Specifica Tecnica

Informazioni sul documento	
Versione	1.0.0
Redazione	Alberto De Agostini
	Andrea Giacomo Baldan
	Davide Trevisan
	Francesco Agostini
	Giacomo Vanin
	Marco Boseggia
Verifica	Alberto De Agostini
	Andrea Giacomo Baldan
	Davide Trevisan
	Michael Munaro
Approvazione	Marco Boseggia
Uso	Esterno
Lista di Distribuzione	ScalateKids
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Diario delle modifiche

Versione	Autore	Ruolo	Data	Descrizione
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 **7**
- 4.1 actorbase . . . 8
  - 4.1.1 Descrizione . . . 9
  - 4.1.2 Package contenuti . . . 9
- 4.2 actorbase::actorsystem . . . 10
  - 4.2.1 Descrizione . . . 11
  - 4.2.2 Package contenuti . . . 11
  - 4.2.3 Interazioni con altre componenti . . . 11
- 4.3 actorbase::actorsystem::clientactor . . . 12
  - 4.3.1 Descrizione . . . 12
  - 4.3.2 Package contenuti . . . 12
  - 4.3.3 Classi . . . 12
    - 4.3.3.1 actorbase::actorsystem::clientactor::ClientActor . . . 12
      - 4.3.3.1.1 Descrizione . . . 12
      - 4.3.3.1.2 Utilizzo . . . 13
      - 4.3.3.1.3 Classi ereditate . . . 13
      - 4.3.3.1.4 Interazioni con altre classi . . . 13
- 4.4 actorbase::actorsystem::clientactor::messages . . . 13
  - 4.4.1 Descrizione . . . 13
  - 4.4.2 Classi . . . 13
    - 4.4.2.1 actorbase::actorsystem::clientactor::messages::LoginResponse . . . 13
      - 4.4.2.1.1 Descrizione . . . 13
      - 4.4.2.1.2 Utilizzo . . . 13
    - 4.4.2.2 actorbase::actorsystem::clientactor::messages::PeerClosed . . . 13
      - 4.4.2.2.1 Descrizione . . . 13
      - 4.4.2.2.2 Utilizzo . . . 14

4.4.2.3 actorbase::actorsystem::clientactor::messages::RESPInput . . . 14
4.4.2.3.1 Descrizione . . . 14
4.4.2.3.2 Utilizzo . . . 14
4.4.2.4 actorbase::actorsystem::clientactor::messages::UpdateReadCollections . . . 14
4.4.2.4.1 Descrizione . . . 14
4.4.2.4.2 Utilizzo . . . 14
4.4.2.5 actorbase::actorsystem::clientactor::messages::UpdateCollections . . . 14
4.4.2.5.1 Descrizione . . . 14
4.4.2.5.2 Utilizzo . . . 14
4.4.2.6 actorbase::actorsystem::clientactor::messages::Response . . . 15
4.4.2.6.1 Descrizione . . . 15
4.4.2.6.2 Utilizzo . . . 15
4.5 actorbase::actorsystem::tcpserver . . . 15
4.5.1 Descrizione . . . 15
4.5.2 Package contenuti . . . 16
4.5.3 Classi . . . 16
4.5.3.1 actorbase::actorsystem::tcpserver::TCPServer . . . 16
4.5.3.1.1 Descrizione . . . 16
4.5.3.1.2 Utilizzo . . . 16
4.5.3.1.3 Classi ereditate . . . 16
4.6 actorbase::actorsystem::tcpserver::messages . . . 16
4.6.1 Descrizione . . . 16
4.6.2 Classi . . . 16
4.6.2.1 actorbase::actorsystem::tcpserver::messages::Bound . . . 16
4.6.2.1.1 Descrizione . . . 16
4.6.2.1.2 Utilizzo . . . 16
4.6.2.2 actorbase::actorsystem::tcpserver::messages::CommandFailed . . . 17
4.6.2.2.1 Descrizione . . . 17
4.6.2.2.2 Utilizzo . . . 17
4.6.2.3 actorbase::actorsystem::tcpserver::messages::Connected . . . 17
4.6.2.3.1 Descrizione . . . 17
4.6.2.3.2 Utilizzo . . . 17
4.7 actorbase::actorsystem::serialization . . . 18
4.7.1 Descrizione . . . 18
4.7.2 Interazioni con altre componenti . . . 18
4.7.3 Classi . . . 19
4.7.3.1 actorbase::actorsystem::serialization::SerializationContext . . . 19
4.7.3.1.1 Descrizione . . . 19
4.7.3.1.2 Utilizzo . . . 19
4.7.3.1.3 Interazioni con altre classi . . . 19
4.7.3.2 actorbase::actorsystem::serialization::SerializationStrategy . . . 19
4.7.3.2.1 Descrizione . . . 19
4.7.3.2.2 Utilizzo . . . 19
4.7.3.2.3 Interazioni con altre classi . . . 19
4.7.3.3 actorbase::actorsystem::serialization::RESPSerialization . . . 19

4.7.3.3.1 Descrizione . . . 19
4.7.3.3.2 Utilizzo . . . 20
4.7.3.3.3 Classi ereditate . . . 20
4.7.3.4 actorbase::actorsystem::serialization::PickleSerialization . . . 20
4.7.3.4.1 Descrizione . . . 20
4.7.3.4.2 Utilizzo . . . 20
4.7.3.4.3 Classi ereditate . . . 20
4.7.3.5 actorbase::actorsystem::serialization::DeserializationContext . . . 20
4.7.3.5.1 Descrizione . . . 20
4.7.3.5.2 Utilizzo . . . 20
4.7.3.5.3 Interazioni con altre classi . . . 20
4.7.3.6 actorbase::actorsystem::serialization::DeserializationStrategy . . . 21
4.7.3.6.1 Descrizione . . . 21
4.7.3.6.2 Utilizzo . . . 21
4.7.3.6.3 Interazioni con altre classi . . . 21
4.7.3.7 actorbase::actorsystem::serialization::RESPDeserialization . . . 21
4.7.3.7.1 Descrizione . . . 21
4.7.3.7.2 Utilizzo . . . 21
4.7.3.7.3 Classi ereditate . . . 21
4.7.3.8 actorbase::actorsystem::serialization::PickleDeserialization . . . 21
4.7.3.8.1 Descrizione . . . 21
4.7.3.8.2 Utilizzo . . . 21
4.7.3.8.3 Classi ereditate . . . 22
4.8 actorbase::actorsystem::warehouseman . . . 22
4.8.1 Descrizione . . . 22
4.8.2 Package contenuti . . . 22
4.8.3 Classi . . . 23
4.8.3.1 actorbase::actorsystem::warehouseman::Warehouseman . . . 23
4.8.3.1.1 Descrizione . . . 23
4.8.3.1.2 Utilizzo . . . 23
4.8.3.1.3 Classi ereditate . . . 23
4.8.3.1.4 Interazioni con altre classi . . . 23
4.9 actorbase::actorsystem::warehouseman::messages . . . 23
4.9.1 Descrizione . . . 23
4.9.1.1 actorbase::actorsystem::warehouseman::messages::Init . . . 23
4.9.1.1.1 Descrizione . . . 23
4.9.1.1.2 Utilizzo . . . 23
4.9.1.2 actorbase::actorsystem::warehouseman::messages::Save . . . 24
4.9.1.2.1 Descrizione . . . 24
4.9.1.2.2 Utilizzo . . . 24
4.10 actorbase::actorsystem::main . . . 24
4.10.1 Descrizione . . . 25
4.10.2 Package contenuti . . . 25
4.10.3 Classi . . . 25
4.10.3.1 actorbase::actorsystem::main::Main . . . 25

4.10.3.1.1 Descrizione . . . 25
4.10.3.1.2 Utilizzo . . . 25
4.10.3.1.3 Classi ereditate . . . 25
4.10.3.1.4 Interazioni con altre classi . . . 25
4.10.3.2 actorbase::actorsystem::main::SFRange . . . 25
4.10.3.2.1 Descrizione . . . 25
4.10.3.2.2 Utilizzo . . . 25
4.10.3.2.3 Interazioni con altre classi . . . 26
4.11 actorbase::actorsystem::main::messages . . . 26
4.11.1 Descrizione . . . 26
4.11.2 Package importati . . . 26
4.11.2.1 actorbase::actorsystem::main::messages::CreateCollection . . . 26
4.11.2.1.1 Descrizione . . . 26
4.11.2.1.2 Utilizzo . . . 26
4.11.2.2 actorbase::actorsystem::main::messages::GetCollection . . . 26
4.11.2.2.1 Descrizione . . . 26
4.11.2.2.2 Utilizzo . . . 26
4.11.2.3 actorbase::actorsystem::main::messages::RemoveCollection . . . 26
4.11.2.3.1 Descrizione . . . 26
4.11.2.3.2 Utilizzo . . . 27
4.11.2.4 actorbase::actorsystem::main::messages::GetItemFrom . . . 27
4.11.2.4.1 Descrizione . . . 27
4.11.2.4.2 Utilizzo . . . 27
4.11.2.5 actorbase::actorsystem::main::messages::AddContributor . . . 27
4.11.2.5.1 Descrizione . . . 27
4.11.2.5.2 Utilizzo . . . 27
4.11.2.6 actorbase::actorsystem::main::messages::DuplicateRequestSF . . . 27
4.11.2.6.1 Descrizione . . . 27
4.11.2.6.2 Utilizzo . . . 27
4.11.2.7 actorbase::actorsystem::main::messages::RemoveContributor . . . 28
4.11.2.7.1 Descrizione . . . 28
4.11.2.7.2 Utilizzo . . . 28
4.11.2.8 actorbase::actorsystem::main::messages::InitUserkeeper . . . 28
4.11.2.8.1 Descrizione . . . 28
4.11.2.8.2 Utilizzo . . . 28
4.11.2.9 actorbase::actorsystem::storefinder::messages::Init . . . 28
4.11.2.9.1 Descrizione . . . 28
4.11.2.9.2 Utilizzo . . . 28
4.11.2.10 actorbase::actorsystem::storefinder::messages::DuplicateRequest . . . 28
4.11.2.10.1 Descrizione . . . 28
4.11.2.10.2 Utilizzo . . . 29
4.11.2.11 actorbase::actorsystem::storefinder::messages::GetItem . . . 29
4.11.2.11.1 Descrizione . . . 29
4.11.2.11.2 Utilizzo . . . 29
4.11.2.12 actorbase::actorsystem::storefinder::messages::RemoveItem . . . 29

4.11.2.12.1 Descrizione . . . 29
4.11.2.12.2 Utilizzo . . . 29
4.11.2.13 actorbase::actorsystem::storefinder::messages::Insert . . . 29
4.11.2.13.1 Descrizione . . . 29
4.11.2.13.2 Utilizzo . . . 29
4.12 actorbase::actorsystem::userkeeper . . . 30
4.12.1 Descrizione . . . 30
4.12.2 Package contenuti . . . 30
4.12.3 Classi . . . 30
4.12.3.1 actorbase::actorsystem::userkeeper::Userkeeper . . . 30
4.12.3.1.1 Descrizione . . . 30
4.12.3.1.2 Utilizzo . . . 31
4.12.3.1.3 Classi ereditate . . . 31
4.12.3.1.4 Interazioni con altre classi . . . 31
4.13 actorbase::actorsystem::userkeeper::messages . . . 31
4.13.1 Descrizione . . . 31
4.13.2 Classi . . . 31
4.13.2.1 actorbase::actorsystem::userkeeper::messages::GetCollections . . . 31
4.13.2.1.1 Descrizione . . . 31
4.13.2.1.2 Utilizzo . . . 31
4.13.2.2 actorbase::actorsystem::userkeeper::messages::GetReadCollection . . . 31
4.13.2.2.1 Descrizione . . . 31
4.13.2.2.2 Utilizzo . . . 32
4.13.2.3 actorbase::actorsystem::userkeeper::messages::ChangePassword . . . 32
4.13.2.3.1 Descrizione . . . 32
4.13.2.3.2 Utilizzo . . . 32
4.13.2.4 actorbase::actorsystem::userkeeper::messages::GetPassword . . . 32
4.13.2.4.1 Descrizione . . . 32
4.13.2.4.2 Utilizzo . . . 32
4.13.2.5 actorbase::actorsystem::userkeeper::messages::RemoveCollection . . . 32
4.13.2.5.1 Descrizione . . . 32
4.13.2.5.2 Utilizzo . . . 32
4.13.2.6 actorbase::actorsystem::userkeeper::messages::AddCollection . . . 33
4.13.2.6.1 Descrizione . . . 33
4.13.2.6.2 Utilizzo . . . 33
4.13.2.7 actorbase::actorsystem::userkeeper::messages::RemoveReadCollection . . . 33
4.13.2.7.1 Descrizione . . . 33
4.13.2.7.2 Utilizzo . . . 33
4.13.2.8 actorbase::actorsystem::userkeeper::messages::AddReadCollection . . . 33
4.13.2.8.1 Descrizione . . . 33
4.13.2.8.2 Utilizzo . . . 33
4.13.2.9 actorbase::actorsystem::userkeeper::messages::BindClient . . . 33
4.13.2.9.1 Descrizione . . . 33
4.13.2.9.2 Utilizzo . . . 34
4.14 actorbase::actorsystem::storefinder . . . 34

4.14.1 Descrizione . . . 34
4.14.2 Package contenuti . . . 35
4.14.3 Classi . . . 35
4.14.3.1 actorbase::actorsystem::storefinder::Storefinder . . . 35
4.14.3.1.1 Descrizione . . . 35
4.14.3.1.2 Utilizzo . . . 35
4.14.3.1.3 Classi ereditate . . . 35
4.14.3.1.4 Interazioni con altre classi . . . 35
4.14.3.2 actorbase::actorsystem::storefinder::SKRange . . . 35
4.14.3.2.1 Descrizione . . . 35
4.14.3.2.2 Utilizzo . . . 35
4.14.3.2.3 Interazioni con altre classi . . . 35
4.15 actorbase::actorsystem::storefinder::messages . . . 36
4.15.1 Descrizione . . . 36
4.15.2 Classi . . . 36
4.15.2.1 actorbase::actorsystem::storefinder::messages::Init . . . 36
4.15.2.1.1 Descrizione . . . 36
4.15.2.1.2 Utilizzo . . . 36
4.15.2.2 actorbase::actorsystem::storefinder::messages::DuplicateRequest . . . 36
4.15.2.2.1 Descrizione . . . 36
4.15.2.2.2 Utilizzo . . . 36
4.15.2.3 actorbase::actorsystem::storefinder::messages::GetItem . . . 36
4.15.2.3.1 Descrizione . . . 36
4.15.2.3.2 Utilizzo . . . 36
4.15.2.4 actorbase::actorsystem::storefinder::messages::RemoveItem . . . 37
4.15.2.4.1 Descrizione . . . 37
4.15.2.4.2 Utilizzo . . . 37
4.15.2.5 actorbase::actorsystem::storefinder::messages::Insert . . . 37
4.15.2.5.1 Descrizione . . . 37
4.15.2.5.2 Utilizzo . . . 37
4.16 actorbase::actorsystem::storekeeper . . . 38
4.16.1 Descrizione . . . 38
4.16.2 Package contenuti . . . 38
4.16.3 Classi . . . 39
4.16.3.1 actorbase::actorsystem::storekeeper::Storekeeper . . . 39
4.16.3.1.1 Descrizione . . . 39
4.16.3.1.2 Utilizzo . . . 39
4.16.3.1.3 Classi ereditate . . . 39
4.16.3.1.4 Interazioni con altre classi . . . 39
4.17 actorbase::actorsystem::storekeeper::messages . . . 39
4.17.1 Descrizione . . . 39
4.17.2 Classi . . . 39
4.17.2.1 actorbase::actorsystem::storekeeper::messages::Init . . . 39
4.17.2.1.1 Descrizione . . . 39
4.17.2.1.2 Utilizzo . . . 39

4.17.2.2 actorbase::actorsystem::storekeeper::messages::GetItem . . . 40
4.17.2.2.1 Descrizione . . . 40
4.17.2.2.2 Utilizzo . . . 40
4.17.2.3 actorbase::actorsystem::storekeeper::messages::GetAllItem . . . 40
4.17.2.3.1 Descrizione . . . 40
4.17.2.3.2 Utilizzo . . . 40
4.17.2.4 actorbase::actorsystem::storekeeper::messages::Insert . . . 40
4.17.2.4.1 Descrizione . . . 40
4.17.2.4.2 Utilizzo . . . 40
4.17.2.5 actorbase::actorsystem::storekeeper::messages::RemoveItem . . . 40
4.17.2.5.1 Descrizione . . . 40
4.17.2.5.2 Utilizzo . . . 40
4.18 actorbase::actorsystem::ninja . . . 41
4.18.1 Descrizione . . . 41
4.18.2 Package contenuti . . . 41
4.18.3 Classi . . . 42
4.18.3.1 actorbase::actorsystem::ninja::Ninja . . . 42
4.18.3.1.1 Descrizione . . . 42
4.18.3.1.2 Utilizzo . . . 42
4.18.3.1.3 Classi ereditate . . . 42
4.18.3.1.4 Interazioni con altre classi . . . 42
4.19 actorbase::actorsystem::ninja::messages . . . 42
4.19.0.0.1 Descrizione . . . 42
4.19.0.1 actorbase::actorsystem::ninja::messages::Update . . . 42
4.19.0.1.1 Descrizione . . . 42
4.19.0.1.2 Utilizzo . . . 42
4.19.0.2 actorbase::actorsystem::ninja::messages::BecomeSK . . . 42
4.19.0.2.1 Descrizione . . . 42
4.19.0.2.2 Utilizzo . . . 43
4.20 actorbase::actorsystem::manager . . . 43
4.20.1 Descrizione . . . 43
4.20.2 Package contenuti . . . 43
4.20.3 Classi . . . 43
4.20.3.1 actorbase::actorsystem::manager::Manager . . . 43
4.20.3.1.1 Descrizione . . . 43
4.20.3.1.2 Utilizzo . . . 44
4.20.3.1.3 Classi ereditate . . . 44
4.20.3.1.4 Interazioni con altre classi . . . 44
4.21 actorbase::actorsystem::manager::messages . . . 44
4.21.0.0.1 Descrizione . . . 44
4.21.0.1 actorbase::actorsystem::manager::messages::DuplicationRequestSK . . . 44
4.21.0.1.1 Descrizione . . . 44
4.21.0.1.2 Utilizzo . . . 44
4.21.0.2 actorbase::actorsystem::manager::messages::DuplicationRequestSF . . . 44
4.21.0.2.1 Descrizione . . . 44

