ereditate	86
4.30.2.15 actorbase::cli::models::HelpCommand	86
4.30.2.15.1 Descrizione	86
4.30.2.15.2 Utilizzo	86
4.30.2.16 actorbase::cli::models::RemoveItemCommand	86
4.30.2.16.1 Descrizione	86
4.30.2.16.2 Utilizzo	87
4.30.2.16.3 Classi ereditate	87
4.30.2.17 actorbase::cli::models::RemoveUserCommand	87
4.30.2.17.1 Descrizione	87
4.30.2.17.2 Utilizzo	87
4.30.2.17.3 Classi ereditate	87
4.30.2.18 actorbase::cli::models::ImportCommand	87
4.30.2.18.1 Descrizione	87
4.30.2.18.2 Utilizzo	87
4.30.2.18.3 Classi ereditate	87
4.30.2.19 actorbase::cli::models::InsertItemCommand	88
4.30.2.19.1 Descrizione	88
4.30.2.19.2 Utilizzo	88
4.30.2.19.3 Classi ereditate	88
4.30.2.20 actorbase::cli::models::CreateCollectionCommand	88
4.30.2.20.1 Descrizione	88
4.30.2.20.2 Utilizzo	88
4.30.2.20.3 Classi ereditate	88
4.30.2.21 actorbase::cli::models::ExportCommand	88



4.30.2.21.1 Descrizione	88
4.30.2.21.2 Utilizzo	88
4.30.2.21.3 Classi ereditate	89
4.30.2.22 actorbase::cli::models::CommandReceiver	89
4.30.2.22.1 Descrizione	89
4.30.2.22.2 Utilizzo	89
4.30.2.22.3 Interazioni con altre classi	89
5 Diagrammi di Attività	90
5.1 Visione generale	90
5.2 Operazioni su collezioni e/o item	92
5.2.1 Creazione collezione	92
5.2.2 Cancellazione collezione	92
5.2.3 Visualizza collezioni	93
5.2.4 Modifica nome collezione	94
5.2.5 Inserimento item	95
5.2.6 Rimozione item	96
5.2.7 Aggiunta collaboratore	97
5.2.8 Rimozione collaboratore	98
5.2.9 Import	99
5.3 Interrogazione del database	100
5.4 Modifica password	101
5.5 Gestione utenti	102
6 Design Pattern	104
6.1 Design Pattern Architetturali	105
6.1.1 MVC	105
6.1.2 Dependancy Injection	106
6.2 Design Pattern Creazionali	106
6.2.1 Singleton	106
6.2.2 Builder	107
6.3 Design Pattern Strutturali	108
6.3.1 Decorator	108
6.4 Design Pattern Comportamentali	109
6.4.1 Command Pattern	109
6.4.2 Observer Pattern	110
7 Stime di fattibilità e di bisogno di risorse	112
8 Tracciamento	113
8.1 Tracciamento Componenti-Requisiti	113
8.2 Tracciamento Requisiti-Componenti	122
A Descrizione Design Pattern	137
A.1 Design Pattern Architetturali	137
A.1.1 MVC	137



A.1.2 Dependency injection	138
A.2 Design Pattern Creazionali	139
A.2.1 Singleton	139
A.2.2 Builder	139
A.3 Design Pattern Strutturali	140
A.3.1 Decorator	140
A.4 Design Pattern Comportamentali	141
A.4.1 Command	141
A.4.2 Observer	142



1 Sommario

1.1 Scopo del documento

Il seguente documento ha lo scopo di descrivere la progettazione ad alto livello che il gruppo *ScalateKids* ha scelto per il progetto **Actorbase**.

Sarà descritta l'architettura_G generale del progetto, in particolare la scelta dei design pattern_G e delle componenti che andranno a comporre il software.

1.2 Scopo del Prodotto

Implementazione di un database NoSQL_G di tipo key-value_G orientato alla gestione di grandi moli di dati utilizzando il modello ad attori_G su JVM_G, comprensivo di un *Domain Specific Language* (DSL_G) da utilizzare da riga di comando per poter interagire con il database.

Il progetto dovrà essere pubblicato su *GitHub* sotto licenza *MIT*.

1.3 Glossario

Tutti i termini di carattere tecnico o frantendibile e gli acronimi sono raccolti nel file [Glossario v3.0.0](#); ogni occorrenza di parole nel *Glossario* è indicata da una "G" in pedice.

1.4 Riferimenti

1.4.1 Normativi

- **Capitolato d'appalto C1:** *Actorbase: a NoSQL DB based on the Actor model;*
<http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C1.pdf>
- **Norme di Progetto:** [Norme di Progetto v4.0.0](#).

1.4.2 Informativi

- **Piano di Progetto:** [Piano di Progetto v4.0.0](#)
- **Dispense fornite dall'insegnamento Ingegneria del Software mod.A:**
<http://www.math.unipd.it/~tullio/IS-1/2015/>
- **Documentazione di Akka_G su Scala_G:** <http://doc.akka.io/docs/akka/2.4.2/scala.html>
- **SBT - Scala build tool:** <http://www.scala-sbt.org/index.html>



- **Scala Pickling - Fast, Customizable, Boilerplate-Free Serialization for Scala:** Libreria di serializzazione e deserializzazione multiformato, pieno supporto JSON_G (JavaScript Object Notation) e serializzazione in binario <http://lampwww.epfl.ch/~hmiller/pickling>
- **Spray - an open-source toolkit for building REST/HTTP-based integration layers on top of Scala and Akka:** Libreria per la gestione di richieste HTTP_G(HyperText Transfert Protocol) in modo asincrono, basata sul modello ad attori offerto dalla libreria Akka. <http://spray.io/>
- **REST - Representational State Transfer:** Architettura software per sistemi di ipertesto_G distribuiti, rappresenta un sistema di localizzazione e richiesta risorse stateless_G, poggia su HTTP_G come protocollo di comunicazione https://en.wikipedia.org/wiki/Representational_state_transfer



2 Tecnologie utilizzate

In questa sezione saranno descritte le tecnologie scelte per lo sviluppo del progetto **Actorbase** e le motivazioni che ci hanno spinto a sceglierle.

2.1 Akka

La scelta della libreria Akka_G è stata dettata dal capitolato, tuttavia l'avremmo scelta comunque per i seguenti motivi:

- **Implementazione del modello ad attori_G:** Akka_G rende semplice la creazione e l'utilizzo degli attori_G. Grazie a ciò è possibile creare facilmente architetture ad eventi estremamente concorrenti e viene naturale la distribuibilità del progetto e l'asincronia tra i processi;
- **Tolleranza agli errori:** Akka_G implementa un sistema di gerarchia di supervisori. Questa gerarchia consente ai supervisori di un attore_G, nel caso in cui quest'ultimo lanci un'eccezione, di mandarlo in crash_G e farlo poi ripartire. In questo modo il sistema è resiliente agli errori;
- **Persistenza:** Ogni messaggio ricevuto da ciascun attore_G può essere salvato in maniera tale che al riavvio esso possa riprendere dallo stato precedente al suo spegnimento.

2.2 Scala

Il capitolato richiede l'utilizzo di un linguaggio di programmazione a scelta tra Scala_G e Java_G. Abbiamo scelto Scala_G per i seguenti motivi:

- **Programmazione funzionale_G:** Questa tecnica di programmazione evita che ci siano degli effetti collaterali tra funzioni, rendendole quindi thread-safe;
- **Implementazione di Akka:** Akka_G risulta più facilmente implementabile in Scala_G che in Java_G. Questo perché la programmazione funzionale_G con i suoi paradigmi si integra meglio con il modello ad attori_G;
- **Implementazione di DSL:** Scala si presta alla creazione di DSL_G in maniera nativa, sia di tipo esterno che di tipo interno, senza necessariamente utilizzare librerie esterne o appoggiarsi a framework appositi;
- **Integrazione con librerie Java:** In Scala è possibile usufruire di molte librerie sviluppate in ambiente Java in quanto utilizza lo stesso modello di compilazione generando dei file_G in bytecode_G.

Tuttavia presenta alcuni svantaggi:

- **Curva di apprendimento:** Scala introduce paradigmi di programmazione funzionale applicati al paradigma ad oggetti, inoltre ha una sintassi molto espressiva e questo può rallentare significativamente il processo di apprendimento per i neofiti;
- **Librerie Java:** La possibilità di usufruire di molte librerie Java, pur essendo considerato positivo, ha anche il lato negativo di doversi appoggiare a documentazioni specifiche per linguaggio Java, e questo può portare a difficili comprensioni del comportamento della libreria in Scala, specie nel caso in cui la compatibilità non sia al 100%.



2.3 sbt

Sbt (Simple Build Tool) è un ambiente di build_G simile alla controparte Java Maven_G o Ant_G. Fornito di console_G e strumenti di compilazione ed esecuzione per progetti Scala, è stato scelto per i seguenti vantaggi:

- Semplice configurazione mediante file;
- REPL_G Scala integrata;
- Facilmente integrabile all'interno di IDE_G e editor;
- Gestione semplificata delle dipendenze;
- Gestione semplificata del processo di build_G.



3 Descrizione architettura

3.1 Metodo e formalismo di specifica

L'esposizione dell'architettura segue la metodologia top-down_G, descrivendo l'architettura iniziando dalle componenti generali e scendendo nel dettaglio. Seguirà quindi la descrizione dei package_G seguiti dalla descrizione in dettaglio delle classi e interfacce che vi appartengono, specificando per ognuna la funzione e le relazioni in ingresso ed uscita. Il formalismo cromatico utilizzato sarà conforme a quanto specificato in *Norme Di Progetto*. Successivamente si illustreranno i Design Pattern_G e la loro applicazione all'interno delle componenti, con un approfondimento del loro funzionamento in [appendice A](#).

Nel trattare le componenti, si chiarisce che sono da intendersi come package_G e i due termini verranno quindi usati come sinonimi.

3.2 Architettura generale

Macroscopicamente il sistema si suddivide in due componenti principali, assimilabili ad un paradigma Client_G-Server_G, il sistema ad attori dove risiedono la struttura della base di dati ed un server_G HTTP (Hyper Text Transport Protocol) pronto a ricevere comandi dall'esterno rappresentano la componente server_G, la Command Line Interface (CLI_G) rappresenta la componente client_G mediante la quale è possibile inviare comandi al sistema e ricevere l'output prodotto da essi. Essa farà uso della componente driver_G per comunicare con il sistema lato server_G mediante il protocollo di comunicazione REST.

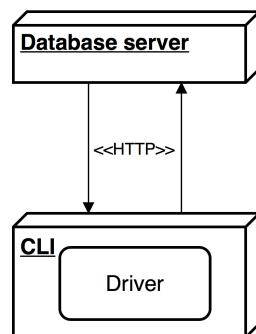


Figura 1: Diagramma architettura concettuale

3.3 Protocollo di comunicazione Client-Server

Actorbase è stato concepito come sistema RESTful_G con possibilità di utilizzo come web-service_G. La connessione è quindi di tipo Stateless, in quanto ogni richiesta non ha nulla a che fare con lo storico di richieste precedenti.



4 Componenti

L'architettura principale del progetto è stata suddivisa in tre macro-componenti, rispettivamente **cli**, **driver** e **actorsystem**; contenute all'interno del package generale **actorbase**.

4.1 actorbase

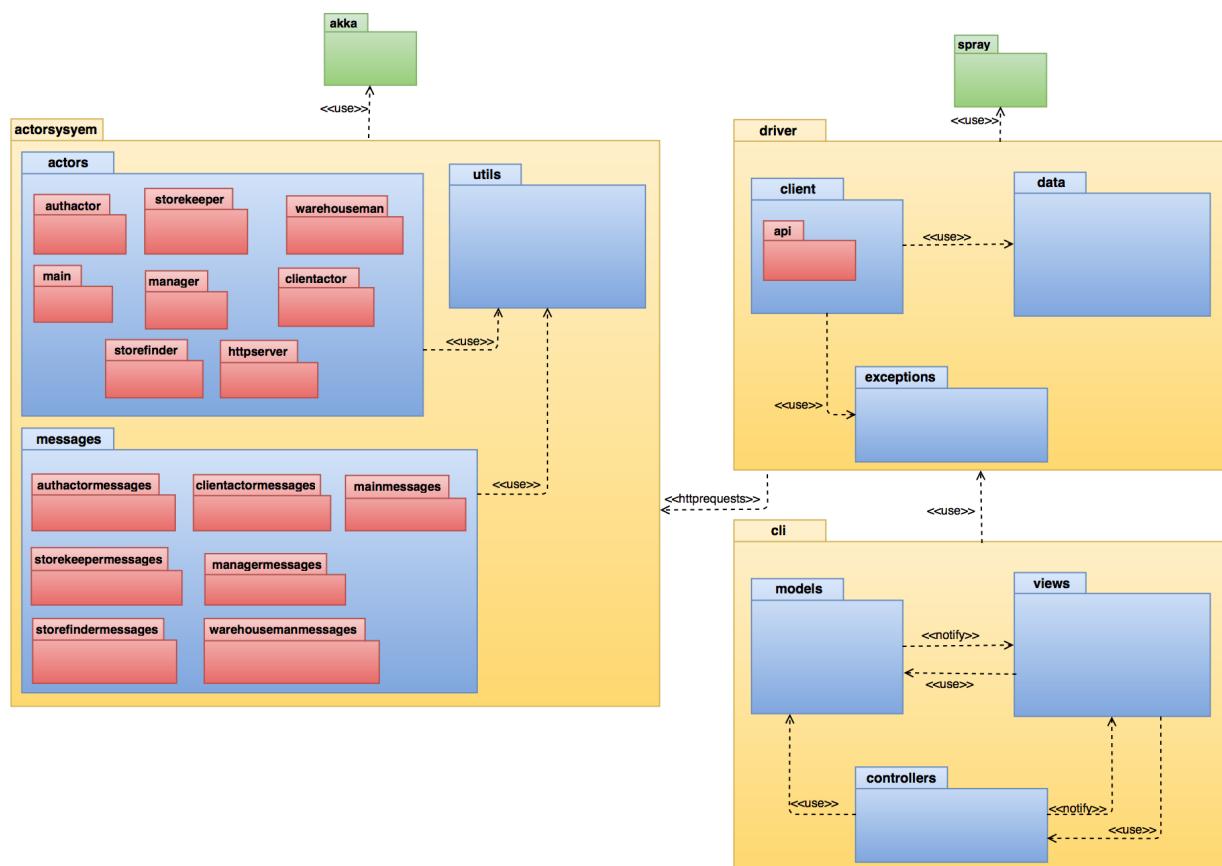


Figura 2: Diagramma componenti principali

4.1.1 Descrizione

La componente principale rappresentata dal package globale **actorbase**.



4.1.2 Package contenuti

- `actorbase::cli`;
- `actorbase::driver`;
- `actorbase::actorsystem`.

Rappresentano le tre macro-componenti del sistema.

4.2 actorbase::actorsystem

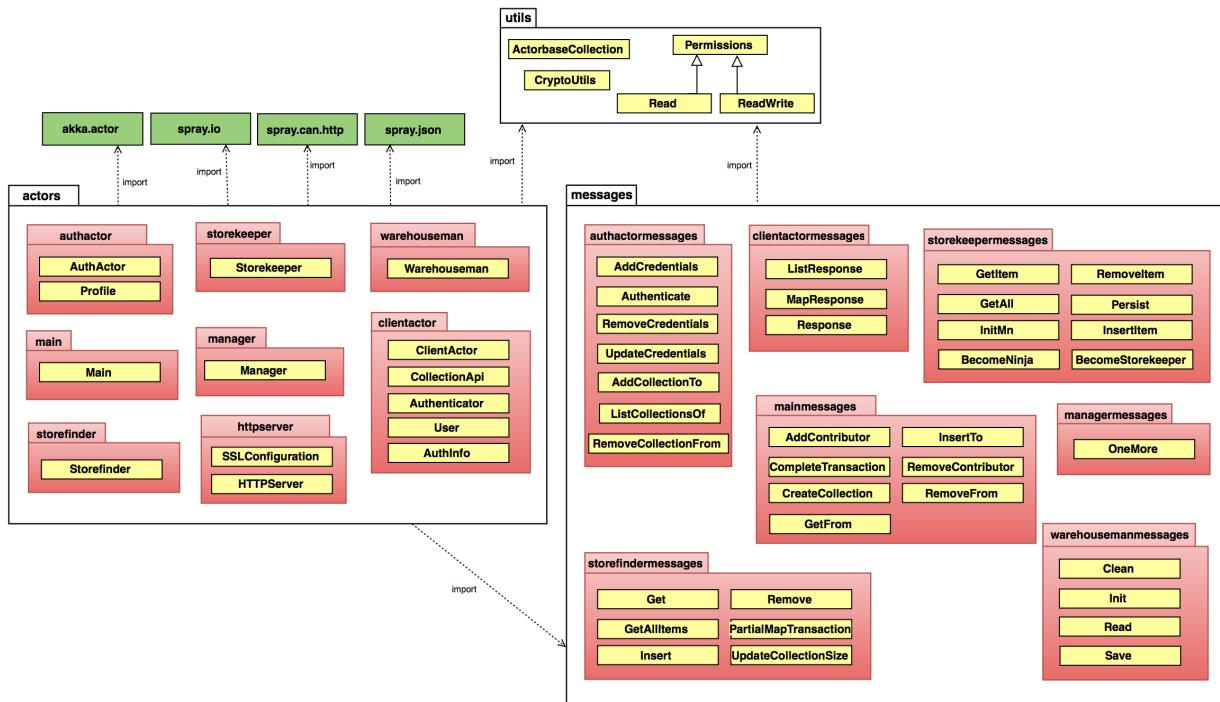


Figura 3: Actorsystem, visione generale

4.2.1 Descrizione

Package_G che raggruppa tutte le componenti del sistema che rappresentano il server_G.

4.2.2 Package contenuti

- `actorbase::actorsystem::actors`;



- `actorbase::actorsystem::messages;`
- `actorbase::actorsystem::utils.`

4.2.3 Interazioni con altre componenti

- `actorbase::driver;`
- Akka.



4.3 actorbase::actorsystem::utils

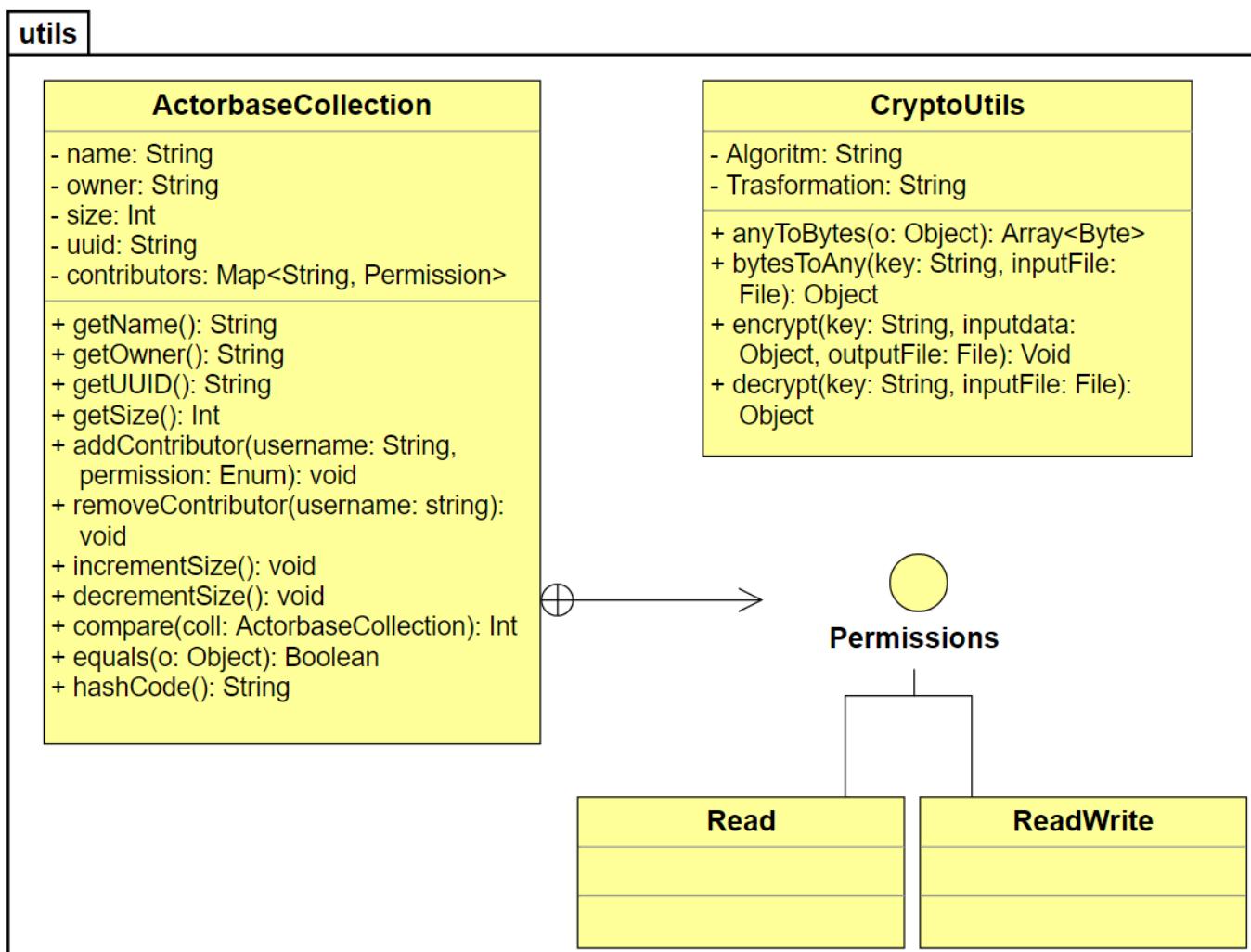


Figura 4: Package actorsystem::utils

4.3.1 Descrizione

Package_G contenente diverse classi di utilità. Utilizzato da gran parte degli attori



4.3.2 Interazioni con altre componenti

- actorbase::actorsystem::actors::httpserver
- actorbase::actorsystem::actors::storekeeper;
- actorbase::actorsystem::actors::storefinder;
- actorbase::actorsystem::actors::authactor;
- actorbase::actorsystem::actors::warehouseman;
- actorbase::actorsystem::actors::main;
- actorbase::actorsystem::actors::clientactor.

4.3.3 Classi

4.3.3.1 CryptoUtils

4.3.3.1.1 Descrizione

Classe atta a criptare e decriptare i dati per il salvataggio e la lettura dei dati su disco.

4.3.3.1.2 Utilizzo

Viene utilizzata dall'attore di tipo Warehouseman_G per il salvataggio e la lettura del database su disco. Anche l'attore actorbase::actorsystem::actors::httpserver utilizza questa classe per decriptare i dati all'accensione del sistema per eseguire un ripopolamento del database.

4.3.3.1.3 Interazioni con altre classi

- actorbase::actorsystem::actors::httpserver::HTTPServer;
- actorbase::actorsystem::actors::warehouseman::Warehouseman.

4.3.3.2 ActorbaseCollection

4.3.3.2.1 Descrizione

Classe atta a modellare una collezione del sistema actorbase. In particolare contiene il creatore della collezione, il nome, la dimensione e la lista di tutti i collaboratori che hanno accesso ad essa.



4.3.3.2.2 Utilizzo

Viene utilizzata da diversi attori del sistema. Questa classe espone metodi getter e un metodo per il confronto con un altro oggetto ActorbaseCollection.

Viene utilizzata dalla maggior parte degli attori per eseguire operazioni sulle collezioni.

4.3.3.2.3 Realizza

- trait scala.math.Ordered

4.3.3.2.4 Interazioni con altre classi

- actorbase::actorsystem::actors::storekeeper::Storekeeper;
- actorbase::actorsystem::actors::storefinder::Storefinder;
- actorbase::actorsystem::actors::authactor::AuthActor;
- actorbase::actorsystem::actors::main::Main;
- actorbase::actorsystem::actors::clientactor::ClientActor.



4.4 actorbase::actorsystem::actors

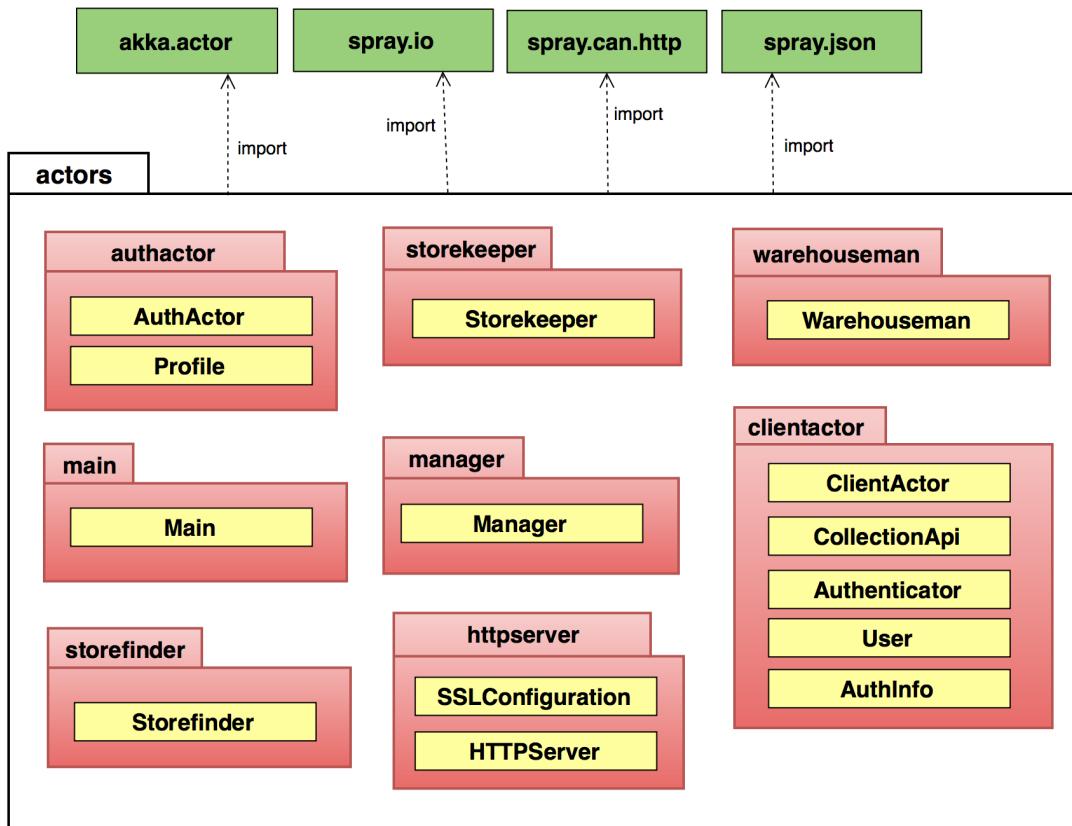


Figura 5: Package_G actors, visione generale

4.4.1 Descrizione

Package_G che raggruppa tutti gli attori necessari al funzionamento del database.

4.4.2 Package contenuti

- `actorbase::actorsystem::actors::clientactor;`
- `actorbase::actorsystem::actors::httpserver;`
- `actorbase::actorsystem::actors::storefinder;`
- `actorbase::actorsystem::actors::storekeeper;`



- actorbase::actorsystem::actors::warehouseman;
- actorbase::actorsystem::actors::manager;
- actorbase::actorsystem::actors::main;
- actorbase::actorsystem::actors::authactor.

4.5 actorbase::actorsystem::actors::clientactor

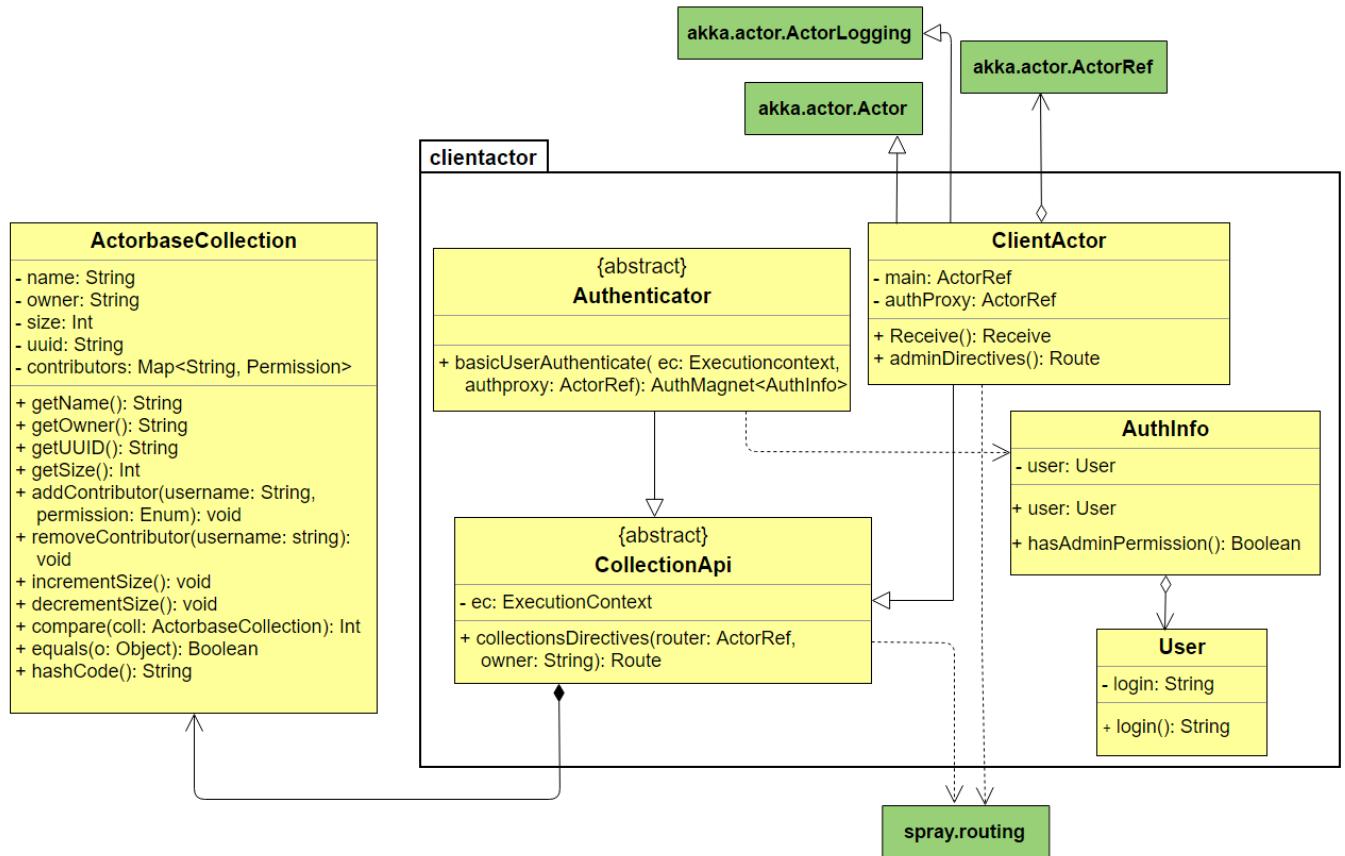


Figura 6: ACtorsystem: package Clientactor

4.5.1 Descrizione

Package_G per l'attore_G con cui si interfacerà il driver_G.