4.21.0.2.2 Utilizzo . . . 45
4.22 actorbase::driver . . . 45
4.22.1 Descrizione . . . 45
4.22.2 Interazioni con altre componenti . . . 45
4.22.3 Package contenuti . . . 46
4.23 actorbase::driver::client . . . 46
4.23.1 Descrizione . . . 46
4.23.2 Interazioni con altre componenti . . . 47
4.23.3 Classi . . . 47
4.23.3.1 actorbase::driver::client::ActorbaseClient . . . 47
4.23.3.1.1 Descrizione . . . 47
4.23.3.1.2 Utilizzo . . . 47
4.23.3.1.3 Interazioni con altre classi . . . 47
4.23.3.2 actorbase::driver::client::CommandRunner . . . 47
4.23.3.2.1 Descrizione . . . 47
4.23.3.2.2 Utilizzo . . . 47
4.23.3.2.3 Interazioni con altre classi . . . 48
4.23.3.3 actorbase::driver::client::Connection . . . 48
4.23.3.3.1 Descrizione . . . 48
4.23.3.3.2 Utilizzo . . . 48
4.23.3.3.3 Interazioni con altre classi . . . 48
4.24 actorbase::driver::serializer . . . 49
4.24.1 Descrizione . . . 49
4.24.2 Interazioni con altre componenti . . . 49
4.24.3 Classi . . . 50
4.24.3.1 actorbase::driver::serializer::SerializationContext . . . 50
4.24.3.1.1 Descrizione . . . 50
4.24.3.1.2 Utilizzo . . . 50
4.24.3.1.3 Interazioni con altre classi . . . 50
4.24.3.2 actorbase::driver::serializer::SerializationStrategy . . . 50
4.24.3.2.1 Descrizione . . . 50
4.24.3.2.2 Utilizzo . . . 50
4.24.3.2.3 Interazioni con altre classi . . . 50
4.24.3.3 actorbase::driver::serializer::RESPSerialize . . . 50
4.24.3.3.1 Descrizione . . . 50
4.24.3.3.2 Utilizzo . . . 51
4.24.3.3.3 Classi ereditate . . . 51
4.24.3.4 actorbase::driver::serializer::DeserializationContext . . . 51
4.24.3.4.1 Descrizione . . . 51
4.24.3.4.2 Utilizzo . . . 51
4.24.3.4.3 Interazioni con altre classi . . . 51
4.24.3.5 actorbase::driver::serializer::DeserializationStrategy . . . 51
4.24.3.5.1 Descrizione . . . 51
4.24.3.5.2 Utilizzo . . . 51
4.24.3.5.3 Interazioni con altre classi . . . 51

4.24.3.6 actorbase::driver::serializer::RESPDeserialize . . . 52
4.24.3.6.1 Descrizione . . . 52
4.24.3.6.2 Utilizzo . . . 52
4.24.3.6.3 Classi ereditate . . . 52
4.24.3.6.4 Interazioni con altre classi . . . 52
4.25 actorbase::driver::actorbasedata . . . 52
4.25.1 Descrizione . . . 53
4.25.2 Interazioni con altre componenti . . . 53
4.25.3 Classi . . . 53
4.25.3.1 actorbase::driver::actorbasedata::ActorbaseObject . . . 53
4.25.3.1.1 Descrizione . . . 53
4.25.3.1.2 Utilizzo . . . 53
4.25.3.1.3 Interazioni con altre classi . . . 53
4.25.3.2 actorbase::driver::actorbasedata::ActorbaseItem . . . 53
4.25.3.2.1 Descrizione . . . 53
4.25.3.2.2 Utilizzo . . . 53
4.25.3.2.3 Classi ereditate . . . 54
4.25.3.3 actorbase::driver::actorbasedata::ActorbaseObjectSeq . . . 54
4.25.3.3.1 Descrizione . . . 54
4.25.3.3.2 Utilizzo . . . 54
4.25.3.3.3 Interazioni con altre classi . . . 54
4.25.3.3.4 Classi ereditate . . . 54
4.25.3.4 actorbase::driver::actorbasedata::ActorbaseCollection . . . 54
4.25.3.4.1 Descrizione . . . 54
4.25.3.4.2 Utilizzo . . . 54
4.25.3.4.3 Classi ereditate . . . 54
4.25.3.4.4 Interazioni con altre classi . . . 55
4.25.3.5 actorbase::driver::actorbasedata::ActorbaseCollectionSeq . . . 55
4.25.3.5.1 Descrizione . . . 55
4.25.3.5.2 Utilizzo . . . 55
4.25.3.5.3 Classi ereditate . . . 55
4.25.3.5.4 Interazioni con altre classi . . . 55
4.25.3.6 actorbase::driver::actorbasedata::Cursor . . . 55
4.25.3.6.1 Descrizione . . . 55
4.25.3.6.2 Utilizzo . . . 55
4.25.3.7 actorbase::driver::actorbasedata::ItemCursor . . . 56
4.25.3.7.1 Descrizione . . . 56
4.25.3.7.2 Utilizzo . . . 56
4.25.3.7.3 Classi ereditate . . . 56
4.25.3.7.4 Interazioni con altre classi . . . 56
4.25.3.8 actorbase::driver::actorbasedata::CollectionCursor . . . 56
4.25.3.8.1 Descrizione . . . 56
4.25.3.8.2 Utilizzo . . . 56
4.25.3.8.3 Classi ereditate . . . 56
4.25.3.8.4 Interazioni con altre classi . . . 56

4.25.3.9 actorbase::driver::actorbasedata::RESPArrayString . . . 57
4.25.3.9.1 Descrizione . . . 57
4.25.3.9.2 Utilizzo . . . 57
4.25.3.9.3 Interazioni con altre classi . . . 57
4.25.3.10 actorbase::driver::actorbasedata::RESPSimpleString . . . 57
4.25.3.10.1 Descrizione . . . 57
4.25.3.10.2 Utilizzo . . . 57
4.25.3.10.3 Interazioni con altre classi . . . 57
4.25.3.11 actorbase::driver::actorbasedata::RESPBulkString . . . 57
4.25.3.11.1 Descrizione . . . 57
4.25.3.11.2 Utilizzo . . . 58
4.25.3.11.3 Classi ereditate . . . 58
4.25.3.12 actorbase::driver::actorbasedata::RESPErrorString . . . 58
4.25.3.12.1 Descrizione . . . 58
4.25.3.12.2 Utilizzo . . . 58
4.25.3.12.3 Classi ereditate . . . 58
4.26 actorbase::driver::exceptions . . . 58
4.26.1 Descrizione . . . 59
4.26.2 Interazione con altre componenti . . . 59
4.26.3 Classi . . . 59
4.26.3.1 actorbase::driver::exceptions::WrongCredentialExc . . . 59
4.26.3.1.1 Descrizione . . . 59
4.26.3.1.2 Utilizzo . . . 59
4.26.3.1.3 Interazioni con altre classi . . . 59
4.26.3.2 actorbase::driver::exceptions::WrongPasswordExc . . . 59
4.26.3.2.1 Descrizione . . . 59
4.26.3.2.2 Utilizzo . . . 59
4.26.3.2.3 Interazioni con altre classi . . . 59
4.26.3.3 actorbase::driver::exceptions::WrongNewPasswordExc . . . 60
4.26.3.3.1 Descrizione . . . 60
4.26.3.3.2 Utilizzo . . . 60
4.26.3.3.3 Interazioni con altre classi . . . 60
4.26.3.4 actorbase::driver::exceptions::CollectionAlreadyExistsExc . . . 60
4.26.3.4.1 Descrizione . . . 60
4.26.3.4.2 Utilizzo . . . 60
4.26.3.4.3 Interazioni con altre classi . . . 60
4.26.3.5 actorbase::driver::exceptions::UndefinedCollectionExc . . . 60
4.26.3.5.1 Descrizione . . . 60
4.26.3.5.2 Utilizzo . . . 60
4.26.3.5.3 Interazioni con altre classi . . . 61
4.26.3.6 actorbase::driver::exceptions::UndefinedUsernameExc . . . 61
4.26.3.6.1 Descrizione . . . 61
4.26.3.6.2 Utilizzo . . . 61
4.26.3.6.3 Interazioni con altre classi . . . 61
4.26.3.7 actorbase::driver::exceptions::UsernameAlreadyExistsExc . . . 61

4.26.3.7.1 Descrizione . . . 61
4.26.3.7.2 Utilizzo . . . 61
4.26.3.7.3 Interazioni con altre classi . . . 61
4.26.3.8 actorbase::driver::exceptions::DuplicateKeyExc . . . 61
4.26.3.8.1 Descrizione . . . 61
4.26.3.8.2 Utilizzo . . . 62
4.26.3.8.3 Interazioni con altre classi . . . 62
4.26.3.9 actorbase::driver::exceptions::UndefinedFileExc . . . 62
4.26.3.9.1 Descrizione . . . 62
4.26.3.9.2 Utilizzo . . . 62
4.26.3.9.3 Interazioni con altre classi . . . 62
4.26.3.10 actorbase::driver::exceptions::MalformedFileExc . . . 62
4.26.3.10.1 Descrizione . . . 62
4.26.3.10.2 Utilizzo . . . 62
4.26.3.10.3 Interazioni con altre classi . . . 62
4.27 actorbase::cli . . . 63
4.27.1 Descrizione . . . 63
4.27.2 Interazioni con altre componenti . . . 63
4.27.3 Package contenuti . . . 63
4.28 actorbase::cli::views . . . 64
4.28.1 Descrizione . . . 64
4.28.2 Interazioni con altre componenti . . . 64
4.28.3 Classi . . . 64
4.28.3.1 actorbase::cli::views::CommandLoop . . . 64
4.28.3.1.1 Descrizione . . . 64
4.28.3.1.2 Utilizzo . . . 65
4.28.3.1.3 Interazioni con altre classi . . . 65
4.28.3.2 actorbase::cli::views::ActorbaseBanner . . . 65
4.28.3.2.1 Descrizione . . . 65
4.28.3.2.2 Utilizzo . . . 65
4.28.3.2.3 Interazioni con altre classi . . . 65
4.28.3.3 actorbase::cli::views::PromptProvider . . . 65
4.28.3.3.1 Descrizione . . . 65
4.28.3.3.2 Utilizzo . . . 65
4.28.3.3.3 Interazioni con altre classi . . . 65
4.28.3.4 actorbase::cli::views::ActorbasePrompt . . . 66
4.28.3.4.1 Descrizione . . . 66
4.28.3.4.2 Utilizzo . . . 66
4.28.3.4.3 Classi ereditate . . . 66
4.28.3.5 actorbase::cli::views::Observer . . . 66
4.28.3.5.1 Descrizione . . . 66
4.28.3.5.2 Utilizzo . . . 66
4.28.3.5.3 Interazioni con altre classi . . . 66
4.28.3.6 actorbase::cli::views::ResultView . . . 66
4.28.3.6.1 Descrizione . . . 66

4.28.3.6.2 Utilizzo . . . 66
4.28.3.6.3 Classi ereditate . . . 67
4.28.3.6.4 Interazioni con altre classi . . . 67
4.29 actorbase::cli::controllers . . . 67
4.29.1 Descrizione . . . 67
4.29.2 Interazioni con altre componenti . . . 67
4.29.3 Classi . . . 67
4.29.3.1 actorbase::cli::controllers::GrammarParser . . . 67
4.29.3.1.1 Descrizione . . . 67
4.29.3.1.2 Utilizzo . . . 67
4.29.3.1.3 Interazioni con altre classi . . . 67
4.30 actorbase::cli::models . . . 68
4.30.1 Descrizione . . . 68
4.30.2 Classi . . . 68
4.30.2.1 actorbase::cli::models::Observable . . . 68
4.30.2.1.1 Descrizione . . . 68
4.30.2.1.2 Utilizzo . . . 68
4.30.2.1.3 Interazioni con altre classi . . . 68
4.30.2.2 actorbase::cli::models::CommandInvoker . . . 68
4.30.2.2.1 Descrizione . . . 68
4.30.2.2.2 Utilizzo . . . 68
4.30.2.2.3 Classi ereditate . . . 68
4.30.2.2.4 Interazioni con altre classi . . . 69
4.30.2.3 actorbase::cli::models::Command . . . 69
4.30.2.3.1 Descrizione . . . 69
4.30.2.3.2 Utilizzo . . . 69
4.30.2.3.3 Interazioni con altre classi . . . 69
4.30.2.4 actorbase::cli::models::FindCommand . . . 69
4.30.2.4.1 Descrizione . . . 69
4.30.2.4.2 Utilizzo . . . 69
4.30.2.4.3 Classi ereditate . . . 69
4.30.2.5 actorbase::cli::models::LogoutCommand . . . 69
4.30.2.5.1 Descrizione . . . 69
4.30.2.5.2 Utilizzo . . . 70
4.30.2.5.3 Classi ereditate . . . 70
4.30.2.6 actorbase::cli::models::ResetPasswordCommand . . . 70
4.30.2.6.1 Descrizione . . . 70
4.30.2.6.2 Utilizzo . . . 70
4.30.2.6.3 Classi ereditate . . . 70
4.30.2.7 actorbase::cli::models::ListCommand . . . 70
4.30.2.7.1 Descrizione . . . 70
4.30.2.7.2 Utilizzo . . . 70
4.30.2.7.3 Classi ereditate . . . 70
4.30.2.8 actorbase::cli::models::AddContributorCommand . . . 71
4.30.2.8.1 Descrizione . . . 71

4.30.2.8.2 Utilizzo . . . 71
4.30.2.8.3 Classi ereditate . . . 71
4.30.2.9 actorbase::cli::models::LoginCommand . . . 71
4.30.2.9.1 Descrizione . . . 71
4.30.2.9.2 Utilizzo . . . 71
4.30.2.9.3 Classi ereditate . . . 71
4.30.2.10 actorbase::cli::models::RemoveCollectionCommand . . . 71
4.30.2.10.1 Descrizione . . . 71
4.30.2.10.2 Utilizzo . . . 71
4.30.2.10.3 Classi ereditate . . . 72
4.30.2.11 actorbase::cli::models::RemoveContributorCommand . . . 72
4.30.2.11.1 Descrizione . . . 72
4.30.2.11.2 Utilizzo . . . 72
4.30.2.11.3 Classi ereditate . . . 72
4.30.2.12 actorbase::cli::models::RenameCollectionCommand . . . 72
4.30.2.12.1 Descrizione . . . 72
4.30.2.12.2 Utilizzo . . . 72
4.30.2.12.3 Classi ereditate . . . 72
4.30.2.13 actorbase::cli::models::AddUserCommand . . . 72
4.30.2.13.1 Descrizione . . . 72
4.30.2.13.2 Utilizzo . . . 73
4.30.2.13.3 Classi ereditate . . . 73
4.30.2.14 actorbase::cli::models::HelpCommand . . . 73
4.30.2.14.1 Descrizione . . . 73
4.30.2.14.2 Utilizzo . . . 73
4.30.2.15 actorbase::cli::models::RemoveItemCommand . . . 73
4.30.2.15.1 Descrizione . . . 73
4.30.2.15.2 Utilizzo . . . 73
4.30.2.15.3 Classi ereditate . . . 73
4.30.2.16 actorbase::cli::models::RemoveUserCommand . . . 73
4.30.2.16.1 Descrizione . . . 73
4.30.2.16.2 Utilizzo . . . 74
4.30.2.16.3 Classi ereditate . . . 74
4.30.2.17 actorbase::cli::models::ImportCommand . . . 74
4.30.2.17.1 Descrizione . . . 74
4.30.2.17.2 Utilizzo . . . 74
4.30.2.17.3 Classi ereditate . . . 74
4.30.2.18 actorbase::cli::models::InsertItemCommand . . . 74
4.30.2.18.1 Descrizione . . . 74
4.30.2.18.2 Utilizzo . . . 74
4.30.2.18.3 Classi ereditate . . . 74
4.30.2.19 actorbase::cli::models::CreateCollectionCommand . . . 75
4.30.2.19.1 Descrizione . . . 75
4.30.2.19.2 Utilizzo . . . 75
4.30.2.19.3 Classi ereditate . . . 75

4.30.2.20 actorbase::cli::models::ExportCommand . . . 75
4.30.2.20.1 Descrizione . . . 75
4.30.2.20.2 Utilizzo . . . 75
4.30.2.20.3 Classi ereditate . . . 75
4.30.2.21 actorbase::cli::models::CommandReceiver . . . 75
4.30.2.21.1 Descrizione . . . 75
4.30.2.21.2 Utilizzo . . . 75
4.30.2.21.3 Interazioni con altre classi . . . 76

5 Diagrammi di Attività **77**
5.1 Visione generale . . . 77
5.2 Operazioni su collezioni e/o item . . . 79
5.2.1 Creazione collezione . . . 79
5.2.2 Cancellazione collezione . . . 79
5.2.3 Visualizza collezioni . . . 80
5.2.4 Modifica nome collezione . . . 81
5.2.5 Inserimento item . . . 82
5.2.6 Rimozione item . . . 83
5.2.7 Aggiunta collaboratore . . . 84
5.2.8 Rimozione collaboratore . . . 85
5.2.9 Import . . . 86
5.3 Interrogazione del database . . . 87
5.4 Modifica password . . . 88
5.5 Gestione utenti . . . 89

6 Diagrammi di Sequenza **91**
6.1 Interazioni generali tra componenti . . . 91
6.2 Inserimento di un Item . . . 92
6.3 Inserimento di un Item con Storekeeper pieno . . . 92
6.4 Autenticazione . . . 93
6.5 Salvataggio su filesystem . . . 94
6.6 Aggiornamento di un attore ninja . . . 95
6.7 Aggiunta collaboratore (in sola lettura) . . . 96
6.8 Connessione al server tramite Driver . . . 97

7 Design Pattern **99**
7.1 Design Pattern Architetturali . . . 99
7.1.1 MVC . . . 99
7.2 Design Pattern Creazionali . . . 100
7.2.1 Singleton . . . 100
7.3 Design Pattern Strutturali . . . 100
7.3.1 Facade . . . 100
7.4 Design Pattern Comportamentali . . . 100
7.4.1 Command Pattern . . . 100
7.4.2 Iterator Pattern . . . 101
7.4.3 Observer Pattern . . . 101

7.4.4 Strategy 102

8 Stime di fattibilità e di bisogno di risorse **104**

9 Tracciamento **105**
9.1 Tracciamento Componenti-Requisiti 105
9.2 Tracciamento Requisiti-Componenti 107

A Descrizione Design Pattern **119**
A.1 Design Pattern Architetturali 119
A.1.1 MVC 119
A.2 Design Pattern Creazionali 120
A.2.1 Singleton 120
A.3 Design Pattern Strutturali 121
A.3.1 Façade 121
A.4 Design Pattern Comportamentali 122
A.4.1 Command 122
A.4.2 Iterator 123
A.4.3 Observer 124
A.4.4 Strategy 125

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 fraintendibile e gli acronimi sono raccolti nel file Glossario v1.0.0; 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 v2.0.0.

1.4.2 Informativi

- **Piano di Progetto:** Piano di Progetto v2.0.0
- **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:** `http://lampwww.epfl.ch/~hmiller/pickling`

- **RESP - Redis Protocol Specification:** Protocollo di comunicazione testuale TCP$_G$ (Transmission Control Protocol) http://redis.io/topics/protocol

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$_G$;
- **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$_G$ TCP (Transmission Control 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 testuale RESP.

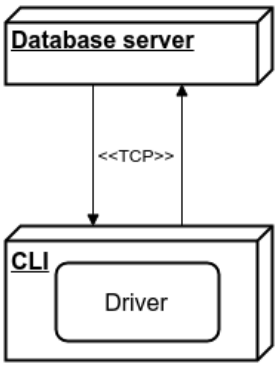

Figura 1: Diagramma architettura concettuale

3.3 Protocollo di comunicazione Client-Server

Actorbase è stato concepito come sistema da utilizzare in ambiente affidabile, e non dovrebbe essere esposto direttamente alla rete internet, ma utilizzato all'interno di sottoreti protette dall'esterno, per questo è stato

deciso di riutilizzare un protocollo di comunicazione piuttosto semplice, leggero e facilmente comprensibile.
Il protocollo **RESP**$_G$ (REdis Serialization Protocol) implementato dal database$_G$ REDIS$_G$ si presta esattamente allo scopo, utilizza 5 tipi di dato differenziati dal primo byte$_G$ che li compone, e delimitati da una sequenza di terminazione formata da CR$_G$ (Carriage Return) e LF$_G$ (Line Feed) che rappresenta il carattere "a capo":

- **Simple Strings** rappresentano stringhe dedicate alle risposte semplici (ack$_G$ dei comandi ricevuti), non possono contenere caratteri di terminazione al loro interno; il primo byte$_G$ è "+";
- **Errors** rappresentano stringhe di errore, formate da:

  ```
  -TIPOERRORE MESSAGGIOERRORE\r\n
  ```

  verranno utilizzate per generare le eccezioni lato client$_G$; il primo byte$_G$ è "-";
- **Integers** rappresentano numeri interi, il primo byte$_G$ è ":";
- **Bulk Strings** rappresentano stringhe dedicate al payload$_G$, formate da:

  ```
  \$LUNGHEZZAPAYLOAD\r\nPAYLOAD\r\n
  ```

  il primo byte$_G$ è "$" e può contenere caratteri di "a capo";
- **Arrays** rappresentano array$_G$ di comandi e **Bulk String**, formati da:

  ```
  *LUNGHEZZASTRINGA\r\nRESPSTRING\r\n
  ```

  il primo byte$_G$ è "*", possono contenere qualsiasi altro tipo di stringa RESP$_G$, inclusi altri array$_G$.

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

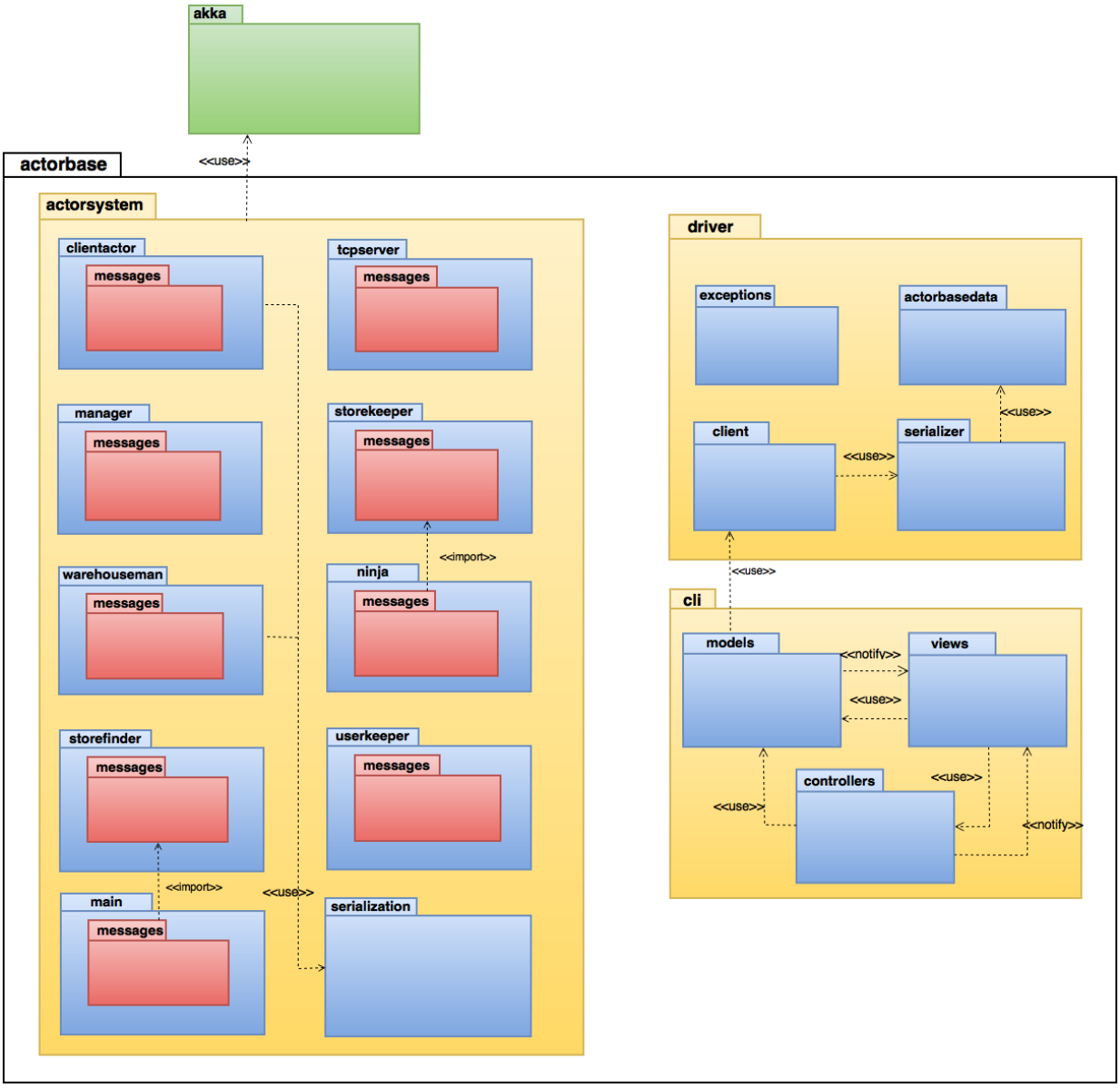

Figura 2: Diagramma componenti principali

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

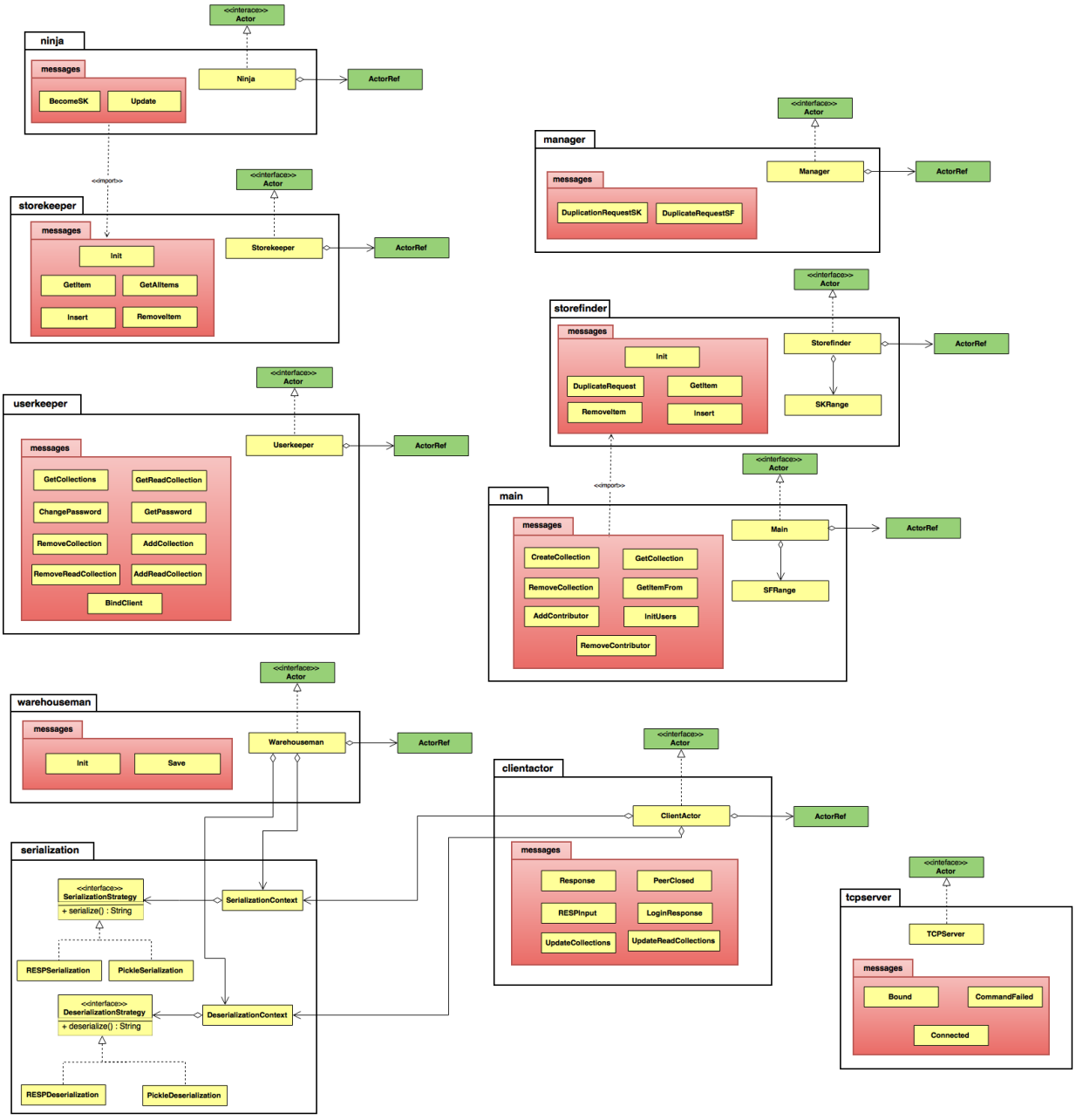

Figura 3: Actorsystem, visione generale

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::clientactor;
- actorbase::actorsystem::tcpserver;
- actorbase::actorsystem::storefinder;
- actorbase::actorsystem::storekeeper;
- actorbase::actorsystem::warehouseman;
- actorbase::actorsystem::ninja;
- actorbase::actorsystem::manager;
- actorbase::actorsystem::userkeeper;
- actorbase::actorsystem::main;
- actorbase::actorsystem::serialization.

4.2.3 Interazioni con altre componenti

- actorbase::driver;
- Akka.

4.3 actorbase::actorsystem::clientactor

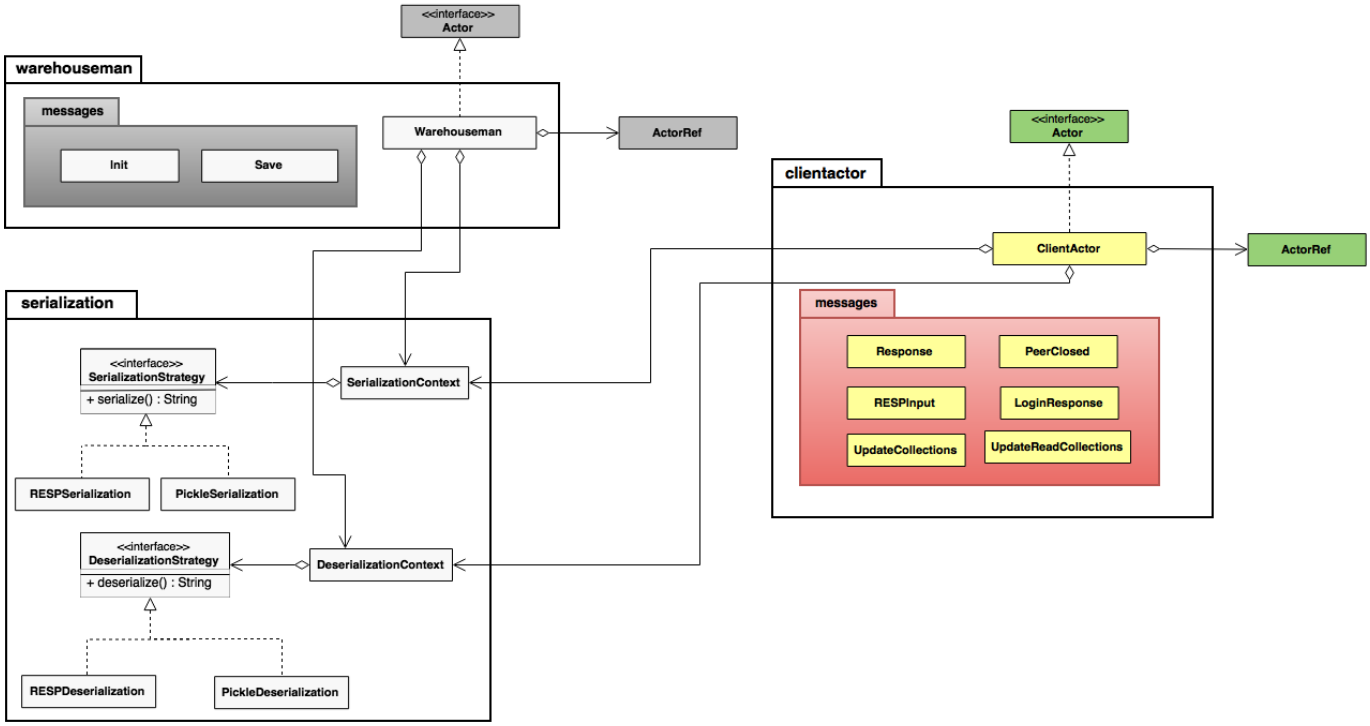

Figura 4: Clientactor e interazioni con main e package serialization

4.3.1 Descrizione

Package$_G$ per l'attore$_G$ con cui si interfaccerà il driver$_G$.

4.3.2 Package contenuti

- actorbase::actorsystem::clientactor::messages.

4.3.3 Classi

4.3.3.1 actorbase::actorsystem::clientactor::ClientActor

4.3.3.1.1 Descrizione

Classe che rappresenta l'attore$_G$ con cui si interfaccia il driver$_G$ dopo la connessione.

4.3.3.1.2 Utilizzo

Questo attore$_G$ riceve le richieste da actorbase::driver::client::Connection e si occupa di inviare messaggi a actorbase::actorsystem::main::Main.

4.3.3.1.3 Classi ereditate

- akka::actor::Actor.

4.3.3.1.4 Interazioni con altre classi

- actorbase::actorsystem::serialization::SerializationContext;
- actorbase::actorsystem::serialization::DeserializationContext;
- akka::actor::ActorRef.

4.4 actorbase::actorsystem::clientactor::messages

4.4.1 Descrizione

Package$_G$ che contiene tutti i messaggi che possono essere ricevuti da actorbase::actorsystem::clientactor::ClientActor.

4.4.2 Classi

4.4.2.1 actorbase::actorsystem::clientactor::messages::LoginResponse

4.4.2.1.1 Descrizione

Messaggio che contiene la password di un utente che ha richiesto il login.

4.4.2.1.2 Utilizzo

In seguito alla ricezione di questo messaggio l'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor controlla che la password ricevuta sia uguale a quella immessa dall'utente e, in caso affermativo, cambia il proprio stato e salva il riferimento dell'attore$_G$ di tipo actorbase::actorsystem::userkeeper::Userkeeper che aveva inviato questo messaggio.

4.4.2.2 actorbase::actorsystem::clientactor::messages::PeerClosed

4.4.2.2.1 Descrizione

Messaggio che indica la volontà di chiudere la connessione con il driver$_G$.

4.4.2.2.2 Utilizzo

In seguito alla ricezione di questo messaggio l'attore$_G$ di tipo actorbase::actorsystem::actorclient::ActorClient chiude la connessione con il driver$_G$ e termina.

4.4.2.3 actorbase::actorsystem::clientactor::messages::RESPInput

4.4.2.3.1 Descrizione

Messaggio che rappresenta l'input ricevuto dal driver$_G$.

4.4.2.3.2 Utilizzo

In seguito alla ricezione di questo messaggio l'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor deserializza il messaggio sfruttando la componente actorbase::actorsystem::serialization::DeserializationContext. Dopo aver fatto ciò crea il messaggio corrispondente al tipo di richiesta che gli era arrivato e lo manda all'attore$_G$ actorbase::actorsystem::main::Main.

4.4.2.4 actorbase::actorsystem::clientactor::messages::UpdateReadCollections

4.4.2.4.1 Descrizione

Messaggio che indica la necessità di aggiornare l'insieme delle collezioni$_G$ cui l'utente ha permessi di sola lettura.

4.4.2.4.2 Utilizzo

Quando actorbase::actorsystem::clientactor::ClientActor riceve questo messaggio aggiorna la sua lista di collezioni$_G$ cui l'utente ha permessi di sola lettura aggiungendo la collezione$_G$ presente nel messaggio.

4.4.2.5 actorbase::actorsystem::clientactor::messages::UpdateCollections

4.4.2.5.1 Descrizione

Messaggio che indica la necessità di aggiornare l'insieme delle collezioni$_G$ cui l'utente ha permessi di lettura e scrittura.

4.4.2.5.2 Utilizzo

Quando actorbase::actorsystem::clientactor::ClientActor riceve questo messaggio aggiorna la sua lista di collezioni$_G$ cui l'utente ha permessi di lettura e scrittura aggiungendo la collezione$_G$ presente nel messaggio.

4.4.2.6 actorbase::actorsystem::clientactor::messages::Response

4.4.2.6.1 Descrizione

Messaggio che rappresenta la risposta da parte del server$_G$ da inoltrare al driver$_G$.

4.4.2.6.2 Utilizzo

Quando actorbase::actorsystem::clientactor::ClientActor riceve questo messaggio il contenuto di quest'ultimo viene serializzato tramite actorbase::actorsystem::serialization::DeserializationContext e mandato a actorbase::driver::client::Connection.

4.5 actorbase::actorsystem::tcpserver

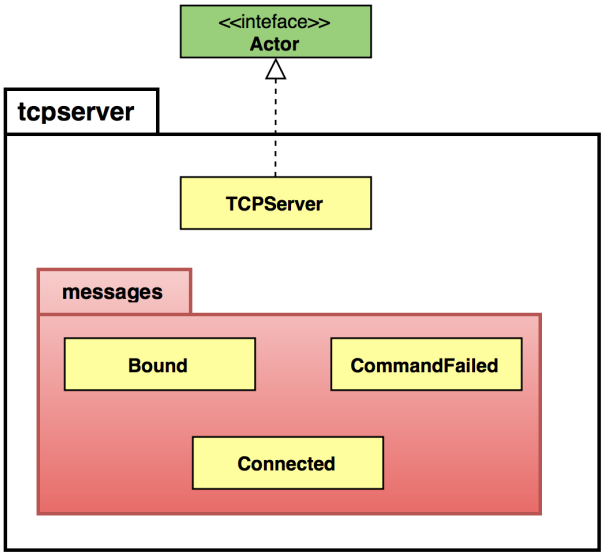

Figura 5: tcpserver, visione generale del package

4.5.1 Descrizione

Package$_G$ per l'attore$_G$ con cui si interfaccerà il driver$_G$ per la connessione iniziale.

4.5.2 Package contenuti

- actorbase::actorsystem::tcpserver::messages.

4.5.3 Classi

4.5.3.1 actorbase::actorsystem::tcpserver::TCPServer

4.5.3.1.1 Descrizione

Classe che rappresenta l'attore$_G$ con cui si interfaccia il driver$_G$ per istanziare la connessione.

4.5.3.1.2 Utilizzo

Questo attore$_G$ riceve la richiesta di connessione da actorbase::driver::client::Connection e si occupa di associare al client$_G$ un attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor per continuare le comunicazioni.

4.5.3.1.3 Classi ereditate

- akka::actor::Actor.

4.6 actorbase::actorsystem::tcpserver::messages

4.6.1 Descrizione

Package$_G$ che contiene tutti i messaggi che possono essere ricevuti da actorbase::actorsystem::tcpserver::TCPServer.

4.6.2 Classi

4.6.2.1 actorbase::actorsystem::tcpserver::messages::Bound

4.6.2.1.1 Descrizione

Messaggio che rappresenta la necessità di mettere il server$_G$ in ascolto.

4.6.2.1.2 Utilizzo

Quando actorbase::actorsystem::tcpserver::TCPServer riceve questo messaggio si mette in ascolto su una porta del server$_G$.

4.6.2.2 actorbase::actorsystem::tcpserver::messages::CommandFailed

4.6.2.2.1 Descrizione

Messaggio che rappresenta un errore nella comunicazione TCP_G.

4.6.2.2.2 Utilizzo

Quando actorbase::actorsystem::tcpserver::TCPServer riceve questo messaggio manderà una notifica a actorbase::driver::client::Connection.

4.6.2.3 actorbase::actorsystem::tcpserver::messages::Connected

4.6.2.3.1 Descrizione

Messaggio che rappresenta una connessione TCP_G riuscita.