In particolare questo attore avrà il compito di ricevere e instradare tutte le richieste entranti dal proprio client.



4.5.2 Interazioni con altre componenti

- `actorbase::actorsystem::actors::httpserver;`
- `actorbase::driver::client::ActorbaseClient;`
- `actorbase::actorsystem::actors::authactor;`
- `spray.json;`
- `spray.routing;`
- `akka.actor;`
- `com.github.t3hnar.bcrypt.`

4.5.3 Classi

4.5.3.1 `actorbase::actorsystem::actors::clientactor::Authenticator`

4.5.3.1.1 Descrizione

Classe astratta usata per effettuare la validazione di un login.

4.5.3.1.2 Utilizzo

Questa classe viene usata dalla classe che la realizza per effettuare il controllo delle credenziali di login.

4.5.3.1.3 Interazioni con altre classi

- `actorbase::actorsystem::actors::authactor::AuthActor.`

4.5.3.2 `actorbase::actorsystem::actors::clientactor::CollectionApi`

4.5.3.2.1 Descrizione

Classe astratta usata per analizzare le route che richiedono operazioni su collezioni e su items.

4.5.3.2.2 Utilizzo

Questa classe viene usata dalla classe che la realizza per effettuare l'analisi delle route che richiedono operazioni su collezioni.



4.5.3.2.3 Interazioni con altre classi

-
- actorbase::actorsystem::actors::utils::ActorbaseCollection;
- spray.routing.

4.5.3.3 actorbase::actorsystem::actors::clientactor::UserApi

4.5.3.3.1 Descrizione

Classe astratta usata per analizzare le route che richiedono operazioni su utenti.

4.5.3.3.2 Utilizzo

Questa classe viene usata dalla classe che la realizza per effettuare l'analisi delle route che richiedono operazioni su utenti.

4.5.3.3.3 Interazioni con altre classi

- spray.routing.

4.5.3.4 actorbase::actorsystem::actors::clientactor::ClientActor

4.5.3.4.1 Descrizione

Classe che rappresenta l'attore_G con cui si interfaccia il driver_G dopo la connessione. Questo attore avrà il compito di ricevere ed instradare tutte le richieste entranti dal proprio client.

4.5.3.4.2 Utilizzo

Questo attore_G riceve le richieste da actorbase::driver::client::Connection e si occupa di inviare messaggi al Main.

Questa classe inoltre si occupa di capire che tipo di richiesta è stata fatta dal driver_G basandosi sulla route di quest'ultima.

4.5.3.4.3 Classi ereditate

- akka::actor::Actor;
- com::actorbase::actorsystem::actors::clientactor::Authenticator;
- com::actorbase::actorsystem::actors::clientactor::CollectionApi;



- com::actorbase::actorsystem::actors::clientactor::UserApi;.

4.5.3.4.4 Interazioni con altre classi

- actorbase::actorsystem::actors::httpserver::HTTPServer;
- actorbase::driver::client::ActorbaseClient.

4.6 actorbase::actorsystem::actors::httpserver

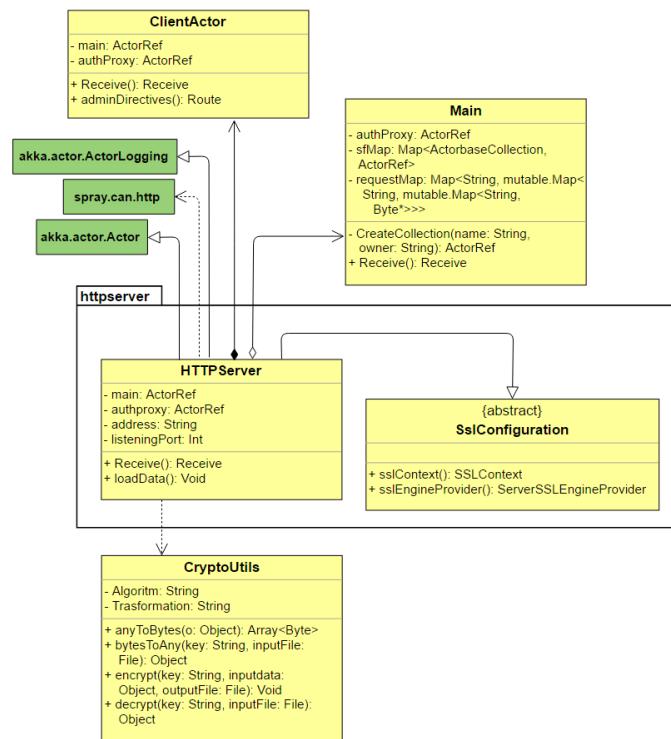


Figura 7: httpserver, visione generale del package

4.6.1 Descrizione

Package_G per l'attore_G con cui si interfacerà il driver_G per la connessione iniziale.



4.6.2 Interazioni con altre componenti

- `actorbase::actorsystem::actors::clientactor;`
- `actorbase::actorsystem::actors::main.`

4.6.3 Classi

4.6.3.1 `actorbase::actorsystem::actors::httpserver::HTTPServer`

4.6.3.1.1 Descrizione

Classe che rappresenta l'attore_G con cui si interfaccia il driver_G per istanziare la connessione.

4.6.3.1.2 Utilizzo

Questo attore_G riceve la richiesta di connessione da `ActorbaseClient` e si occupa di associare al client_G un attore_G di tipo `ClientActor` per continuare le comunicazioni.

4.6.3.1.3 Classi ereditate

- `actorbase::actorsystem::actors::httpserver::SslConfiguration;`
- `akka::actor::Actor.`

4.6.4 Interazioni con altre componenti

- `actorbase::actorsystem::actors::clientactor::ClientActor;`
- `actorbase::actorsystem::actors::main::Main.`

4.6.4.1 `actorbase::actorsystem::actors::httpserver::SslConfiguration`

4.6.4.1.1 Descrizione

Classe che contiene metodi di utilità per applicare algoritmi di crittografia ssl.

4.6.4.1.2 Utilizzo

Questa classe verrà utilizzata per applicare la crittografia SSL.

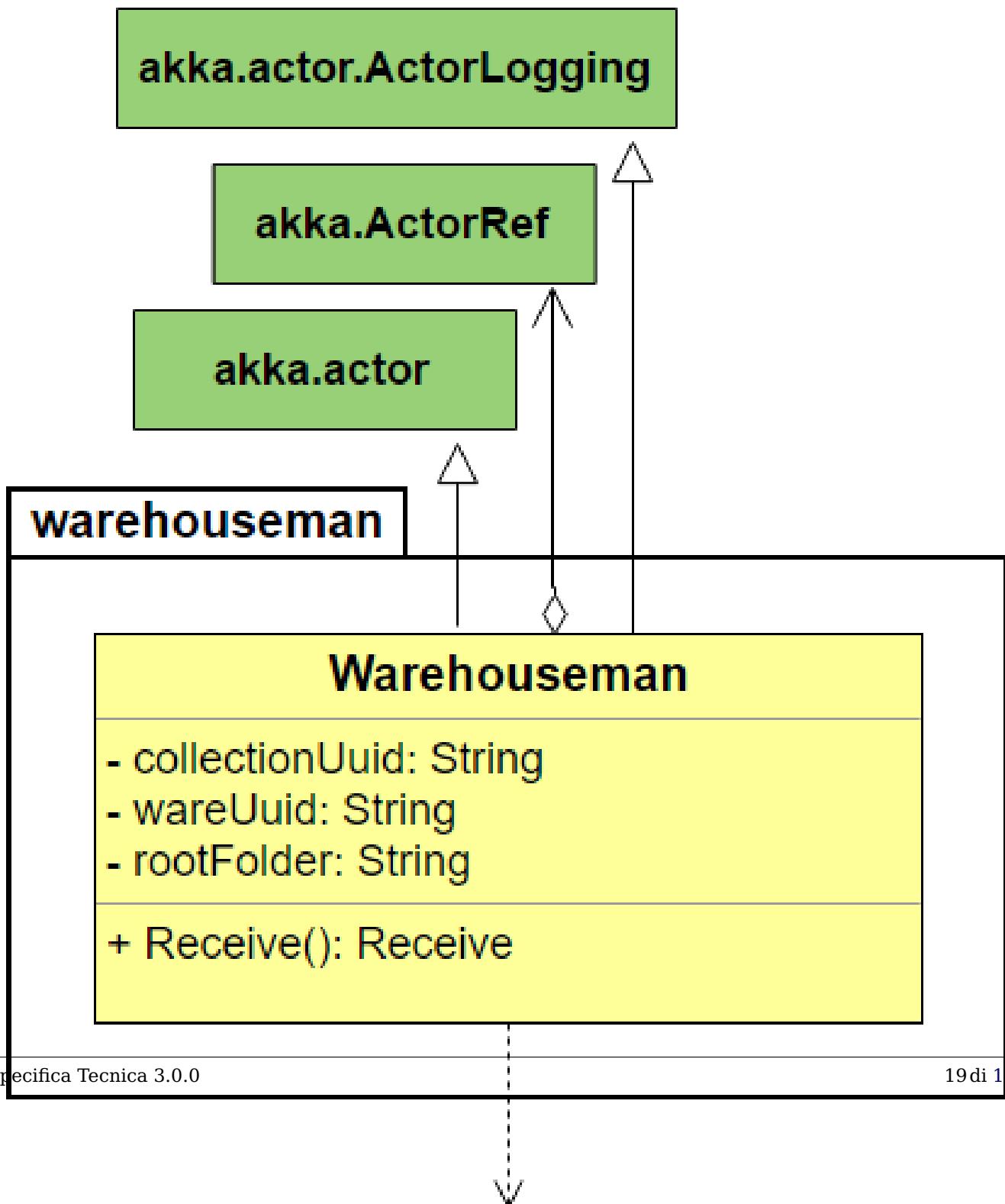


ScalateKids

4 COMPONENTI



4.7 actorbase::actorsystem::actors::warehouseman





4.7.1 Descrizione

Package_G che rappresenta l'attore_G che si occuperà della persistenza_G su disco dei dati. In particolare questo attore si occuperà di persistere su disco tutte le coppie chiave valore del database.

4.7.2 Interazioni con altre componenti

- actorbase::actorsystem::utils;
- actorbase::actorsystem::actors::storekeeper;
- actorbase::actorsystem::actors::storefinder.

4.7.3 Classi

4.7.3.1 actorbase::actorsystem::actors::warehouseman::Warehouseman

4.7.3.1.1 Descrizione

Classe che rappresenta un attore_G di tipo Warehouseman_G.

4.7.3.1.2 Utilizzo

Questa classe viene utilizzata per effettuare il salvataggio su filesystem di tutti i dati salvati nel database.

4.7.3.1.3 Classi ereditate

- akka::actor::Actor.

4.7.3.1.4 Interazioni con altre classi

- actorbase::actorsystem::CryptoUtils;
- actorbase::actorsystem::actors::storekeeper::Storekeeper;
- actorbase::actorsystem::actors::storefinder::Storefinder.



4.8 actorbase::actorsystem::actors::warehouseman::messages

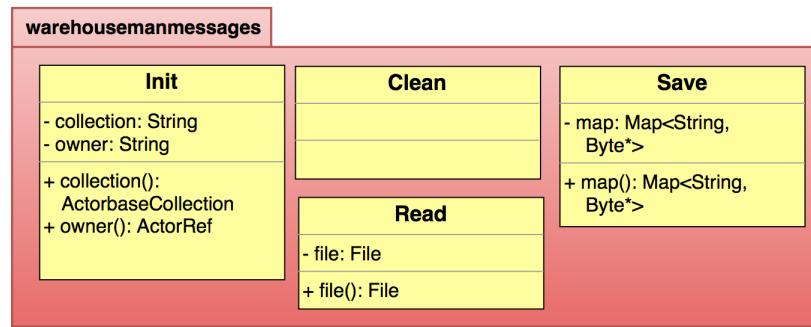


Figura 9: Actorsystem: package messages del warehouseman

4.8.1 Descrizione

Package_G che racchiude tutti i messaggi che gli attori di tipo Warehouseman_G possono ricevere.

4.8.2 Interazioni con altre componenti

- actorbase::actorsystem::utils.



4.9 actorbase::actorsystem::actors::main

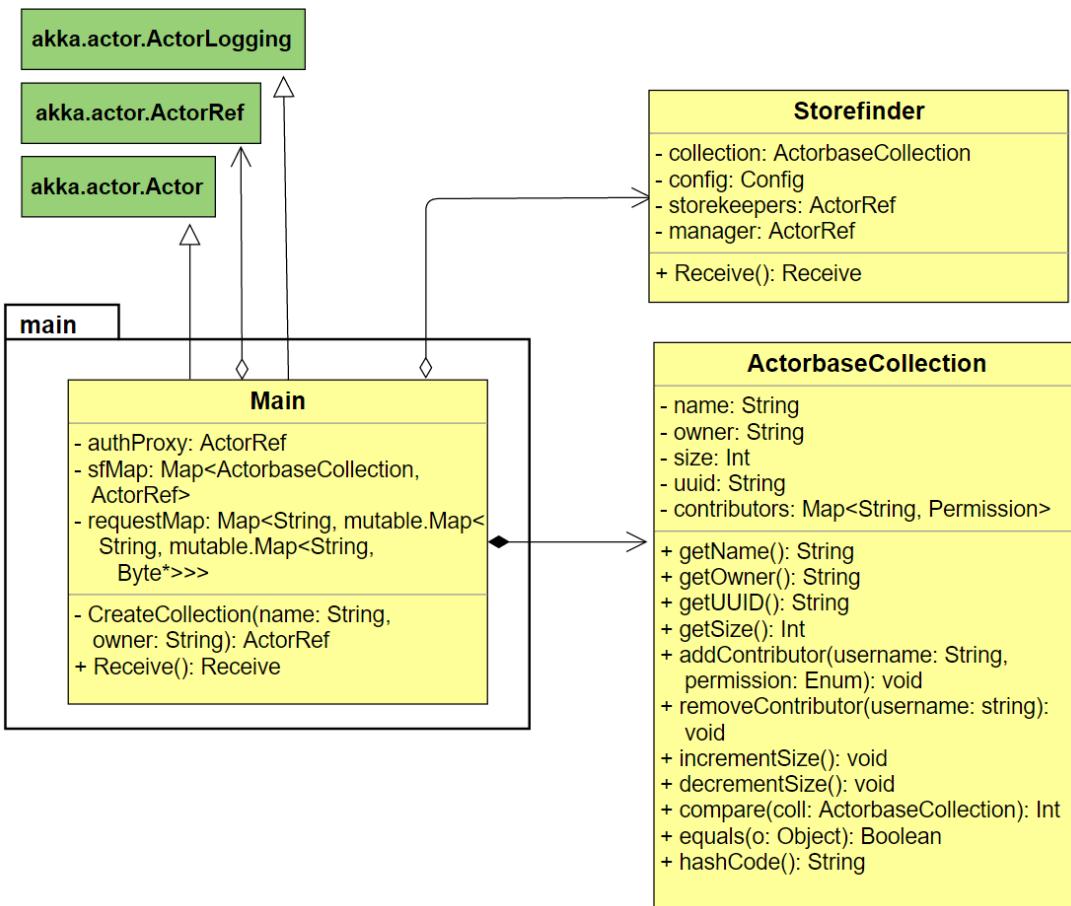


Figura 10: Package dell'attore di tipo Main

4.9.1 Descrizione

Package_G che rappresenta l'attore_G che si occuperà di gestire le richieste al server_G.

4.9.2 Interazioni con altre componenti

- `actorbase::actorsystem::actors::httpserver`;
- `actorbase::actorsystem::utils`;
- `actorbase::actorsystem::actors::storefinder`;



- `actorbase::actorsystem::actors::authactor.`

4.9.3 Classi

4.9.3.1 `actorbase::actorsystem::actors::main::Main`

4.9.3.1.1 Descrizione

Classe che rappresenta un attore_G di tipo Main_G.

4.9.3.1.2 Utilizzo

Questa classe viene utilizzata per gestire le richieste ricevute al server_G. In particolare questo attore conterrà riferimenti alle mappe degli items inseriti, così da poter instradare le richieste ai giusti attori quando si presenterà la necessità.

4.9.3.1.3 Classi ereditate

- `akka::actor::Actor.`

4.9.3.1.4 Interazioni con altre classi

- `actorbase::actorsystem::actors::httpserver::HTTPServer;`
- `actorbase::actorsystem::utils::ActorbaseCollection;`
- `actorbase::actorsystem::actors::storefinder::Storefinder;`
- `actorbase::actorsystem::actors::authactor::AuthActor.`

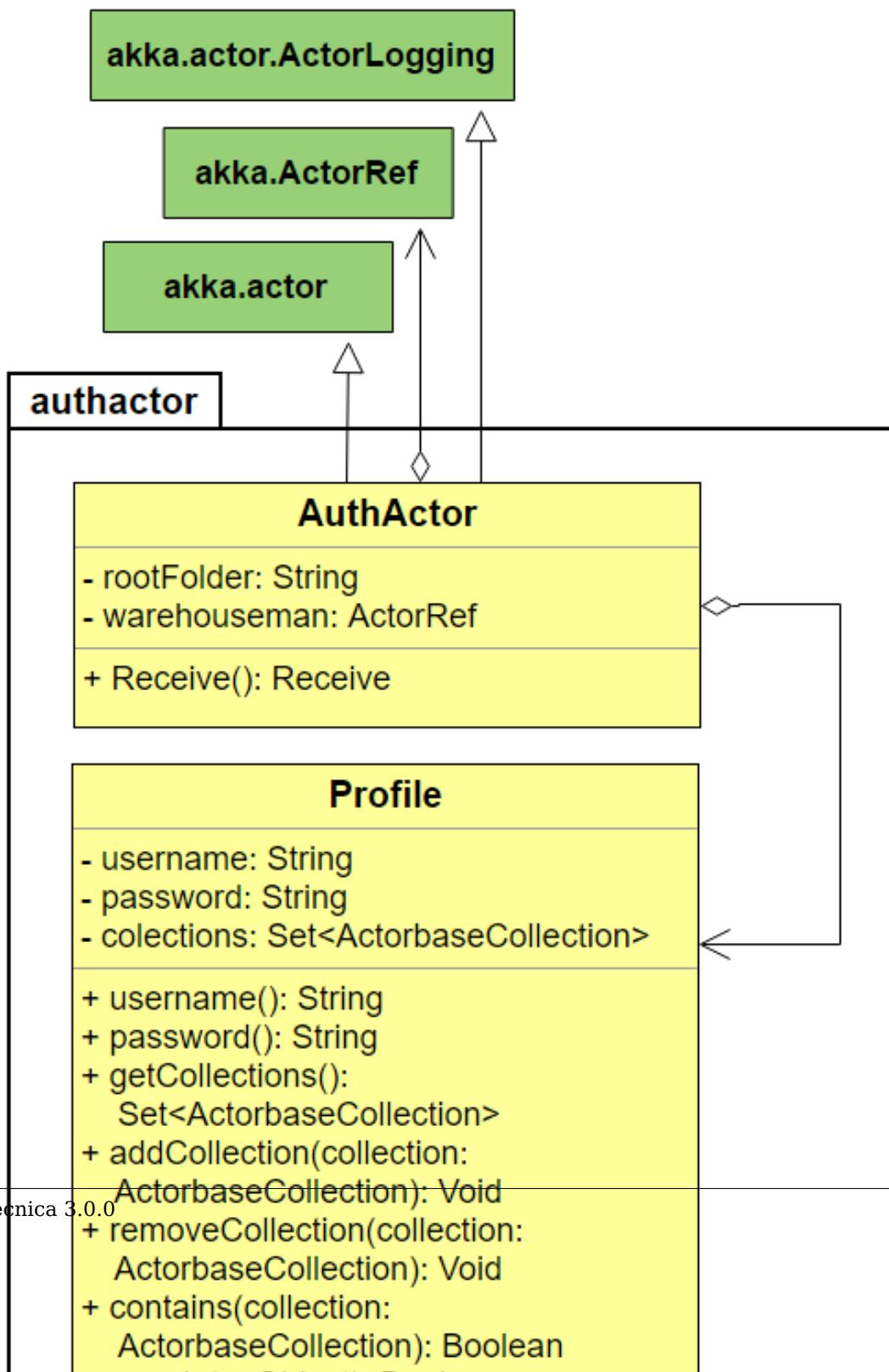


ScalateKids

4 COMPONENTI



4.10 actorbase::actorsystem::actors::authactor





4.10.1 Descrizione

Classe che rappresenta un attore_G di tipo Authactor_G.

4.10.2 Classi

4.10.2.1 actorbase::actorsystem::actors::authactor::Authactor

4.10.2.1.1 Descrizione

Classe che rappresenta un attore_G di tipo Authactor_G.

4.10.2.1.2 Utilizzo

Questa classe viene utilizzata per gestire la parte di autenticazione del server.

4.10.2.1.3 Classi ereditate

- akka::actor::Actor.

4.10.2.1.4 Interazioni con altre classi

- [actorbase::actorsystem::actors::authactor::Profile](#).

4.10.2.2 actorbase::actorsystem::actors::authactor::Profile

4.10.2.2.1 Descrizione

Classe che modella un utente registrato al sistema actorbase.

4.10.2.2.2 Utilizzo

Questa classe viene utilizzata da [Authactor](#) per gestire l'autenticazione del server.

4.10.2.2.3 Interazioni con altre classi

- [actorbase::actorsystem::authactor::Authactor](#).



4.11 actorbase::actorsystem::actors::storefinder

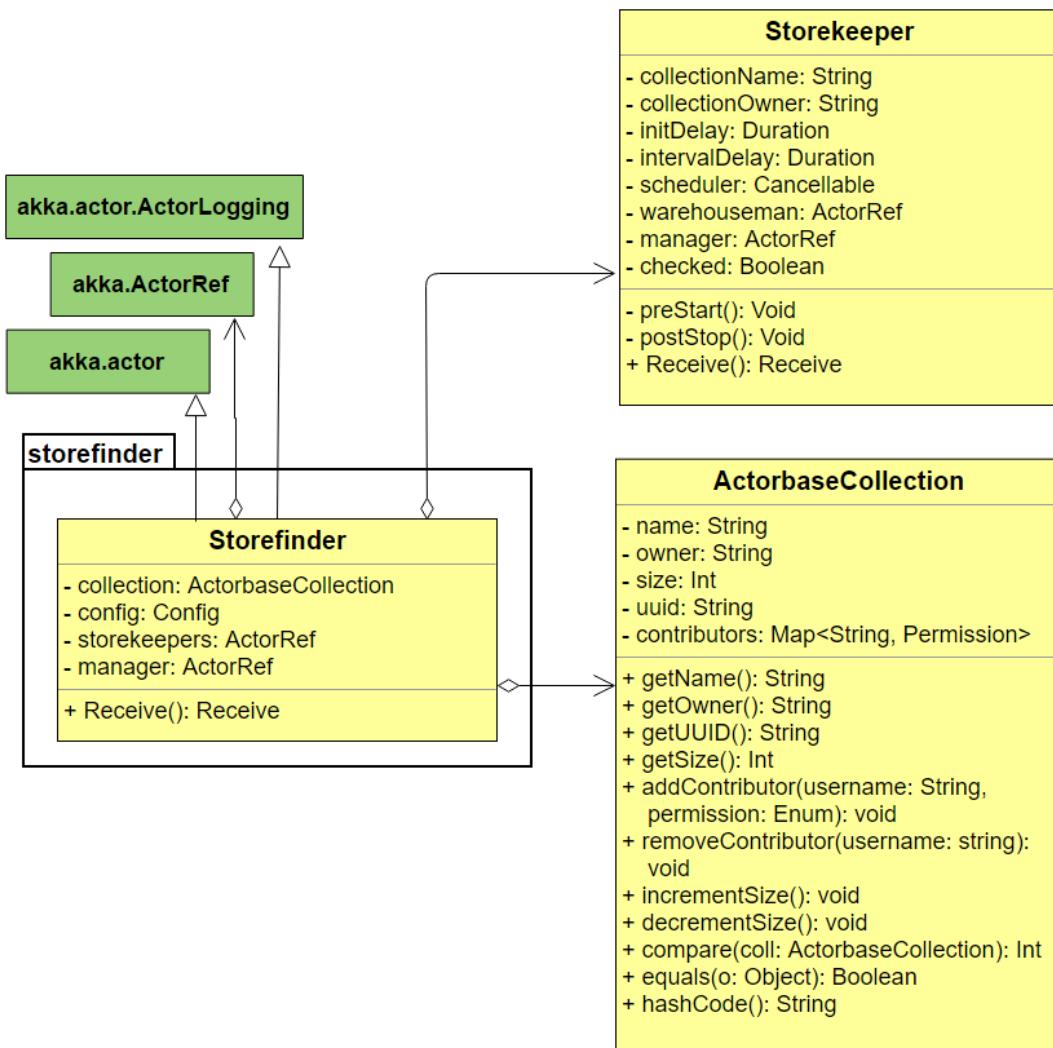


Figura 12: Storefinder, visione generale del package

4.11.1 Descrizione

Classe che rappresenta un attore_G di tipo Storefinder_G.



4.11.2 Interazioni con altre componenti

- `actorbase::actorsystem::actors::main;`
- `actorbase::actorsystem::utils;`
- `actorbase::actorsystem::actors::storekeeper;`
- `actorbase::actorsystem::actors::warehouseman.`

4.11.3 Classi

4.11.3.1 `actorbase::actorsystem::actors::storefinder::Storefinder`

4.11.3.1.1 Descrizione

Classe che rappresenta un attore_G di tipo `StorefinderG`.

4.11.3.1.2 Utilizzo

Questa classe viene utilizzata per tener traccia degli attori di tipo `actorbase::actorsystem::actors::storekeeper::Storekeeper` potendo così scegliere il giusto attore per soddisfare le richieste in ingresso dai clients.

4.11.3.1.3 Classi ereditate

- `akka::actor::Actor.`

4.11.3.1.4 Interazioni con altre classi

- `actorbase::actorsystem::actors::main::Main;`
- `actorbase::actorsystem::utils::ActorbaseCollection;`
- `actorbase::actorsystem::actors::storekeeper::Storekeeper;`
- `actorbase::actorsystem::actors::warehouseman::Warehouseman.`



4.12 actorbase::actorsystem::actors::storekeeper

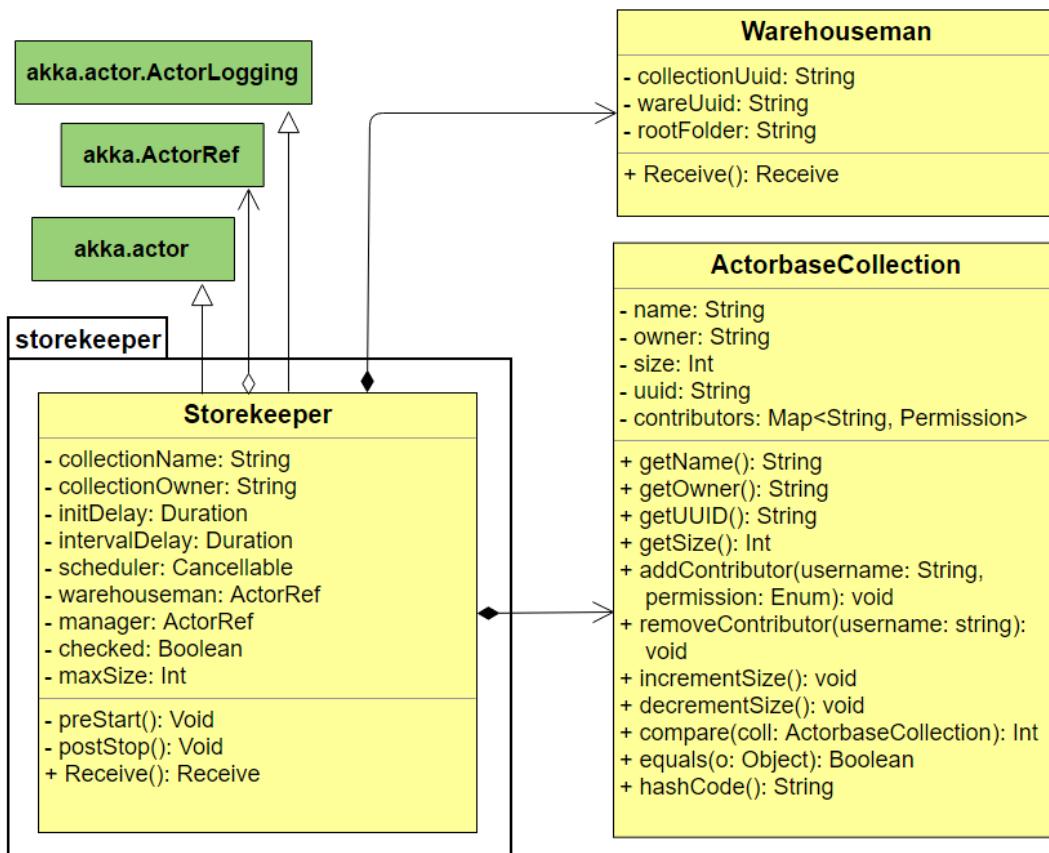


Figura 13: Storekeeper, visione generale del package

4.12.1 Descrizione

Package_G che rappresenta l'attore_G di tipo Storekeeper_G.

4.12.2 Interazioni con altre componenti

- `actorbase::actorsystem::utils`;
- `actorbase::actorsystem::actors::storefinder`;
- `actorbase::actorsystem::actors::warehouseman`.



4.12.3 Classi

4.12.3.1 actorbase::actorsystem::actors::storekeeper::Storekeeper

4.12.3.1.1 Descrizione

Classe che rappresenta un attore_G di tipo Storekeeper_G.

4.12.3.1.2 Utilizzo

Questa classe viene utilizzata per salvare nel database_G gli items.

4.12.3.1.3 Classi ereditate

- akka::actor::Actor.

4.12.3.1.4 Interazioni con altre classi

- actorbase::actorsystem::utils::ActorbaseCollection;
- actorbase::actorsystem::actors::storefinder::Storefinder;
- actorbase::actorsystem::actors::warehouseman::Warehouseman.



4.13 actorbase::actorsystem::actors::manager

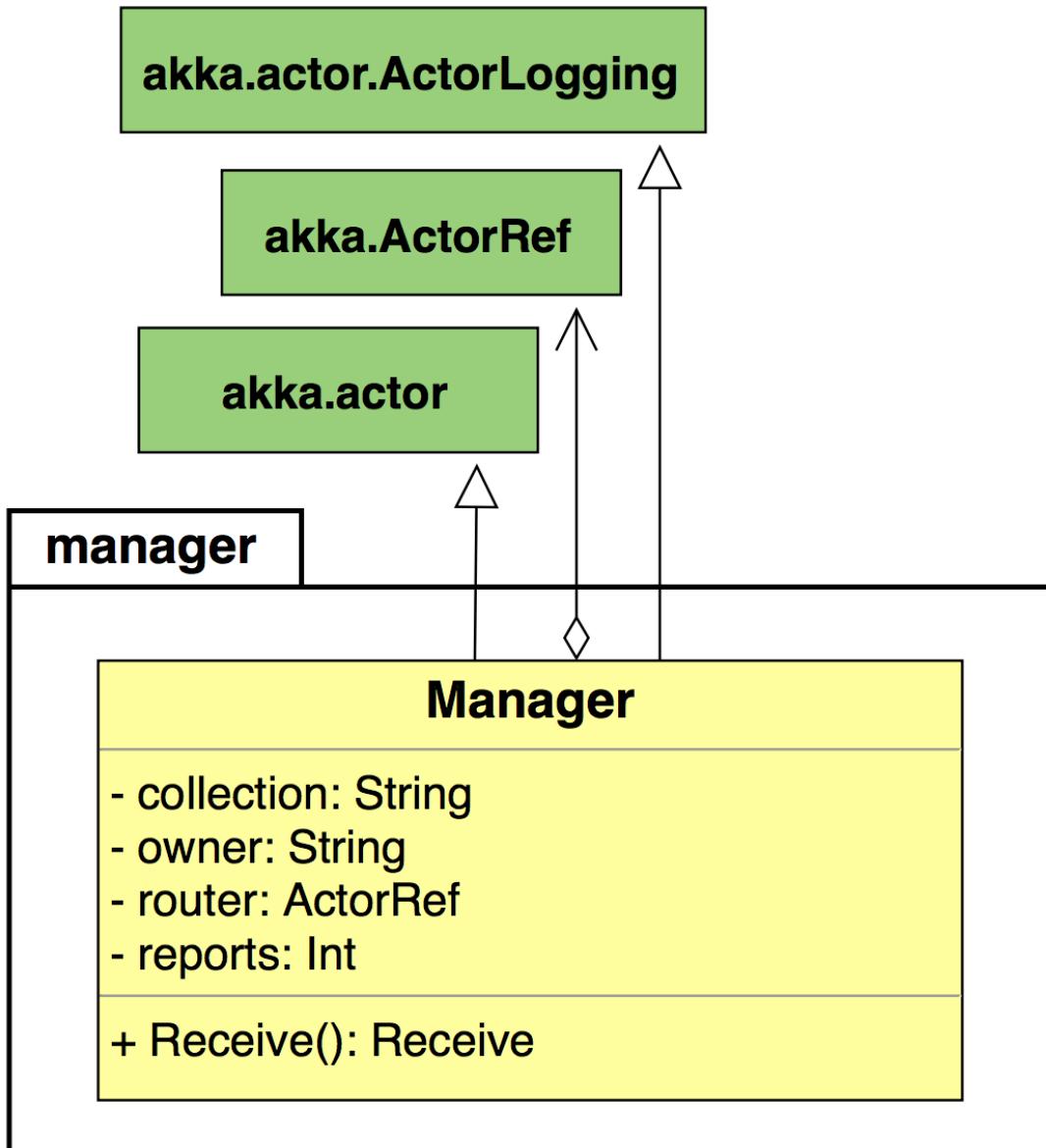


Figura 14: Manager, visione generale del package



4.13.1 Descrizione

Package_G che rappresenta l'attore_G di tipo Manager_G.

4.13.2 Interazioni con altre componenti

- `actorbase::actorsystem::utils`.

4.13.3 Classi

4.13.3.1 `actorbase::actorsystem::actors::manager::Manager`

4.13.3.1.1 Descrizione

Classe che rappresenta un attore_G di tipo Manager_G.

4.13.3.1.2 Utilizzo

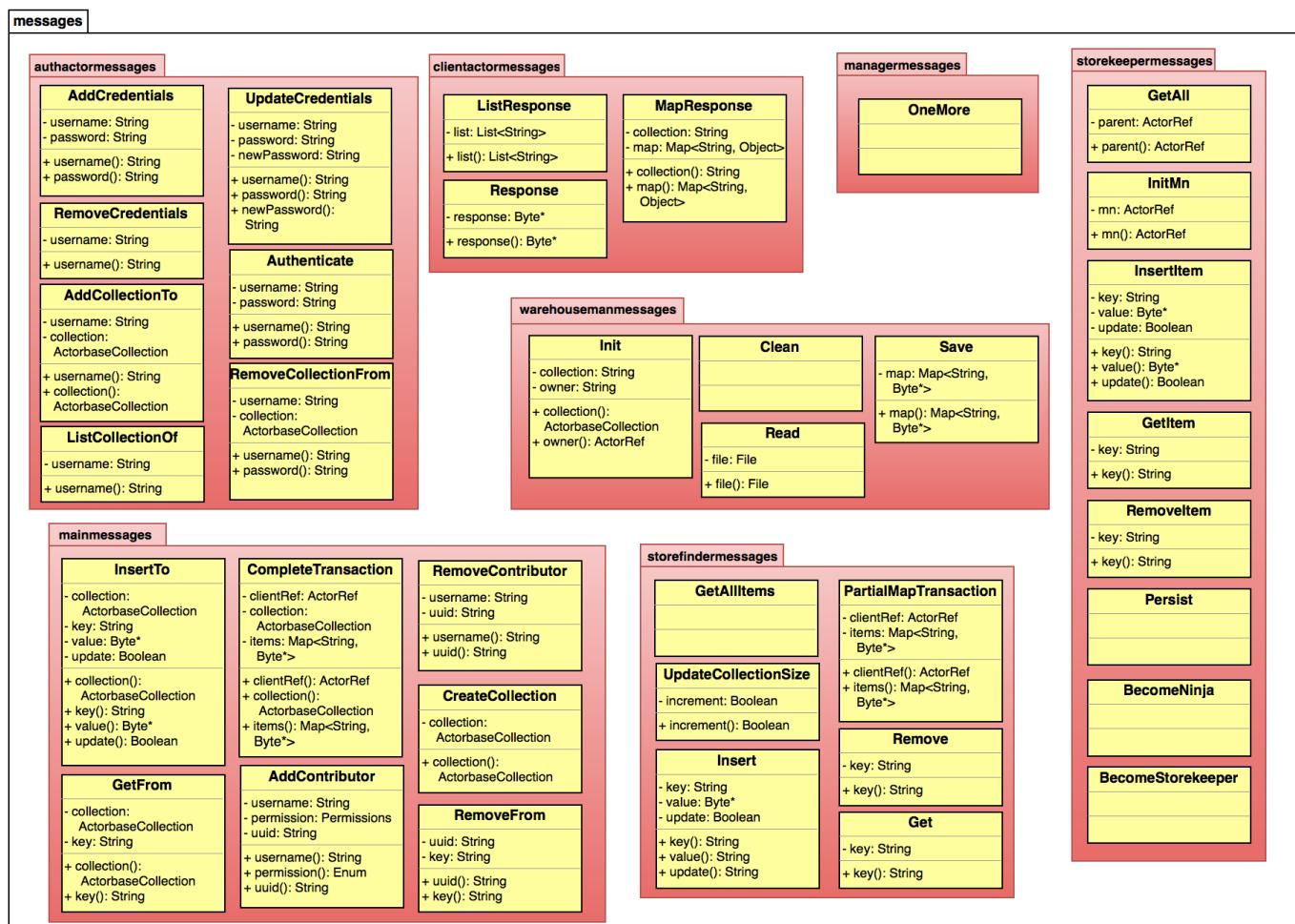
Questa classe viene utilizzata per gestire le richieste di creazione di nuovi Storekeeper quando questi raggiungono il limite di item previsto dal sistema.

4.13.3.1.3 Classi ereditate

- `akka::actor::Actor`.



4.14 actorbase::actorsystem::messages

Figura 15: Package_G messages

4.14.1 Descrizione

Package_G che raggruppa tutti i messaggi del server_G.

4.14.2 Package contenuti

- actorbase::actorsystem::messages::authactormessages;



- actorbase::actorsystem::messages::clientactormessages;
- actorbase::actorsystem::messages::mainmessages;
- actorbase::actorsystem::messages::storefindermessages;
- actorbase::actorsystem::messages::storekeepermessages;
- actorbase::actorsystem::messages::warehousemanmessages;
- actorbase::actorsystem::messages::manangermessages.

4.15 actorbase::actorsystem::messages::authactormessages

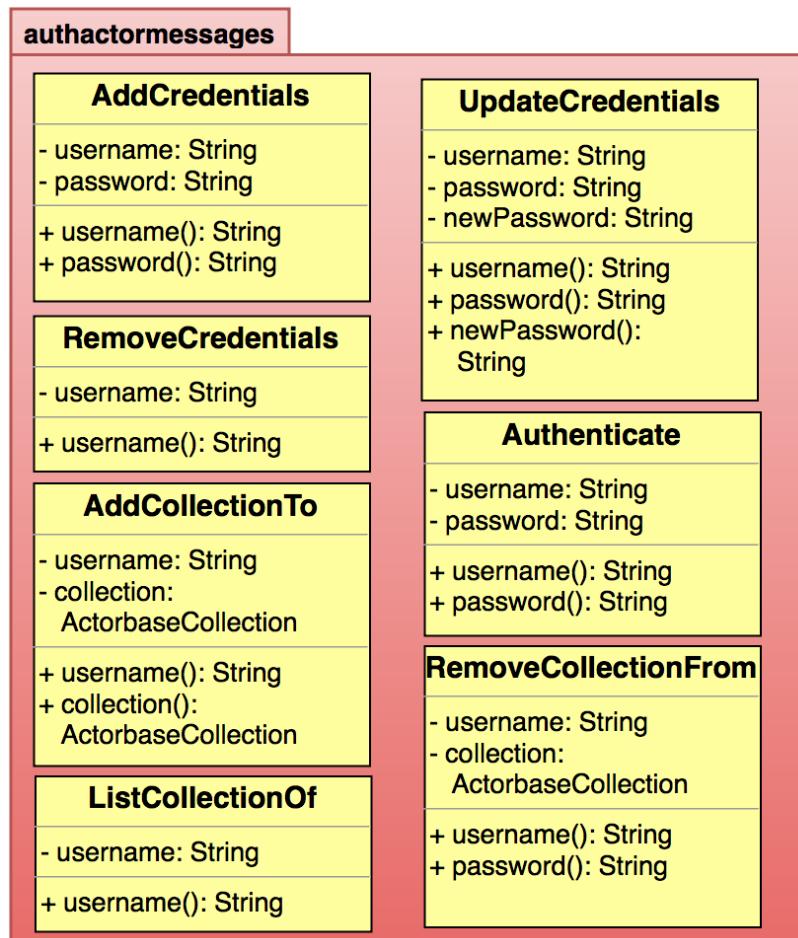


Figura 16: Package `actorsystem::messages::authactormessages`



4.15.1 Descrizione

Package_G contenente i messaggi processabili dall'attore AuthActor.

4.15.2 Classi

4.15.2.1 actorbase::actorsystem::messages::authactormessages::AddCredentials

4.15.2.1.1 Descrizione

Classe che rappresenta una richiesta di aggiunta di credenziali al sistema.

Implementata utilizzando la specifica funzionalità *case class* di Scala_G, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.15.2.1.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta all'attore_G AuthActor di aggiunta di credenziali al sistema, all'interno del metodo *Receive* che la utilizza avviene inoltre il controllo della password, in modo che rispetti i vincoli imposti dal sistema.

4.15.2.2 actorbase::actorsystem::messages::authactormessages::UpdateCredentials

4.15.2.2.1 Descrizione

Classe che rappresenta una richiesta di modifica di password di un utente al sistema.

Implementata utilizzando la specifica funzionalità *case class* di Scala_G, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.15.2.2.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta all'attore AuthActor di modifica della password di un utente.

4.15.2.3 actorbase::actorsystem::messages::authactormessages::RemoveCredentials

4.15.2.3.1 Descrizione

Classe che rappresenta una richiesta di rimozione di un utente dal sistema.

Implementata utilizzando la specifica funzionalità *case class* di Scala_G, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.



4.15.2.3.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta all'attore_G AuthActor di rimozione di un utente.

4.15.2.4 actorbase::actorsystem::messages::authactormessages::Authenticate

4.15.2.4.1 Descrizione

Classe che rappresenta una richiesta di login di un utente al sistema.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.15.2.4.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta all'attore AuthActor di accesso al sistema. L'attore AuthActor dovrà controllare se le credenziali immesse dall'utente sono corrette.

4.15.2.5 actorbase::actorsystem::messages::authactormessages::AddCollectionTo

4.15.2.5.1 Descrizione

Classe che rappresenta una richiesta di inserimento di una collezione_G ad uno specifico utente del sistema.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.15.2.5.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta all'attore AuthActor di inserimento nuova collezione_G all'utente specificato. L'attore AuthActor dovrà controllare che l'utente a cui aggiungere la collezione_G sia presente nel sistema e che la collezione_G da inserire non sia già presente associata all'utente stesso.

4.15.2.6 actorbase::actorsystem::messages::authactormessages::RemoveCollectionFrom

4.15.2.6.1 Descrizione

Classe che rappresenta una richiesta di rimozione di una collezione_G associata ad uno specifico utente del sistema.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.



4.15.2.6.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta all'attore AuthActor di rimozione di una collezione_G associata all'utente specificato. L'attore AuthActor dovrà controllare che l'utente a cui rimuovere la collezione_G sia presente nel sistema e che la collezione_G da rimuovere sia effettivamente presente e associata all'utente stesso.

4.15.2.7 actorbase::actorsystem::messages::authactormessages::ListCollectionsOf

4.15.2.7.1 Descrizione

Classe che rappresenta una richiesta di visualizzazione delle collezioni_G associate ad uno specifico utente del sistema.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.15.2.7.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta all'attore AuthActor di visualizzazione delle collezioni_G associate all'utente specificato. L'attore AuthActor dovrà controllare che l'utente specificato sia presente nel sistema.



4.16 actorbase::actorsystem::messages::clientactormessages

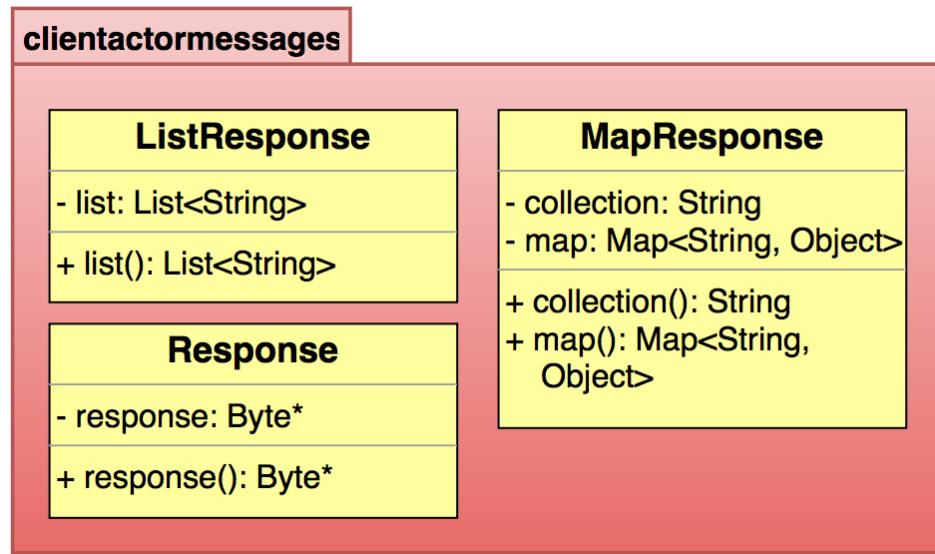


Figura 17: Package actorsystem::messages::clientactormessages

4.16.1 Descrizione

Package_G contenente i messaggi processabili dall'attore [ClientActor](#).

4.16.2 Classi

4.16.2.1 actorbase::actorsystem::messages::clientactormessages::ListResponse

4.16.2.1.1 Descrizione

Classe che rappresenta un messaggio utilizzato per consegnare al [ClientActor](#) una lista di stringhe, utilizzato per adempiere ad alcune richieste dell'utente.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.16.2.1.2 Utilizzo

Questa classe viene utilizzata per mandare una lista di stringhe all'attore [ClientActor](#). Questo viene utilizzato per adempiere ad alcune richieste dell'utente come la richiesta di lista collezioni.



4.16.2.2 `actorbase::actorsystem::messages::clientactormessages::MapResponse`

4.16.2.2.1 Descrizione

Classe che rappresenta un messaggio utilizzato per consegnare al `ClientActor` una collezione dati, utilizzato per adempiere ad alcune richieste dell'utente.

Implementata utilizzando la specifica funzionalità `case class` di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching₆.

4.16.2.2.2 Utilizzo

Questa classe viene utilizzata per mandare una collezione dati all'attore `ClientActor`. Questo viene utilizzato per adempiere ad alcune richieste dell'utente come la richiesta di restituzione di una collezione.

4.16.2.3 `actorbase::actorsystem::messages::clientactormessages::Response`

4.16.2.3.1 Descrizione

Classe che rappresenta un messaggio utilizzato per consegnare al `ClientActor` una stringa, utilizzato per adempiere ad alcune richieste dell'utente.

Implementata utilizzando la specifica funzionalità `case class` di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching₆.

4.16.2.3.2 Utilizzo

Questa classe viene utilizzata per mandare una stringa all'attore `ClientActor`. Questo viene utilizzato per adempiere ad alcune richieste dell'utente.



4.17 actorbase::actorsystem::messages::mainmessages

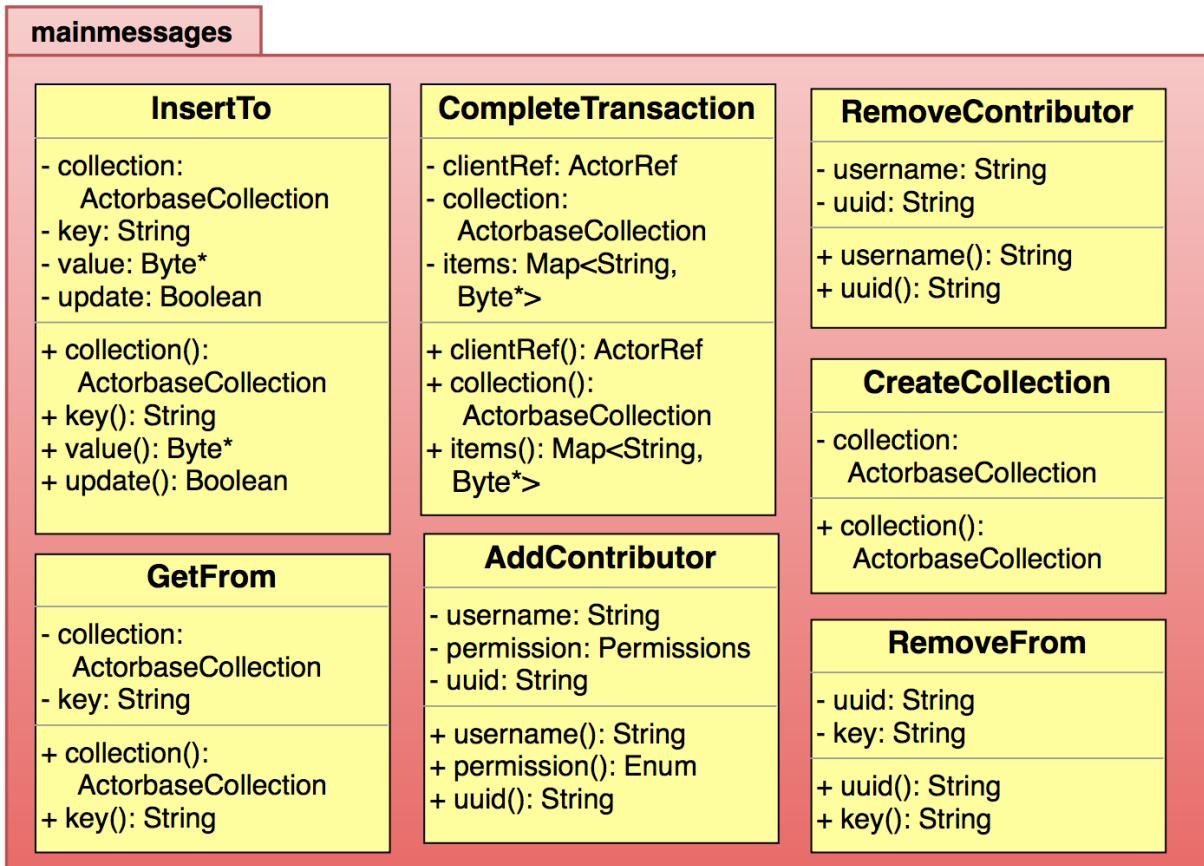


Figura 18: Package actorsystem::messages::mainmessages

4.17.1 Descrizione

Package_G contenente i messaggi processabili dall'attore `Main`.



4.17.2 Classi

4.17.2.1 actorbase::actorsystem::messages::mainmessages::InsertTo

4.17.2.1.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere al [Main](#) l'inserimento di un item. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching₆.

4.17.2.1.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta di inserimento di un item all'attore [Main](#). Questo attore provverà ad inoltrare la richiesta al giusto attore [Storefinder](#) che rappresenta la collezione_G in cui inserire l'item. Se non è presente la collezione in cui inserire l'item questa verrà creata istanziando un nuovo attore [Storefinder](#), infine verrà inoltrata la richiesta di inserimento a quest'ultimo.

4.17.2.2 actorbase::actorsystem::messages::mainmessages::GetFrom

4.17.2.2.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere al [Main](#) di ritornare items o collezioni. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching₆.

4.17.2.2.2 Utilizzo

Questa classe viene utilizzata per mandare una richiesta di ritorno di un item o di una collezione all'attore [Main](#). Questo attore provverà ad inoltrare la richiesta al giusto attore [Storefinder](#). Il parametro in ingresso *key* determina se l'utente richiede un singolo item o una intera collezione.

4.17.2.3 actorbase::actorsystem::messages::mainmessages::CompleteTransaction

4.17.2.3.1 Descrizione

Classe che rappresenta un messaggio utilizzato per assemblare parti di collezioni_G provenienti dagli attori_G di tipo [Storekeeper](#) verso l'attore_G [Main](#). Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching₆.



4.17.2.3.2 Utilizzo

Questa classe viene utilizzata per assemblare le parti di collezione_G provenienti dagli attori_G di tipo **Storekeeper** verso l'attore_G **Main**.

Quest'ultimo dovrà attendere tutti i messaggi **CompleteTransaction** necessari per costruire un intera collezione_G da restituire all'utente.

Questo messaggio è utilizzato quando un utente richiede un intera collezione_G.

4.17.2.4 actorbase::actorsystem::messages::mainmessages::RemoveFrom

4.17.2.4.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere la rimozione di item all'attore **Main**.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.17.2.4.2 Utilizzo

Questa classe viene utilizzata per rimuovere items dal sistema. L'attore **Main** dovrà inoltrare la richiesta al giusto attore **Storefinder**, ovvero colui che rappresenta la collezione in cui effettuare l'operazione.

4.17.2.5 actorbase::actorsystem::messages::mainmessages::AddContributor

4.17.2.5.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere l'aggiunta di un collaboratore ad una collezione_G all'attore **Main**.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.17.2.5.2 Utilizzo

Questa classe viene utilizzata per richiedere l'aggiunta di collaboratori ad una collezione. L'attore **Main** dovrà aggiungere lo username del collaboratore designato alla collezione richiesta.

4.17.2.6 actorbase::actorsystem::messages::mainmessages::RemoveContributor

4.17.2.6.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere la rimozione di un collaboratore ad una collezione all'attore **Main**.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.



4.17.2.6.2 Utilizzo

Questa classe viene utilizzata per richiedere la rimozione di collaboratori ad una collezione. L'attore **Main** dovrà rimuovere lo username del collaboratore designato alla collezione richiesta.

4.17.2.7 actorbase::actorsystem::messages::mainmessages::CreateCollection

4.17.2.7.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere la creazione di una collezione all'attore **Main**. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching.

4.17.2.7.2 Utilizzo

Questa classe viene utilizzata per richiedere la creazione di una collezione.

L'attore **Main** provverà a creare la nuova collezione e a creare un attore **Storefinder** adibito alla gestione di quest'ultima.



4.18 actorbase::actorsystem::messages::storefindermessages

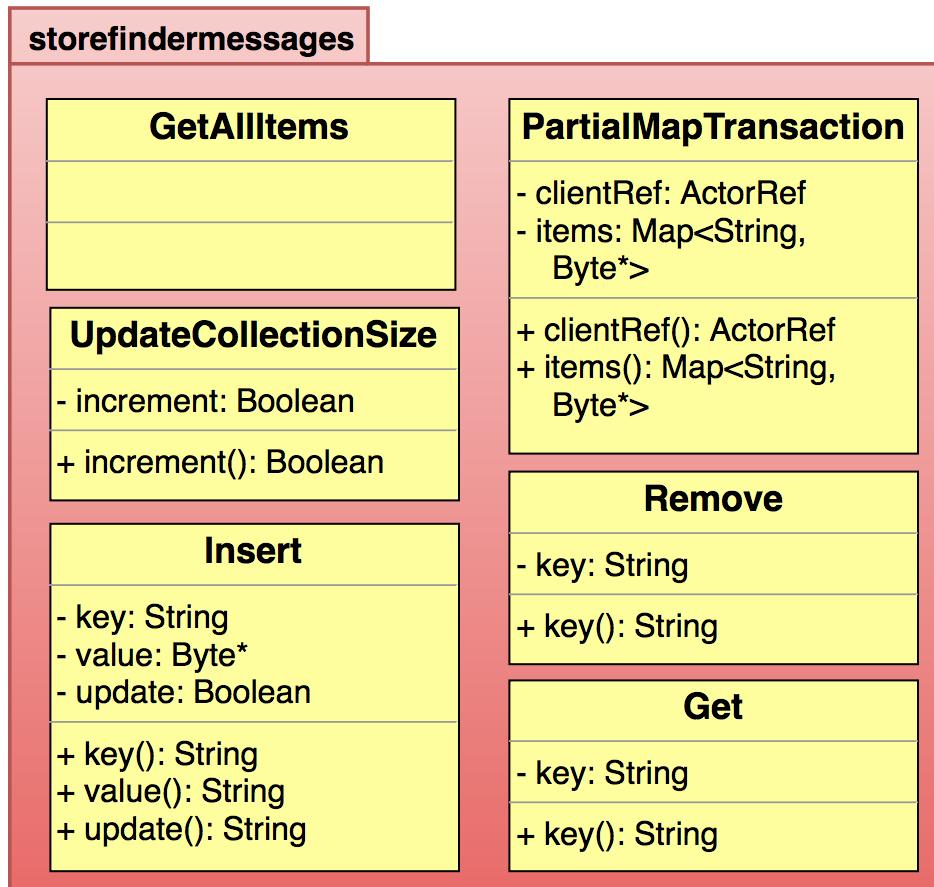


Figura 19: Package `actorsystem::messages::storefindermessages`

4.18.1 Descrizione

Package_G contenente i messaggi processabili dall'attore `Storefinder`.



4.18.2 Classi

4.18.2.1 actorbase::actorsystem::messages::storefinder::GetAllItems

4.18.2.1.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere allo [Storefinder](#) di ritornare tutti gli item mappati dai propri [Storekeeper](#).

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.18.2.1.2 Utilizzo

Questa classe viene utilizzata per richiedere tutti gli items mappati dallo [Storefinder](#).

Questo attore provvederà a mandare richieste Get agli attori [Storekeeper](#).

Questo tipo di messaggio è ricevuto quando un utente intende richiedere intere collezioni.

4.18.2.2 actorbase::actorsystem::messages::storefinder::UpdateCollectionSize

4.18.2.2.1 Descrizione

Classe che rappresenta un messaggio utilizzato per aggiornare il numero di item contenuti in una collezione.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.18.2.2.2 Utilizzo

Questa classe viene utilizzata per tenere aggiornate le dimensioni delle collezioni quando si inseriscono o rimuovono items da esse.

4.18.2.3 actorbase::actorsystem::messages::storefinder::Get

4.18.2.3.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere un item.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.18.2.3.2 Utilizzo

Questa classe viene utilizzata per richiedere un item da restituire all'utente.

Quando riceve questo messaggio l'attore provvederà a inoltrare la richiesta allo [Storekeeper](#) esatto, ovvero colui che contiene l'item mappato dalla chiave inserita.



4.18.2.4 actorbase::actorsystem::messages::storefinder::PartialMapTransaction

4.18.2.4.1 Descrizione

Classe che rappresenta un messaggio utilizzato per ricevere parti di collezione_G da inviare all'attore parent_G. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.18.2.4.2 Utilizzo

Questa classe viene utilizzata per ricevere parti di collezione_G a seguito di una richiesta di un' intera collezione_G da parte di un utente.

Quando riceve questo messaggio l'attore_G provvederà a inoltrare la mappa di item_G all'attore_G **Main**, il quale provvederà ad assemblare tutta la collezione_G da restituire all'utente.

4.18.2.5 actorbase::actorsystem::messages::storefinder::Remove

4.18.2.5.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere la rimozione di un item_G.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.18.2.5.2 Utilizzo

Questa classe viene utilizzata per richiedere la rimozione di un item_G dal sistema.

Quando riceve questo messaggio l'attore provvederà a inoltrare la richiesta allo **Storekeeper** giusto.

4.18.2.6 actorbase::actorsystem::messages::storefinder::Insert

4.18.2.6.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere l'aggiunta di un item.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.18.2.6.2 Utilizzo

Questa classe viene utilizzata per richiedere l'aggiunta di un item dal sistema.

Quando riceve questo messaggio l'attore provvederà a inoltrare la richiesta allo **Storekeeper** giusto.



4.19 actorbase::actorsystem::messages::storekeepermessages

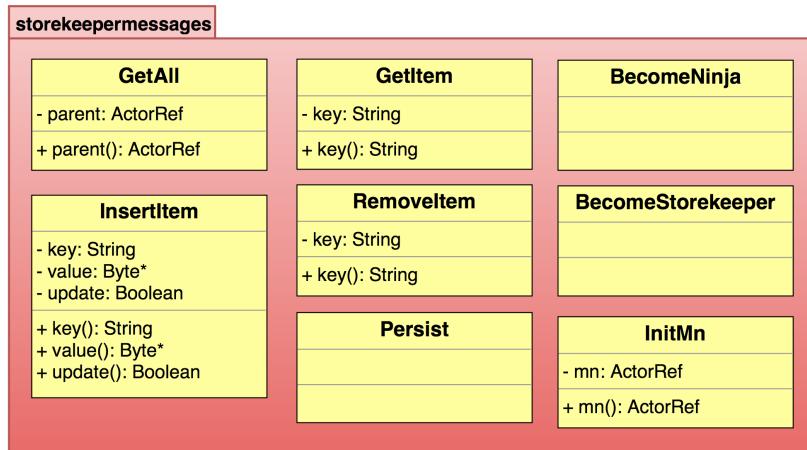


Figura 20: Package `actorsystem::messages::storekeepermessages`

4.19.1 Descrizione

Package_G contenente i messaggi processabili dall'attore `Storekeeper`.

4.19.2 Classi

4.19.2.1 actorbase::actorsystem::messages::storekeeper::Persist

4.19.2.1.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere allo `Storekeeper` di persistere i dati su disco. Implementata utilizzando la specifica funzionalità `case class` di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.19.2.1.2 Utilizzo

Questa classe viene utilizzata per richiedere di effettuare persistenza su disco dei dati di questo attore. Questo attore provvederà a inoltrare la richiesta all'attore `Warehouseman`.