4.6.2.3.2 Utilizzo

Quando actorbase::actorsystem::tcpserver::TCPServer riceve questo messaggio associa al $client_G$ un $attore_G$ di tipo actorbase::actorsystem::clientactor::ClientActor.

4.7 actorbase::actorsystem::serialization

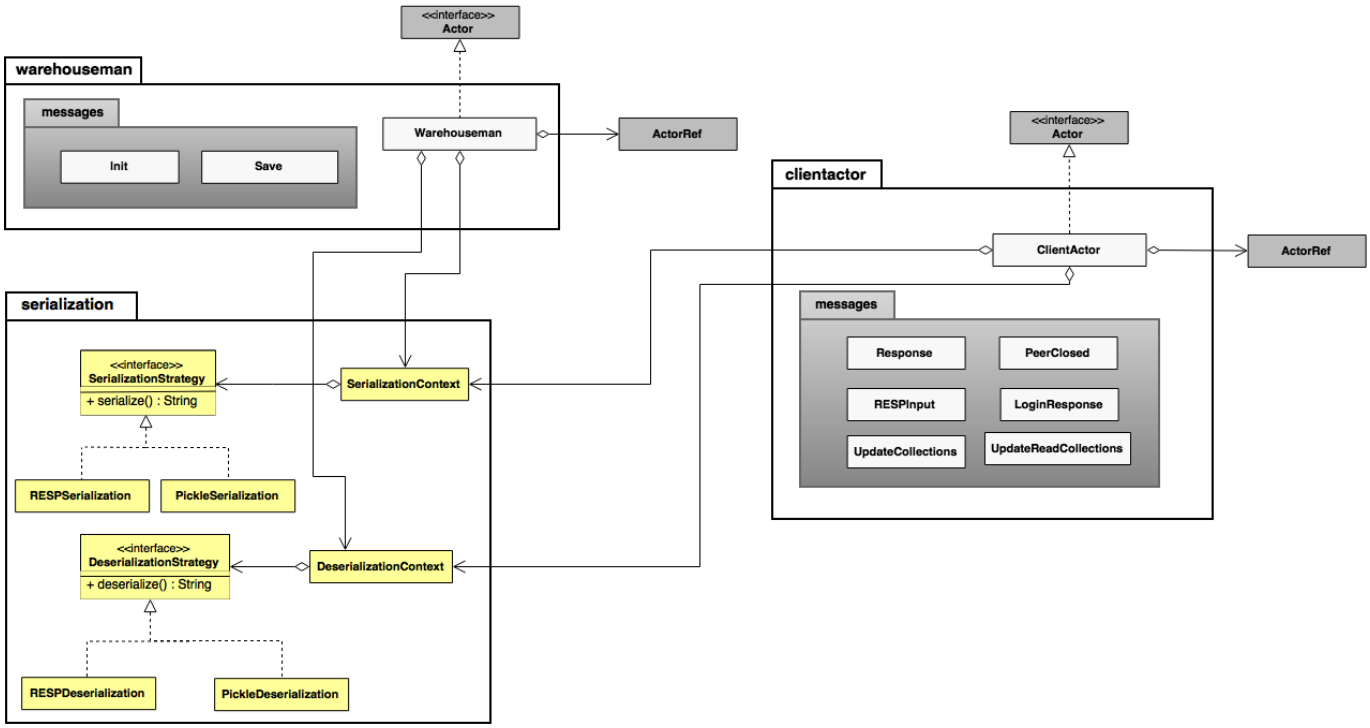

Figura 6: Serialization e interazioni con main e clientactor

4.7.1 Descrizione

Package$_G$ che si occupa di serializzare$_G$ e deserializzare$_G$ i messaggi mandati o ricevuti dal driver$_G$.

4.7.2 Interazioni con altre componenti

- actorbase::actorsystem::warehouseman;
- actorbase::actorsystem::clientactor.

4.7.3 Classi

4.7.3.1 actorbase::actorsystem::serialization::SerializationContext

4.7.3.1.1 Descrizione

Classe che offre dei metodi per la scelta su quale tipo di serializzazione$_G$ effettuare.

4.7.3.1.2 Utilizzo

Questa classe viene utilizzata da actorbase::actorsystem::warehouseman::Warehouseman e actorbase::actorsystem::clientactor::ClientActor per serializzare$_G$ i dati scegliendo una tra le strategie indicate dalle classi che estendono actorbase::actorsystem::serialization::SerializationStrategy.

4.7.3.1.3 Interazioni con altre classi

- actorbase::actorsystem::warehouseman::Warehouseman;
- actorbase::actorsystem::clientactor::ClientActor;
- actorbase::actorsystem::serialization::SerializationStrategy.

4.7.3.2 actorbase::actorsystem::serialization::SerializationStrategy

4.7.3.2.1 Descrizione

Interfaccia che offre una strategia generale per serializzare$_G$ i dati.

4.7.3.2.2 Utilizzo

Questa interfaccia viene utilizzata per l'applicazione del designpattern$_G$ strategy$_G$. Offre un metodo generalizzato per la serializzazione$_G$ di dati che verrà poi implementato dalle classi che implementeranno questa interfaccia.

4.7.3.2.3 Interazioni con altre classi

- actorbase::actorsystem::serialization::SerializationContext.

4.7.3.3 actorbase::actorsystem::serialization::RESPSerialization

4.7.3.3.1 Descrizione

Classe che si occupa di serializzare i dati in stringhe di formato RESP$_G$.

4.7.3.3.2 Utilizzo

Questa classe viene utilizzata per l'implementazione di una strategia di serializzazione$_G$ in formato RESP$_G$. Viene utilizzata principalmente per dati che dovranno essere mandati al driver$_G$.

4.7.3.3.3 Classi ereditate

- actorbase::actorsystem::serialization::SerializationStrategy.

4.7.3.4 actorbase::actorsystem::serialization::PickleSerialization

4.7.3.4.1 Descrizione

Classe che si occupa di serializzare i dati in formato Pickle$_G$.

4.7.3.4.2 Utilizzo

Questa classe viene utilizzata per l'implementazione di una strategia di serializzazione$_G$ in formato Pickle$_G$. Viene utilizzata principalmente per dati che dovranno essere salvati su disco.

4.7.3.4.3 Classi ereditate

- actorbase::actorsystem::serialization::SerializationStrategy.

4.7.3.5 actorbase::actorsystem::serialization::DeserializationContext

4.7.3.5.1 Descrizione

Classe che offre dei metodi per la scelta su quale tipo di deserializzazione$_G$ effettuare.

4.7.3.5.2 Utilizzo

Questa classe viene utilizzata da actorbase::actorsystem::warehouseman::Warehouseman e actorbase::actorsystem::clientactor::ClientActor per deserializzare$_G$ i dati scegliendo una tra le strategie indicate dalle classi che estendono actorbase::actorsystem::serialization::DeserializationStrategy.

4.7.3.5.3 Interazioni con altre classi

- actorbase::actorsystem::warehouseman::Warehouseman;
- actorbase::actorsystem::clientactor::ClientActor;
- actorbase::actorsystem::serialization::DeserializationStrategy.

4.7.3.6 actorbase::actorsystem::serialization::DeserializationStrategy

4.7.3.6.1 Descrizione

Interfaccia che offre una strategia generale per deserializzare$_G$ i dati.

4.7.3.6.2 Utilizzo

Questa interfaccia viene utilizzata per l'applicazione del design pattern$_G$ strategy$_G$. Offre un metodo generalizzato per la deserializzazione$_G$ di dati che verrà poi implementato dalle classi che implementeranno questa interfaccia.

4.7.3.6.3 Interazioni con altre classi

- actorbase::actorsystem::serialization::DeserializationContext.

4.7.3.7 actorbase::actorsystem::serialization::RESPDeserialization

4.7.3.7.1 Descrizione

Classe che si occupa di deserializzare i dati da stringhe in formato RESP$_G$.

4.7.3.7.2 Utilizzo

Questa classe viene utilizzata per l'implementazione di una strategia di deserializzazione$_G$ dal formato RESP$_G$. Viene utilizzato principalmente per i dati ricevuti dal driver$_G$.

4.7.3.7.3 Classi ereditate

- actorbase::actorsystem::serialization::DeserializationStrategy.

4.7.3.8 actorbase::actorsystem::serialization::PickleDeserialization

4.7.3.8.1 Descrizione

Classe che si occupa di deserializzare$_G$ i dati dal formato Pickle$_G$.

4.7.3.8.2 Utilizzo

Questa classe viene utilizzata per l'implementazione di una strategia di deserializzazione$_G$ dal formato Pickle$_G$. Viene utilizzato principalmente per la lettura di dati da disco.

4.7.3.8.3 Classi ereditate

- actorbase::actorsystem::serialization::DeserializationStrategy.

4.8 actorbase::actorsystem::warehouseman

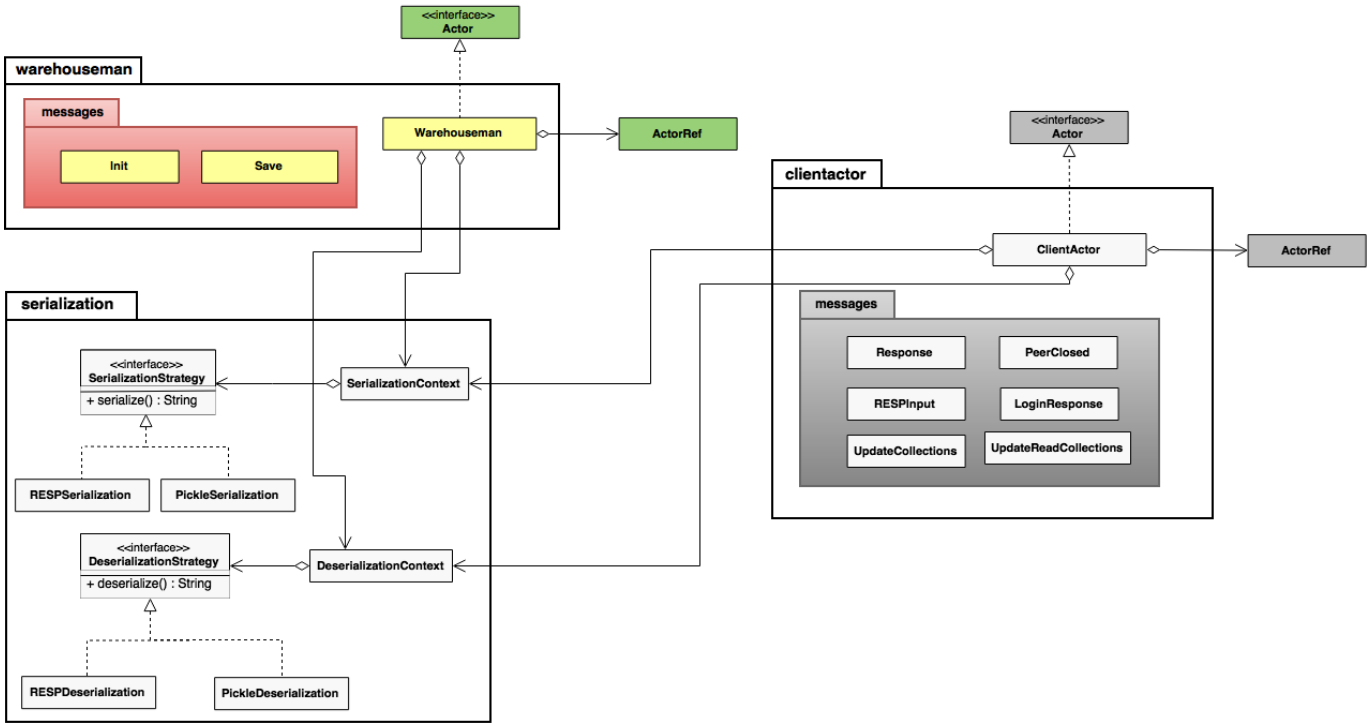

Figura 7: Warehouseman e interazioni con main e serialization

4.8.1 Descrizione

Package$_G$ che rappresenta l'attore$_G$ che si occuperà della persistenza$_G$ su disco dei dati.

4.8.2 Package contenuti

- actorbase::actorsystem::warehouseman::messages.

4.8.3 Classi

4.8.3.1 actorbase::actorsystem::warehouseman::Warehouseman

4.8.3.1.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Warehouseman$_G$.

4.8.3.1.2 Utilizzo

Questa classe viene utilizzata per effettuare il salvataggio su filesystem del database$_G$ e per caricare i dati da filesystem.

4.8.3.1.3 Classi ereditate

- akka::actor::Actor.

4.8.3.1.4 Interazioni con altre classi

- actorbase::actorsystem::serialization::SerializationContext;
- actorbase::actorsystem::serialization::DeserializationContext;
- akka::actor::ActorRef.

4.9 actorbase::actorsystem::warehouseman::messages

4.9.1 Descrizione

Package$_G$ che racchiude tutti i messaggi che gli attori di tipo Warehouseman$_G$ possono ricevere.

4.9.1.1 actorbase::actorsystem::warehouseman::messages::Init

4.9.1.1.1 Descrizione

Messaggio che porta alla lettura dei dati da disco.

4.9.1.1.2 Utilizzo

Quando actorbase::actorsystem::warehouseman::Warehouseman riceve questo messaggio inizializza uno actorbase::actorsystem::storekeeper::StoreKeeper con i dati letti da disco.

4.9.1.2 actorbase::actorsystem::warehouseman::messages::Save

4.9.1.2.1 Descrizione

Messaggio che porta al salvataggio dei dati su disco.

4.9.1.2.2 Utilizzo

Quando actorbase::actorsystem::warehouseman::Warehouseman riceve questo messaggio salverà i dati su disco sfruttando actorbase::actorsystem::serialization::SerializationContext.

4.10 actorbase::actorsystem::main

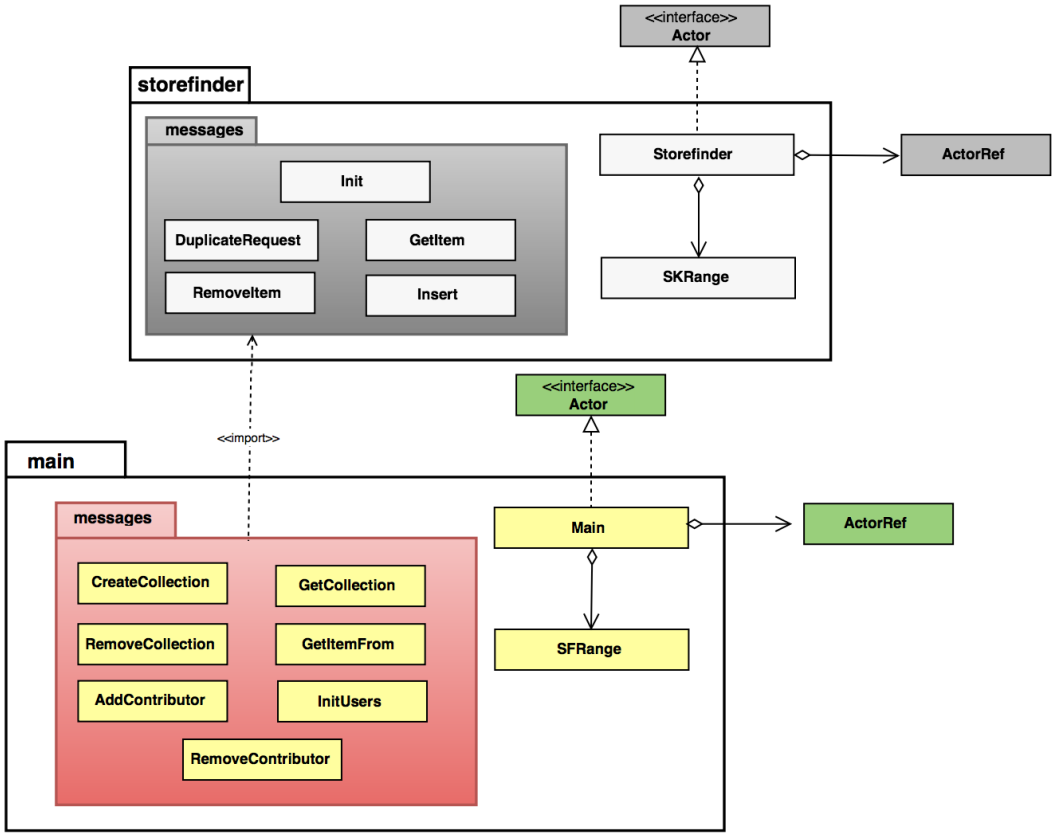

Figura 8: Main e interazioni con storefinder

4.10.1 Descrizione

Package$_G$ che rappresenta l'attore$_G$ che si occuperà di gestire le richieste al server$_G$.

4.10.2 Package contenuti

- actorbase::actorsystem::main::messages.

4.10.3 Classi

4.10.3.1 actorbase::actorsystem::main::Main

4.10.3.1.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Main$_G$.

4.10.3.1.2 Utilizzo

Questa classe viene utilizzata per gestire le richieste ricevute al server$_G$.

4.10.3.1.3 Classi ereditate

- akka::actor::Actor.

4.10.3.1.4 Interazioni con altre classi

- akka::actor::ActorRef;
- actorbase::actorsystem::main::SFRange.

4.10.3.2 actorbase::actorsystem::main::SFRange

4.10.3.2.1 Descrizione

Classe che rappresenta i range di indici contenuti in ogni attore$_G$ di tipo Storekeeper$_G$.

4.10.3.2.2 Utilizzo

Questa classe viene utilizzata per tener traccia del contenuto degli attori di tipo actorbase::actorsystem::storefinder::Storefinder.

4.10.3.2.3 Interazioni con altre classi

- actorbase::actorsystem::main::Main.

4.11 actorbase::actorsystem::main::messages

4.11.1 Descrizione

Package$_G$ che racchiude tutti i messaggi che gli attori di tipo Warehouseman$_G$ possono ricevere.

4.11.2 Package importati

- actorbase::actorsystem::storefinder::messages

4.11.2.1 actorbase::actorsystem::main::messages::CreateCollection

4.11.2.1.1 Descrizione

Messaggio che porta alla creazione di una collezione$_G$.

4.11.2.1.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio inizializza uno actorbase::actorsystem::storefinder::Storefinder.

4.11.2.2 actorbase::actorsystem::main::messages::GetCollection

4.11.2.2.1 Descrizione

Messaggio che indica una richiesta di una collezione$_G$.

4.11.2.2.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio cercherà l'attore$_G$ di tipo actorbase::actorsystem::storefinder::Storefinder corrispondente alla collezione$_G$ cercata.

4.11.2.3 actorbase::actorsystem::main::messages::RemoveCollection

4.11.2.3.1 Descrizione

Messaggio che indica una richiesta di rimozione di una collezione$_G$.

4.11.2.3.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio cercherà l'attore$_G$ di tipo actorbase::actorsystem::storefinder::Storefinder corrispondente alla collezione$_G$ cercata e ne avvierà la procedura di rimozione dal database$_G$.

4.11.2.4 actorbase::actorsystem::main::messages::GetItemFrom

4.11.2.4.1 Descrizione

Messaggio che indica una richiesta di ricerca di un item$_G$ da una o più collezione$_G$.

4.11.2.4.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio cercherà gli attori di tipo actorbase::actorsystem::storefinder::Storefinder corrispondenti alle collezioni$_G$ su cui effettuare la ricerca e inoltrerà a loro la richiesta dell'item$_G$ con la chiave da cercare.

4.11.2.5 actorbase::actorsystem::main::messages::AddContributor

4.11.2.5.1 Descrizione

Messaggio che indica una richiesta di aggiunta di un collaboratore ad una collezione$_G$.