4.19.2.2 actorbase::actorsystem::messages::storekeeper::InitMn

4.19.2.2.1 Descrizione

Classe che rappresenta un messaggio utilizzato per inizializzare un riferimento ad un attore Manager. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.19.2.2.2 Utilizzo

Questa classe viene utilizzata per ricevere un riferimento ad un attore Manager. Questo messaggio verrà inviato allo Storekeeper subito dopo la sua creazione.

L'attore userà il riferimento al Manager per informarlo quando raggiunge una certa soglia di items. Il Manager creerà altri Storekeeper per distribuire il carico.

4.19.2.3 actorbase::actorsystem::messages::storekeeper::GetAll

4.19.2.3.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere tutti gli items contenuti nell'attore. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.19.2.3.2 Utilizzo

Questa classe viene utilizzata per ricevere una richiesta di restituzione di tutti gli items mappati da questo attore. Quando questo attore riceve questo messaggio provvederà a inviare tutti gli items contenuti in esso al riferimento all'attore passato come parametro.

Questo messaggio serve a restituire all'utente intere collezioni.

4.19.2.4 actorbase::actorsystem::messages::storekeeper::GetItem

4.19.2.4.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere un item contenuto nell'attore. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.19.2.4.2 Utilizzo

Questa classe viene utilizzata per richiedere la restituzione di un item mappato da questo attore. Quando questo attore riceve questo messaggio provvederà a inviare l'item richiesto all'attore ClientActor che ha effettuato la richiesta.



4.19.2.5 actorbase::actorsystem::messages::storekeeper::InsertItem

4.19.2.5.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere l'inserimento di un item. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching₆.

4.19.2.5.2 Utilizzo

Questa classe viene utilizzata per richiedere l'inserimento di un item. Quando questo attore riceve questo messaggio provvederà ad aggiungere l'item in questione nella propria struttura dati. L'inserimento può avvenire con sovrascrittura o senza a seconda del valore dell'attributo update passato nel messaggio.

4.19.2.6 actorbase::actorsystem::messages::storekeeper::RemoveItem

4.19.2.6.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere la rimozione di un item contenuto nell'attore. Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching₆.

4.19.2.6.2 Utilizzo

Questa classe viene utilizzata per richiedere la rimozione di un item mappato da questo attore. Quando questo attore riceve questo messaggio provvederà a rimuovere dalla propria struttura dati l'item con la chiave passata come parametro.



4.20 actorbase::actorsystem::messages::warehousemanmessages

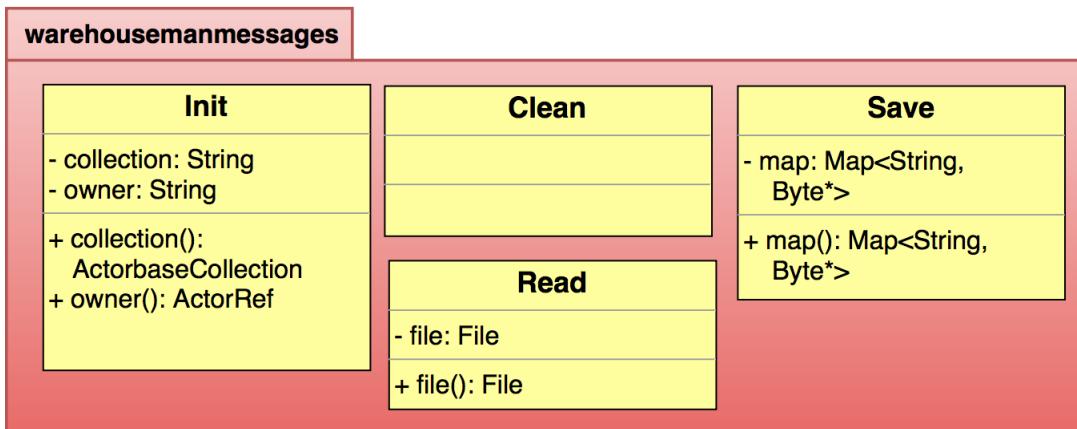


Figura 21: Package actorsystem::messages::warehousemanmessages

4.20.1 Descrizione

Package_G contenente i messaggi processabili dall'attore [Warehouseman](#).

4.20.2 Classi

4.20.2.1 actorbase::actorsystem::messages::warehouseman::Clean

4.20.2.1.1 Descrizione

Classe che rappresenta un messaggio utilizzato per richiedere allo [Warehouseman](#) di rimuovere i dati rappresentati da esso dal disco.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.20.2.1.2 Utilizzo

Questa classe viene utilizzata per richiedere di rimuovere dal disco tutti i file contenenti dati rappresentati dall'attore in questione.

Questo messaggio è utilizzato ad esempio quando si rimuove una collezione dal sistema.



4.20.2.2 actorbase::actorsystem::messages::warehouseman::Init

4.20.2.2.1 Descrizione

Classe che rappresenta un messaggio utilizzato per inizializzare l'attore.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.20.2.2.2 Utilizzo

Questa classe viene utilizzata per inizializzare l'attore con le informazioni necessarie come la collezione che rappresenta.

Questo messaggio viene ricevuto subito dopo la propria creazione in modo da inizializzarlo immediatamente.

4.20.2.3 actorbase::actorsystem::messages::warehouseman::Save

4.20.2.3.1 Descrizione

Classe che rappresenta un messaggio utilizzato per persistere i dati su disco.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.20.2.3.2 Utilizzo

Questa classe viene utilizzata per richiedere all'attore di persistere dati su disco. Quando l'attore riceve questo messaggio provvederà a salvare su filesystem i dati passati come parametri in file che seguono una gerarchia simile alla gerarchia delle collezioni del sistema.

I dati salvati vengono prima criptati tramite l'ausilio della classe [CryptoUtils](#)

4.20.2.4 actorbase::actorsystem::messages::warehouseman::Read

4.20.2.4.1 Descrizione

Classe che rappresenta un messaggio utilizzato per leggere i dati da disco.

Implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.20.2.4.2 Utilizzo

Questa classe viene utilizzata per richiedere all'attore di leggere dati da disco. Quando l'attore riceve questo messaggio provvederà a decriptare, leggere e restituire i dati all'attore richiedente.

Questo messaggio è usato ad esempio quando il sistema subisce un riavvio per ripopolare la struttura coi dati scritti su disco.



4.21 actorbase::actorsystem::messages::managermessages

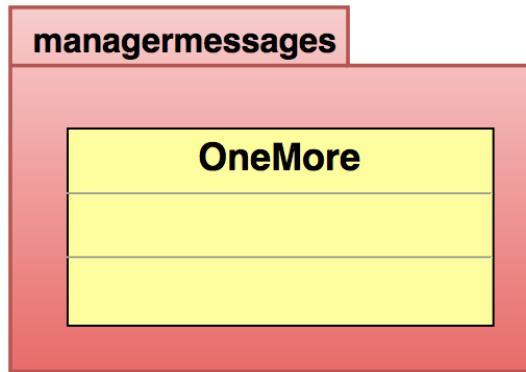


Figura 22: Package actorsystem::messages::managermessages

4.21.1 Descrizione

Package_G contenente i messaggi processabili dall'attore Manager.

4.21.2 Classi

4.21.2.1 actorbase::actorsystem::messages::managermessages::OneMore

4.21.2.1.1 Descrizione

Classe che rappresenta un messaggio utilizzato per alleggerire il carico in ingresso degli attori_G di tipo Storekeeper, creandone di nuovi e aggiungendoli alla pool_G dell'attore_G di tipo Storefinder che li coordina.

4.21.2.1.2 Utilizzo

Viene utilizzato dagli attori_G di tipo Storekeeper per notificare il superamento della soglia di coppie chiave-valore impostata da configurazione, che determina quindi un sovraccarico dell'attore_G in questione.

Il messaggio si occupa dunque di generare nuovi attori_G di tipo Storekeeper in modo da redistribuire il carico in ingresso.



4.22 actorbase::driver

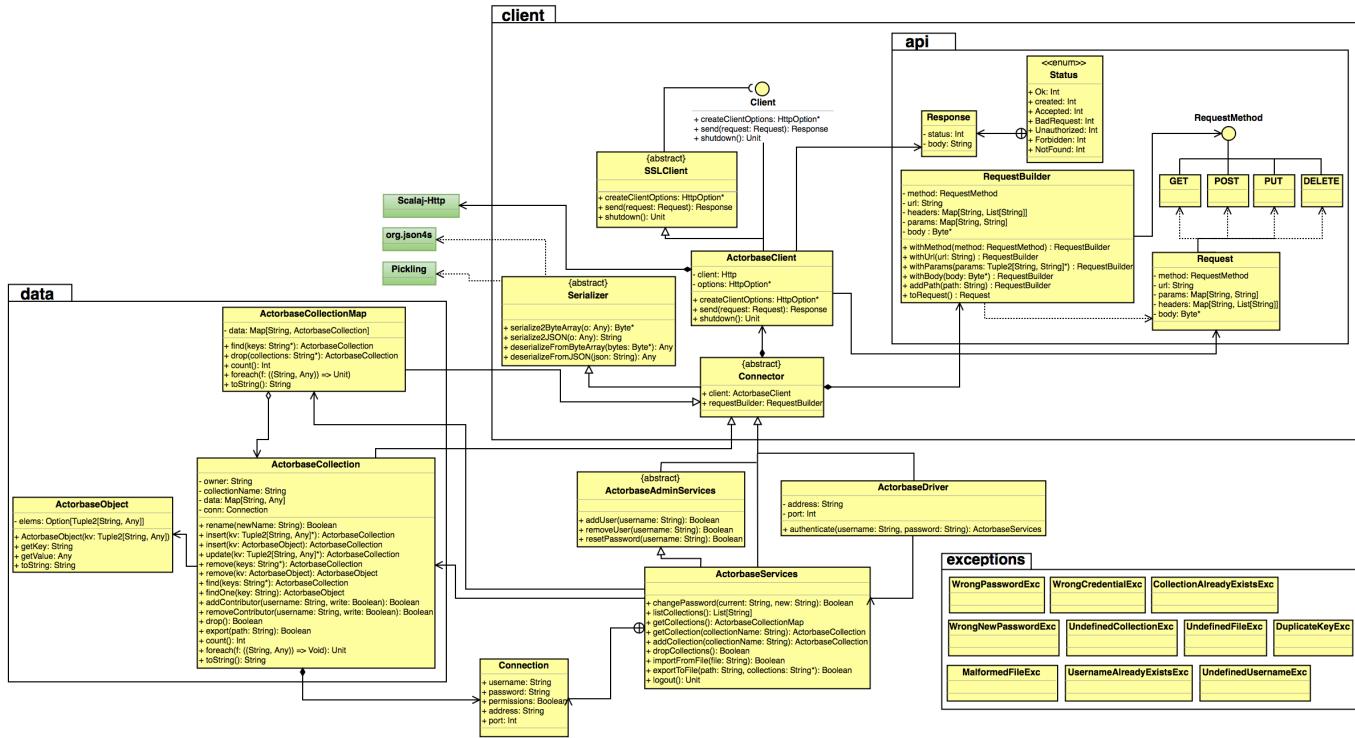


Figura 23: CLI:architettura MVC variante push model

4.22.1 Descrizione

Package_G per la componente driver_G da utilizzare in un programma Scala_G, rappresenta l'interfaccia contenente i metodi designati all'utilizzo finale in un programma Scala_G.

4.22.2 Interazioni con altre componenti

- actorbase::cli;
- actorbase::actorsystem.

4.22.3 Package contenuti

- actorbase::driver::client;



- actorbase::driver::api;
- actorbase::driver::data;
- actorbase::driver::exceptions.

4.22.4 Classi

4.22.4.1 ActorbaseDriver

4.22.4.1.1 Descrizione

Classe principale della componente driver_G , rappresenta l'interfaccia di connessione e autenticazione con il server $_G$.

4.22.4.1.2 Utilizzo

Espone i metodi di connessione e autenticazione al server $_G$ **Actorbase**, e restituisce un oggetto di tipo ActorbaseServices in base ai privilegi del proprio profilo nel sistema.

4.22.4.1.3 Eredita

- actorbase::driver::client::Connector.

4.22.4.1.4 Interazioni con le altre classi

- actorbase::driver::ActorbaseServices

4.22.4.2 ActorbaseServices

4.22.4.2.1 Descrizione

Classe principale della componente driver_G , rappresenta l'interfaccia contenente i metodi designati all'utilizzo di **Actorbase** in un programma Scala $_G$.

4.22.4.2.2 Utilizzo

Espone i metodi di interrogazione, ricerca e gestione dei dati remoti all'interno del database $_G$.

4.22.4.2.3 Eredita

- actorbase::driver::client::Connector.



4.22.4.2.4 Interazioni con le altre classi

- actorbase::driver::client::ActorbaseCollection;
- actorbase::driver::client::ActorbaseCollectionMap.

4.22.4.3 ActorbaseAdminServices

4.22.4.3.1 Descrizione

Classe principale della componente driver_G , rappresenta l'interfaccia contenente i metodi designati all'utilizzo di **Actorbase** in un programma Scala $_G$.

4.22.4.3.2 Utilizzo

Aggiunge funzionalità amministrative all'oggetto [ActorbaseServices](#), ossia operazioni di gestione degli utenti.

4.22.4.3.3 Eredità

- actorbase::driver::client::ActorbaseServices.

4.22.4.3.4 Interazioni con le altre classi

- actorbase::driver::client::ActorbaseCollection;
- actorbase::driver::client::ActorbaseCollectionMap.

4.22.4.4 Connection

4.22.4.4.1 Descrizione

Classe interna all'oggetto [ActorbaseServices](#) contenente i parametri di connessione al server $_G$ e le credenziali del profilo **Actorbase**.

4.22.4.4.2 Utilizzo

Viene utilizzata come oggetto contenente i parametri di connessione, da utilizzare nelle classi che necessitano di collegamenti al server $_G$.

4.22.4.4.3 Interazioni con le altre classi

- actorbase::driver::client::data::ActorbaseCollection.



4.23 actorbase::driver::client

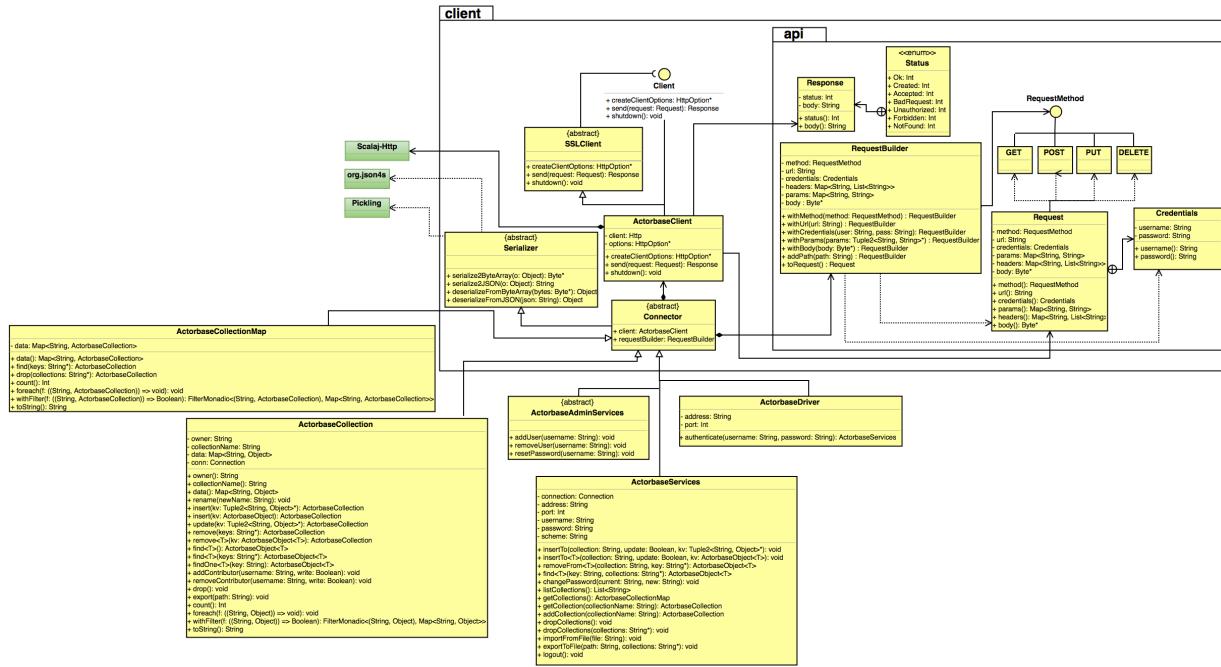


Figura 24: Driver: Packager client

4.23.1 Descrizione

Package_G che si occupa di comunicare sia con la componente cli_G che con la componente server_G del sistema.

4.23.2 Interazioni con altre componenti

- actorbase::cli::models;
- actorbase::actorsystem::actors::clientactor.

4.23.3 Classi

4.23.3.1 Client

4.23.3.1.1 Descrizione

Interfaccia che rappresenta il comportamento di un client attivo alla connessione con un server.



4.23.3.1.2 Utilizzo

Viene utilizzata per delineare il comportamento generale di un generico client, le classi che la realizzano hanno libertà di implementazione specifica dei metodi.

4.23.3.2 SSLClient (abstract)

4.23.3.2.1 Descrizione

Classe astratta che rappresenta il comportamento di un client attivo alla connessione protetta con un server.

4.23.3.2.2 Utilizzo

Viene utilizzata per delineare il comportamento generale di un generico client, funge da Decorator_G in quanto aggiunge funzionalità di protezione della connessione mediante protocolli TLS/SSL_G, le classi che la realizzano hanno libertà di implementazione specifica dei metodi.

4.23.3.2.3 Realizza

- actorbase::driver::client::Client.

4.23.3.3 actorbase::driver::client::ActorbaseClient

4.23.3.3.1 Descrizione

Classe che rappresenta il client utilizzato per la connessione con la componente server_G del sistema.

4.23.3.3.2 Utilizzo

Viene utilizzata dalla componente Connector per fornire la comunicazione con la componente server_G del sistema alle classi che lo necessitano.

4.23.3.3.3 Realizza

- actorbase::driver::client::Client.

4.23.3.3.4 Eredita

- actorbase::driver::client::SSLClient.



4.23.3.4 Serializer (abstract)

4.23.3.4.1 Descrizione

Classe astratta atta a fornire metodi di serializzazione e deserializzazione nelle varianti JSON_G e array_G di byte_G .

4.23.3.4.2 Utilizzo

Viene utilizzata mediante estensione dalle classi che necessitano di operazioni di serializzazione e deserializzazione in formato JSON_G o array_G di byte o entrambi.

4.23.3.4.3 Ereditata da

- `actorbase::driver::client::Connector`

4.23.3.5 actorbase::driver::client::Connector (abstract)

4.23.3.5.1 Descrizione

Classe astratta che si occupa di istanziare un oggetto di tipo `ActorbaseClient` e uno di tipo `RequestBuilder` in modo da fornire possibilità di comunicazione con il server alle classi che la estendono.

4.23.3.5.2 Utilizzo

Viene estesa dalle classi che necessitano di comunicare con il server.

4.23.3.5.3 Eredita

- `actorbase::driver::client::Serializer`



4.24 actorbase::driver::client::api

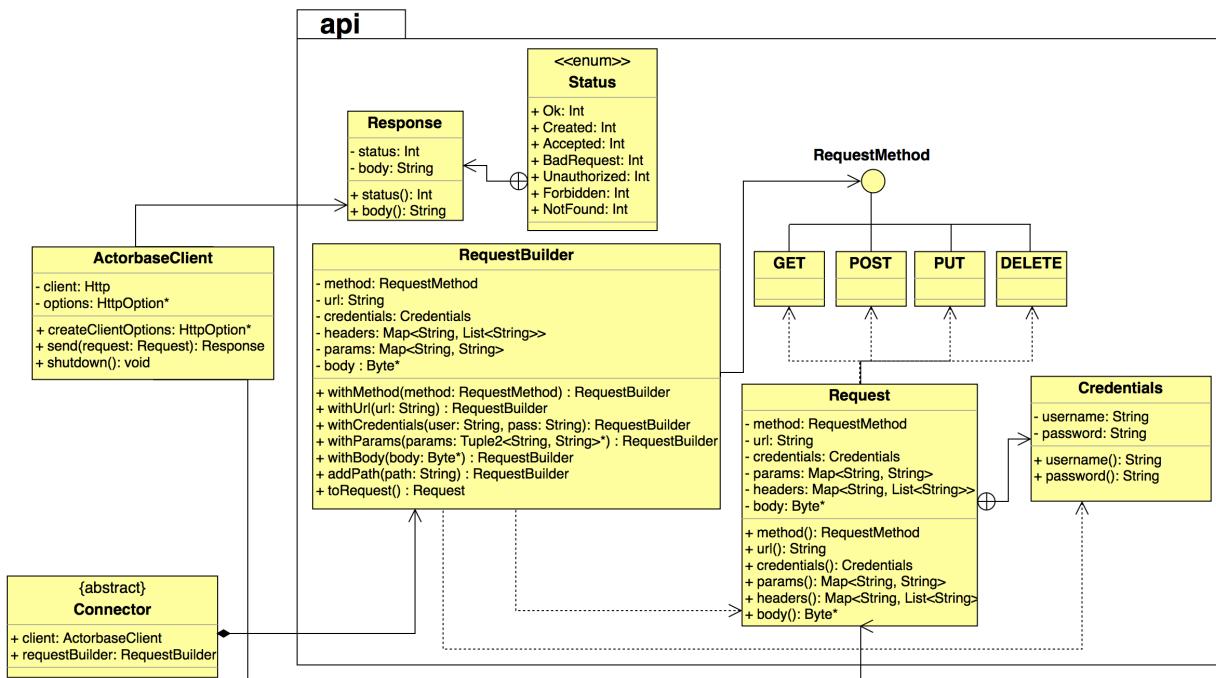


Figura 25: Driver: Package api e interazioni con package client

4.24.1 Descrizione

Package_G che offre le strutture dati necessarie per la comunicazione con il server_G, con la cli_G o eventuali programmi esterni. Si occupa inoltre della gestione delle richieste REST_G da inviare utilizzando il protocollo HTTP_G.

4.24.2 Interazioni con altre componenti

- `actorbase::driver::client;`



4.24.3 Classi

4.24.3.1 actorbase::driver::client::api::RequestMethod (abstract)

4.24.3.1.1 Descrizione

Classe astratta che rappresenta una generica richiesta HTTP_G , implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.24.3.1.2 Utilizzo

Viene utilizzata come classe padre per l'implementazione delle richieste HTTP_G da utilizzare nella comunicazione con il server.

4.24.3.1.3 Interazioni con le altre classi

- `actorbase::driver::client::api::RequestBuilder`.

4.24.3.2 actorbase::driver::client::api::GET (abstract)

4.24.3.2.1 Descrizione

Classe astratta che rappresenta una generica richiesta HTTP_G di tipo GET_G , implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.24.3.2.2 Utilizzo

Viene utilizzata come classe per l'implementazione delle richieste HTTP_G di tipo GET_G da utilizzare nella comunicazione con il server.

4.24.3.2.3 Interazioni con le altre classi

- `actorbase::driver::client::api::Request`.

4.24.3.2.4 Eredità

- `actorbase::driver::client::api::RequestMethod`



4.24.3.3 `actorbase::driver::client::api::POST (abstract)`

4.24.3.3.1 Descrizione

Classe astratta che rappresenta una generica richiesta HTTP_G di tipo POST_G , implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.24.3.3.2 Utilizzo

Viene utilizzata come classe per l'implementazione delle richieste HTTP_G di tipo POST_G da utilizzare nella comunicazione con il server.

4.24.3.3.3 Interazioni con le altre classi

- `actorbase::driver::client::api::Request`.

4.24.3.3.4 Eredità

- `actorbase::driver::client::api::RequestMethod`

4.24.3.4 `actorbase::driver::client::api::PUT (abstract)`

4.24.3.4.1 Descrizione

Classe astratta che rappresenta una generica richiesta HTTP_G di tipo PUT_G , implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.24.3.4.2 Utilizzo

Viene utilizzata come classe per l'implementazione delle richieste HTTP_G di tipo PUT_G da utilizzare nella comunicazione con il server.

4.24.3.4.3 Interazioni con le altre classi

- `actorbase::driver::client::api::Request`.

4.24.3.4.4 Eredità

- `actorbase::driver::client::api::RequestMethod`



4.24.3.5 actorbase::driver::client::api::DELETE (abstract)

4.24.3.5.1 Descrizione

Classe astratta che rappresenta una generica richiesta HTTP_G di tipo DELETE_G , implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G.

4.24.3.5.2 Utilizzo

Viene utilizzata come classe per l'implementazione delle richieste HTTP_G di tipo DELETE_G da utilizzare nella comunicazione con il server.

4.24.3.5.3 Interazioni con le altre classi

- `actorbase::driver::client::api::Request`.

4.24.3.5.4 Eredità

- `actorbase::driver::client::api::RequestMethod`

4.24.3.6 actorbase::driver::client::api::Request

4.24.3.6.1 Descrizione

Classe che rappresenta una generica richiesta HTTP_G , implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G, racchiude al suo interno i parametri di richiesta, quali il metodo, di tipo `RequestMethod`, l' URL_G a cui inviare la richiesta, gli header_G HTTP_G associati, i parametri e il payload_G.

4.24.3.6.2 Utilizzo

Viene utilizzata come classe per l'implementazione delle richieste HTTP_G di tipo DELETE_G da utilizzare nella comunicazione con il server. Viene creata mediante il design pattern `Builder`_G nella classe `RequestBuilder`.

4.24.3.6.3 Interazioni con le altre classi

- `actorbase::driver::client::api::RequestBuilder`.
- `actorbase::driver::client::ActorbaseClient`.



4.24.3.7 actorbase::driver::client::api::Response

4.24.3.7.1 Descrizione

Classe che rappresenta una generica risposta HTTP_G, implementata utilizzando la specifica funzionalità *case class* di Scala, in modo da poter usufruirne mediante costrutti e algoritmi di pattern matching_G, racchiude al suo interno i parametri di risposta, quali il codice di stato, rappresentato da una enum interna **Status** e il payload_G di risposta in formato stringa.

4.24.3.7.2 Utilizzo

Viene utilizzata come classe per l'incapsulamento delle risposte HTTP_G da utilizzare nella comunicazione con il server.

4.24.3.7.3 Interazioni con le altre classi

- `actorbase::driver::client::api::Status`;
- `actorbase::driver::client::ActorbaseClient`.

4.24.3.8 Status

4.24.3.8.1 Descrizione

Enum interna a `hyperref[sec:actorbase::driver::client::api::Response]Response`, rappresenta il codice di ritorno della richiesta HTTP_G inviata al server.

4.24.3.8.2 Utilizzo

Viene utilizzata come classe per l'incapsulamento dei codici di ritorno HTTP_G da utilizzare nella comunicazione con il server.

4.24.3.8.3 Interazioni con le altre classi

- `actorbase::driver::client::api::Response`.

4.24.3.9 RequestBuilder

4.24.3.9.1 Descrizione

Rappresenta una classe builder per la creazione di richieste HTTP_G, racchiude al suo interno i parametri di richiesta, quali il metodo, di tipo `RequestMethod`, l'URL_G a cui inviare la richiesta, gli header_G HTTP_G associati, i parametri e il payload_G e i metodi `with` per aggiungere attributi alla richiesta di tipo `Request` da generare.



4.24.3.9.2 Utilizzo

Viene utilizzata come classe buider per la costruzione delle richieste HTTP_G di tipo **Request** da utilizzare nella comunicazione con il server.

4.24.3.9.3 Interazioni con le altre classi

- actorbase::driver::client::api::Request;
- actorbase::driver::client::Connector.

4.25 actorbase::driver::client::api::data

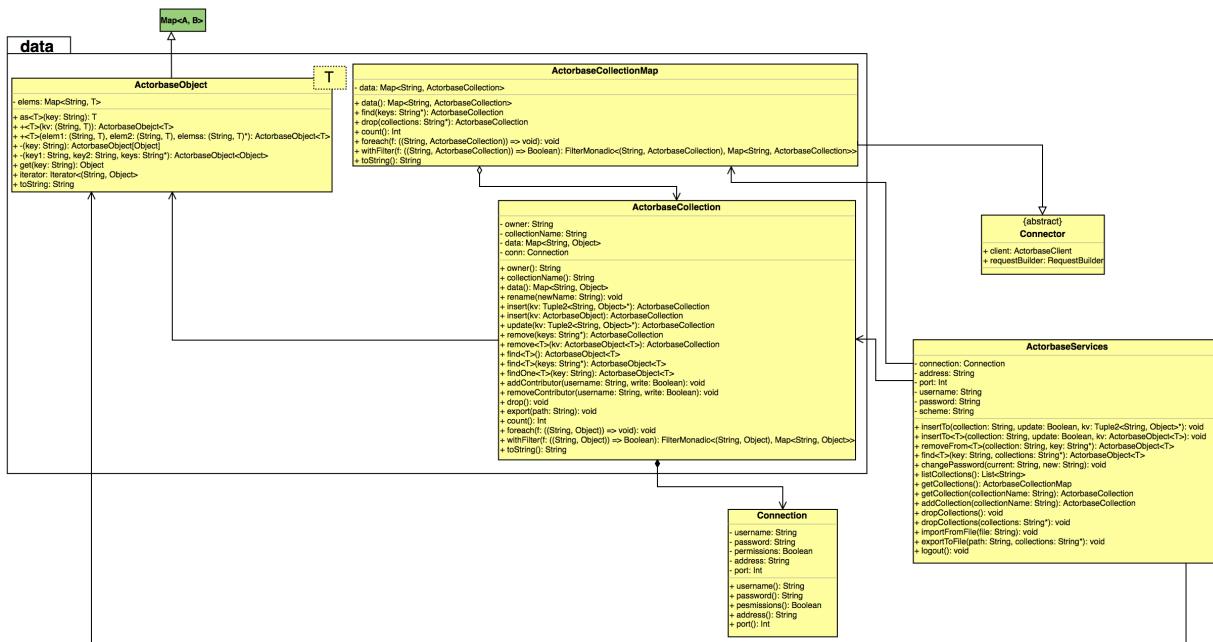


Figura 26: Driver: Package data e interazioni con package padre

4.25.1 Descrizione

Package_G che offre le strutture dati utili alla gestione dei contenuti della base di dati in locale, permettendo di eseguire le comuni operazioni di CRUD_G e ricerche più accurate in maniera semplificata, riflettendo le eventuali modifiche locali sul server remoto che contiene i dati originali.



4.25.2 Interazioni con altre componenti

- `actorbase::driver::client`.

4.25.3 Classi

4.25.3.1 ActorbaseObject

4.25.3.1.1 Descrizione

Interfaccia che rappresenta un oggetto generico contenente una coppia chiave-valore ricevuto dal server_G.

4.25.3.1.2 Utilizzo

Classe utilizzata per agevolare le ricerche specifiche all'interno delle collezioni_G del database_G.

4.25.3.1.3 Interazioni con le altre classi

- `actorbase::driver::client::data::ActorbaseCollection`.

4.25.3.2 actorbase::driver::data::ActorbaseCollection

4.25.3.2.1 Descrizione

Classe che rappresenta una singola collezione_G del database_G.

4.25.3.2.2 Utilizzo

Questa classe viene utilizzata per la rappresentazione di una singola collezione_G vista come insieme di oggetti di tipo coppia chiave-valore. Il valore può essere qualsiasi tipo, inclusi oggetti arbitrari creati all'interno di un programma Scala_G. Offre metodi di utilità e navigabilità dei contenuti.

4.25.3.2.3 Eredità

- `actorbase::driver::data::Serializer`.

4.25.3.2.4 Interazioni con le altre classi

- `actorbase::driver::client::data::ActorbaseObject`.
- `actorbase::driver::client::data::ActorbaseCollectionMap`.



4.25.3.3 **actorbase::driver::data::ActorbaseCollectionMap**

4.25.3.3.1 Descrizione

Classe che rappresenta una collezione_G di collezione_G del sistema.

4.25.3.3.2 Utilizzo

Questa classe viene utilizzata per la rappresentazione dell'aggregazione di collezioni_G, è formata da una mappa di coppie nomecollezione - [ActorbaseCollection](#).

4.25.3.3.3 Eredità

- actorbase::driver::data::Serializer.

4.25.3.3.4 Interazioni con le altre classi

- actorbase::driver::client::ActorbaseDriver.
- actorbase::driver::client::data::ActorbaseCollection.

4.26 **actorbase::driver::exceptions**

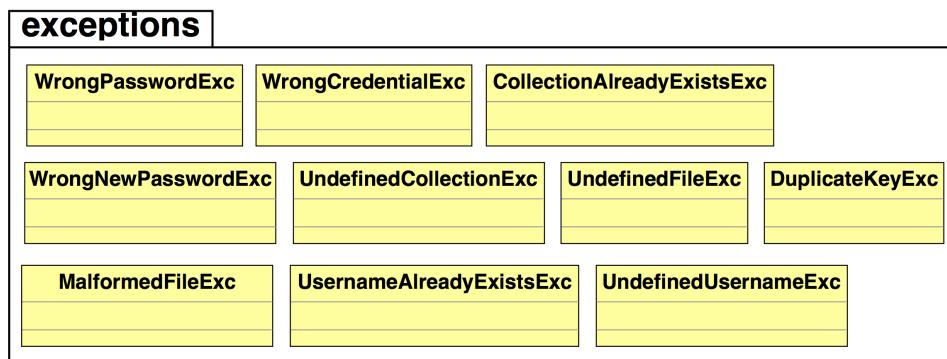


Figura 27: Driver: Package exceptions

4.26.1 Descrizione

Package_G che contiene le possibili eccezioni che si possono verificare nel driver_G



4.26.2 Interazione con altre componenti

- `actorbase::driver::client;`

4.26.3 Classi

4.26.3.1 `actorbase::driver::exceptions::WrongCredentialExc`

4.26.3.1.1 Descrizione

Classe che rappresenta l'eccezione del tentativo di accesso con username o password sbagliati.

4.26.3.1.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire errori nelle credenziali di accesso.

4.26.3.1.3 Interazioni con altre classi

- `actorbase::driver::client::ActorbaseClient.`

4.26.3.2 `actorbase::driver::exceptions::WrongPasswordExc`

4.26.3.2.1 Descrizione

Classe che rappresenta l'eccezione dell'inserimento sbagliato della vecchia password.

4.26.3.2.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui alla modifica della propria password, quando dovrei inserire la vecchia password questa non corrisponda.

4.26.3.2.3 Interazioni con altre classi

- `actorbase::driver::client::ActorbaseClient.`

4.26.3.3 `actorbase::driver::exceptions::WrongNewPasswordExc`

4.26.3.3.1 Descrizione

Classe che rappresenta l'eccezione dell'inserimento sbagliato della nuova password.



4.26.3.3.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui alla modifica della propria password, quando dovrei inserire la nuova password questa non soddisfi i requisiti richiesti.

4.26.3.3 Interazioni con altre classi

- actorbase::driver::client::ActorbaseClient.

4.26.3.4 actorbase::driver::exceptions::CollectionAlreadyExistsExc

4.26.3.4.1 Descrizione

Classe che rappresenta l'eccezione del conflitto di nomi tra una collezione esistente e una nuova.

4.26.3.4.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui alla creazione di una nuova collezione, il nome della nuova collezione corrisponda al nome di una collezione già esistente.

4.26.3.4.3 Interazioni con altre classi

- actorbase::driver::client::ActorbaseClient.

4.26.3.5 actorbase::driver::exceptions::UndefinedCollectionExc

4.26.3.5.1 Descrizione

Classe che rappresenta l'eccezione di mancato inserimento del nome di una nuova collezione.

4.26.3.5.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui alla creazione di una nuova collezione, venga omesso il nome di quest'ultima.

4.26.3.5.3 Interazioni con altre classi

- actorbase::driver::client::ActorbaseClient.



4.26.3.6 **actorbase::driver::exceptions::UndefinedUsernameExc**

4.26.3.6.1 Descrizione

Classe che rappresenta l'eccezione di mancato inserimento dell'username utente.

4.26.3.6.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui alla registrazione di un nuovo utente, venga omesso l'username.

4.26.3.6.3 Interazioni con altre classi

- [actorbase::driver::client::ActorbaseClient](#).

4.26.3.7 **actorbase::driver::exceptions::UsernameAlreadyExistsExc**

4.26.3.7.1 Descrizione

Classe che rappresenta l'eccezione del conflitto di username tra un utente già esistente e uno nuovo.

4.26.3.7.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui alla registrazione di un nuovo utente, l'username di quest'ultimo corrisponda all'username di un utente già esistente.

4.26.3.7.3 Interazioni con altre classi

- [actorbase::driver::client::ActorbaseClient](#).

4.26.3.8 **actorbase::driver::exceptions::DuplicateKeyExc**

4.26.3.8.1 Descrizione

Classe che rappresenta l'eccezione del conflitto tra una chiave già esistente e una nuova chiave.

4.26.3.8.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui all'inserimento di una nuova chiave, questa corrisponda ad una chiave già esistente nella stessa collezione.



4.26.3.8.3 Interazioni con altre classi

- `actorbase::driver::client::ActorbaseClient`.

4.26.3.9 `actorbase::driver::exceptions::UndefinedFileExc`

4.26.3.9.1 Descrizione

Classe che rappresenta l'eccezione di inserimento di un path sbagliato di un file o del mancato inserimento del path.

4.26.3.9.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui all'inserimento di un nuovo item o di una serie di item da file, si sbagli il path del file o si ometta di inserirlo.

4.26.3.9.3 Interazioni con altre classi

- `actorbase::driver::client::ActorbaseClient`.

4.26.3.10 `actorbase::driver::exceptions::MalformedFileExc`

4.26.3.10.1 Descrizione

Classe che rappresenta l'eccezione di inserimento di un path di un file non conforme.

4.26.3.10.2 Utilizzo

Questa classe viene utilizzata dal driver_G per gestire il caso in cui all'inserimento di un nuovo item o di una serie di item da file, si inserisca il path di un file in un formato sbagliato o il cui testo non rispetta le regole di formattazione.

4.26.3.10.3 Interazioni con altre classi

- `actorbase::driver::client::ActorbaseClient`.



4.27 actorbase::cli

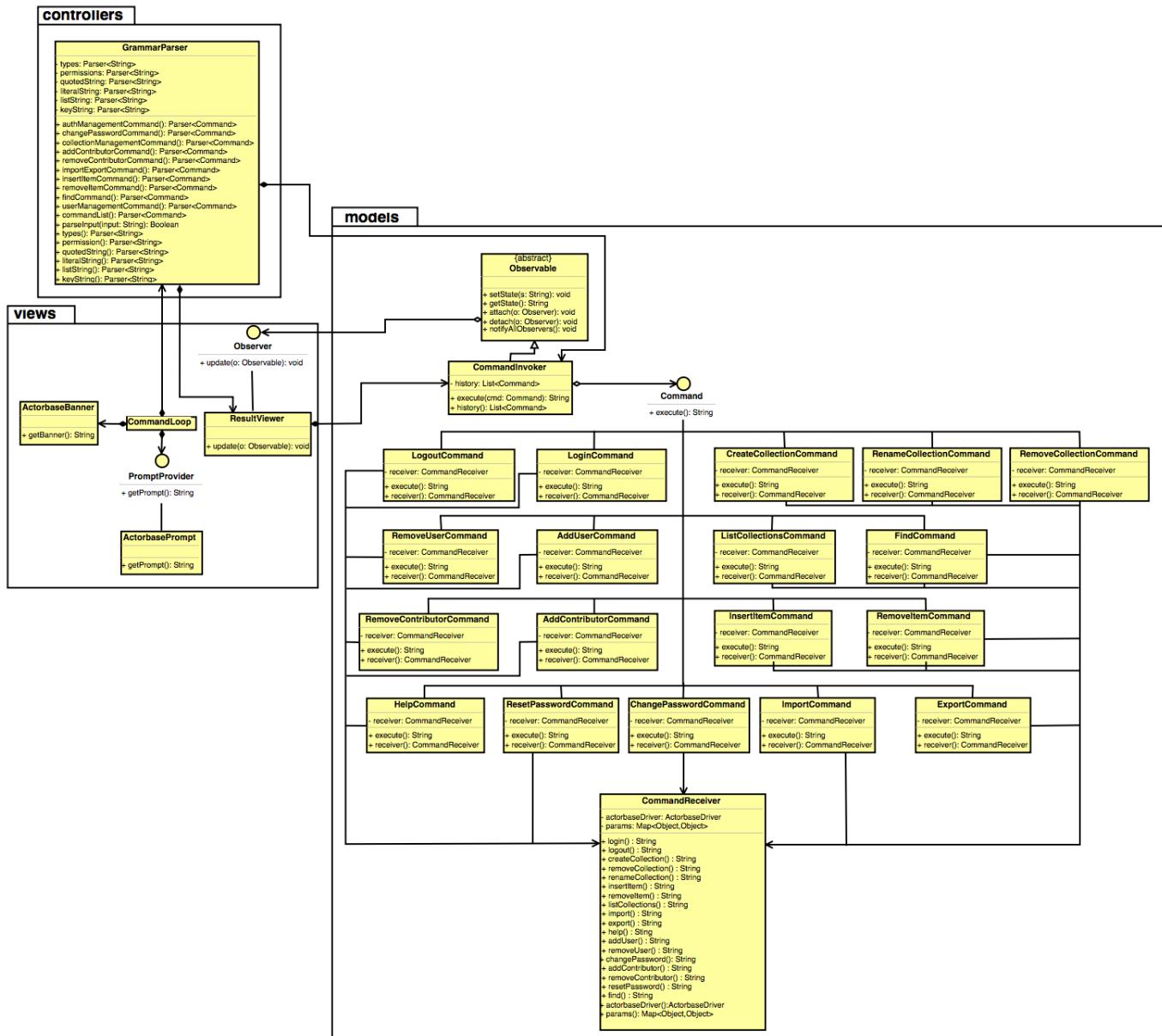


Figura 28: CLI, architettura MVC variante push model



4.27.1 Descrizione

Package_G per la parte di client_G rappresentata dalla cli_G. Questa componente viene rappresentata usando il design pattern_G MVC_G in versione push model_G.

4.27.2 Interazioni con altre componenti

- [actorbase::driver](#).

4.27.3 Package contenuti

- [actorbase::cli::views](#);
- [actorbase::cli::controllers](#);
- [actorbase::cli::models](#).



ScalateKids

4 COMPONENTI



4.28 actorbase::cli::views

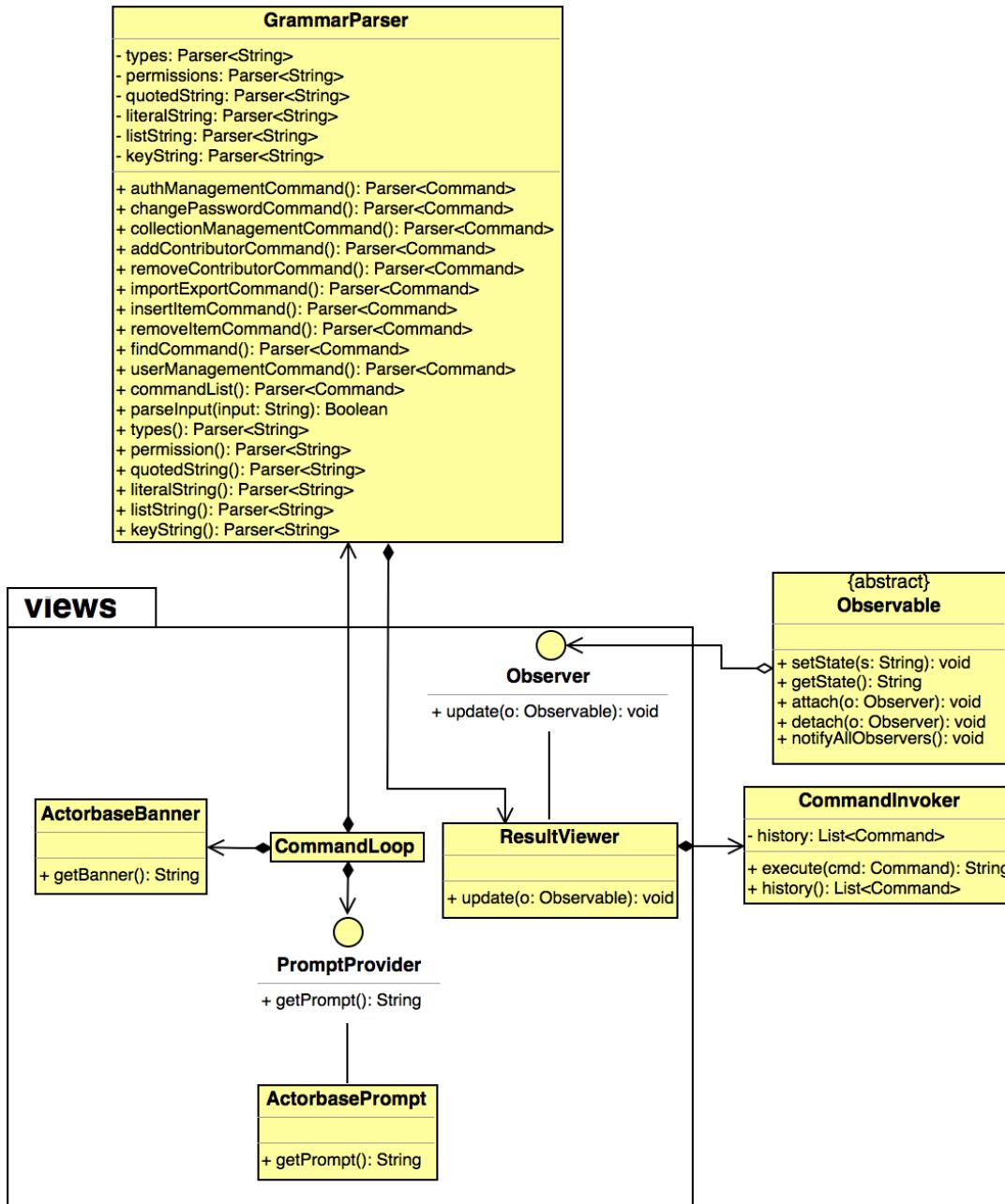


Figura 29: CLI, views package, interazioni con controllers e models



4.28.1 Descrizione

Package_G per la parte view_G della cli_G. Questa componente gestisce l'output e offre l'interfaccia tramite cui l'utente può inserire i comandi.

4.28.2 Interazioni con altre componenti

- actorbase::cli::controllers;
- actorbase::cli::models.

4.28.3 Classi

4.28.3.1 actorbase::cli::views::CommandLoop

4.28.3.1.1 Descrizione

Questa classe verrà utilizzata per leggere i comandi inseriti dall'utente.

4.28.3.1.2 Utilizzo

Viene utilizzata per la lettura dei comandi inseriti dall'utente.

4.28.3.1.3 Interazioni con altre classi

- actorbase::cli::views::PromptProvider;
- actorbase::cli::views::ActorbaseBanner;
- actorbase::cli::views::ResultView;
- actorbase::cli::controllers::GrammarParser.

4.28.3.2 actorbase::cli::views::ActorbaseBanner

4.28.3.2.1 Descrizione

Classe per la creazione del banner_G da mostrare all'avvio della CLI_G.

4.28.3.2.2 Utilizzo

Viene utilizzata per la creazione di un banner_G di presentazione all'avvio della CLI_G.



4.28.3.2.3 Interazioni con altre classi

- `actorbase::cli::views::CommandLoop`.

4.28.3.3 `actorbase::cli::views::PromptProvider`

4.28.3.3.1 Descrizione

Interfaccia per la creazione del prompt_G dei comandi.

4.28.3.3.2 Utilizzo

Offre un'interfaccia per la creazione di un prompt_G generico.

4.28.3.3.3 Interazioni con altre classi

- `actorbase::cli::views::CommandLoop`.

4.28.3.4 `actorbase::cli::views::ActorbasePrompt`

4.28.3.4.1 Descrizione

Classe che implementa l'interfaccia `actorbase::cli::views::PromptProvider` per la creazione di un prompt_G dei comandi.

4.28.3.4.2 Utilizzo

Viene utilizzata per la creazione di un prompt_G per l'inserimento dei comandi e la visualizzazione delle informazioni relative alla connessione effettuata.

4.28.3.4.3 Classi ereditate

- `actorbase::cli::views::PromptProvider`.

4.28.3.5 `actorbase::cli::views::Observer`

4.28.3.5.1 Descrizione

Interfaccia per l'implementazione del design pattern_G Observer_G che permette di aggiornare la view_G ad ogni risposta della componente driver_G.



4.28.3.5.2 Utilizzo

Offre un'interfaccia per permettere di aggiornare la view_G ad ogni risposta ricevuta dal driver_G .

4.28.3.5.3 Interazioni con altre classi

- `actorbase::cli::models::Observable`.

4.28.3.6 `actorbase::cli::views::ResultView`

4.28.3.6.1 Descrizione

Classe per la gestione e la formattazione in output dei risultati ottenuti dall'esecuzione del comando ricevuto in input tramite il design pattern_G Observer_G.

4.28.3.6.2 Utilizzo

Viene utilizzata per la gestione e la formattazione in output dei risultati ottenuti dal server mediante l'inserimento dei comandi tramite il design pattern_G Observer_G.

4.28.3.6.3 Classi ereditate

- `actorbase::cli::views::Observer`.

4.28.3.6.4 Interazioni con altre classi

- `actorbase::cli::views::CommandLoop`.



4.29 actorbase::cli::controllers

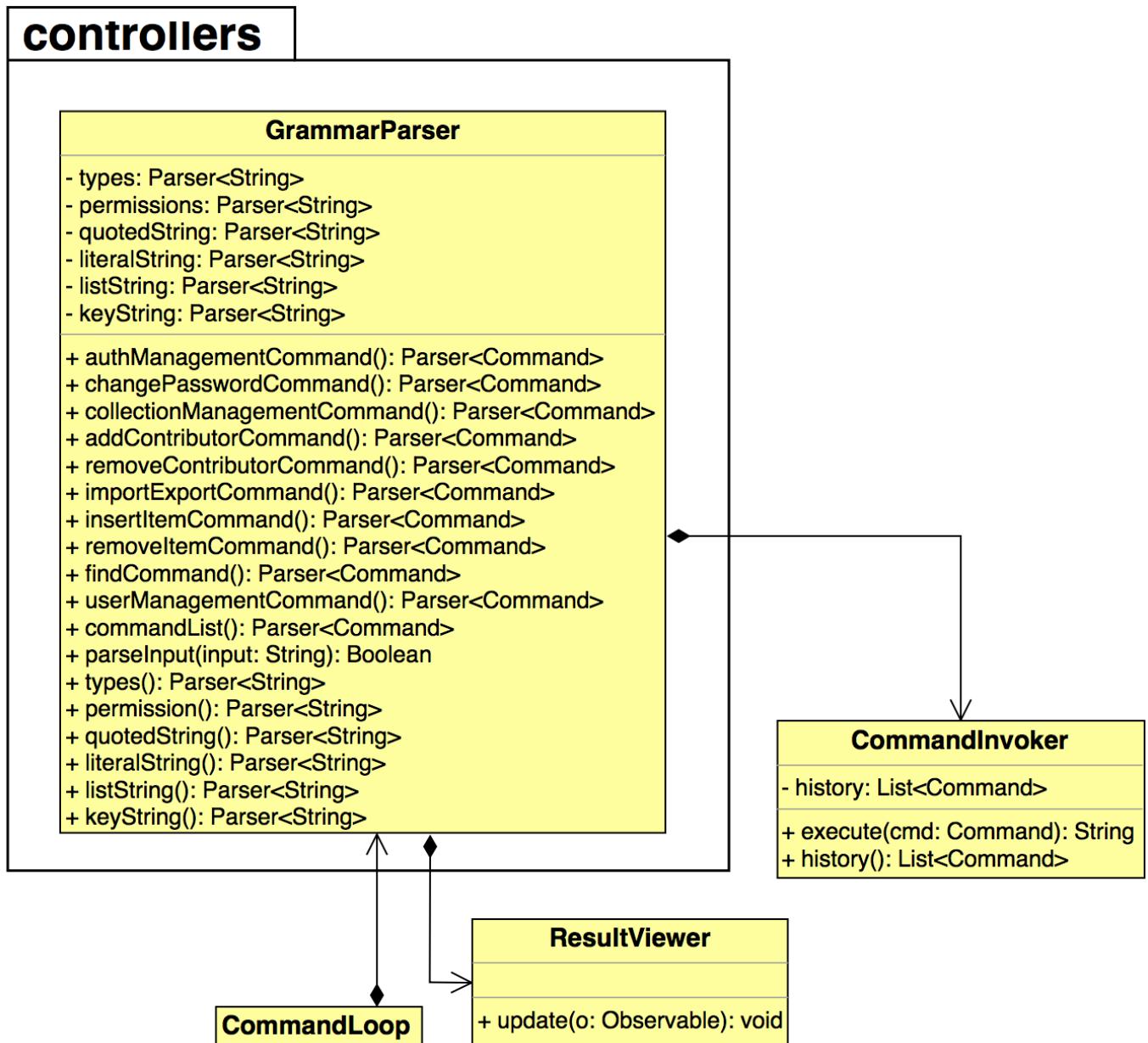


Figura 30: CLI: package controllers e interazioni con package models e views



4.29.1 Descrizione

Package_G per la parte controller_G della cli_G. Questa componente si occupa di passare il comando ricevuto dalla view_G al model_G.

4.29.2 Interazioni con altre componenti

- actorbase::cli::views;
- actorbase::cli::models.

4.29.3 Classi

4.29.3.1 actorbase::cli::controllers::GrammarParser

4.29.3.1.1 Descrizione

Classe che si occupa di effettuare il parsing_G del comando ricevuto dalla componente view_G e di chiamare la componente model_G con i parametri opportuni.

4.29.3.1.2 Utilizzo

Viene utilizzata per effettuare il parsing_G del comando ricevuto da **actorbase::cli::views::CommandLoop** e, in seguito, chiamare **actorbase::cli::models::CommandInvoker** con i parametri corretti per l'esecuzione del comando.

4.29.3.1.3 Interazioni con altre classi

- actorbase::cli::views::CommandLoop;
- actorbase::cli::models::CommandInvoker.



4.30 actorbase::cli::models

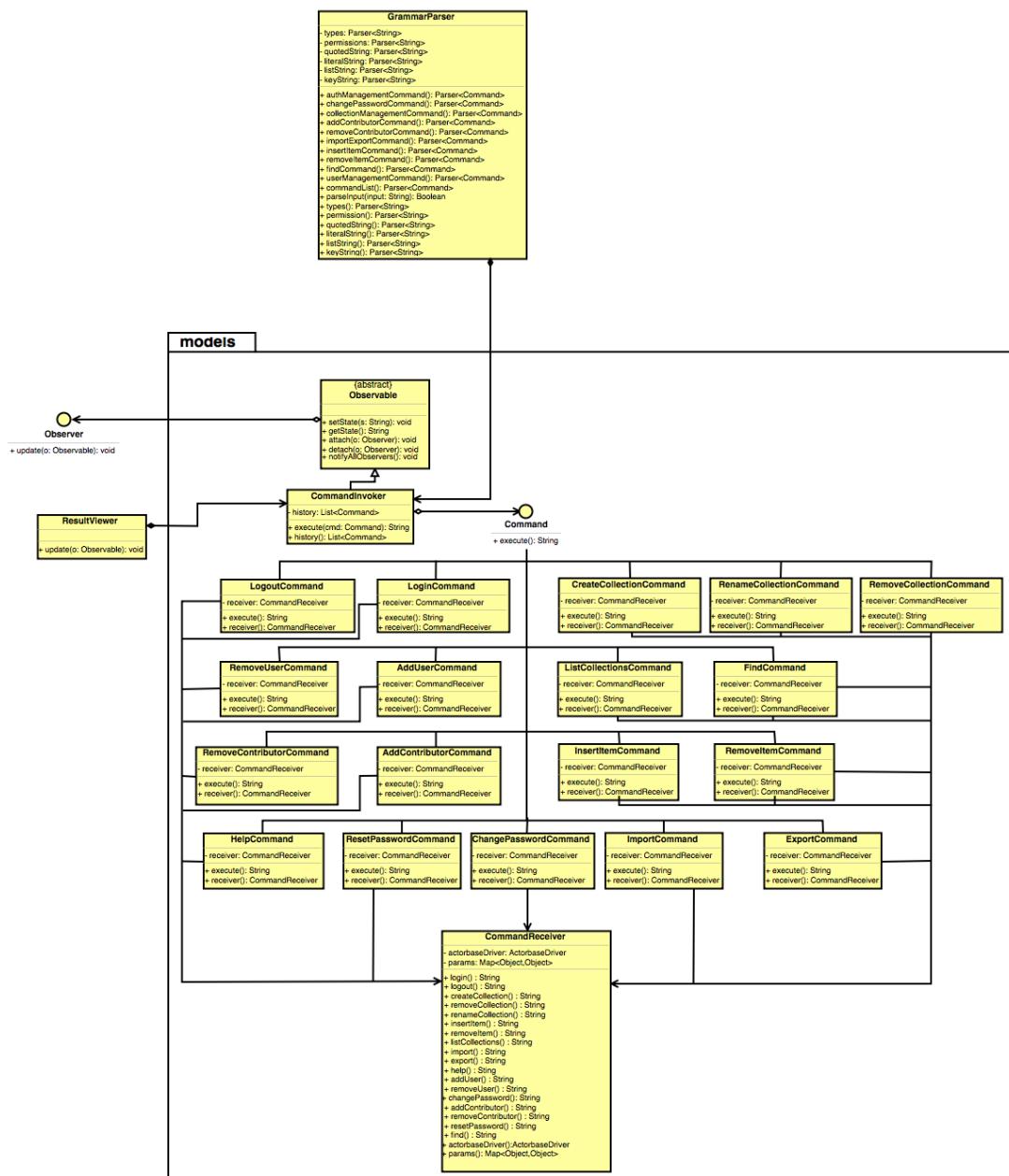


Figura 31: CLI: package models e interazioni con controllers e views



4.30.1 Descrizione

Package_G per la parte model_G della cli_G. Questa componente si occupa della comunicazione con la componente driver_G e con la componente view_G tramite il push model_G del design pattern_G observer_G. Per la gestione dei comandi si è scelto di utilizzare il command pattern_G.

4.30.2 Classi

4.30.2.1 actorbase::cli::models::Observable

4.30.2.1.1 Descrizione

Questa classe rappresenta l'oggetto osservato nell'ambito del design pattern_G observer_G.

4.30.2.1.2 Utilizzo

Viene utilizzata per notificare la classe actorbase::cli::views::Observer quando avviene una modifica al proprio stato, ossia quando il driver_G ha ritornato un risultato.

4.30.2.1.3 Interazioni con altre classi

- actorbase::cli::views::Observer.

4.30.2.2 actorbase::cli::models::CommandInvoker

4.30.2.2.1 Descrizione

Questa classe rappresenta l'Invoker_G del command pattern_G. Essa si occupa di eseguire il comando ricevuto dalla componente controller_G.

4.30.2.2.2 Utilizzo

Viene utilizzata per eseguire il comando ricevuto da actorbase::cli::controllers::GrammarParser chiamando il metodo opportuno che implementa l'interfaccia actorbase::cli::models::Command.

4.30.2.2.3 Classi ereditate

- actorbase::cli::models::Observable.



4.30.2.2.4 Interazioni con altre classi

- [actorbase::cli::controllers::GrammarParser](#);
- [actorbase::cli::models::Command](#).

4.30.2.3 actorbase::cli::models::Command

4.30.2.3.1 Descrizione

Interfaccia generica che rappresenta un comando.

4.30.2.3.2 Utilizzo

Viene utilizzata per offrire un'interfaccia comune a tutti i comandi.

Ha un riferimento a [actorbase::cli::models::CommandReceiver](#) per l'esecuzione del comando.

4.30.2.3.3 Interazioni con altre classi

- [actorbase::cli::models::CommandInvoker](#);
- [actorbase::view::cli::models::CommandReceiver](#).

4.30.2.4 actorbase::cli::models::FindCommand

4.30.2.4.1 Descrizione

Classe per il comando di ricerca.

4.30.2.4.2 Utilizzo

Viene utilizzata per chiamare il metodo di [actorbase::cli::models::CommandReceiver](#) per la ricerca con i parametri immessi dall'utente.

4.30.2.4.3 Classi ereditate

- [actorbase::cli::models::Command](#).

4.30.2.5 actorbase::cli::models::LogoutCommand

4.30.2.5.1 Descrizione

Classe per il comando di logout.



4.30.2.5.2 Utilizzo

Viene utilizzata per chiamare il metodo di [actorbase::cli::models::CommandReceiver](#) per il logout.

4.30.2.5.3 Classi ereditate

- [actorbase::cli::models::Command](#).

4.30.2.6 actorbase::cli::models::ResetPasswordCommand

4.30.2.6.1 Descrizione

Classe per il comando di reset della password.

4.30.2.6.2 Utilizzo

Viene utilizzata per chiamare il metodo di [actorbase::cli::models::CommandReceiver](#) per il reset della password dell'utente specificato in input.

4.30.2.6.3 Classi ereditate

- [actorbase::cli::models::Command](#).

4.30.2.7 actorbase::cli::models::ChangePasswordCommand

4.30.2.7.1 Descrizione

Classe per il comando di modifica della password.

4.30.2.7.2 Utilizzo

Viene utilizzata per chiamare il metodo di [actorbase::cli::models::CommandReceiver](#) per la modifica della password..

4.30.2.7.3 Classi ereditate

- [actorbase::cli::models::Command](#).



4.30.2.8 **actorbase::cli::models::ListCommand**

4.30.2.8.1 Descrizione

Classe per il comando di visualizzazione dell'elenco dei nomi di tutte le collezioni presenti nel database_G.

4.30.2.8.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per la visualizzazione dell'elenco dei nomi di tutte le collezioni presenti nel database_G.

4.30.2.8.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.9 **actorbase::cli::models::AddContributorCommand**

4.30.2.9.1 Descrizione

Classe per il comando di aggiunta collaboratore a una collezione.

4.30.2.9.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per l'aggiunta di un collaboratore a una collezione con il nome della collezione_G e lo username specificati in input.

4.30.2.9.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.10 **actorbase::cli::models::LoginCommand**

4.30.2.10.1 Descrizione

Classe per il comando di login.