4.11.2.5.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio cercherà gli attori di tipo actorbase::actorsystem::storefinder::Storefinder che mappano gli attori di tipo actorbase::actorsystem::userkeeper::Userkeeper. e inoltrerà loro la richiesta di aggiunta del collaboratore.

4.11.2.6 actorbase::actorsystem::main::messages::DuplicateRequestSF

4.11.2.6.1 Descrizione

Messaggio che indica una richiesta di sdoppiamento di uno Storefinder$_G$.

4.11.2.6.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio provvederà a sdoppiare l'attore$_G$ di tipo actorbase::actorsystem::storefinder::Storefinder.

4.11.2.7 actorbase::actorsystem::main::messages::RemoveContributor

4.11.2.7.1 Descrizione

Messaggio che indica una richiesta di rimozione di un collaboratore ad una collezione$_G$.

4.11.2.7.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio cercherà gli attori di tipo actorbase::actorsystem::storefinder::Storefinder che mappano gli attori di tipo actorbase::actorsystem::userkeeper::Userkeeper e inoltrerà loro la richiesta di rimozione del collaboratore.

4.11.2.8 actorbase::actorsystem::main::messages::InitUserkeeper

4.11.2.8.1 Descrizione

Messaggio che indica una richiesta di creazione di uno Userkeeper$_G$.

4.11.2.8.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio provvederà a inoltrare agli attori di tipo actorbase::actorsystem::storefinder::Storefinder che mappano gli attori di tipo actorbase::actorsystem::userkeeper::Userkeeper la creazione di un attore$_G$ di questo tipo.

4.11.2.9 actorbase::actorsystem::storefinder::messages::Init

4.11.2.9.1 Descrizione

Messaggio importato dal package actorbase::actorsystem::storefinder::messages che, se ricevuto dall'attore$_G$ di tipo Main$_G$, indica la necessità di inizializzare degli Storefinder$_G$.

4.11.2.9.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio provvederà a inizializzare gli attori di tipo actorbase::actorsystem::storefinder::Storefinder presenti nella cartella indicata dal messaggio.

4.11.2.10 actorbase::actorsystem::storefinder::messages::DuplicateRequest

4.11.2.10.1 Descrizione

Messaggio importato dal package actorbase::actorsystem::storefinder::messages che, se ricevuto dall'attore$_G$ di tipo Main$_G$, indica l'avvenuta divisione di uno Storefinder$_G$.

4.11.2.10.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio provvederà ad aggiornare i suoi indici.

4.11.2.11 actorbase::actorsystem::storefinder::messages::GetItem

4.11.2.11.1 Descrizione

Messaggio importato dal package actorbase::actorsystem::storefinder::messages che indica la richiesta di un item$_G$.

4.11.2.11.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio provvederà a inoltrare la richiesta all'attore$_G$ di tipo actorbase::actorsystem::storefinder::Storefinder che contiene l'item$_G$ cercato.

4.11.2.12 actorbase::actorsystem::storefinder::messages::RemoveItem

4.11.2.12.1 Descrizione

Messaggio che indica la richiesta di rimozione di un item$_G$.

4.11.2.12.2 Utilizzo

Quando actorbase::actorsystem::storefinder::Storefinder riceve questo messaggio provvederà a inoltrare la richiesta all'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper contenente l'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper contenente l'item$_G$ da rimuovere.

4.11.2.13 actorbase::actorsystem::storefinder::messages::Insert

4.11.2.13.1 Descrizione

Messaggio importato dal package actorbase::actorsystem::storefinder::messages che indica la richiesta di inserimento di un item$_G$.

4.11.2.13.2 Utilizzo

Quando actorbase::actorsystem::main::Main riceve questo messaggio provvederà a inoltrare la richiesta all'attore$_G$ di tipo actorbase::actorsystem::storefinder::Storefinder contenente l'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper destinato a ricevere l'item$_G$ da inserire.

4.12 actorbase::actorsystem::userkeeper

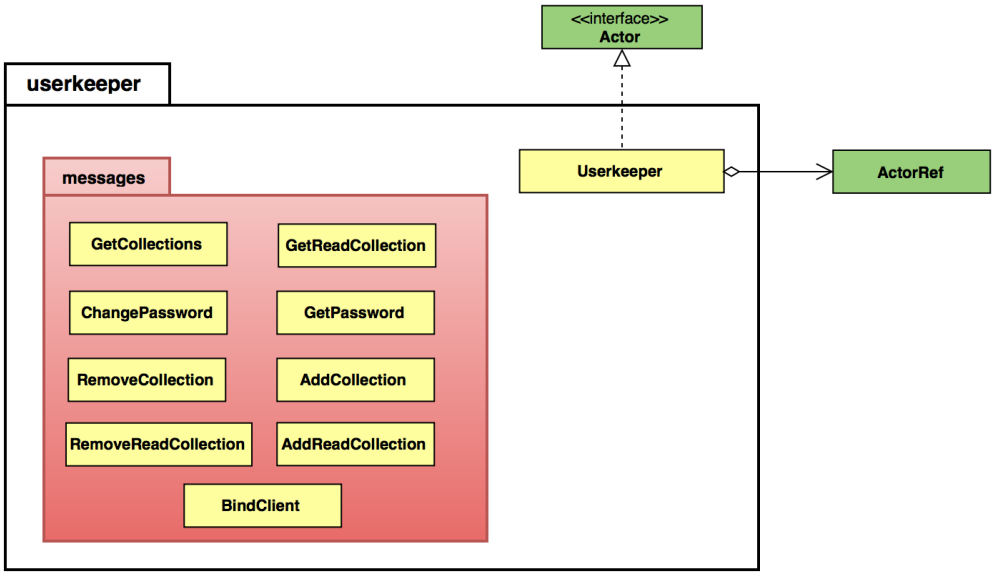

Figura 9: Userkeeper, visione generale del package

4.12.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Userkeeper$_G$.

4.12.2 Package contenuti

- actorbase::actorsystem::userkeeper::messages.

4.12.3 Classi

4.12.3.1 actorbase::actorsystem::userkeeper::Userkeeper

4.12.3.1.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Userkeeper$_G$.

4.12.3.1.2 Utilizzo

Questa classe viene utilizzata per salvare gli utenti nel database$_G$.

4.12.3.1.3 Classi ereditate

- akka::actor::Actor.

4.12.3.1.4 Interazioni con altre classi

- akka::actor::ActorRef.

4.13 actorbase::actorsystem::userkeeper::messages

4.13.1 Descrizione

Package$_G$ che contiene tutti i messaggi che possono essere ricevuti da actorbase::actorsystem::userkeeper::Userkeeper.

4.13.2 Classi

4.13.2.1 actorbase::actorsystem::userkeeper::messages::GetCollections

4.13.2.1.1 Descrizione

Messaggio che indica la richiesta della lista di collezioni$_G$ di cui l'utente ha permessi in lettura e scrittura.

4.13.2.1.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà a restituire all'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor la lista di collezioni$_G$ di cui l'utente ha permessi in lettura e scrittura.

4.13.2.2 actorbase::actorsystem::userkeeper::messages::GetReadCollection

4.13.2.2.1 Descrizione

Messaggio che indica la richiesta della lista di collezioni$_G$ di cui l'utente ha permessi in lettura.

4.13.2.2.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà a restituire all'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor la lista di collezioni$_G$ di cui l'utente ha permessi in lettura.

4.13.2.3 actorbase::actorsystem::userkeeper::messages::ChangePassword

4.13.2.3.1 Descrizione

Messaggio che indica la richiesta di cambiamento della password.

4.13.2.3.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà a modificare la password dell'utente.

4.13.2.4 actorbase::actorsystem::userkeeper::messages::GetPassword

4.13.2.4.1 Descrizione

Messaggio che indica la richiesta della password.

4.13.2.4.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà a inviare la password all'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor.

4.13.2.5 actorbase::actorsystem::userkeeper::messages::RemoveCollection

4.13.2.5.1 Descrizione

Messaggio che indica la richiesta di rimozione di una collezione$_G$ dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura e scrittura.

4.13.2.5.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà a rimuovere la collezione$_G$ desiderata dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura e scrittura.

4.13.2.6 actorbase::actorsystem::userkeeper::messages::AddCollection

4.13.2.6.1 Descrizione

Messaggio che indica la richiesta di aggiunta di una collezione$_G$ dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura e scrittura.

4.13.2.6.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà ad aggiungere la collezione$_G$ desiderata dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura e scrittura.

4.13.2.7 actorbase::actorsystem::userkeeper::messages::RemoveReadCollection

4.13.2.7.1 Descrizione

Messaggio che indica la richiesta di rimozione di una collezione$_G$ dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura.

4.13.2.7.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà a rimuovere la collezione$_G$ desiderata dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura.

4.13.2.8 actorbase::actorsystem::userkeeper::messages::AddReadCollection

4.13.2.8.1 Descrizione

Messaggio che indica la richiesta di aggiunta di una collezione$_G$ dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura.

4.13.2.8.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà ad aggiungere la collezione$_G$ desiderata dalla lista contenente le collezioni$_G$ di cui l'utente ha permessi in lettura.

4.13.2.9 actorbase::actorsystem::userkeeper::messages::BindClient

4.13.2.9.1 Descrizione

Messaggio che indica la richiesta di collegamento con un ClientActor$_G$.

4.13.2.9.2 Utilizzo

Quando actorbase::actorsystem::userkeeper::Userkeeper riceve questo messaggio provvederà a salvarsi il riferimento dell'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor che ha inviato questo messaggio.

4.14 actorbase::actorsystem::storefinder

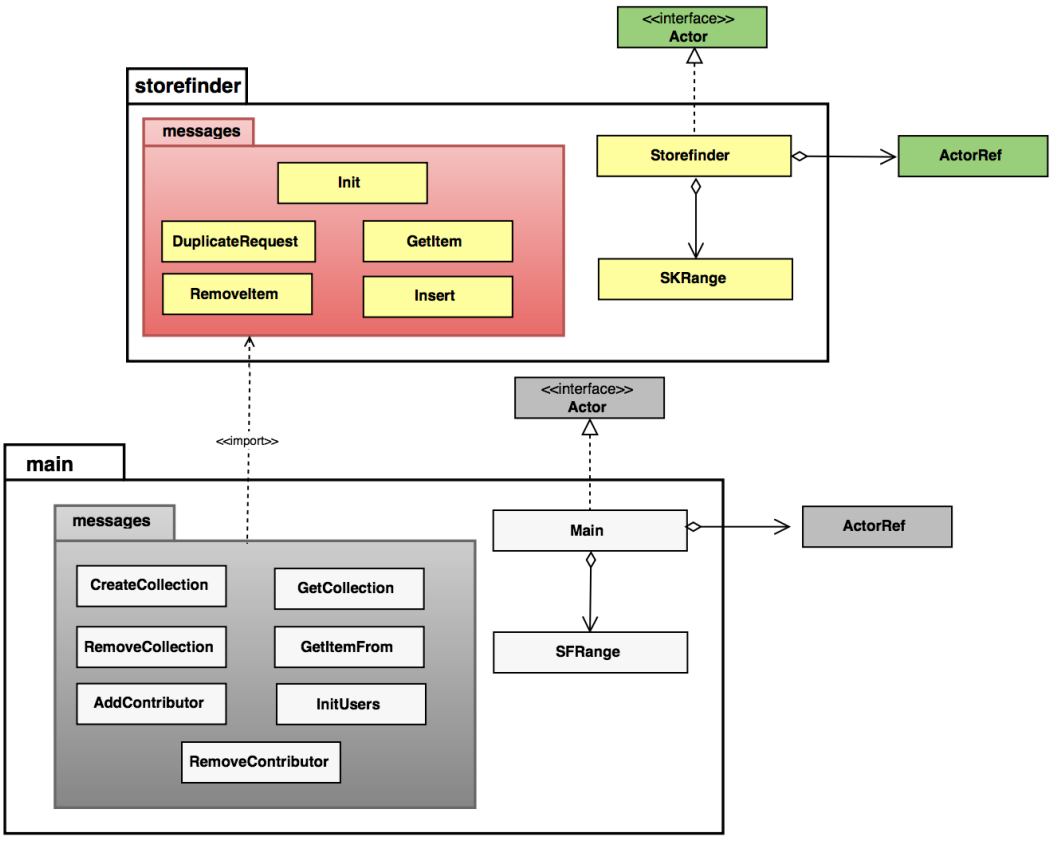

Figura 10: Storefinder e interazioni con main

4.14.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Storefinder$_G$.

4.14.2 Package contenuti

- actorbase::actorsystem::storefinder::messages.

4.14.3 Classi

4.14.3.1 actorbase::actorsystem::storefinder::Storefinder

4.14.3.1.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Storefinder$_G$.

4.14.3.1.2 Utilizzo

Questa classe viene utilizzata per tener traccia degli attori di tipo actorbase::actorsystem::storekeeper::Storekeeper.

4.14.3.1.3 Classi ereditate

- akka::actor::Actor.

4.14.3.1.4 Interazioni con altre classi

- akka::actor::ActorRef;
- actorbase::actorsystem::storefinder::SKRange.

4.14.3.2 actorbase::actorsystem::storefinder::SKRange

4.14.3.2.1 Descrizione

Classe che rappresenta i range di indici contenuti in ogni attore$_G$ di tipo Storekeeper$_G$.

4.14.3.2.2 Utilizzo

Questa classe viene utilizzata per tener traccia del contenuto degli attori di tipo actorbase::actorsystem::storekeeper::Storekeeper.

4.14.3.2.3 Interazioni con altre classi

- actorbase::actorsystem::storefinder::Storefinder.

4.15 actorbase::actorsystem::storefinder::messages

4.15.1 Descrizione

Package$_G$ che contiene tutti i messaggi che possono essere ricevuti da actorbase::actorsystem::storefinder::Storefinder.

4.15.2 Classi

4.15.2.1 actorbase::actorsystem::storefinder::messages::Init

4.15.2.1.1 Descrizione

Messaggio che, se ricevuto dall'attore$_G$ di tipo Storefinder$_G$, indica la necessità di inizializzare degli Storekeeper$_G$.

4.15.2.1.2 Utilizzo

Quando actorbase::actorsystem::storefinder::Storefinder riceve questo messaggio provvederà a inizializzare gli attori di tipo actorbase::actorsystem::storekeeper::Storekeeper presenti nella cartella indicata dal messaggio.

4.15.2.2 actorbase::actorsystem::storefinder::messages::DuplicateRequest

4.15.2.2.1 Descrizione

Messaggio che, se ricevuto dall'attore$_G$ di tipo Storefinder$_G$, indica l'avvenuta divisione di uno Storekeeper$_G$.

4.15.2.2.2 Utilizzo

Quando actorbase::actorsystem::storefinder::Storefinder riceve questo messaggio provvederà ad aggiornare i suoi indici.

4.15.2.3 actorbase::actorsystem::storefinder::messages::GetItem

4.15.2.3.1 Descrizione

Messaggio che indica la richiesta di un item$_G$.

4.15.2.3.2 Utilizzo

Quando actorbase::actorsystem::storefinder::Storefinder riceve questo messaggio provvederà a inoltrare la richiesta all'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper che contiene l'item$_G$ cercato.

4.15.2.4 actorbase::actorsystem::storefinder::messages::RemoveItem

4.15.2.4.1 Descrizione

Messaggio che indica la richiesta di rimozione di un item$_G$.

4.15.2.4.2 Utilizzo

Quando actorbase::actorsystem::storefinder::Storefinder riceve questo messaggio provvederà a inoltrare la richiesta all'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper che contiene l'item$_G$ da rimuovere.

4.15.2.5 actorbase::actorsystem::storefinder::messages::Insert

4.15.2.5.1 Descrizione

Messaggio che indica la richiesta di inserimento di un item$_G$.

4.15.2.5.2 Utilizzo

Quando actorbase::actorsystem::storefinder::Storefinder riceve questo messaggio provvederà a inoltrare la richiesta all'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper destinato a ricevere l'item$_G$ da inserire.

4.16 actorbase::actorsystem::storekeeper

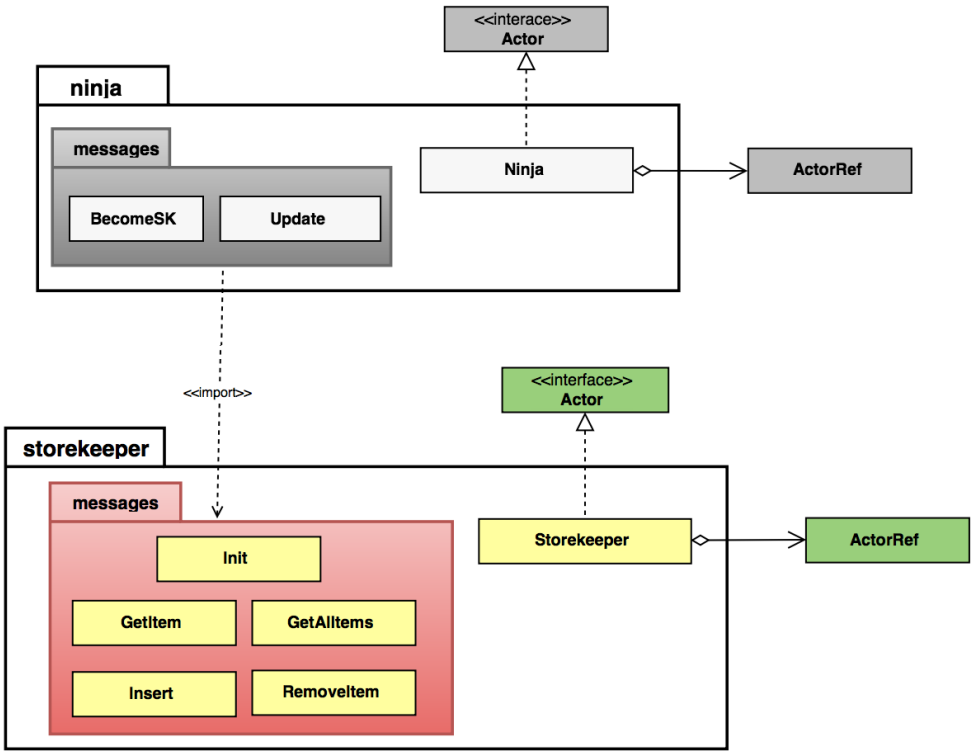

Figura 11: Storekeeper e interazioni con ninja

4.16.1 Descrizione

Package$_G$ che rappresenta l'attore$_G$ di tipo Storekeeper$_G$.

4.16.2 Package contenuti

- actorbase::actorsystem::storekeeper::messages.

4.16.3 Classi

4.16.3.1 actorbase::actorsystem::storekeeper::Storekeeper

4.16.3.1.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Storekeeper$_G$.

4.16.3.1.2 Utilizzo

Questa classe viene utilizzata per salvare nel database$_G$ gli item.

4.16.3.1.3 Classi ereditate

- akka::actor::Actor.

4.16.3.1.4 Interazioni con altre classi

- akka::actor::ActorRef.

4.17 actorbase::actorsystem::storekeeper::messages

4.17.1 Descrizione

Package$_G$ che contiene tutti i messaggi che possono essere ricevuti dall'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper.

4.17.2 Classi

4.17.2.1 actorbase::actorsystem::storekeeper::messages::Init

4.17.2.1.1 Descrizione

Messaggio che richiede l'inizializzazione dello Storekeeper$_G$.

4.17.2.1.2 Utilizzo

Quando actorbase::actorsystem::storekeeper::Storekeeper riceve questo messaggio manda un messaggio all'attore$_G$ di tipo actorbase::actorsystem::warehouseman::Warehouseman per la deserializzazione$_G$ dei dati da leggere da disco.