4.30.2.10.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per il login con le credenziali immesse dall'utente.



4.30.2.10.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.11 `actorbase::cli::models::RemoveCollectionCommand`

4.30.2.11.1 Descrizione

Classe per il comando di rimozione una collezione_G.

4.30.2.11.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per la rimozione di una collezione con i parametri specificati in input.

4.30.2.11.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.12 `actorbase::cli::models::RemoveContributorCommand`

4.30.2.12.1 Descrizione

Classe per il comando di rimozione di un collaboratore da una collezione.

4.30.2.12.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per la rimozione di un collaboratore da una collezione con il nome della collezione_G e lo username specificati in input.

4.30.2.12.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.13 `actorbase::cli::models::RenameCollectionCommand`

4.30.2.13.1 Descrizione

Classe per il comando rinominazione di una collezione_G.



4.30.2.13.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per la rinominazione di una collezione_G con i parametri specificati in input.

4.30.2.13.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.14 actorbase::cli::models::AddUserCommand

4.30.2.14.1 Descrizione

Classe per il comando di aggiunta di un utente.

4.30.2.14.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per l'aggiunta di un utente con i parametri specificati in input.

4.30.2.14.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.15 actorbase::cli::models::HelpCommand

4.30.2.15.1 Descrizione

Classe per la richiesta di aiuto.

4.30.2.15.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per la richiesta di aiuto con i parametri immessi dall'utente.

4.30.2.16 actorbase::cli::models::RemoveItemCommand

4.30.2.16.1 Descrizione

Classe per il comando di rimozione di un item_G.



4.30.2.16.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per la rimozione di un item_G con i parametri immessi dall'utente.

4.30.2.16.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.17 `actorbase::cli::models::RemoveUserCommand`

4.30.2.17.1 Descrizione

Classe per il comando di rimozione di un utente.

4.30.2.17.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per la rimozione di un utente con i parametri immessi dall'utente.

4.30.2.17.3 Classi ereditate

- `actorbase::cli::models::Command`.

4.30.2.18 `actorbase::cli::models::ImportCommand`

4.30.2.18.1 Descrizione

Classe per il comando di importazione da un file JSON_G.

4.30.2.18.2 Utilizzo

Viene utilizzata per chiamare il metodo di `actorbase::cli::models::CommandReceiver` per l'importazione da un file JSON_G con i parametri immessi dall'utente.

4.30.2.18.3 Classi ereditate

- `actorbase::cli::models::Command`.



4.30.2.19 actorbase::cli::models::InsertItemCommand

4.30.2.19.1 Descrizione

Classe per il comando di inserimento di un item_G in una collezione_G.

4.30.2.19.2 Utilizzo

Viene utilizzata per chiamare il metodo di actorbase::cli::models::CommandReceiver per l' inserimento di un item_G in una collezione_G con i parametri immessi dall'utente.

4.30.2.19.3 Classi ereditate

- actorbase::cli::models::Command.

4.30.2.20 actorbase::cli::models::CreateCollectionCommand

4.30.2.20.1 Descrizione

Classe per il comando di creazione di una nuova collezione_G.

4.30.2.20.2 Utilizzo

Viene utilizzata per chiamare il metodo di actorbase::cli::models::CommandReceiver per la creazione di una nuova collezione_G con i parametri immessi dall'utente.

4.30.2.20.3 Classi ereditate

- actorbase::cli::models::Command.

4.30.2.21 actorbase::cli::models::ExportCommand

4.30.2.21.1 Descrizione

Classe per il comando di esportazione in file JSON_G.

4.30.2.21.2 Utilizzo

Viene utilizzata per chiamare il metodo di actorbase::cli::models::CommandReceiver per l'esportazione in file JSON_G con i parametri immessi dall'utente.



4.30.2.21.3 Classi ereditate

- [actorbase::cli::models::Command](#).

4.30.2.22 actorbase::cli::models::CommandReceiver

4.30.2.22.1 Descrizione

Classe contenente i metodi per richiamare la componente driver_G per l'esecuzione dei comandi o per restituire l'aiuto se richiesto.

4.30.2.22.2 Utilizzo

Viene utilizzata da [actorbase::cli::models::Command](#) per chiamare [actorbase::driver::client::ActorbaseClient](#) o per ricevere un aiuto da visualizzare se richiesto dall'utente.

4.30.2.22.3 Interazioni con altre classi

- [actorbase::cli::models::Command](#);
- [actorbase::driver::client::ActorbaseClient](#).



5 Diagrammi di Attività

In questa sezione, vengono illustrati i diagrammi di attività che descrivono l’interazione dell’utente con **Actor-base** tramite l’utilizzo della CLI_G ad esso associata. È stato disegnato un diagramma ad alto livello in cui sono rappresentate le operazioni possibili sul $database_G$. Esse vengono poi illustrate tramite dei sotto-diagrammi per poterne comprendere meglio il funzionamento e facilitarne la lettura.

5.1 Visione generale

L’utente, dopo aver aperto la CLI_G , ha la possibilità di autenticarsi al $database_G$ oppure chiedere dell’aiuto. Una volta autenticato, esso può operare su collezioni e/o item, interrogare il $database_G$, modificare la propria password, effettuare il logout oppure, se si tratta di un amministratore, gestire gli utenti del $database_G$.

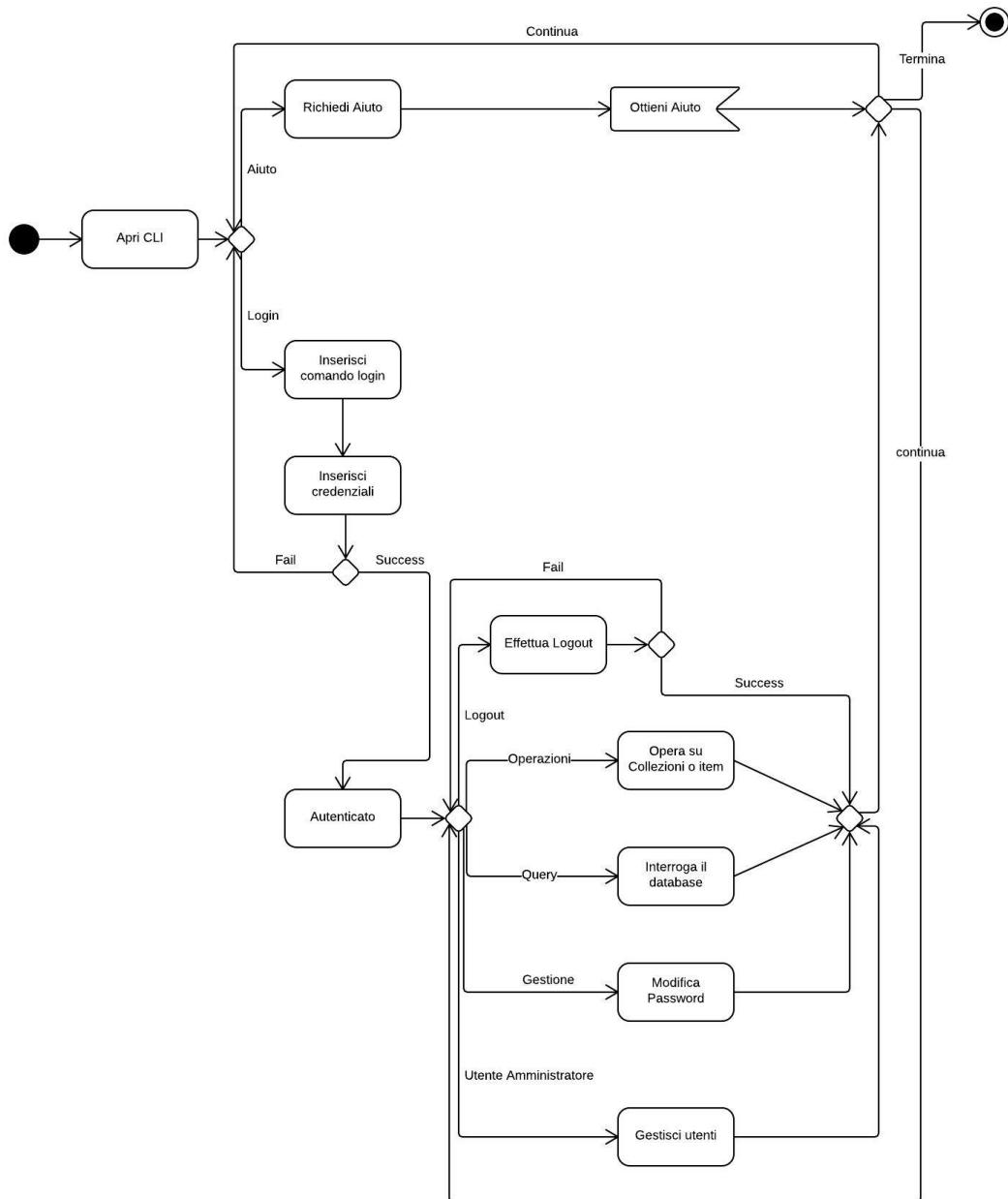


Figura 32: Diagrammi attività - Visione generale



5.2 Operazioni su collezioni e/o item

5.2.1 Creazione collezione

Questo tipo di operazione permette di inserire una collezione all'interno del database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per la creazione della collezione, il nome della collezione stessa e i suoi parametri. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

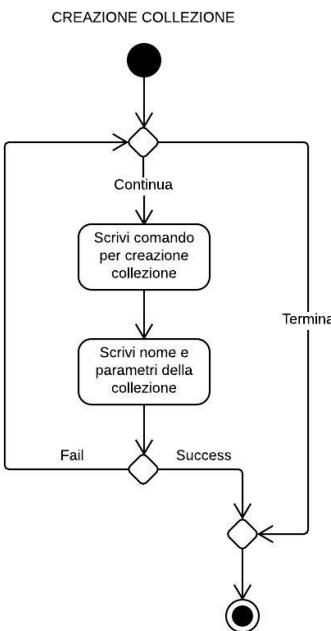


Figura 33: Diagrammi attività - Creazione collezione

5.2.2 Cancellazione collezione

Questo tipo di operazione permette di cancellare una collezione dal database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per la cancellazione di una collezione e il nome della collezione stessa. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

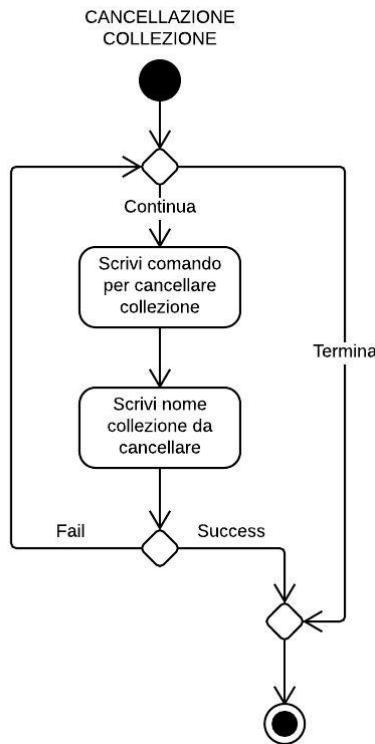


Figura 34: Diagrammi attività - Cancellazione collezione

5.2.3 Visualizza collezioni

Questo tipo di operazione permette di visualizzare le collezioni presenti all'interno del database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per la visualizzazione delle collezioni. Una volta premuto il tasto invio, l'operazione andrà a buon termine (ricevendo la lista delle collezioni) se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

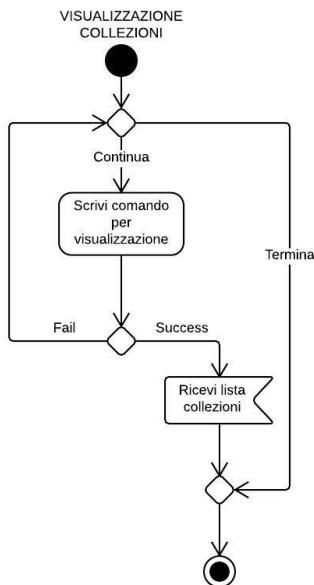


Figura 35: Diagrammi attività - Visualizzazione collezioni

5.2.4 Modifica nome collezione

Questo tipo di operazione permette di modificare il nome di una collezione presente nel database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per la rinominazione della collezione, il nome della collezione stessa e il nuovo nome per essa. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

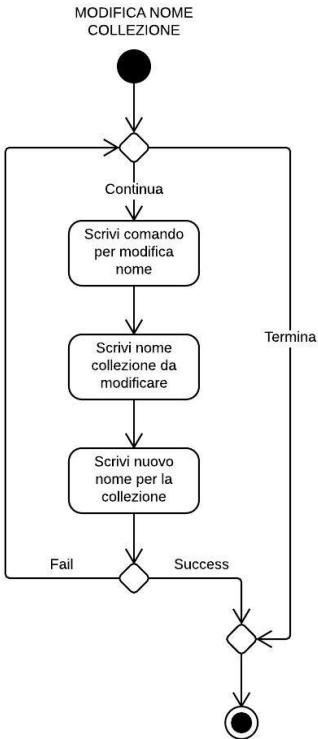


Figura 36: Diagrammi attività - Modifica nome collezione

5.2.5 Inserimento item

Questo tipo di operazione permette di inserire un item all'interno di una collezione del database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per l'inserimento item, il valore e parametri dell'item stesso e il nome della collezione dove inserire l'item. Una volta premuto il tasto invio, l'operazione andrà a buon termine (ricevendo la lista delle collezioni) se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

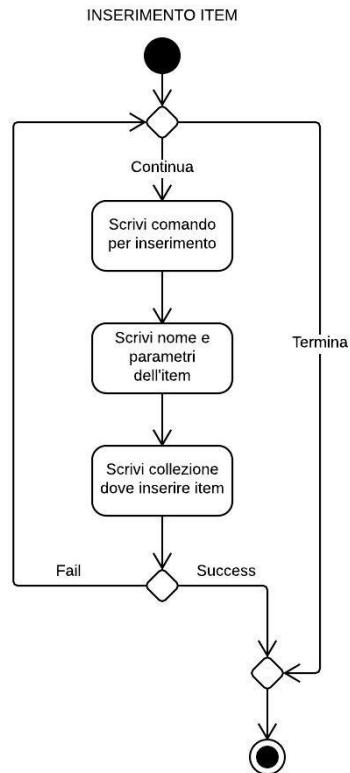


Figura 37: Diagrammi attività - Inserimento item

5.2.6 Rimozione item

Questo tipo di operazione permette di rimuovere un item dall'interno di una collezione_G del database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per l'eliminazione di un item, il nome della collezione da dove rimuovere l'item e il nome dell'item stesso. Una volta premuto il tasto invio, l'operazione andrà a buon termine (ricevendo la lista delle collezioni) se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

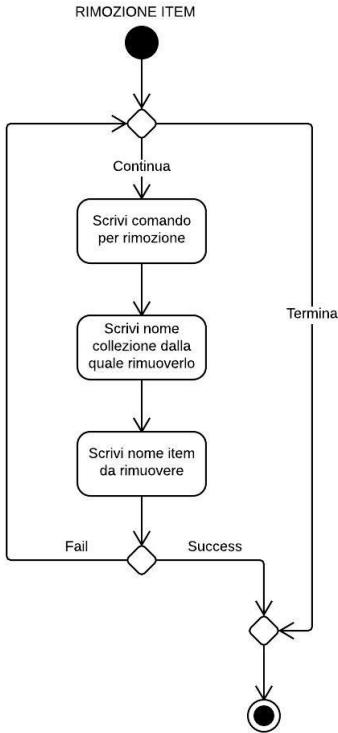


Figura 38: Diagrammi attività - Rimozione item

5.2.7 Aggiunta collaboratore

Questo tipo di operazione permette di aggiungere un collaboratore_G ad una collezione presente nel database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per l'aggiunta di un collaboratore ad una collezione, lo username del collaboratore e il nome della collezione alla quale aggiungerlo. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

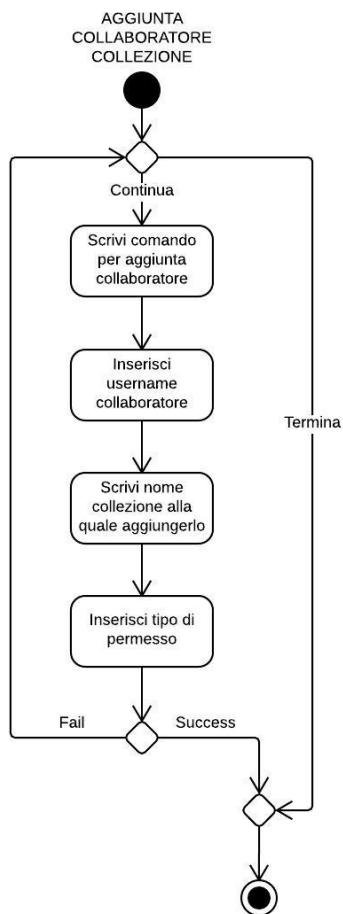


Figura 39: Diagrammi attività - Aggiunta collaboratore

5.2.8 Rimozione collaboratore

Questo tipo di operazione permette di rimuovere un collaboratore da una collezione presente nel database_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per la rimozione di un collaboratore da una collezione, lo username del collaboratore e il nome della collezione dalla quale rimuoverlo. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

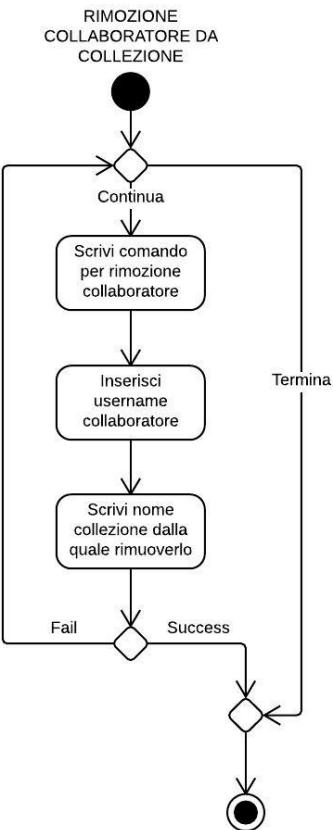


Figura 40: Diagrammi attività - Rimozione collaboratore

5.2.9 Import

Questo tipo di operazione permette di importare nel database_G collezioni o item tramite file JSON_G. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per l'importazione e il path che porta al file desiderato. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

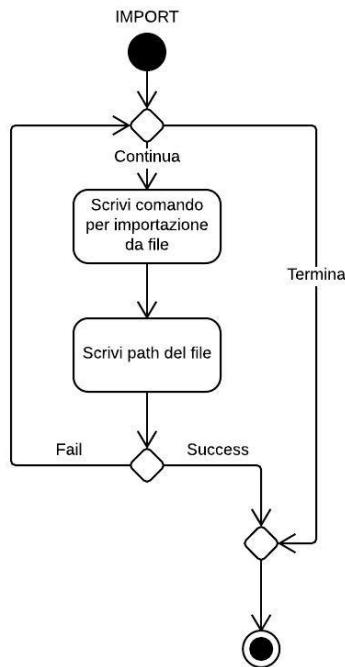


Figura 41: Diagrammi attività - Import

5.3 Interrogazione del database

Questo tipo di operazione permette di interrogare il database_G tramite delle query. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per l'interrogazione del database_G e i parametri per la ricerca. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando, ricevendo il risultato della query, altrimenti verrà visualizzato un messaggio di errore.

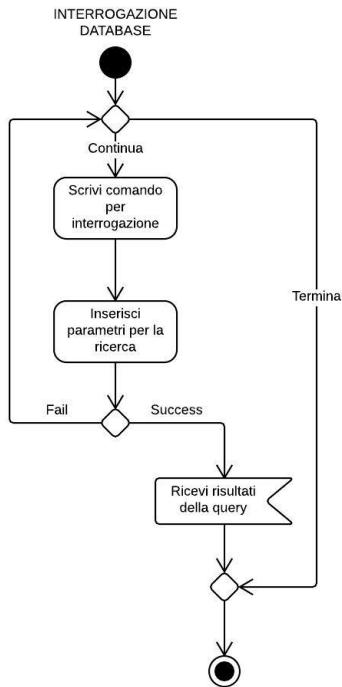


Figura 42: Diagrammi attività - Interrogazione del database_G

5.4 Modifica password

Questo tipo di operazione permette di modificare la propria password. Come si vede dal grafico che segue, l'utente dovrà inserire il comando per la modifica della password, la vecchia password, la nuova password e confermare quest'ultima. Una volta premuto il tasto invio, l'operazione andrà a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

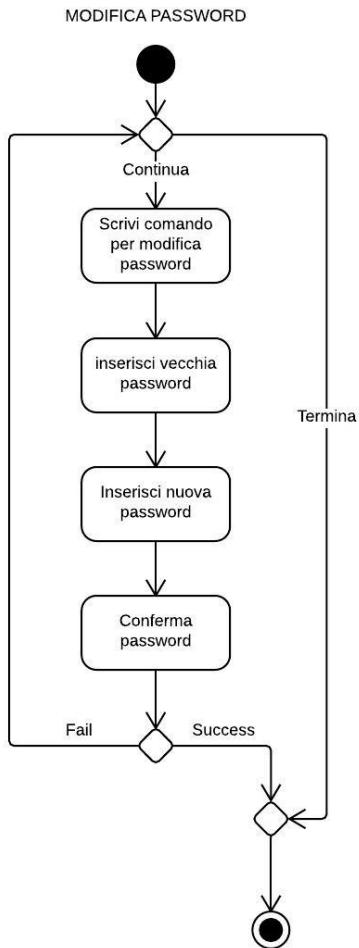


Figura 43: Diagrammi attività - Modifica password

5.5 Gestione utenti

La gestione utenti, possibile solo ad un utente amministratore, prevede la possibilità di aggiungere o rimuovere un utente dal database_G e la possibilità di resettare la password di un utente. Per aggiungere o rimuovere un utente, l'amministratore dovrà inserire il rispettivo comando, lo username dell'utente da aggiungere/rimuovere dal database_G e una conferma. Per resettare la password di un utente, l'amministratore dovrà scrivere il comando per il reset della password, lo username dell'utente interessato e una conferma. Una volta premuto il tasto invio, le operazioni andranno a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

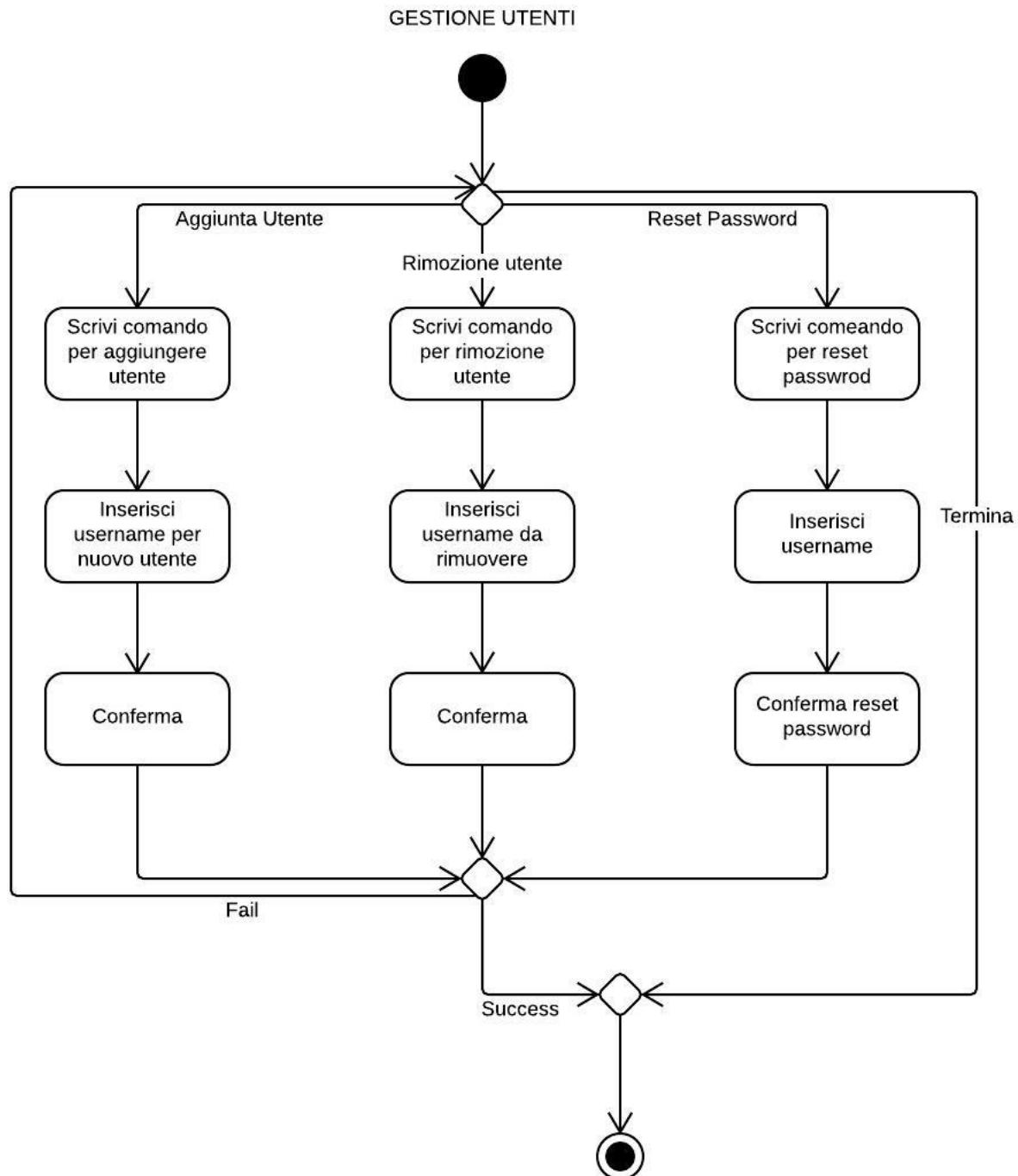


Figura 44: Diagrammi attività - Gestione utenti



6 Design Pattern

I Design Pattern_G rappresentano soluzioni progettuali generali ad un problema ricorrente. Si tratta di modelli logici da applicare per la risoluzione di un problema ancor prima della definizione dell'algoritmo risolutivo vero e proprio, favoriscono il riutilizzo del codice e rendono l'architettura più manutenibile.

I Design Pattern_G possono essere suddivisi in:

- **architetturali:** operano al livello più alto, esprimono schemi di base per impostare l'organizzazione strutturale di un sistema software;
- **creazionali:** nascondono i costruttori delle classi e mettono dei metodi al loro posto creando un'interfaccia, in questo modo forniscono un'astrazione del processo di instanziazione degli oggetti;
- **strutturali:** consentono di riutilizzare degli oggetti esistenti fornendo agli utilizzatori un'interfaccia più adatta alle loro esigenze;
- **comportamentali:** definiscono soluzioni per le interazioni tra oggetti.

Si rimanda all'Appendice A per un approfondimento dei Design Pattern_G utilizzati nel progetto **Aktorbase**.



6.1 Design Pattern Architetturali

6.1.1 MVC

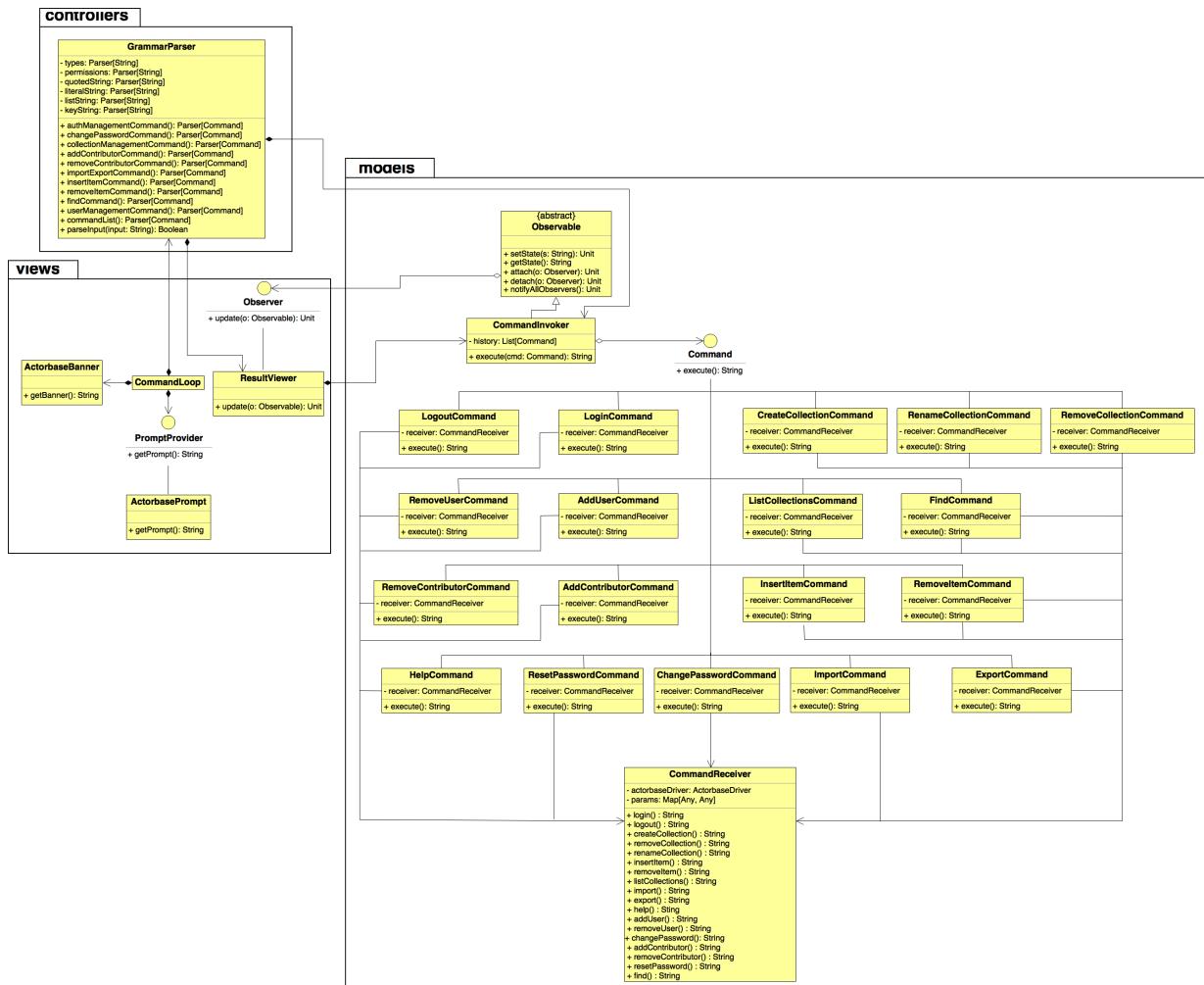


Figura 45: MVC pattern in actorbase

- **Scopo:** È stato scelto l'utilizzo del pattern $\text{Model}_G \text{ View}_G \text{ Controller}_G$ (MVC_G) per l'alto grado di separazione tra la parte logica dell'applicazione e parte grafica che offre;
- **Utilizzo:** Viene utilizzato per delineare l'architettura generale della componente CLI_G (Command Line Interface) del progetto. La parte View_G funge da interfaccia con l'utente, si occupa di leggere l'input e inviarlo al Controller, il quale si occupa di validare l'input e inviare il comando estrappolato al Model_G .