4.17.2.2 actorbase::actorsystem::storekeeper::messages::GetItem

4.17.2.2.1 Descrizione

Messaggio che richiede un item$_G$ contenuto dall'attore$_G$ Storekeeper$_G$ che lo riceve.

4.17.2.2.2 Utilizzo

Quando actorbase::actorsystem::storekeeper::Storekeeper riceve questo messaggio risponde mandando l'item$_G$ cercato all'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor che aveva effettuato la richiesta.

4.17.2.3 actorbase::actorsystem::storekeeper::messages::GetAllItem

4.17.2.3.1 Descrizione

Messaggio che richiede gli item$_G$ contenuti dall'attore$_G$ Storekeeper$_G$ che lo riceve.

4.17.2.3.2 Utilizzo

Quando actorbase::actorsystem::storekeeper::Storekeeper riceve questo messaggio risponde mandando gli item$_G$ all'attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor che aveva effettuato la richiesta.

4.17.2.4 actorbase::actorsystem::storekeeper::messages::Insert

4.17.2.4.1 Descrizione

Messaggio che richiede l'inserimento di un item$_G$.

4.17.2.4.2 Utilizzo

Quando actorbase::actorsystem::storekeeper::Storekeeper riceve questo messaggio inserisce l'item$_G$.

4.17.2.5 actorbase::actorsystem::storekeeper::messages::RemoveItem

4.17.2.5.1 Descrizione

Messaggio che richiede la rimozione di un item$_G$.

4.17.2.5.2 Utilizzo

Quando actorbase::actorsystem::storekeeper::Storekeeper riceve questo messaggio rimuove l'item$_G$.

4.18 actorbase::actorsystem::ninja

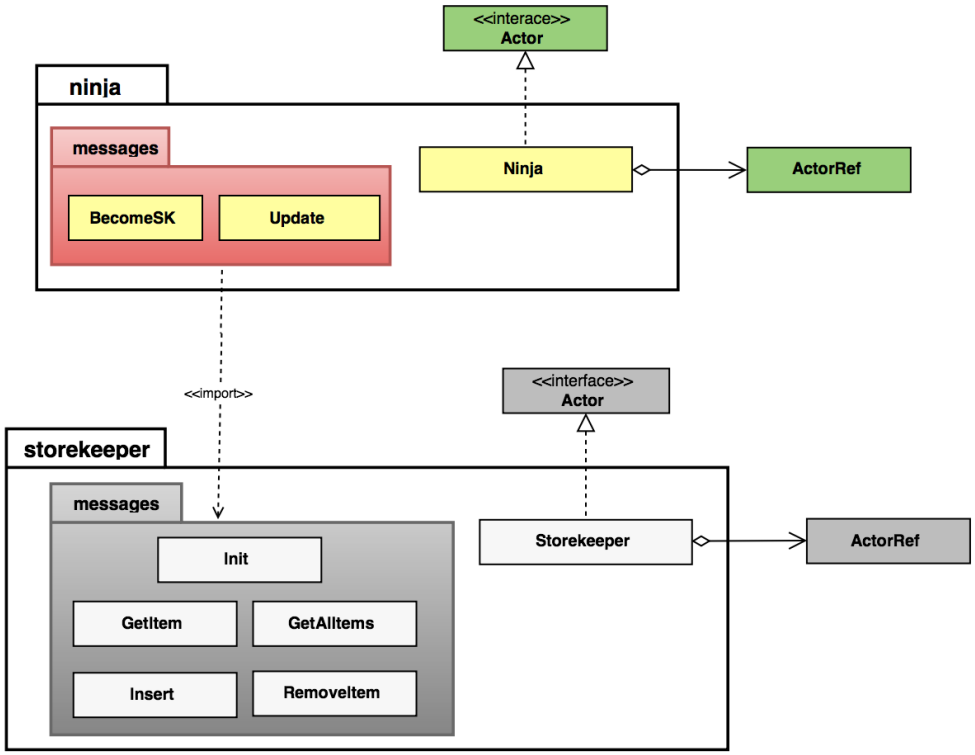

Figura 12: Ninja e interazioni con storekeeper

4.18.1 Descrizione

Package$_G$ che rappresenta l'attore$_G$ di tipo Ninja$_G$.

4.18.2 Package contenuti

- actorbase::actorsystem::ninja::messages.

4.18.3 Classi

4.18.3.1 actorbase::actorsystem::ninja::Ninja

4.18.3.1.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Ninja$_G$.

4.18.3.1.2 Utilizzo

Questa classe viene utilizzata per avere un backup$_G$ per gli attori di tipo actorbase::actorsystem::storekeeper::Storekeeper.

4.18.3.1.3 Classi ereditate

- akka::actor::Actor.

4.18.3.1.4 Interazioni con altre classi

- akka::actor::ActorRef.

4.19 actorbase::actorsystem::ninja::messages

4.19.0.0.1 Descrizione

Package$_G$ che contiene tutti i messaggi che possono essere ricevuti dall'attore$_G$ di tipo actorbase::actorsystem::ninja::Ninja

4.19.0.1 actorbase::actorsystem::ninja::messages::Update

4.19.0.1.1 Descrizione

Messaggio che richiede l'update dell'attore$_G$ che lo riceve.

4.19.0.1.2 Utilizzo

Quando actorbase::actorsystem::ninja::Ninja riceve questo messaggio aggiorna gli item$_G$ che contiene.

4.19.0.2 actorbase::actorsystem::ninja::messages::BecomeSK

4.19.0.2.1 Descrizione

Messaggio che richiede che l'attore$_G$ di tipo Ninja$_G$ diventi di tipo Storekeeper$_G$.

4.19.0.2.2 Utilizzo

Quando actorbase::actorsystem::ninja::Ninja riceve questo messaggio cambia il proprio contesto importando tutti i messaggi di actorbase::actorsystem::storekeeper::messages e comportandosi come se fosse un actorbase::actorsystem::storekeeper::Storekeeper.

4.20 actorbase::actorsystem::manager

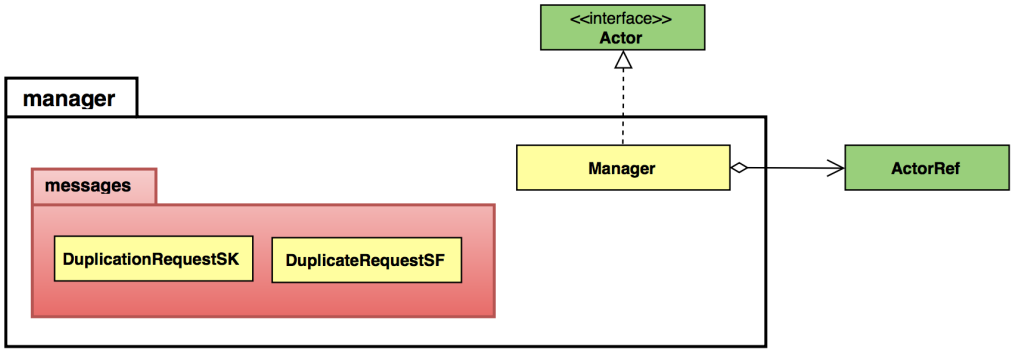

Figura 13: Manager, visione generale del package

4.20.1 Descrizione

Package$_G$ che rappresenta l'attore$_G$ di tipo Manager$_G$.

4.20.2 Package contenuti

- actorbase::actorsystem::manager::messages.

4.20.3 Classi

4.20.3.1 actorbase::actorsystem::manager::Manager

4.20.3.1.1 Descrizione

Classe che rappresenta un attore$_G$ di tipo Manager$_G$.

4.20.3.1.2 Utilizzo

Questa classe viene utilizzata per gestire le richieste di divisione di attori di tipo sec:actorbase::actorsystem::storekeeper::Storekeeper e sec:actorbase::actorsystem::storefinder::Storefinder.

4.20.3.1.3 Classi ereditate

- akka::actor::Actor.

4.20.3.1.4 Interazioni con altre classi

- akka::actor::ActorRef.

4.21 actorbase::actorsystem::manager::messages

4.21.0.0.1 Descrizione

Package$_G$ che contiene tutti i messaggi che possono essere ricevuti dall'attore$_G$ di tipo actorbase::actorsystem::manager::Manager

4.21.0.1 actorbase::actorsystem::manager::messages::DuplicationRequestSK

4.21.0.1.1 Descrizione

Messaggio che richiede la divisione a metà dell'attore$_G$ di tipo Storekeeper$_G$.

4.21.0.1.2 Utilizzo

Quando actorbase::actorsystem::manager::Manager riceve questo messaggio provvederà a dividere a metà l'attore$_G$ di tipo actorbase::actorsystem::storekeeper::Storekeeper che dovrà essere diviso.Manderà inoltre un messaggio all'attore$_G$ di tipo actorbase::actorsystem::storefinder::Storefinder per aggiornare i suoi riferimenti.

4.21.0.2 actorbase::actorsystem::manager::messages::DuplicationRequestSF

4.21.0.2.1 Descrizione

Messaggio che richiede la divisione a metà dell'attore$_G$ di tipo Storefinder$_G$.

4.21.0.2.2 Utilizzo

Quando actorbase::actorsystem::manager::Manager riceve questo messaggio provvederà a dividere a metà l'attore$_G$ di tipo actorbase::actorsystem::storefinder::Storefinder che dovrà essere diviso. Manderà inoltre un messaggio all'attore$_G$ di tipo actorbase::actorsystem::main::Main per aggiornare i suoi riferimenti.

4.22 actorbase::driver

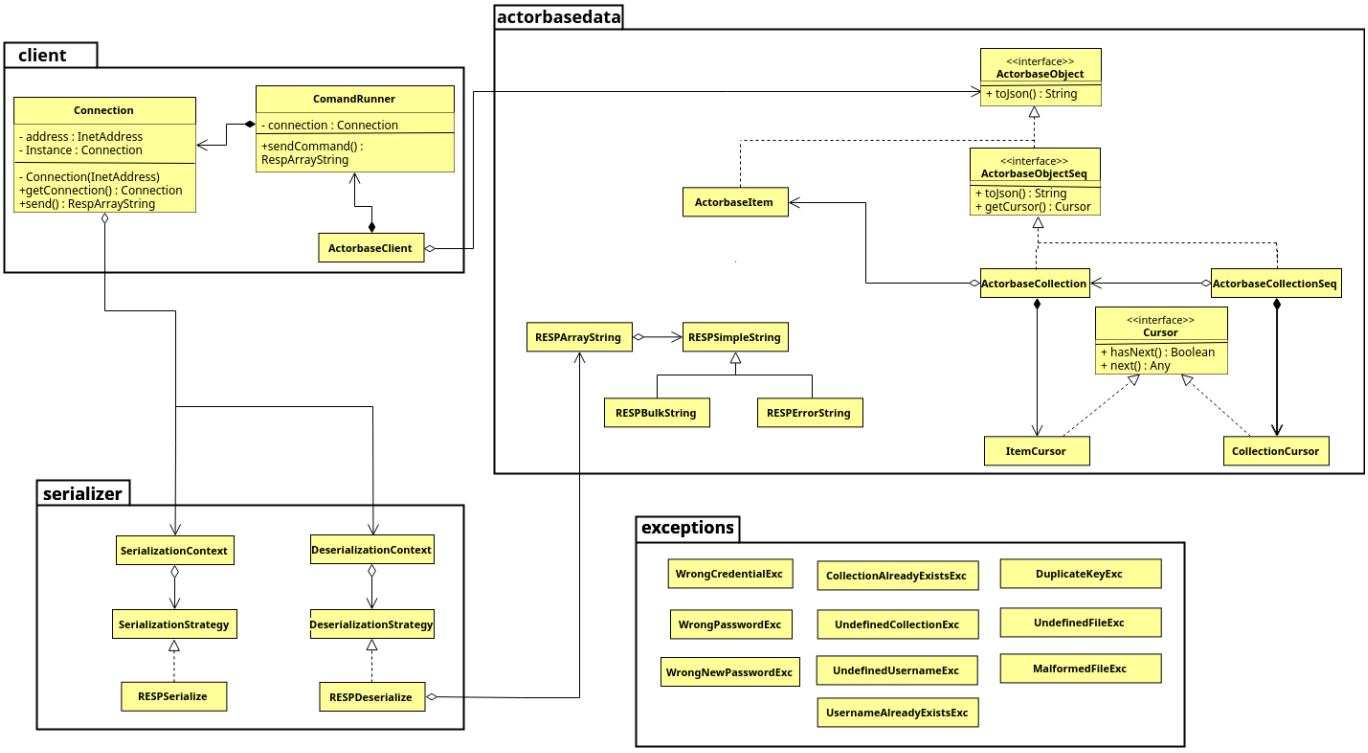

Figura 14: CLI:architettura MVC variante push model

4.22.1 Descrizione

Package$_G$ per la componente driver$_G$ dell'architettura MVC$_G$.

4.22.2 Interazioni con altre componenti

- actorbase::cli;

- actorbase::actorsystem.

4.22.3 Package contenuti

- actorbase::driver::client;
- actorbase::driver::serializer;
- actorbase::driver::actorbasedata;
- actorbase::driver::exceptions.

4.23 actorbase::driver::client

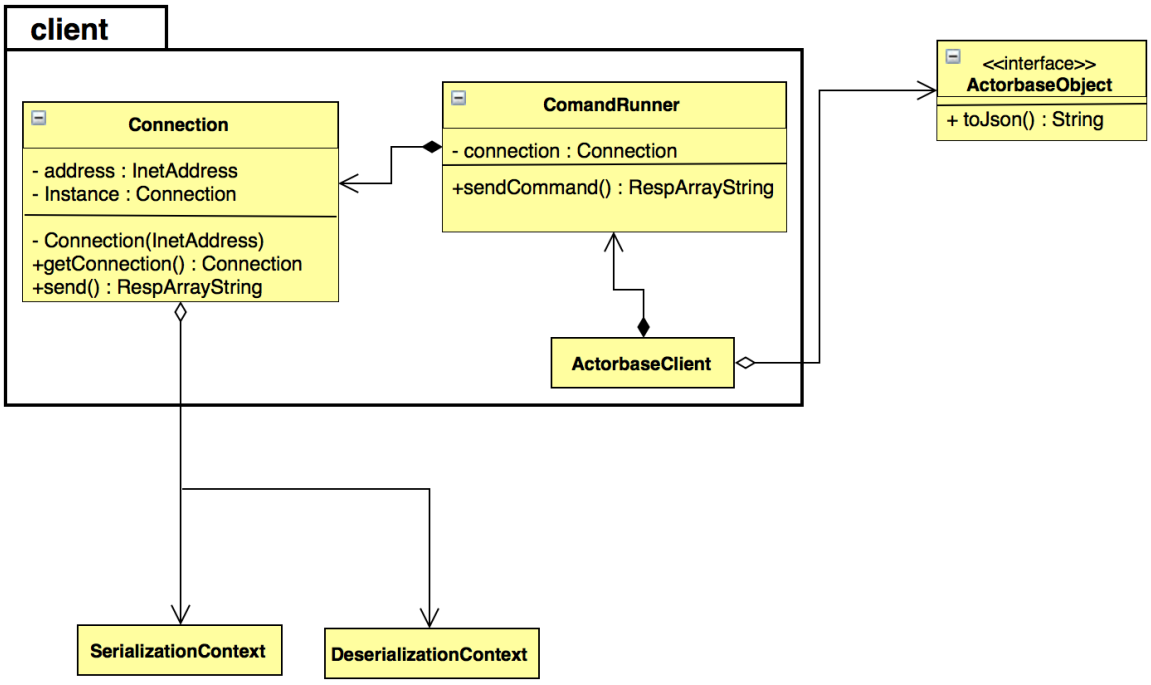

Figura 15: CLI, architettura MVC variante push model

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::clientactor.

4.23.3 Classi

4.23.3.1 actorbase::driver::client::ActorbaseClient

4.23.3.1.1 Descrizione

Classe che rappresenta un'istanza del driver$_G$.

4.23.3.1.2 Utilizzo

La componente actorbase::cli::models::CommandReceiver ha un riferimento a questa classe tramite il quale le vengono passati i comandi che sono stati già sottoposti al parsing$_G$. Questa classe si occupa di mandare il comando alla componente actorbase::driver::client::CommandRunner, che rappresenta una classe façade$_G$, per farne proseguire l'esecuzione.
Il risultato delle operazioni viene restituito a actorbase::cli::models::CommandReceiver tramite oggetti di tipo actorbase::driver::actorbasedata::ActorbaseObject.

4.23.3.1.3 Interazioni con altre classi

- actorbase::cli::models::CommandReceiver;
- actorbase::driver::client::CommandRunner;
- actorbase::driver::actorbasedata::ActorbaseObject.

4.23.3.2 actorbase::driver::client::CommandRunner

4.23.3.2.1 Descrizione

Classe façade$_G$ che si occupa di mettere in comunicazione il driver$_G$ con il server$_G$.

4.23.3.2.2 Utilizzo

Questa classe riceve dalla componente actorbase::driver::client::ActorbaseClient il comando da eseguire e le restituisce il risultato nascondendole tutte le operazioni eseguite.

4.23.3.2.3 Interazioni con altre classi

- actorbase::driver::client::ActorbaseClient;
- actorbase::driver::client::Connection.

4.23.3.3 actorbase::driver::client::Connection

4.23.3.3.1 Descrizione

Classe per la connessione tra il driver$_G$ e il server$_G$.

4.23.3.3.2 Utilizzo

Questa classe si incarica della connessione e comunicazione con il server$_G$. Una volta connesso alla componente actorbase::actorsystem::tcpserver::TCPServer le verrà assegnato un oggetto di tipo actorbase::actorsystem::clientactor::ClientActor.
Questa classe inoltre chiamerà le componenti necessarie per la serializzazione$_G$ e deserializzazione$_G$ delle stringhe ricevute dal server$_G$ o da actorbase::driver::client::CommandRunner, ossia actorbase::driver::serializer::SerializationContext e actorbase::driver::serializer::DeserializationContext.

4.23.3.3.3 Interazioni con altre classi

- actorbase::driver::serializer::SerializationContext;
- actorbase::driver::serializer::DeserializationContext;
- actorbase::driver::client::CommandRunner;
- actorbase::actorsystem::tcpserver::TCPServer;
- actorbase::actorsystem::clientactor::ClientActor.

4.24 actorbase::driver::serializer

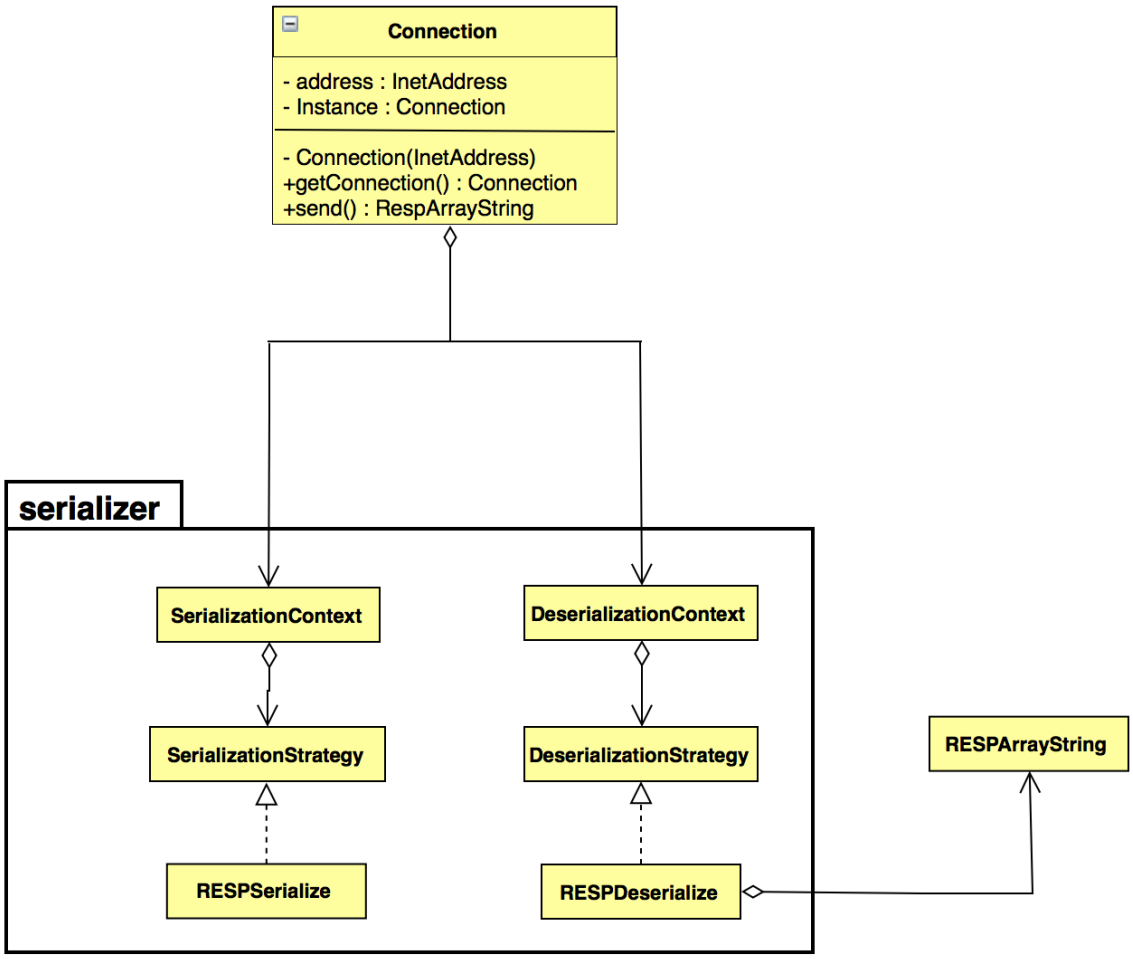

Figura 16: Driver, package serializer e interazioni con package client e package actorbasedata

4.24.1 Descrizione

Package$_G$ che si occupa di serializzare$_G$ e deserializzare$_G$ i messaggi mandati o ricevuti dal server$_G$.

4.24.2 Interazioni con altre componenti

- actorbase::driver::client;

- actorbase::driver::actorbasedata.

4.24.3 Classi

4.24.3.1 actorbase::driver::serializer::SerializationContext

4.24.3.1.1 Descrizione

Classe che offre dei metodi per la scelta su quale tipo di serializzazione$_G$ effettuare.

4.24.3.1.2 Utilizzo

Questa classe viene utilizzata da actorbase::driver::client::Connection per serializzare$_G$ i dati scegliendo una tra le strategie indicate dalle classi che estendono actorbase::driver::serializer::SerializationStrategy.

4.24.3.1.3 Interazioni con altre classi

- actorbase::driver::client::Connection;
- actorbase::driver::serializer::SerializationStrategy.

4.24.3.2 actorbase::driver::serializer::SerializationStrategy

4.24.3.2.1 Descrizione

Interfaccia che offre una strategia generale per serializzare$_G$ i dati.

4.24.3.2.2 Utilizzo

Questa interfaccia viene utilizzata per l'applicazione del design pattern$_G$ strategy$_G$. Offre un metodo generalizzato per la serializzazione$_G$ di dati che verrà poi implementato dalle classi che implementeranno questa interfaccia.

4.24.3.2.3 Interazioni con altre classi

- actorbase::driver::serializer::SerializationContext.

4.24.3.3 actorbase::driver::serializer::RESPSerialize

4.24.3.3.1 Descrizione

Classe che si occupa di serializzare i dati in stringhe di formato RESP$_G$.

4.24.3.3.2 Utilizzo

Questa classe viene utilizzata per l'implementazione di una strategia di serializzazione$_G$ in formato RESP$_G$.

4.24.3.3.3 Classi ereditate

- actorbase::driver::serializer::SerializationStrategy.

4.24.3.4 actorbase::driver::serializer::DeserializationContext

4.24.3.4.1 Descrizione

Classe che offre dei metodi per la scelta su quale tipo di deserializzazione$_G$ effettuare.

4.24.3.4.2 Utilizzo

Questa classe viene utilizzata da actorbase::driver::client::Connection per deserializzare$_G$ i dati scegliendo una tra le strategie indicate dalle classi che estendono actorbase::driver::serializer::DeserializationStrategy.

4.24.3.4.3 Interazioni con altre classi

- actorbase::driver::serializer::DeserializationStrategy.

4.24.3.5 actorbase::driver::serializer::DeserializationStrategy

4.24.3.5.1 Descrizione

Interfaccia che offre una strategia generale per deserializzare$_G$ i dati.

4.24.3.5.2 Utilizzo

Questa interfaccia viene utilizzata per l'applicazione del design pattern$_G$ strategy$_G$. Offre un metodo generalizzato per la deserializzazione$_G$ di dati che verrà poi implementato dalle classi che implementeranno questa interfaccia.

4.24.3.5.3 Interazioni con altre classi

- actorbase::driver::serializer::DeserializationContext.

4.24.3.6 actorbase::driver::serializer::RESPDeserialize

4.24.3.6.1 Descrizione

Classe che si occupa di deserializzare i dati da stringhe in formato RESP$_G$.

4.24.3.6.2 Utilizzo

Questa classe viene utilizzata per l'implementazione di una strategia di deserializzazione$_G$ dal formato RESP$_G$.

4.24.3.6.3 Classi ereditate

- actorbase::driver::serializer::DeserializationStrategy.

4.24.3.6.4 Interazioni con altre classi

- actorbase::driver::actorbasedata::RESPArrayString.

4.25 actorbase::driver::actorbasedata

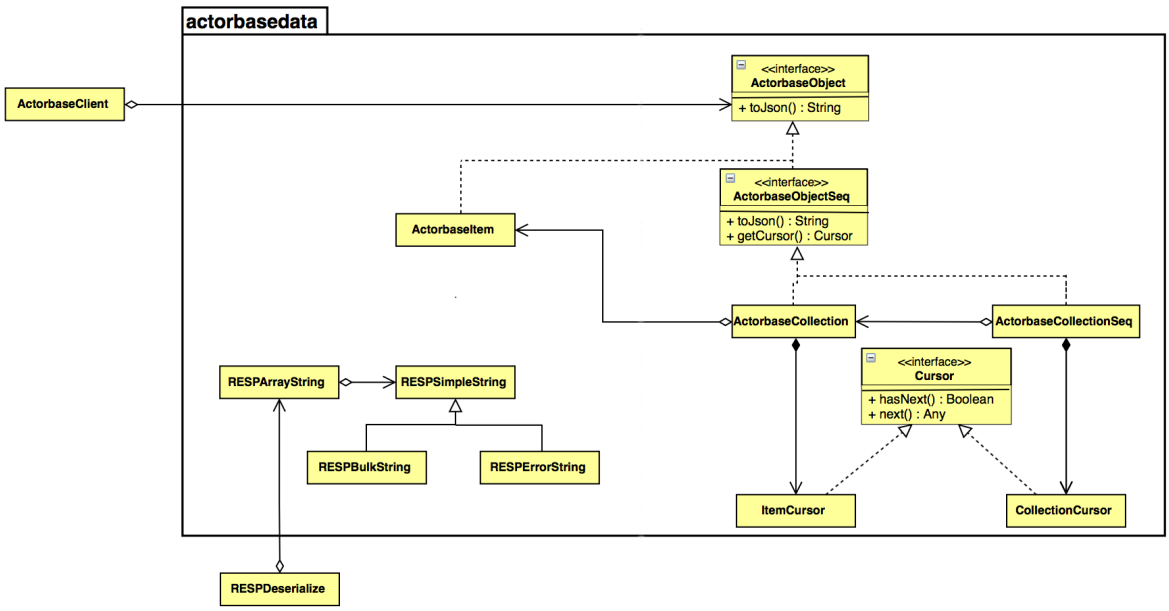

Figura 17: Driver, package actorbasedata e interazioni con package serializer e package client

4.25.1 Descrizione

Package$_G$ che offre le strutture dati necessarie per la comunicazione con il server$_G$, con la cli$_G$ o eventuali programmi esterni.

4.25.2 Interazioni con altre componenti

- actorbase::cli::models;
- actorbase::driver::client;
- actorbase::driver::serializer.

4.25.3 Classi

4.25.3.1 actorbase::driver::actorbasedata::ActorbaseObject

4.25.3.1.1 Descrizione

Interfaccia che rappresenta un oggetto per ritornare i dati ricevuti dal server$_G$ alla cli$_G$ o a un programma esterno.

4.25.3.1.2 Utilizzo

Questa interfaccia viene utilizzata dalle classi che la implementano per rappresentare i dati che verranno ritornati alla cli$_G$ o al programma esterno.

4.25.3.1.3 Interazioni con altre classi

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

4.25.3.2 actorbase::driver::actorbasedata::ActorbaseItem

4.25.3.2.1 Descrizione

Classe che implementa actorbase::driver::actorbasedata::ActorbaseObject in modo da rappresentare un singolo item$_G$.

4.25.3.2.2 Utilizzo

Questa classe viene utilizzata per la rappresentazione di un singolo item$_G$ che verrà restituito alla cli$_G$ o a un programma esterno.

4.25.3.2.3 Classi ereditate

- actorbase::driver::actorbasedata::ActorbaseObject.

4.25.3.3 actorbase::driver::actorbasedata::ActorbaseObjectSeq

4.25.3.3.1 Descrizione

Interfaccia che rappresenta una struttura dati generica per ritornare i dati ricevuti dal server$_G$ alla cli$_G$ o a un programma esterno.

4.25.3.3.2 Utilizzo

Questa interfaccia viene utilizzata dalle classi che la implementano per rappresentare un insieme di collezioni$_G$ o item$_G$.

4.25.3.3.3 Interazioni con altre classi

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

4.25.3.3.4 Classi ereditate

- actorbase::driver::actorbasedata::ActorbaseObject.

4.25.3.4 actorbase::driver::actorbasedata::ActorbaseCollection

4.25.3.4.1 Descrizione

Classe che implementa actorbase::driver::actorbasedata::ActorbaseObjectSeq in modo da rappresentare una singola collezione$_G$.

4.25.3.4.2 Utilizzo

Questa classe viene utilizzata per la rappresentazione di una singola collezione$_G$ vista come insieme di oggetti di tipo actorbase::driver::actorbasedata::ActorbaseItem che verrà restituito alla cli$_G$ o a un programma esterno. Essa offrirà anche un cursore$_G$ di tipo actorbase::driver::actorbasedata::ItemCursor per scorrere gli item$_G$ al suo interno.

4.25.3.4.3 Classi ereditate

- actorbase::driver::actorbasedata::ActorbaseObjectSeq.

4.25.3.4.4 Interazioni con altre classi

- actorbase::driver::actorbasedata::ActorbaseItem;
- actorbase::driver::actorbasedata::ItemCursor.

4.25.3.5 actorbase::driver::actorbasedata::ActorbaseCollectionSeq

4.25.3.5.1 Descrizione

Classe che implementa actorbase::driver::actorbasedata::ActorbaseObjectSeq in modo da rappresentare un insieme di collezioni$_G$.

4.25.3.5.2 Utilizzo

Questa classe viene utilizzata per la rappresentazione di un insieme di collezioni$_G$ viste come insieme di oggetti di tipo actorbase::driver::actorbasedata::ActorbaseCollection che verrà restituito alla cli$_G$ o a un programma esterno. Essa offrirà anche un cursore$_G$ di tipo actorbase::driver::actorbasedata::CollectionCursor per scorrere le collezioni$_G$ al suo interno.

4.25.3.5.3 Classi ereditate

- actorbase::driver::actorbasedata::ActorbaseObjectSeq.

4.25.3.5.4 Interazioni con altre classi

- actorbase::driver::actorbasedata::ActorbaseCollection;
- actorbase::driver::actorbasedata::CollectionCursor.

4.25.3.6 actorbase::driver::actorbasedata::Cursor

4.25.3.6.1 Descrizione

Interfaccia che offre un cursore per lo scorrimento di una struttura dati secondo il design pattern$_G$ iterator$_G$.

4.25.3.6.2 Utilizzo

Questa interfaccia contiene i metodi necessari allo scorrimento di una struttura dati.

4.25.3.7 actorbase::driver::actorbasedata::ItemCursor

4.25.3.7.1 Descrizione

Classe che offre al driver$_G$ un cursore per lo scorrimento di una struttura dati che rappresenta una singola collezione$_G$ implementando l'interfaccia actorbase::driver::actorbasedata::Cursor.

4.25.3.7.2 Utilizzo

Questa classe viene utilizzata dalla classe actorbase::driver::actorbasedata::ActorbaseCollection per scorrere gli item$_G$ interni alla collezione$_G$ rappresentata.

4.25.3.7.3 Classi ereditate

- actorbase::driver::actorbasedata::Cursor.

4.25.3.7.4 Interazioni con altre classi

- actorbase::driver::actorbasedata::ActorbaseCollection.

4.25.3.8 actorbase::driver::actorbasedata::CollectionCursor

4.25.3.8.1 Descrizione

Classe che offre al driver$_G$ un cursore per lo scorrimento di una struttura dati che rappresenta un'insieme di collezioni$_G$ implementando l'interfaccia actorbase::driver::actorbasedata::Cursor.

4.25.3.8.2 Utilizzo

Questa classe viene utilizzata dalla classe actorbase::driver::actorbasedata::ActorbaseCollection per scorrere le collezioni$_G$ interni all'insieme di collezioni$_G$ rappresentato.

4.25.3.8.3 Classi ereditate

- actorbase::driver::actorbasedata::Cursor.

4.25.3.8.4 Interazioni con altre classi

- actorbase::driver::actorbasedata::ActorbaseCollectionSeq.

4.25.3.9 actorbase::driver::actorbasedata::RESPArrayString

4.25.3.9.1 Descrizione

Questa classe rappresenta un array$_G$ di actorbase::driver::actorbasedata::RESPSimpleString.

4.25.3.9.2 Utilizzo

Questa classe viene utilizzata da actorbase::driver::serializer::RESPDeserialize per deserializzare$_G$ la risposta ricevuta dal server$_G$.

4.25.3.9.3 Interazioni con altre classi

- actorbase::driver::actorbasedata::RESPSimpleString;
- actorbase::driver::serializer::RESPDeserialize.

4.25.3.10 actorbase::driver::actorbasedata::RESPSimpleString

4.25.3.10.1 Descrizione

Questa classe rappresenta una stringa senza l'indicazione di quanto sia lunga la stringa. Essa termina quando si trova un carattere CR (Carriage Return)$_G$ e LF (Line Feed)$_G$.

4.25.3.10.2 Utilizzo

Questa classe viene utilizzata per deserializzare$_G$ principalmente le risposte ottenute dal server$_G$ che non comprendano item$_G$ o collezioni$_G$.

4.25.3.10.3 Interazioni con altre classi

- actorbase::driver::actorbasedata::RESPArrayString.

4.25.3.11 actorbase::driver::actorbasedata::RESPBulkString

4.25.3.11.1 Descrizione

Questa classe rappresenta una stringa con l'indicazione di quanto sia lunga la stringa. Può contenere qualsiasi carattere.

4.25.3.11.2 Utilizzo

Questa classe viene utilizzata per deserializzare$_G$ principalmente le risposte ottenute dal server$_G$ che comprendano item$_G$ o collezioni$_G$.

4.25.3.11.3 Classi ereditate

- actorbase::driver::actorbasedata::RESPSimpleString.

4.25.3.12 actorbase::driver::actorbasedata::RESPErrorString

4.25.3.12.1 Descrizione

Questa classe rappresenta una stringa di tipo actorbase::driver::actorbasedata::RESPSimpleString con i primi caratteri che indicano il tipo di errore.

4.25.3.12.2 Utilizzo

Questa classe viene utilizzata per deserializzare$_G$ principalmente le risposte ottenute dal server$_G$ che comprendano messaggi di errore.

4.25.3.12.3 Classi ereditate

- actorbase::driver::actorbasedata::RESPSimpleString.

4.26 actorbase::driver::exceptions

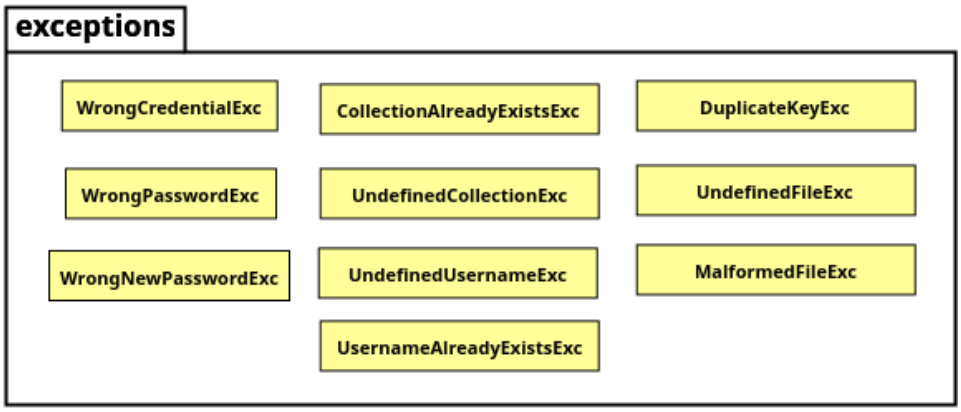

Figura 18: Driver:Package exceptions

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.

4.26.3.3.3 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

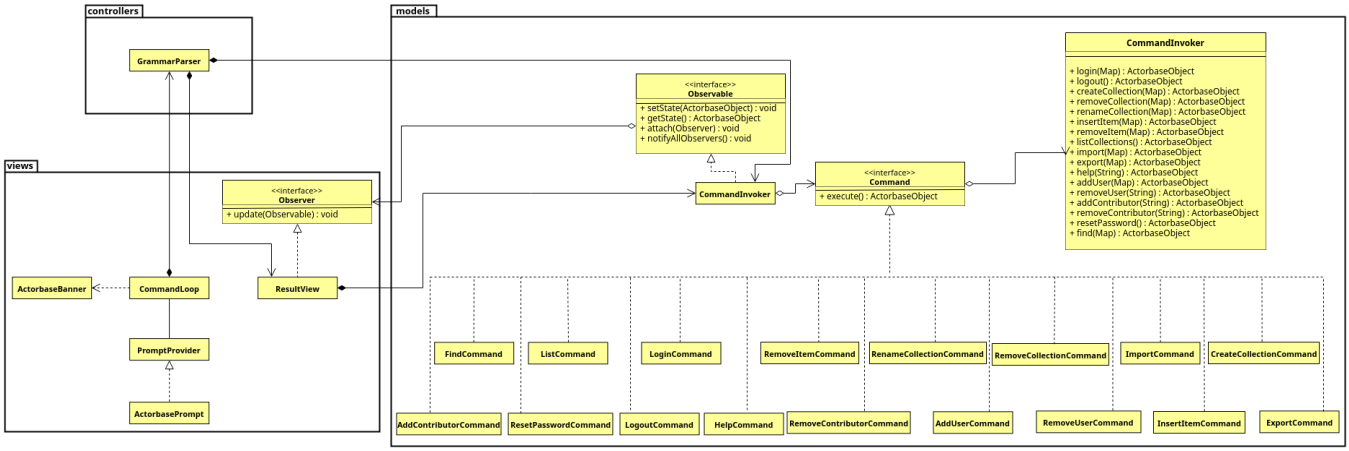

Figura 19: CLI, architettura MVC variante push model

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.