Il Model_G mediante un Command Pattern_G (vedi Command Pattern) esegue il comando richiesto e notifica la View_G utilizzando un Observer Pattern_G (vedi Observer Pattern) secondo la variante Push model_G.

6.1.2 Dependancy Injection

- **Scopo:** Separa il comportamento di una componente dalla risoluzione delle sue dipendenze, offrendo un maggiore grado di astrazione;
- **Utilizzo:** Viene utilizzato nella componente driver, più precisamente nella classe `ActorbaseServices`, sfruttando la variante cake-pattern_G offerta dal linguaggio Scala_G per aggiungere dipendenze amministrative fornite da `ActorbaseAdminServices` alla classe dedicata alle operazioni comuni. Tale pattern_G viene anche utilizzato nella classe `ActorbaseClient` per fornire crittografia TLS/SSL_G (Transport Layer Security / Secure Socket Layer) al client_G utilizzato per la connessione.

6.2 Design Pattern Creazionali

6.2.1 Singleton

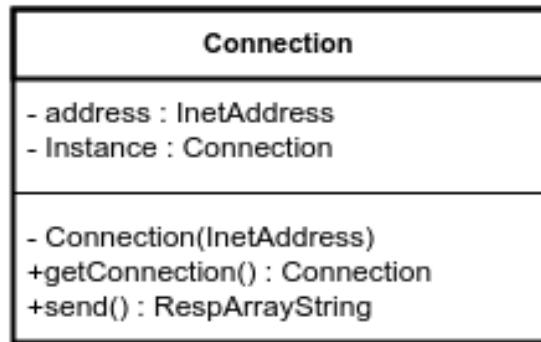


Figura 46: Connection Singleton

- **Scopo:** Viene utilizzato per le classi di cui è preferibile avere un'unica istanza durante l'esecuzione dell'applicazione;
- **Utilizzo:** È stato utilizzato una sola volta per la classe di connessione all'interno della componente driver_G, in modo da offrire un unico punto di accesso al server_G.



6.2.2 Builder

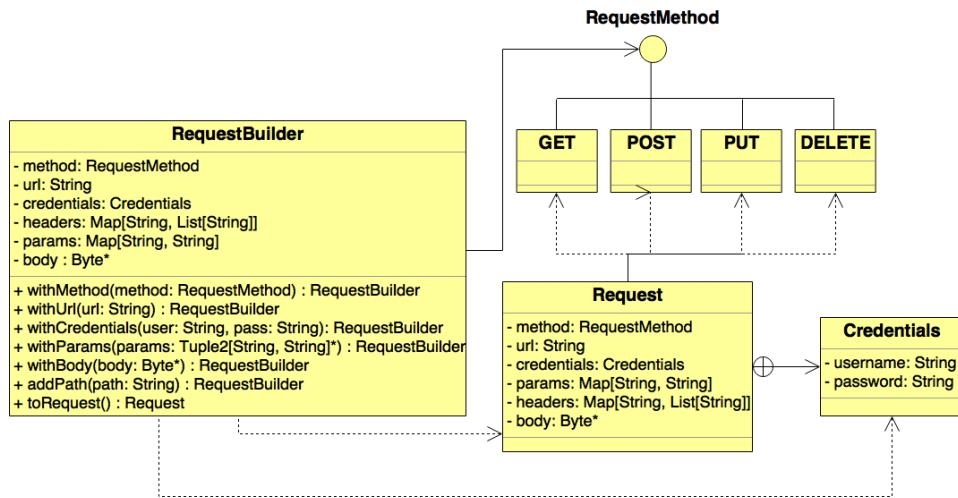


Figura 47: Builder pattern in *actorbase*

- **Scopo:** Il pattern Builder_G viene usato per separare la costruzione di un oggetto complesso dalla sua rappresentazione. L'algoritmo di creazione dell'oggetto complesso è indipendente dalle varie parti che costituiscono l'oggetto e da come vengono assemblate. In questo modo è possibile costruire diverse rappresentazioni dell'oggetto utilizzando lo stesso oggetto Builder_G .
- **Utilizzo:** Viene implementato all'interno della componente driver_G, precisamente dalla classe **RequestBuilder** per la costruzione delle diverse richieste HTTP_G da inviare alla componente server_G.



6.3 Design Pattern Strutturali

6.3.1 Decorator

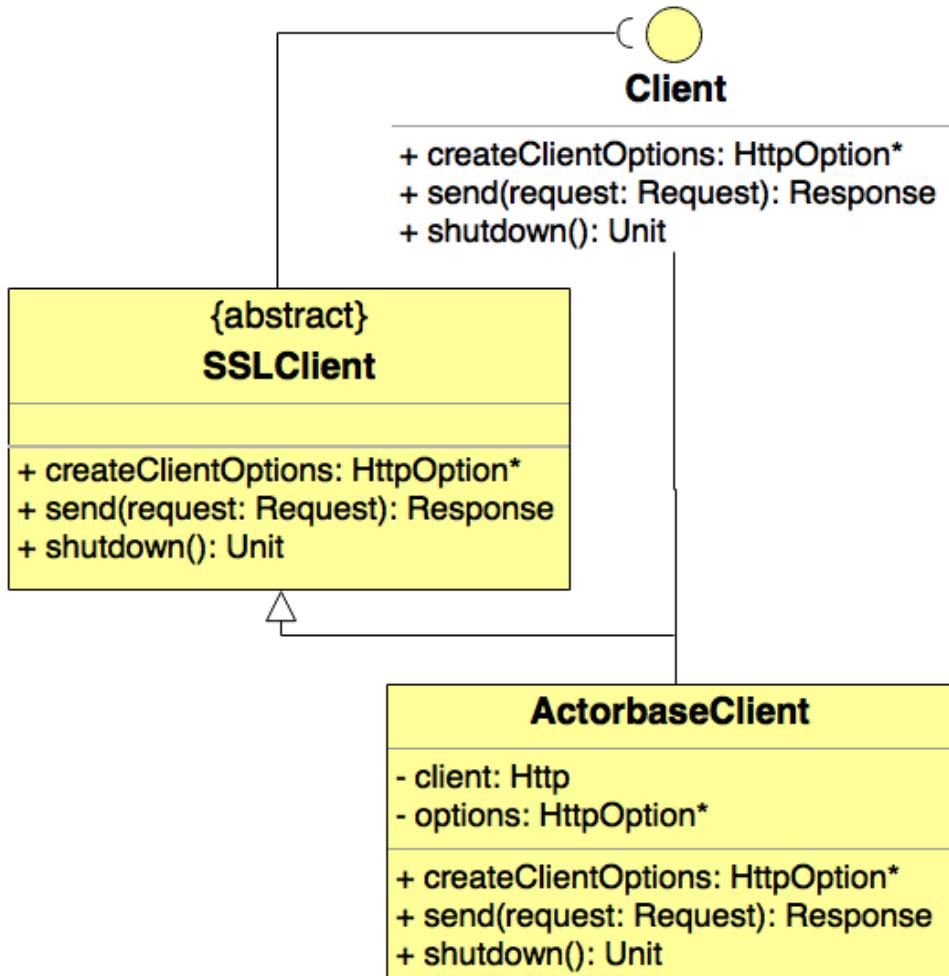


Figura 48: Decorator pattern in `actorbase`

- **Scopo:** Il pattern `DecoratorG` viene usato per aggiungere dinamicamente nuove funzionalità ad oggetti già esistenti, e permette di estenderli più facilmente;
- **Utilizzo:** Viene utilizzato all'interno della componente `driverG`, precisamente dalla classe `ActorbaseClient` per aggiungere funzionalità di protezione `SSL/TLSG` alla comunicazione `HTTPG` con la componente `serverG`.



6.4 Design Pattern Comportamentali

6.4.1 Command Pattern

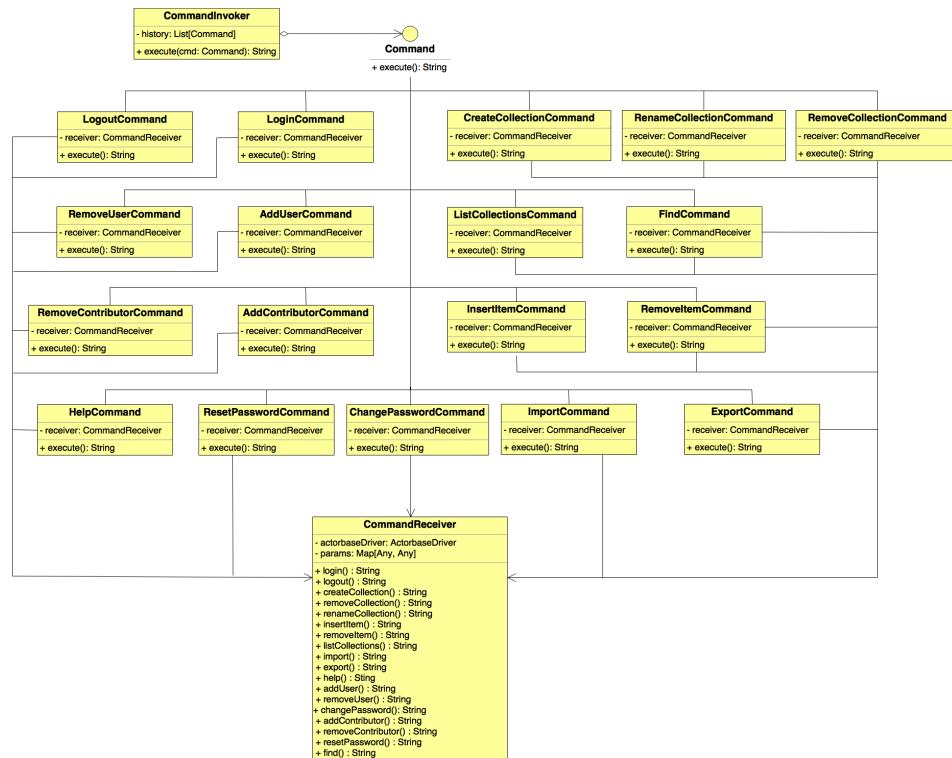


Figura 49: Command pattern in *actorbase*

- **Scopo:** Viene utilizzato per separare l'implementazione di un'azione (Receiver_G) dal richiamante dell'azione stessa (Invoker_G).
- **Utilizzo:** Viene utilizzato per la gestione dei comandi nella componente CLI_G , e funge da Model_G dell'architettura MVC_G utilizzata.

Le classi che implementano i comandi realizzando l'interfaccia `Command` sono:

- `FindCommand`
- `ListCommand`
- `LoginCommand`
- `LogoutCommand`
- `HelpCommand`



- InsertItemCommand
- RemoveItemCommand
- RenameCollectionCommand
- CreateCollectionCommand
- ChangePasswordCommand
- RemoveCollectionCommand
- ImportCommand
- ExportCommand
- AddContributorCommand
- RemoveContributorCommand
- ResetPasswordCommand
- AddUserCommand
- RemoveUserCommand

La classe incaricata di eseguire i comandi è `CommandReceiver`, invocata mediante `CommandInvoker`, utilizzando i parametri ricevuti dalla parte Controller_G dell'architettura MVC_G scelta per la componente CLI_G .

6.4.2 Observer Pattern

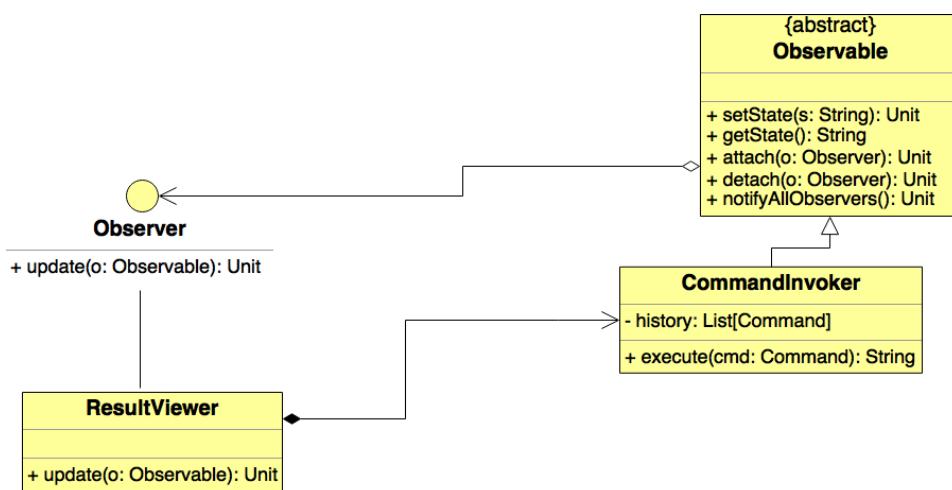


Figura 50: Observer pattern in *actorbase*



- **Scopo:** Viene utilizzato per tenere sotto controllo lo stato di diversi oggetti, ed è parte dell'implementazione del Design Pattern_G MVC_G con logica push model_G, il funzionamento si basa su meccanismi di callback.
- **Utilizzo:** È stato utilizzato nella componente CLI_G del progetto in modo da ottenere una view_G auto-aggiornante sull'output prodotto dai comandi inviati alla componente driver_G.
Più in dettaglio, la classe CommandInvoker realizza l'interfaccia Observable aggiungendo l'oggetto di tipo ResultView alla propria lista di observers_G, all'esecuzione di ogni comando, notifica resultview con l'output prodotto.



7 Stime di fattibilità e di bisogno di risorse

Durante la progettazione dell'architettura, oltre alle tecnologie e librerie consigliate dal proponente, ne sono state ricercate e testate altre in modo da poter usufruire di funzionalità già esistenti.

Dopo un iniziale orientamento verso il framework_G Spring_G, più precisamente nella sua derivazione SpringShell_G, in modo da poter generare una CLI_G personalizzata, abbiamo deciso di rimanere su un'implementazione più leggera in linguaggio Scala_G in quanto la scarsa personalizzazione offerta e il grosso carico di dipendenze che necessitava SpringShell_G per il suo utilizzo è stato ritenuto eccessivamente gravoso sulla realizzazione della componente in questione.

È stato tuttavia deciso di utilizzare il framework_G pickling_G per effettuare serializzazione_G della persistenza su disco; offre una buona documentazione e numerosi esempi di utilizzo, rende inoltre possibile la definizione di protocolli di serializzazione_G personalizzati.

Per la gestione della comunicazione HTTP_G sono stati utilizzate due librerie:

- **Lato server:** Spray permette di gestire efficacemente le richieste in ingresso di tipo HTTP_G asincronamente, appositamente sviluppata per funzionare in congiunzione con il modello ad attori offerto da akka_G;
- **Lato client:** Scalaj-http è una semplice libreria Scala_G che funge da client HTTP_G per la comunicazione con la componente server_G.



8 Tracciamento

8.1 Tracciamento Componenti-Requisiti

Componente	Requisiti
actorbase	DEF3 DEV8 DEV9 OBF1 OBF2 OBQ12 OBV4 OBV5 OBV6 OBV7
actorbase::actorsystem	OBF1
actorbase::actorsystem::clientactor	OBF1.1.10 OBF1.1.10.1 OBF1.1.10.1.1 OBF1.1.10.2 OBF1.1.10.2.1 OBF1.1.10.2.1.1 OBF1.1.10.2.2 OPF1.1.10.2.3 OPF1.1.10.2.3.1 OPF1.1.10.2.3.2 OBF1.1.10.2.4 OBF1.1.10.2.4.1 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.5.3 OBF1.1.10.2.5.4 OBF1.1.10.2.6 OBF1.1.10.2.6.1 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.1 OBF1.1.10.3.2 OBF1.1.10.3.3 OBF1.1.10.3.4 OBF1.1.10.3.5 OBF1.1.10.4 OBF1.1.10.5 OBF1.1.10.5.1 OBF1.1.10.6 OBF1.1.10.7 OBF1.1.10.7.1 OBF1.1.10.7.1.1 OBF1.1.10.7.2 OBF1.1.10.7.2.1 OBF1.1.10.7.3 OBF1.1.10.7.3.1 OBF1.1.2
actorbase::actorsystem::clientactor::messages	OBF1.1.2.1 OBF1.1.2.2
actorbase::actorsystem::httpserver	OBF1.1 OBF1.1.1 OBF1.1.1.1 OBF1.1.10
actorbase::actorsystem::httpserver::messages	OBF1.1.1.1.1



actorbase::actorsystem::main	OBF1.1.10.1 OBF1.1.10.2 OBF1.1.10.2.1 OBF1.1.10.2.1.1 OPF1.1.10.2.3 OBF1.1.10.2.4 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.6 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.1 OBF1.1.10.3.2 OBF1.1.10.3.3 OBF1.1.10.3.4 OBF1.1.10.4 OBF1.1.10.5 OBF1.1.10.7 OBF1.1.10.7.1 OBF1.1.10.7.2 OBF1.1.10.7.2.1 OBF1.1.10.7.3 OBF1.1.10.7.3.1 OBF1.1.3
actorbase::actorsystem::main::messages	OBF1.1.3.1 OBF1.1.3.2 OBF1.1.3.3 OBF1.1.3.4 OBF1.1.3.5 OBF1.1.3.6 OBF1.1.3.7 OBF1.1.3.8 OBF1.1.3.9 OBF1.1.3.11 OBF1.1.3.12 OBF1.1.3.12 OBF1.1.3.13
actorbase::actorsystem::manager	DEF1.1.8 OBF1.1.10.3 OBF1.1.10.3.1
actorbase::actorsystem::manager::messages	DEF1.1.8.1 DEF1.1.8.2
actorbase::actorsystem::ninja	DEF1.1.7 OBF1.1.10.2 OBF1.1.10.2.4 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.6 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.1 OBF1.1.10.3.4
actorbase::actorsystem::ninja::messages	DEF1.1.7.1 DEF1.1.7.2
actorbase::actorsystem::storefinder	OBF1.1.10.1 OBF1.1.10.2 OBF1.1.10.2.1 OPF1.1.10.2.3 OBF1.1.10.2.4 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.5.4 OBF1.1.10.2.6 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.1 OBF1.1.10.3.3 OBF1.1.10.3.4 OBF1.1.10.4 OBF1.1.10.5 OBF1.1.10.7 OBF1.1.10.7.1 OBF1.1.10.7.2 OBF1.1.10.7.2.1 OBF1.1.10.7.3 OBF1.1.10.7.3.1 OBF1.1.4



actorbase::actorsystem::storefinder::messages	OBF1.1.4.1 OBF1.1.4.2 OBF1.1.4.3 OBF1.1.4.4 OBF1.1.4.5 OBF1.1.4.6 OBF1.1.4.7
actorbase::actorsystem::storekeeper	OBF1.1.10.2 OBF1.1.10.2.1 OBF1.1.10.2.4 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.6 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.1 OBF1.1.10.3.3 OBF1.1.10.3.4 OBF1.1.10.4 OBF1.1.5
actorbase::actorsystem::storekeeper::messages	OBF1.1.5.1 OBF1.1.5.2 OBF1.1.5.3 OBF1.1.5.4 OBF1.1.5.5 OBF1.1.5.6 OBF1.1.5.7
actorbase::actorsystem::userfinder	OBF1.1.10.1 OBF1.1.10.1.1 OBF1.1.10.2 OBF1.1.10.2.1 OPF1.1.10.2.3 OPF1.1.10.2.3.1 OBF1.1.10.2.4 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.6 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.2 OBF1.1.10.5 OBF1.1.10.5.1 OBF1.1.10.7 OBF1.1.10.7.1 OBF1.1.10.7.3 OBF1.1.11
actorbase::actorsystem::userfinder::messages	OBF1.1.11.1 OBF1.1.11.2 OBF1.1.11.3 OBF1.1.11.4 OBF1.1.11.5 OBF1.1.11.6 OBF1.1.11.7
actorbase::actorsystem::userkeeper	OBF1.1.10.1 OBF1.1.10.1.1 OBF1.1.10.2 OBF1.1.10.2.1 OPF1.1.10.2.3 OPF1.1.10.2.3.1 OBF1.1.10.2.4 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.6 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.2 OBF1.1.10.5 OBF1.1.10.5.1 OBF1.1.10.7 OBF1.1.10.7.1 OBF1.1.10.7.3 OBF1.1.9
actorbase::actorsystem::userkeeper::messages	OBF1.1.9.1 OBF1.1.9.2 OBF1.1.9.3 OBF1.1.9.4 OBF1.1.9.5 OBF1.1.9.6 OBF1.1.9.9



actorbase::actorsystem::utils	DEF1.1.7 OBF1.1.11 OBF1.1.2 OBF1.1.3 OBF1.1.4 OBF1.1.5 OBF1.1.6 OBF1.1.9
actorbase::actorsystem::warehouseman	OBF1.1.10.2 OBF1.1.10.2.4 OBF1.1.10.2.5 OBF1.1.10.2.5.1 OBF1.1.10.2.5.2 OBF1.1.10.2.6 OBF1.1.10.2.7 OBF1.1.10.3 OBF1.1.10.3.1 OBF1.1.10.3.4 OBF1.1.6
actorbase::actorsystem::warehouseman::messages	OBF1.1.6.1 OBF1.1.6.2 OBF1.1.6.3 OBF1.1.6.4
actorbase::cli	OBF2
actorbase::cli::controllers	DEF2.1.2.7 DEF2.1.3.1.2 DEF2.1.4 DEF2.1.5 OBF2.1 OBF2.1.1 OBF2.1.2 OBF2.1.2.1 OBF2.1.2.2 OBF2.1.2.3 OPF2.1.2.4 OBF2.1.2.5 OBF2.1.2.6 OBF2.1.3 OBF2.1.3.1 OBF2.1.3.1.1 OBF2.1.6 OBF2.1.7 OBF2.1.8 OBF2.1.9 OBF2.1.9.1 OBF2.1.9.2 OBF2.1.9.3



actorbase::cli::models	DEF2.1.2.7 DEF2.1.2.7.1 DEF2.1.3.1.2 DEF2.1.3.1.2.1 DEF2.1.4 DEF2.1.5 DEF2.1.5.1 OBF2.1.1 OBF2.1.1.1 OBF2.1.1.2 OBF2.1.2 OBF2.1.2.1 OBF2.1.2.1.1 OBF2.1.2.2 OBF2.1.2.3 OBF2.1.2.3.1 OPF2.1.2.4 OPF2.1.2.4.1 OPF2.1.2.4.2 OBF2.1.2.5 OBF2.1.2.5.1 OBF2.1.2.5.2 OBF2.1.2.5.3 OBF2.1.2.6 OBF2.1.2.6.1 OBF2.1.2.6.2 OBF2.1.2.8 OBF2.1.3 OBF2.1.3.1 OBF2.1.3.1.1 OBF2.1.3.1.1.1 OBF2.1.3.1.1.2 OBF2.1.3.1.1.3 OBF2.1.3.1.1.4 OBF2.1.3.1.1.5 OBF2.1.3.6 OBF2.1.6 OBF2.1.6.1 OBF2.1.6.2 OBF2.1.7 OBF2.1.7.1 OBF2.1.7.2 OBF2.1.7.3 OBF2.1.8 OBF2.1.9 OBF2.1.9.1 OBF2.1.9.1.1 OBF2.1.9.1.2 OBF2.1.9.2 OBF2.1.9.2.1 OBF2.1.9.2.2 OBF2.1.9.3 OBF2.1.9.3.1 OBF2.1.9.3.2
------------------------	---



actorbase::cli::views	DEF2.1.2.7 DEF2.1.2.7.1 DEF2.1.2.7.2 DEF2.1.3.1.2 DEF2.1.3.1.2.1 DEF2.1.3.2 DEF2.1.3.3 DEF2.1.4 DEF2.1.5 DEF2.1.5.1 OBF2 OBF2.1.1 OBF2.1.1.1 OBF2.1.1.2 OBF2.1.1.3 OBF2.1.2 OBF2.1.2.1 OBF2.1.2.1.1 OPF2.1.2.10 OBF2.1.2.11 OBF2.1.2.12 OBF2.1.2.13 OBF2.1.2.14 OBF2.1.2.2 OBF2.1.2.3 OBF2.1.2.3.1 OPF2.1.2.4 OPF2.1.2.4.1 OPF2.1.2.4.2 OBF2.1.2.5 OBF2.1.2.5.1 OBF2.1.2.5.2 OBF2.1.2.5.3 OBF2.1.2.6 OBF2.1.2.6.1 OBF2.1.2.6.2 OBF2.1.2.8 OPF2.1.2.9 OBF2.1.3 OBF2.1.3.1 OBF2.1.3.1.1 OBF2.1.3.1.1.1 OBF2.1.3.1.1.2 OBF2.1.3.1.1.3 OBF2.1.3.1.1.4 OBF2.1.3.1.1.5 OBF2.1.3.4 OBF2.1.3.6 OBF2.1.3.6.1 OBF2.1.3.6.2 OBF2.1.3.5 OBF2.1.6 OBF2.1.6.1 OBF2.1.6.2 OBF2.1.7 OBF2.1.7.1 OBF2.1.7.2 OBF2.1.7.3 OBF2.1.7.4 OBF2.1.7.5 OBF2.1.7.6 OBF2.1.8 OBF2.1.9 OBF2.1.9.1 OBF2.1.9.1.1 OBF2.1.9.1.2 OBF2.1.9.2 OBF2.1.9.2.1 OBF2.1.9.2.2 OBF2.1.9.3 OBF2.1.9.3.1 OBF2.1.9.3.2 OBF2.1.9.4 OBF2.1.9.5 OBF2.1.9.6
-----------------------	--



actorbase::cli::views::Observer	DEF3.1.2 DEF3.1.3 OPF2.1.2.10 OBF2.1.2.11 OBF2.1.2.12 OBF2.1.2.13 OBF2.1.2.14 OBF2.1.2.8 OPF2.1.2.9 OBF2.1.3.4 OBF2.1.3.5 OBF2.1.7.4 OBF2.1.7.5 OBF2.1.7.6 OBF2.1.9.4 OBF2.1.9.5 OBF2.1.9.6
actorbase::driver	DEF3



actorbase::driver::client	DEF3.1 DEF3.1.1 DEF3.1.2 DEF3.1.3 DEF3.2 DEF3.2.1 DEF3.2.1.1 DEF3.2.1.2 DEF3.2.2 DEF3.2.3 DEF3.2.3.1 DEF3.2.3.2 OPF3.2.4 OPF3.2.4.1 OPF3.2.4.2 OPF3.2.4.3 OPF3.2.4.4 DEF3.2.5 DEF3.2.5.1 DEF3.2.5.2 DEF3.2.5.3 DEF3.2.5.4 DEF3.2.5.5 DEF3.2.6 DEF3.2.6.1 DEF3.2.6.2 DEF3.2.6.3 DEF3.2.7 DEF3.2.7.1 DEF3.2.7.2 DEF3.2.7.3 DEF3.3 DEF3.3.1 DEF3.3.1.1 DEF3.3.1.1.1 DEF3.3.1.1.2 DEF3.3.1.1.3 DEF3.3.1.1.4 DEF3.3.1.2 DEF3.3.1.2.1 DEF3.3.1.2.2 DEF3.3.1.2.3 DEF3.3.1.4 DEF3.3.2 DEF3.3.2.1 DEF3.3.2.2 DEF3.3.2.3 DEF3.4 DEF3.4.1 DEF3.4.2 DEF3.5 DEF3.6 DEF3.6.1 DEF3.6.1.1 DEF3.6.2 DEF3.6.2.1 DEF3.6.3 DEF3.6.3.1 DEF3.6.4 DEF3.6.5 DEF3.6.6 DEF3.7 DEF3.7.1 DEF3.7.2 DEF3.7.3 DEF3.7.4 DEF3.8 DEF3.8.1 DEF3.8.2 DEF3.8.3
---------------------------	---



actorbase::driver::client::api	DEF3.1 DEF3.1.1 DEF3.1.2 DEF3.2 DEF3.2.1 DEF3.2.1.1 DEF3.2.2 DEF3.2.3 DEF3.2.3.1 OPF3.2.4 OPF3.2.4.1 OPF3.2.4.2 DEF3.2.5 DEF3.2.5.1 DEF3.2.5.2 DEF3.2.5.5 DEF3.2.6 DEF3.2.6.1 DEF3.2.6.2 DEF3.2.7 DEF3.2.7.1 DEF3.2.7.2 DEF3.2.7.3 DEF3.3 DEF3.3.1 DEF3.3.1.1 DEF3.3.1.1.1 DEF3.3.1.1.2 DEF3.3.1.1.3 DEF3.3.1.1.4 DEF3.3.1.2 DEF3.3.1.2.1 DEF3.3.2 DEF3.3.2.1 DEF3.3.2.2 DEF3.4 DEF3.4.1 DEF3.4.2 DEF3.5 DEF3.6 DEF3.6.1 DEF3.6.1.1 DEF3.6.2 DEF3.6.2.1 DEF3.6.3 DEF3.6.3.1 DEF3.7 DEF3.7.1 DEF3.7.2
--------------------------------	--



actorbase::driver::data	DEF3.2 DEF3.2.1 DEF3.2.1.1 DEF3.2.1.2 DEF3.2.3 DEF3.2.3.1 OPF3.2.4 OPF3.2.4.1 OPF3.2.4.2 OPF3.2.4.3 OPF3.2.4.4 DEF3.2.5 DEF3.2.5.1 DEF3.2.5.2 DEF3.2.5.3 DEF3.2.5.4 DEF3.2.5.5 DEF3.2.6 DEF3.2.6.1 DEF3.2.6.2 DEF3.2.6.3 DEF3.3 DEF3.3.1 DEF3.3.1.1 DEF3.3.1.1.1 DEF3.3.1.1.2 DEF3.3.1.1.3 DEF3.3.1.1.4 DEF3.3.1.4 DEF3.3.2 DEF3.3.2.1 DEF3.3.2.2 DEF3.3.2.3 DEF3.4 DEF3.4.1 DEF3.4.2 DEF3.8 DEF3.8.1 DEF3.8.2 DEF3.8.3
actorbase::driver::exceptions	DEF3.1.3 DEF3.2.1.2 DEF3.2.3.2 OPF3.2.4.3 OPF3.2.4.4 DEF3.2.5.3 DEF3.2.5.4 DEF3.2.6.3 DEF3.3.1.2.2 DEF3.3.1.2.3 DEF3.3.1.4 DEF3.3.2.3 DEF3.6.4 DEF3.6.5 DEF3.6.6 DEF3.7.3 DEF3.7.4

8.2 Tracciamento Requisiti-Componenti

Requisito	Componenti
DEF1.1.7	actorbase::actorsystem::ninja actorbase::actorsystem::utils
DEF1.1.7.1	actorbase::actorsystem::ninja::messages
DEF1.1.7.2	actorbase::actorsystem::ninja::messages
DEF1.1.8	actorbase::actorsystem::manager
DEF1.1.8.1	actorbase::actorsystem::manager::messages
DEF1.1.8.2	actorbase::actorsystem::manager::messages
DEF2.1.2.7	actorbase::cli::models actorbase::cli::views



DEF2.1.2.7.1	actorbase::cli::models actorbase::cli::views
DEF2.1.2.7.2	actorbase::cli::views
DEF2.1.3.1.2	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
DEF2.1.3.1.2.1	actorbase::cli::models actorbase::cli::views
DEF2.1.3.2	actorbase::cli::views
DEF2.1.3.3	actorbase::cli::views
DEF2.1.4	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
DEF2.1.5	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
DEF2.1.5.1	actorbase::cli::models actorbase::cli::views
DEF3	actorbase actorbase::driver
DEF3.1	actorbase::driver::client actorbase::driver::client::api
DEF3.1.1	actorbase::driver::client actorbase::driver::client::api
DEF3.1.2	actorbase::cli::views::Observer actorbase::driver::client actorbase::driver::client::api
DEF3.1.3	actorbase::cli::views::Observer actorbase::driver::client actorbase::driver::exceptions
DEF3.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.1.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.1.2	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions



DEF3.2.2	actorbase::driver::client actorbase::driver::client::api
DEF3.2.3	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.3.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.3.2	actorbase::driver::client actorbase::driver::exceptions
OPF3.2.4	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
OPF3.2.4.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
OPF3.2.4.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
OPF3.2.4.3	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions
OPF3.2.4.4	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions
DEF3.2.5	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.5.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.5.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.5.3	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions
DEF3.2.5.4	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions
DEF3.2.5.5	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data



DEF3.2.6	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.6.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.6.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.2.6.3	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions
DEF3.2.7	actorbase::driver::client actorbase::driver::client::api
DEF3.2.7.1	actorbase::driver::client actorbase::driver::client::api
DEF3.2.7.2	actorbase::driver::client actorbase::driver::client::api
DEF3.2.7.3	actorbase::driver::client actorbase::driver::client::api
DEF3.3	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.1.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.1.1.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.1.1.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.1.1.3	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.1.1.4	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.1.2	actorbase::driver::client actorbase::driver::client::api



DEF3.3.1.2.1	actorbase::driver::client actorbase::driver::client::api
DEF3.3.1.2.2	actorbase::driver::client actorbase::driver::exceptions
DEF3.3.1.2.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.3.1.4	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions
DEF3.3.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.2.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.2.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.3.2.3	actorbase::driver::client actorbase::driver::data actorbase::driver::exceptions
DEF3.4	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.4.1	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.4.2	actorbase::driver::client actorbase::driver::client::api actorbase::driver::data
DEF3.5	actorbase::driver::client actorbase::driver::client::api
DEF3.6	actorbase::driver::client actorbase::driver::client::api
DEF3.6.1	actorbase::driver::client actorbase::driver::client::api
DEF3.6.1.1	actorbase::driver::client actorbase::driver::client::api
DEF3.6.2	actorbase::driver::client actorbase::driver::client::api
DEF3.6.2.1	actorbase::driver::client actorbase::driver::client::api



DEF3.6.3	actorbase::driver::client actorbase::driver::client::api
DEF3.6.3.1	actorbase::driver::client actorbase::driver::client::api
DEF3.6.4	actorbase::driver::client actorbase::driver::exceptions
DEF3.6.5	actorbase::driver::client actorbase::driver::exceptions
DEF3.6.6	actorbase::driver::client actorbase::driver::exceptions
DEF3.7	actorbase::driver::client actorbase::driver::client::api
DEF3.7.1	actorbase::driver::client actorbase::driver::client::api
DEF3.7.2	actorbase::driver::client actorbase::driver::client::api
DEF3.7.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.7.4	actorbase::driver::client actorbase::driver::exceptions
DEF3.8	actorbase::driver::client actorbase::driver::data
DEF3.8.1	actorbase::driver::client actorbase::driver::data
DEF3.8.2	actorbase::driver::client actorbase::driver::data
DEF3.8.3	actorbase::driver::client actorbase::driver::data
DEV8	actorbase
DEV9	actorbase
OBF1	actorbase actorbase::actorsystem
OBF1.1	actorbase::actorsystem::httpserver
OBF1.1.1	actorbase::actorsystem::httpserver
OBF1.1.1.1	actorbase::actorsystem::httpserver
OBF1.1.1.1.1	actorbase::actorsystem::httpserver::messages
OBF1.1.10	actorbase::actorsystem::clientactor actorbase::actorsystem::httpserver
OBF1.1.10.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper



OBF1.1.10.1.1	actorbase::actorsystem::clientactor actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.2	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman
OBF1.1.10.2.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.2.1.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main
OBF1.1.10.2.2	actorbase::actorsystem::clientactor
OPF1.1.10.2.3	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OPF1.1.10.2.3.1	actorbase::actorsystem::clientactor actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OPF1.1.10.2.3.2	actorbase::actorsystem::clientactor
OBF1.1.10.2.4	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman
OBF1.1.10.2.4.1	actorbase::actorsystem::clientactor



OBF1.1.10.2.5	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman
OBF1.1.10.2.5.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman
OBF1.1.10.2.5.2	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman
OBF1.1.10.2.5.3	actorbase::actorsystem::clientactor
OBF1.1.10.2.5.4	actorbase::actorsystem::clientactor actorbase::actorsystem::storefinder
OBF1.1.10.2.6	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman
OBF1.1.10.2.6.1	actorbase::actorsystem::clientactor
OBF1.1.10.2.7	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman



OBF1.1.10.3	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::manager actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper actorbase::actorsystem::warehouseman
OBF1.1.10.3.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::manager actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::warehouseman
OBF1.1.10.3.2	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.3.3	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper
OBF1.1.10.3.4	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper actorbase::actorsystem::warehouseman
OBF1.1.10.3.5	actorbase::actorsystem::clientactor
OBF1.1.10.4	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper
OBF1.1.10.5	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.5.1	actorbase::actorsystem::clientactor actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.6	actorbase::actorsystem::clientactor



OBF1.1.10.7	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.7.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.7.1.1	actorbase::actorsystem::clientactor
OBF1.1.10.7.2	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder
OBF1.1.10.7.2.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder
OBF1.1.10.7.3	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userfinder actorbase::actorsystem::userkeeper
OBF1.1.10.7.3.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder
OBF1.1.11	actorbase::actorsystem::userfinder actorbase::actorsystem::utils
OBF1.1.11.1	actorbase::actorsystem::userfinder::messages
OBF1.1.11.2	actorbase::actorsystem::userfinder::messages
OBF1.1.11.3	actorbase::actorsystem::userfinder::messages
OBF1.1.11.4	actorbase::actorsystem::userfinder::messages
OBF1.1.11.5	actorbase::actorsystem::userfinder::messages
OBF1.1.11.6	actorbase::actorsystem::userfinder::messages
OBF1.1.11.7	actorbase::actorsystem::userfinder::messages
OBF1.1.2	actorbase::actorsystem::clientactor actorbase::actorsystem::utils
OBF1.1.2.1	actorbase::actorsystem::clientactor::messages
OBF1.1.2.2	actorbase::actorsystem::clientactor::messages
OBF1.1.3	actorbase::actorsystem::main actorbase::actorsystem::utils
OBF1.1.3.1	actorbase::actorsystem::main::messages
OBF1.1.3.10	actorbase::actorsystem::main::messages
OBF1.1.3.11	actorbase::actorsystem::main::messages



OBF1.1.3.12	actorbase::actorsystem::main::messages
OBF1.1.3.13	actorbase::actorsystem::main::messages
OBF1.1.3.2	actorbase::actorsystem::main::messages
OBF1.1.3.3	actorbase::actorsystem::main::messages
OBF1.1.3.4	actorbase::actorsystem::main::messages
OBF1.1.3.5	actorbase::actorsystem::main::messages
OBF1.1.3.6	actorbase::actorsystem::main::messages
OBF1.1.3.7	actorbase::actorsystem::main::messages
OBF1.1.3.8	actorbase::actorsystem::main::messages
OBF1.1.3.9	actorbase::actorsystem::main::messages
OBF1.1.4	actorbase::actorsystem::storefinder actorbase::actorsystem::utils
OBF1.1.4.1	actorbase::actorsystem::storefinder::messages
OBF1.1.4.2	actorbase::actorsystem::storefinder::messages
OBF1.1.4.3	actorbase::actorsystem::storefinder::messages
OBF1.1.4.4	actorbase::actorsystem::storefinder::messages
OBF1.1.4.5	actorbase::actorsystem::storefinder::messages
OBF1.1.4.6	actorbase::actorsystem::storefinder::messages
OBF1.1.4.7	actorbase::actorsystem::storefinder::messages
OBF1.1.5	actorbase::actorsystem::storekeeper actorbase::actorsystem::utils
OBF1.1.5.1	actorbase::actorsystem::storekeeper::messages
OBF1.1.5.2	actorbase::actorsystem::storekeeper::messages
OBF1.1.5.3	actorbase::actorsystem::storekeeper::messages
OBF1.1.5.4	actorbase::actorsystem::storekeeper::messages
OBF1.1.5.5	actorbase::actorsystem::storekeeper::messages
OBF1.1.5.6	actorbase::actorsystem::storekeeper::messages
OBF1.1.5.7	actorbase::actorsystem::storekeeper::messages
OBF1.1.6	actorbase::actorsystem::utils actorbase::actorsystem::warehouseman
OBF1.1.6.1	actorbase::actorsystem::warehouseman::messages
OBF1.1.6.2	actorbase::actorsystem::warehouseman::messages
OBF1.1.6.3	actorbase::actorsystem::warehouseman::messages
OBF1.1.6.4	actorbase::actorystem::warehouseman::messages
OBF1.1.9	actorbase::actorsystem::userkeeper actorbase::actorsystem::utils
OBF1.1.9.1	actorbase::actorsystem::userkeeper::messages
OBF1.1.9.2	actorbase::actorsystem::userkeeper::messages
OBF1.1.9.3	actorbase::actorsystem::userkeeper::messages
OBF1.1.9.4	actorbase::actorsystem::userkeeper::messages
OBF1.1.9.5	actorbase::actorsystem::userkeeper::messages
OBF1.1.9.6	actorbase::actorsystem::userkeeper::messages
OBF1.1.9.9	actorbase::actorsystem::userkeeper::messages



OBF2	actorbase actorbase::cli actorbase::cli::views
OBF2.1	actorbase::cli::controllers
OBF2.1.1	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.1.1	actorbase::cli::models actorbase::cli::views
OBF2.1.1.2	actorbase::cli::models actorbase::cli::views
OBF2.1.1.3	actorbase::cli::views
OBF2.1.2	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.2.1	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.2.1.1	actorbase::cli::models actorbase::cli::views
OPF2.1.2.10	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.2.11	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.2.12	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.2.13	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.2.14	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.2.2	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.2.3	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.2.3.1	actorbase::cli::models actorbase::cli::views
OPF2.1.2.4	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OPF2.1.2.4.1	actorbase::cli::models actorbase::cli::views



OPF2.1.2.4.2	actorbase::cli::models actorbase::cli::views
OBF2.1.2.5	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.2.5.1	actorbase::cli::models actorbase::cli::views
OBF2.1.2.5.2	actorbase::cli::models actorbase::cli::views
OBF2.1.2.5.3	actorbase::cli::models actorbase::cli::views
OBF2.1.2.6	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.2.6.1	actorbase::cli::models actorbase::cli::views
OBF2.1.2.6.2	actorbase::cli::models actorbase::cli::views
OBF2.1.2.8	actorbase::cli::models actorbase::cli::views actorbase::cli::views::Observer
OPF2.1.2.9	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.3	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.3.1	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.3.1.1	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.3.1.1.1	actorbase::cli::models actorbase::cli::views
OBF2.1.3.1.1.2	actorbase::cli::models actorbase::cli::views
OBF2.1.3.1.1.3	actorbase::cli::models actorbase::cli::views
OBF2.1.3.1.1.4	actorbase::cli::models actorbase::cli::views
OBF2.1.3.1.1.5	actorbase::cli::models actorbase::cli::views



OBF2.1.3.4	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.3.5	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.3.6	actorbase::cli::models actorbase::cli::views
OBF2.1.3.6.1	actorbase::cli::views
OBF2.1.3.6.2	actorbase::cli::views
OBF2.1.6	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.6.1	actorbase::cli::models actorbase::cli::views
OBF2.1.6.2	actorbase::cli::models actorbase::cli::views
OBF2.1.7	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.7.1	actorbase::cli::models actorbase::cli::views
OBF2.1.7.2	actorbase::cli::models actorbase::cli::views
OBF2.1.7.3	actorbase::cli::models actorbase::cli::views
OBF2.1.7.4	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.7.5	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.7.6	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.8	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.9	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.9.1	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.9.1.1	actorbase::cli::models actorbase::cli::views
OBF2.1.9.1.2	actorbase::cli::models actorbase::cli::views



OBF2.1.9.2	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.9.2.1	actorbase::cli::models actorbase::cli::views
OBF2.1.9.2.2	actorbase::cli::models actorbase::cli::views
OBF2.1.9.3	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.9.3.1	actorbase::cli::models actorbase::cli::views
OBF2.1.9.3.2	actorbase::cli::models actorbase::cli::views
OBF2.1.9.4	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.9.5	actorbase::cli::views actorbase::cli::views::Observer
OBF2.1.9.6	actorbase::cli::views actorbase::cli::views::Observer
OBQ12	actorbase
OBV4	actorbase
OBV5	actorbase
OBV6	actorbase
OBV7	actorbase



A Descrizione Design Pattern

A.1 Design Pattern Architetturali

A.1.1 MVC

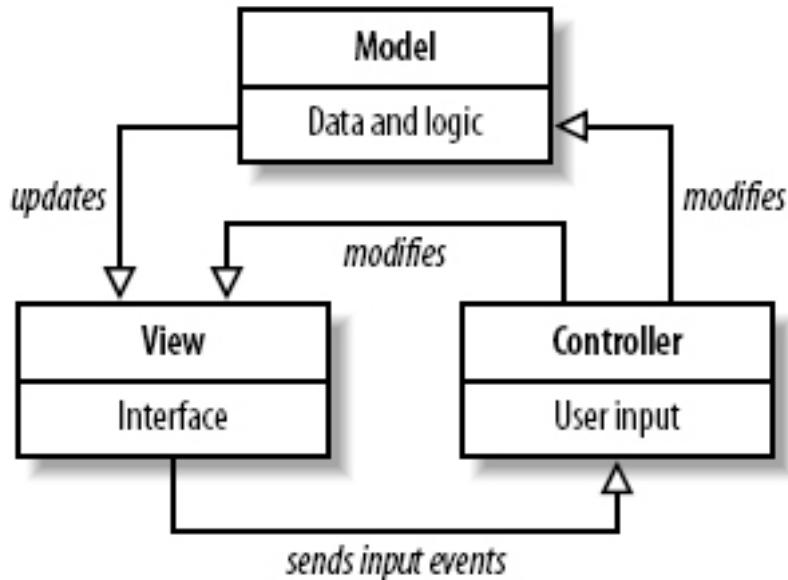


Figura 51: Esempio di applicazione del design pattern MVC

- **Scopo:** Disaccoppiare le tre seguenti componenti:
 - Model: parte che si occupa direttamente di dati, logica e regole dell'applicazione;
 - View: rappresentazione grafica sotto forma delle varie tipologie di output utilizzate per rappresentare i dati dell'applicazione;
 - Controller: parte che accetta gli input e li converte in comandi per il model o la view;
- **Motivazione:** Molte applicazioni hanno la necessità di recuperare dati e di mostrarli in maniera opportuna agli utenti. Poiché si tratta di una comunicazione tra i dati dei dati e la interfaccia utente, bisogna trovare il metodo per far comunicare le due parti senza accorparle assieme, cosa che comporterebbe codice pesante e scarsa manutenibilità, in quanto in genere la parte grafica si evolve più in fretta della parte di model e, viceversa, bisogna rendere l'interfaccia che si offre all'utente quanto più separata possibile dall'implementazione effettiva della gestione dei dati. La soluzione che è stata trovata è costituita dal design pattern₆ Model-View-Controller (MVC) che separa le tre componenti come già illustrato;



- **Applicabilità:** Il pattern MVC è adatto ad essere utilizzato nei seguenti casi:

- se c'è la necessità che un insieme di oggetti debba essere considerato come un oggetto singolo;
- se c'è la necessità di disaccoppiare il model e la view, creando un sistema di notifiche tra le due componenti;
- Se c'è la necessità di avere più view per il medesimo model.

A.1.2 Dependency injection

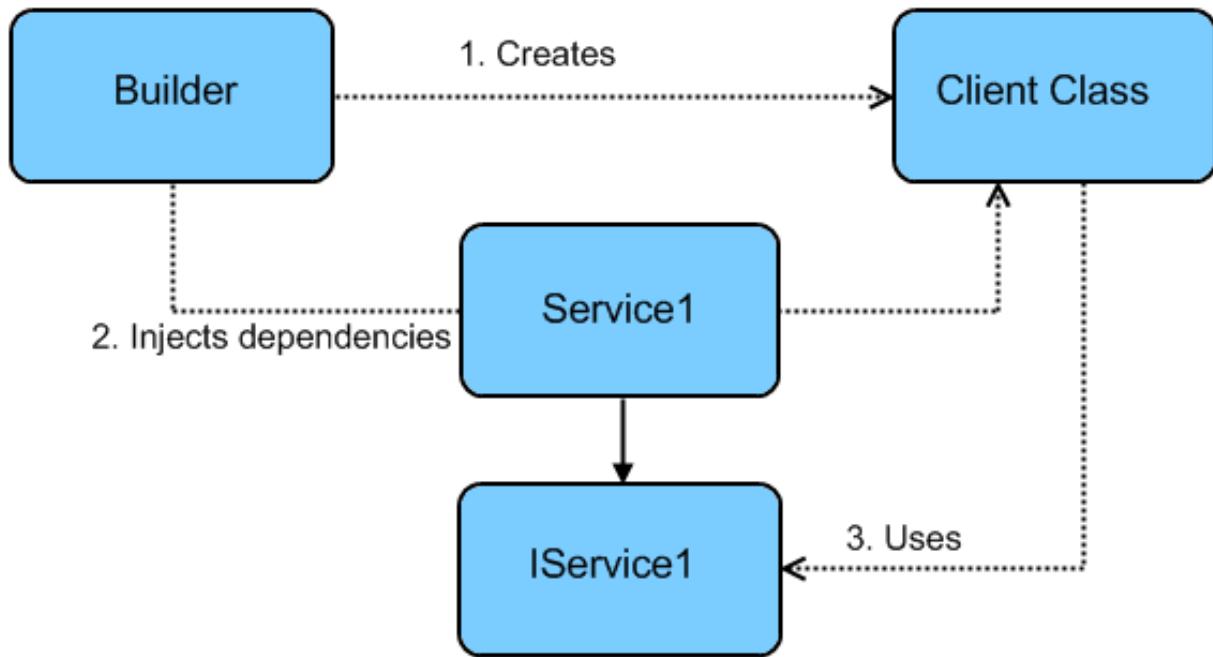


Figura 52: Esempio di applicazione del design pattern Dependency Injection

- **Scopo:** Separare il comportamento di una componente dalla risoluzione delle sue dipendenze;
- **Motivazione:** Collegare due componenti in modo esplicito ne aumenta l'accoppiamento, la progettazione degli unit-test risulta difficoltosa, c'è uno scarso utilizzo della componente e una scarsa manutenibilità. Di conseguenza le dipendenze vanno minimizzate e un metodo per farlo è appunto l'utilizzo del design pattern Dependency Injection;
- **Applicabilità:** Il pattern Dependency injection è adatto ad essere utilizzato nei seguenti casi:
 - se si vuole migliorare la fase di testing;
 - se si vuole avere una maggior resilienza del codice;



- se si vuole avere una migliore flessibilità.

A.2 Design Pattern Creazionali

A.2.1 Singleton

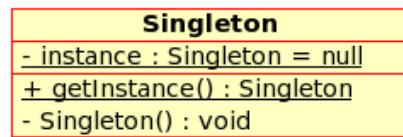


Figura 53: Esempio di applicazione del design pattern singleton

- Scopo:** Assicurare che una classe abbia un'unica istanza ed avere un punto di accesso globale ad essa;
- Motivazione:** Per alcune classi è fondamentale assicurare che non abbiano più di un'istanza. Per questo motivo la classe tipicamente ha un costruttore privato e un metodo per poter accedere all'istanza della classe pubblico;
- Applicabilità:** Questo design pattern_G può essere applicato nei seguenti casi:
 - Quando è essenziale che ci sia solo una istanza di una classe e tale istanza deve essere resa accessibile attraverso un unico punto di accesso;
 - Quando l'unica istanza della classe deve essere estesa in maniera tale da non dover avere client_G con codici diversi.

A.2.2 Builder

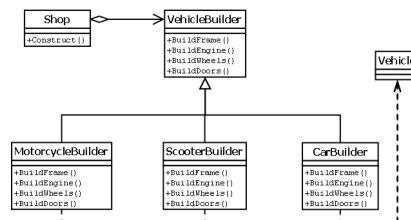


Figura 54: Esempio di applicazione del design pattern builder

- Scopo:** Permettere la crezione di oggetti complessi indipendentemente dalla sua rappresentazione;



- **Motivazione:** Alcuni oggetti richiedono molti parametri di inizializzazione e possono avere molte rappresentazioni, questo porta spesso all'effetto telescoping_G dei costruttori di tale oggetto, e il pattern_G abstract factory_G risulta non essere abbastanza adeguato allo scopo;
- **Applicabilità:** Questo design pattern_G può essere applicato nei seguenti casi:
 - quando è necessaria la creazione di oggetti complessi con molte varianti di rappresentazione.

A.3 Design Pattern Strutturali

A.3.1 Decorator

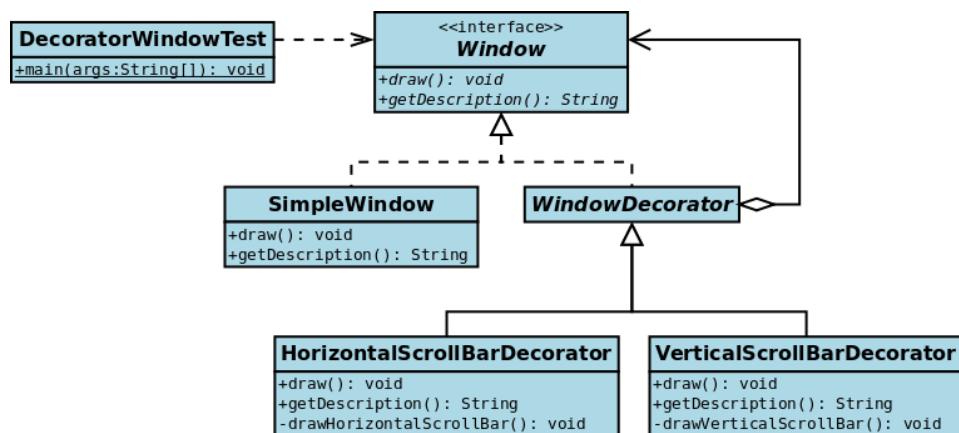


Figura 55: Esempio di applicazione del design pattern decorator

- **Scopo:** Aggiungere responsabilità e funzionalità dinamicamente ad oggetti già esistenti;
- **Motivazione:** Permettere l'estendibilità di oggetti e affinare le funzionalità offerte modificandole o aggiungendone di nuove in modo trasparente.
- **Applicabilità:** Questo design pattern_G può essere applicato nei seguenti casi:
 - Quanto si presenta la necessità di riutilizzare un oggetto per le sue funzionalità, necessitando tuttavia di altre aggiuntive.



A.4 Design Pattern Comportamentali

A.4.1 Command

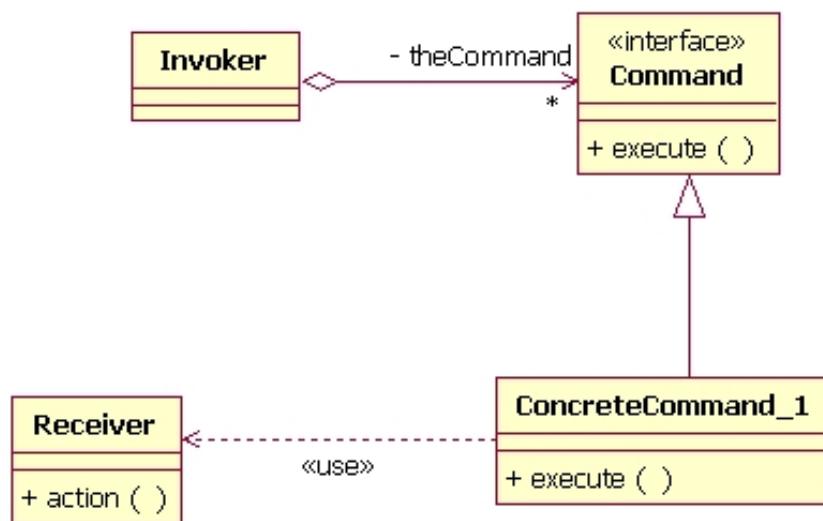


Figura 56: Esempio di applicazione del design pattern command

- **Scopo:** Incapsulare una richiesta in un oggetto, cosicché il client_G sia indipendente dalle richieste, poiché le richieste hanno al loro interno tutti i dati necessari per essere risolte.
- **Motivazione:** C'è la necessità di gestire richieste di cui non si conoscono i particolari poiché i Toolkit associano ai propri elementi, richieste da eseguire. Una classe astratta, **Command**, definisce l'interfaccia per eseguire la richiesta che è un semplice oggetto, per cui è possibile rendere variabile la reazione del client_G senza conoscere i dettagli dell'operazione stessa.
- **Applicabilità:** Il Command pattern si presta bene alla parametrizzazione di oggetti sull'azione da eseguire (Callback function) soprattutto nello specificare, accodare ed eseguire richieste molteplici volte. Vi è inoltre il Supporto alle operazioni di Undo e Redo. Vi è inoltre il supporto a transazione in quanto è possibile encapsulare un'azione in una operazione atomica.



A.4.2 Observer

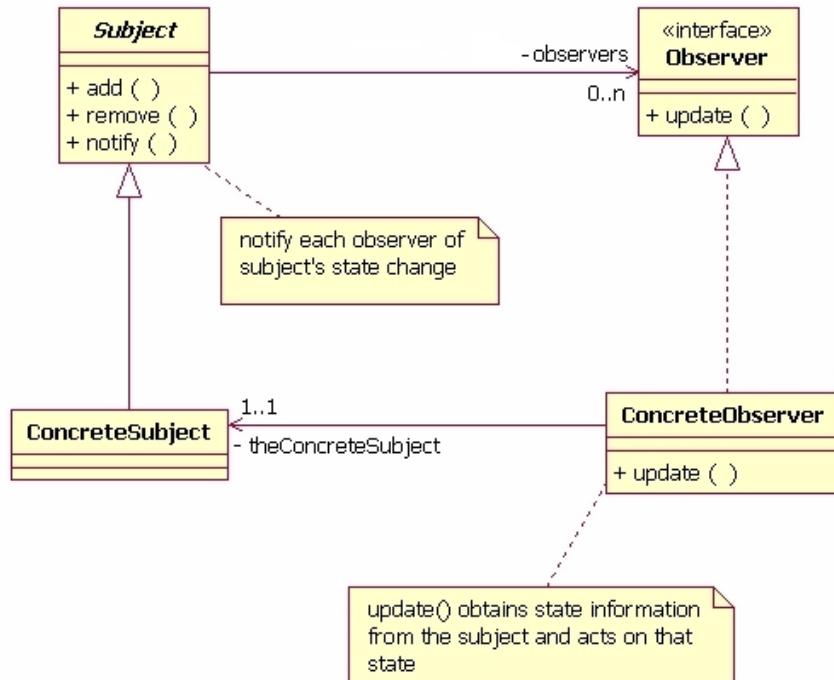


Figura 57: Esempio di applicazione del design pattern observer

- **Scopo:** Definisce una dipendenza “1..n” fra oggetti, riflettendo la modifica di un oggetto su quelli che ad esso dipendono.
- **Motivazione:** Mantenere la consistenza fra oggetti, oltre che al modello e alle viste ad esso collegate. Observer pattern definisce come implementare la relazione di dipendenza: infatti il modello prevede due attori principali che sono:
 - Subject, che effettua le notifiche e contiene le interfacce per registrare e rimuovere gli observer;
 - Observer, che si aggiorna in base alle notifiche che riceve dal subject.
- **Applicabilità:** È utile nei casi in cui si debba associare più “viste” differenti ad una astrazione. Il pattern comporta inoltre un aumento del grado di riuso dei singoli tipi; È indicato inoltre quando il cambiamento di un oggetto richiede il cambiamento di altri oggetti, oppure non si conosce quanti oggetti devono cambiare. È utile anche per notificare oggetti senza fare assunzioni su quali siano questi oggetti ed evita l'accoppiamento “forte”.

Elenco delle tabelle

Elenco delle figure

1	Diagramma architettura concettuale	5
2	Diagramma componenti principali	6
3	Actorsystem, visione generale	7
4	Package actorsystem::utils	9
5	Package _G actors, visione generale	12
6	ACtorsystem: package Clientactor	13
7	httpserver, visione generale del package	16
8	warehouseman, visione generale del package	19
9	Actorsystem: package messages del warehouseman	21
10	Package dell'attore di tipo Main	22
11	Userfinder, visione generale del package	25
12	Storefinder, visione generale del package	27
13	Storekeeper, visione generale del package	29
14	Manager, visione generale del package	31
15	Package _G messages	33
16	Package actorsystem::messages::authactormessages	34
17	Package actorsystem::messages::clientactormessages	38
18	Package actorsystem::messages::mainmessages	40
19	Package actorsystem::messages::storefindermessages	44
20	Package actorsystem::messages::storekeepermessages	47
21	Package actorsystem::messages::warehousemanmessages	50
22	Package actorsystem::messages::managermessages	52
23	CLI:architettura MVC variante push model	53
24	Driver: Packager client	56
25	Driver: Package api e interazioni con package client	59
26	Driver: Package data e interazioni con package padre	64
27	Driver: Package exceptions	66
28	CLI, architettura MVC variante push model	71
29	CLI, views package, interazioni con controllers e models	74
30	CLI: package controllers e interazioni con package models e views	78
31	CLI: package models e interazioni con controllers e views	80
32	Diagrammi attività - Visione generale	91
33	Diagrammi attività - Creazione collezione	92
34	Diagrammi attività - Cancellazione collezione	93
35	Diagrammi attività - Visualizzazione collezioni	94
36	Diagrammi attività - Modifica nome collezione	95
37	Diagrammi attività - Inserimento item	96
38	Diagrammi attività - Rimozione item	97
39	Diagrammi attività - Aggiunta collaboratore	98
40	Diagrammi attività - Rimozione collaboratore	99
41	Diagrammi attività - Import	100
42	Diagrammi attività - Interrogazione del database _G	101
43	Diagrammi attività - Modifica password	102



44	Diagrammi attività - Gestione utenti	103
45	MVC pattern in actorbase	105
46	Connection Singleton	106
47	Builder pattern in actorbase	107
48	Decorator pattern in actorbase	108
49	Command pattern in actorbase	109
50	Observer pattern in actorbase	110
51	Esempio di applicazione del design pattern MVC	137
52	Esempio di applicazione del design pattern Dependency Injection	138
53	Esempio di applicazione del design pattern singleton	139
54	Esempio di applicazione del design pattern builder	139
55	Esempio di applicazione del design pattern decorator	140
56	Esempio di applicazione del design pattern command	141
57	Esempio di applicazione del design pattern observer	142