4.28 actorbase::cli::views

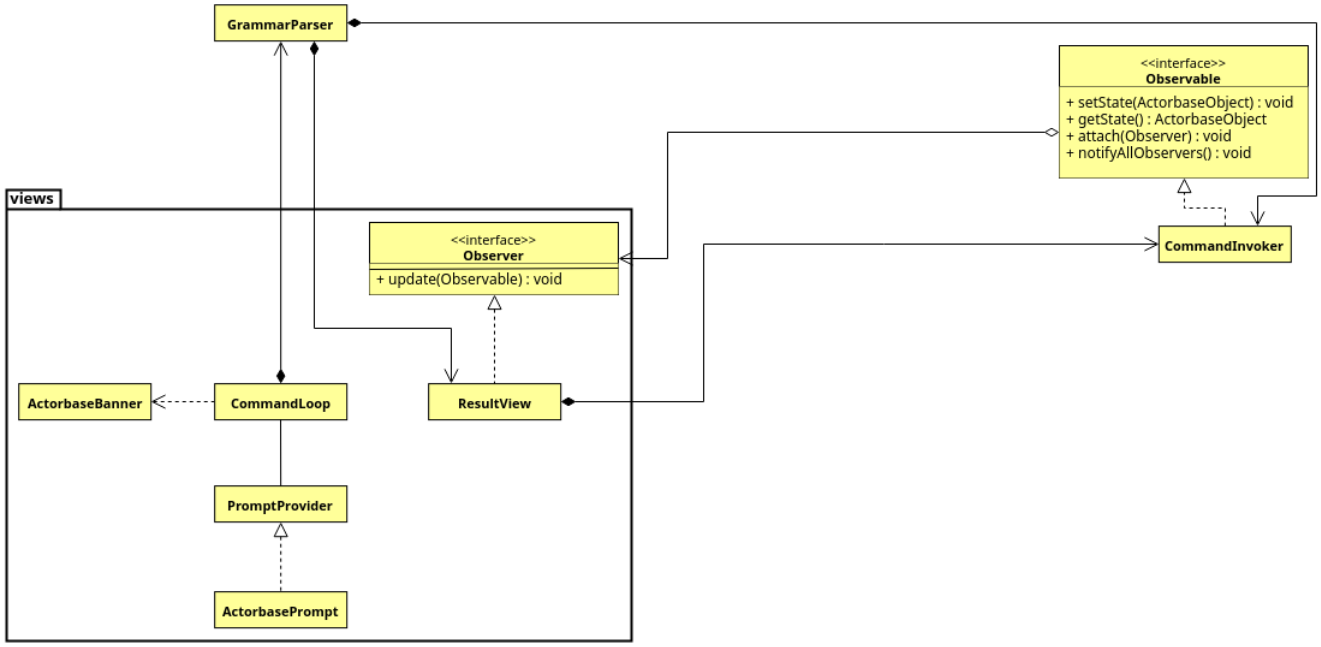

Figura 20: CLI, views package, interazioni con Controllers e Models

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

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

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::ListCommand

4.30.2.7.1 Descrizione

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

4.30.2.7.2 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$_G$.

4.30.2.7.3 Classi ereditate

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

4.30.2.8 actorbase::cli::models::AddContributorCommand

4.30.2.8.1 Descrizione

Classe per il comando di aggiunta collaboratore a una collezione.

4.30.2.8.2 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.

4.30.2.8.3 Classi ereditate

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

4.30.2.9 actorbase::cli::models::LoginCommand

4.30.2.9.1 Descrizione

Classe per il comando di login.

4.30.2.9.2 Utilizzo

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

4.30.2.9.3 Classi ereditate

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

4.30.2.10 actorbase::cli::models::RemoveCollectionCommand

4.30.2.10.1 Descrizione

Classe per il comando di rimozione una collezione$_G$.

4.30.2.10.2 Utilizzo

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

4.30.2.10.3 Classi ereditate

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

4.30.2.11 actorbase::cli::models::RemoveContributorCommand

4.30.2.11.1 Descrizione

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

4.30.2.11.2 Utilizzo

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

4.30.2.11.3 Classi ereditate

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

4.30.2.12 actorbase::cli::models::RenameCollectionCommand

4.30.2.12.1 Descrizione

Classe per il comando rinominazione di una collezione$_G$.

4.30.2.12.2 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.

4.30.2.12.3 Classi ereditate

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

4.30.2.13 actorbase::cli::models::AddUserCommand

4.30.2.13.1 Descrizione

Classe per il comando di aggiunta di un utente.

4.30.2.13.2 Utilizzo

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

4.30.2.13.3 Classi ereditate

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

4.30.2.14 actorbase::cli::models::HelpCommand

4.30.2.14.1 Descrizione

Classe per la richiesta di aiuto.

4.30.2.14.2 Utilizzo

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

4.30.2.15 actorbase::cli::models::RemoveItemCommand

4.30.2.15.1 Descrizione

Classe per il comando di rimozione di un item$_G$.

4.30.2.15.2 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.

4.30.2.15.3 Classi ereditate

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

4.30.2.16 actorbase::cli::models::RemoveUserCommand

4.30.2.16.1 Descrizione

Classe per il comando di rimozione di un utente.

4.30.2.16.2 Utilizzo

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

4.30.2.16.3 Classi ereditate

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

4.30.2.17 actorbase::cli::models::ImportCommand

4.30.2.17.1 Descrizione

Classe per il comando di importazione da un file JSON$_G$.

4.30.2.17.2 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.

4.30.2.17.3 Classi ereditate

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

4.30.2.18 actorbase::cli::models::InsertItemCommand

4.30.2.18.1 Descrizione

Classe per il comando di inserimento di un item$_G$ in una collezione$_G$.

4.30.2.18.2 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.

4.30.2.18.3 Classi ereditate

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

4.30.2.19 actorbase::cli::models::CreateCollectionCommand

4.30.2.19.1 Descrizione

Classe per il comando di creazione di una nuova collezione$_G$.

4.30.2.19.2 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.

4.30.2.19.3 Classi ereditate

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

4.30.2.20 actorbase::cli::models::ExportCommand

4.30.2.20.1 Descrizione

Classe per il comando di esportazione in file JSON$_G$.

4.30.2.20.2 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.

4.30.2.20.3 Classi ereditate

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

4.30.2.21 actorbase::cli::models::CommandReceiver

4.30.2.21.1 Descrizione

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

4.30.2.21.2 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.

4.30.2.21.3 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. É 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.

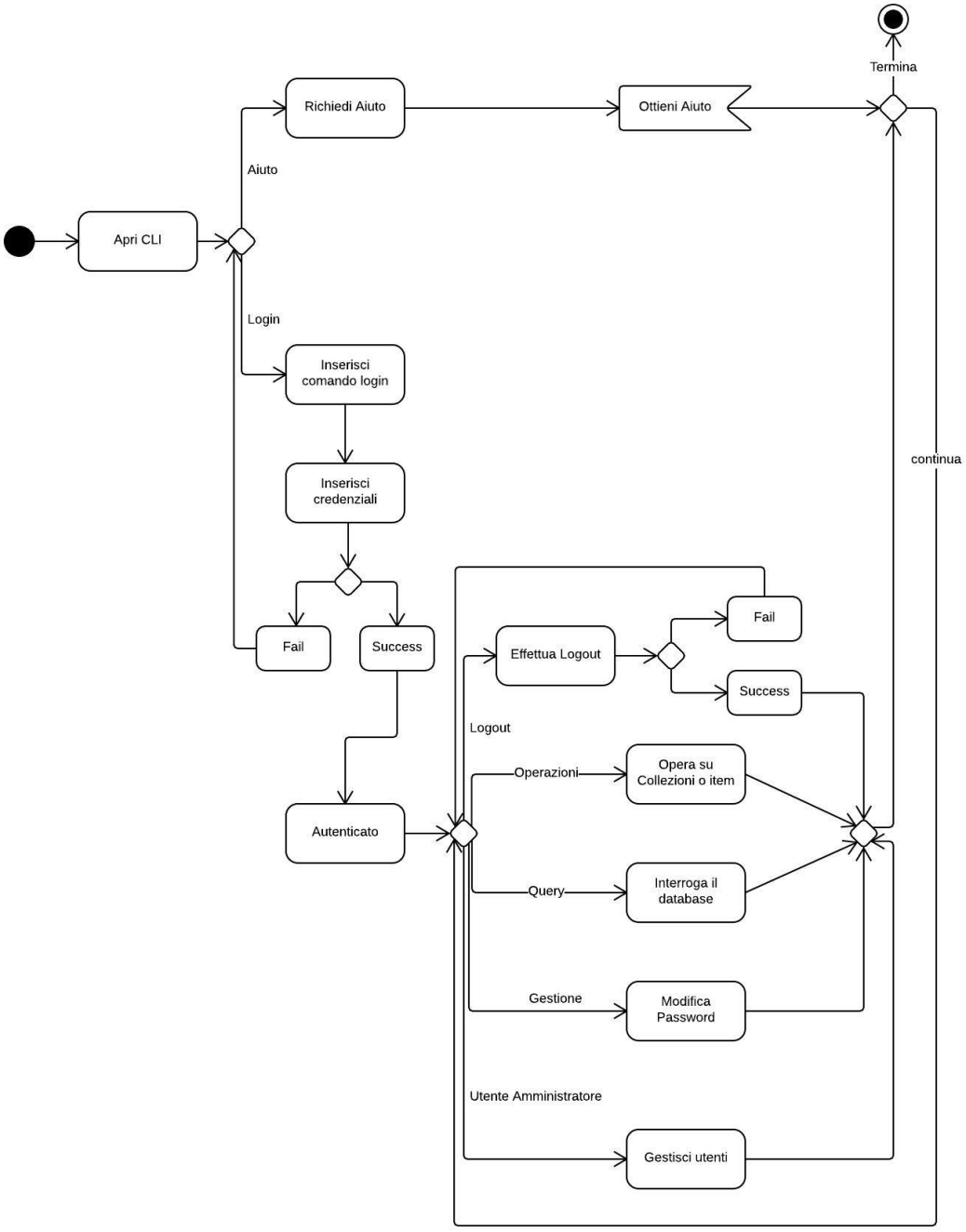

Figura 21: Diagrammi attività - Visione generale

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.

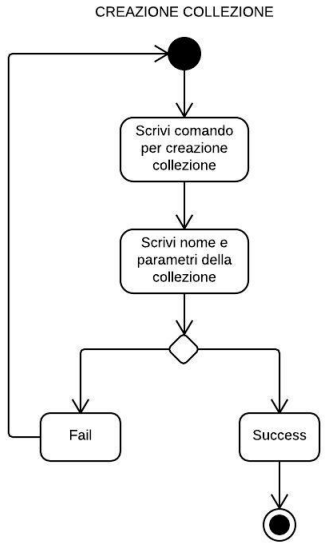

Figura 22: Diagrammi attività - Creazione collezione

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.

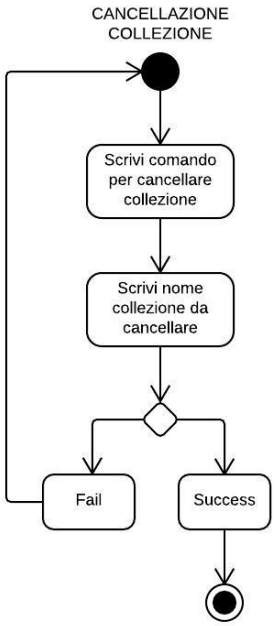

Figura 23: Diagrammi attività - Cancellazione collezione

5.2.3 Visualizza collezioni

Questo tipo di operazione permette di visualizzare le collezioni presenti all'interno dal 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.

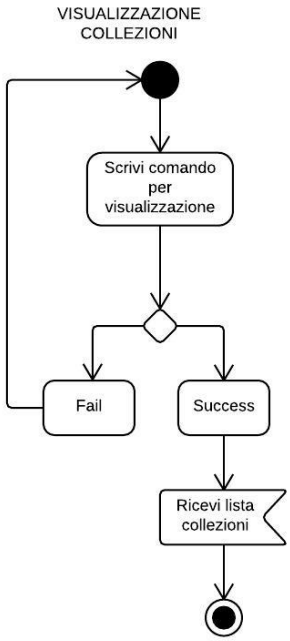

Figura 24: Diagrammi attività - Visualizzazione collezioni

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.

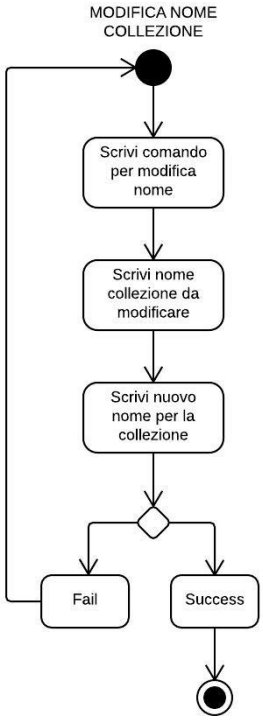

Figura 25: Diagrammi attività - Modifica nome collezione

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.

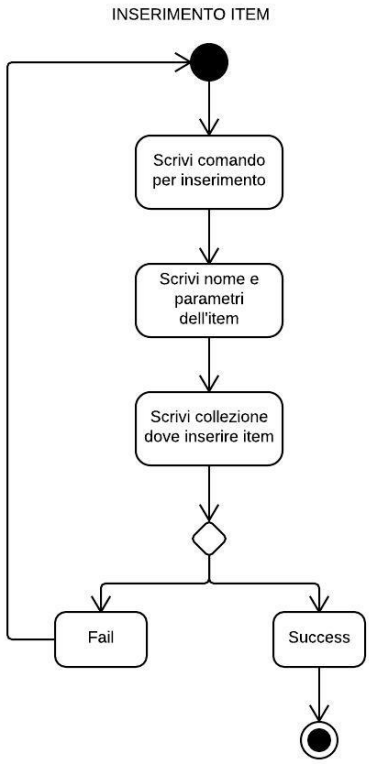

Figura 26: Diagrammi attività - Inserimento item

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.

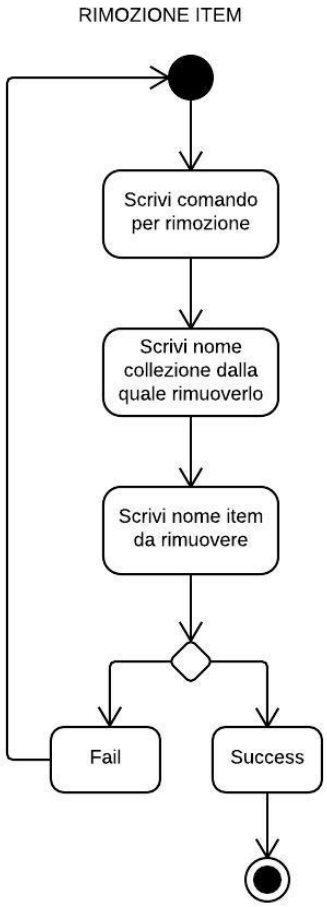

Figura 27: Diagrammi attività - Rimozione item

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.

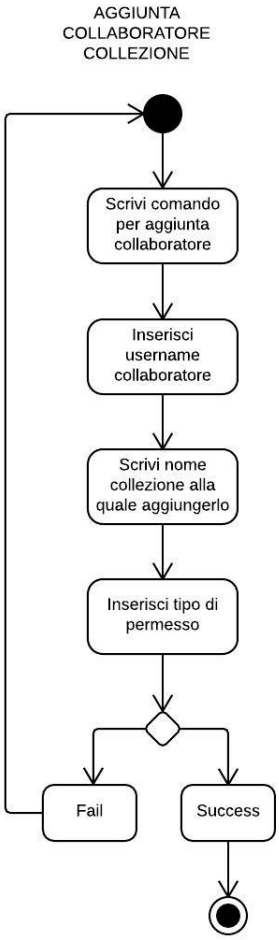

Figura 28: Diagrammi attività - Aggiunta collaboratore

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.

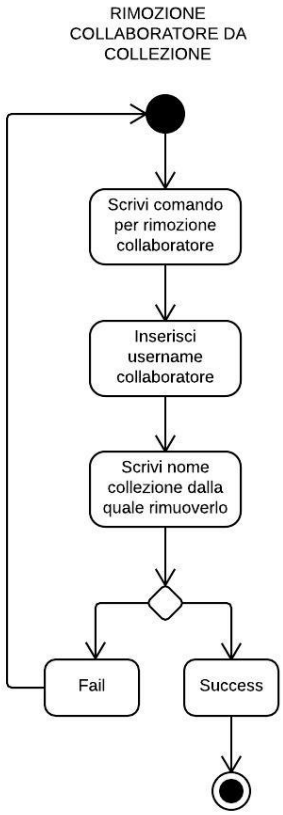

Figura 29: Diagrammi attività - Rimozione collaboratore

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.

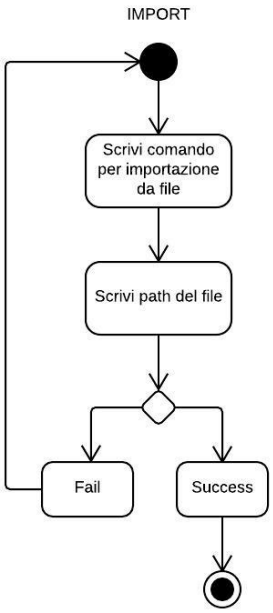

Figura 30: Diagrammi attività - Import

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.

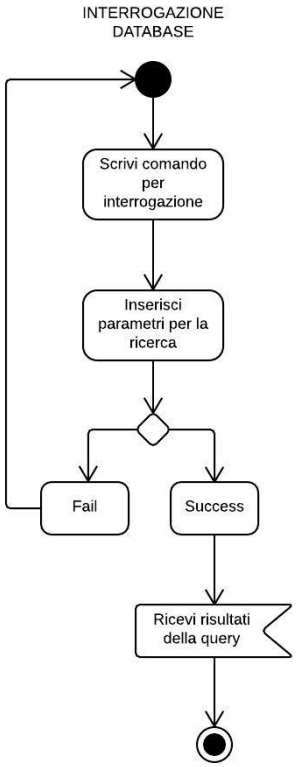

Figura 31: Diagrammi attività - Interrogazione del database$_G$

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.

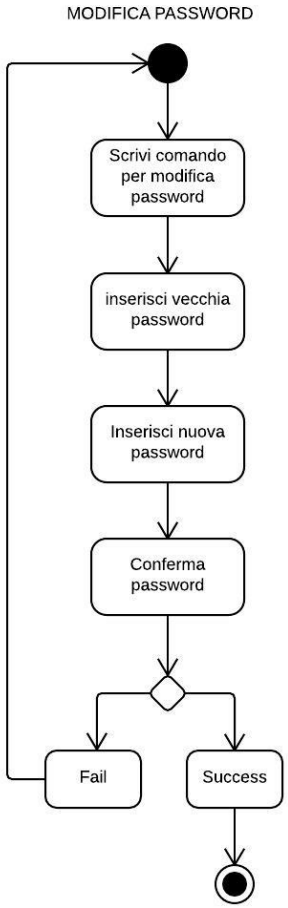

Figura 32: Diagrammi attività - Modifica password

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 operazione andranno a buon termine se l'utente ha scritto correttamente il comando altrimenti verrà visualizzato un messaggio di errore.

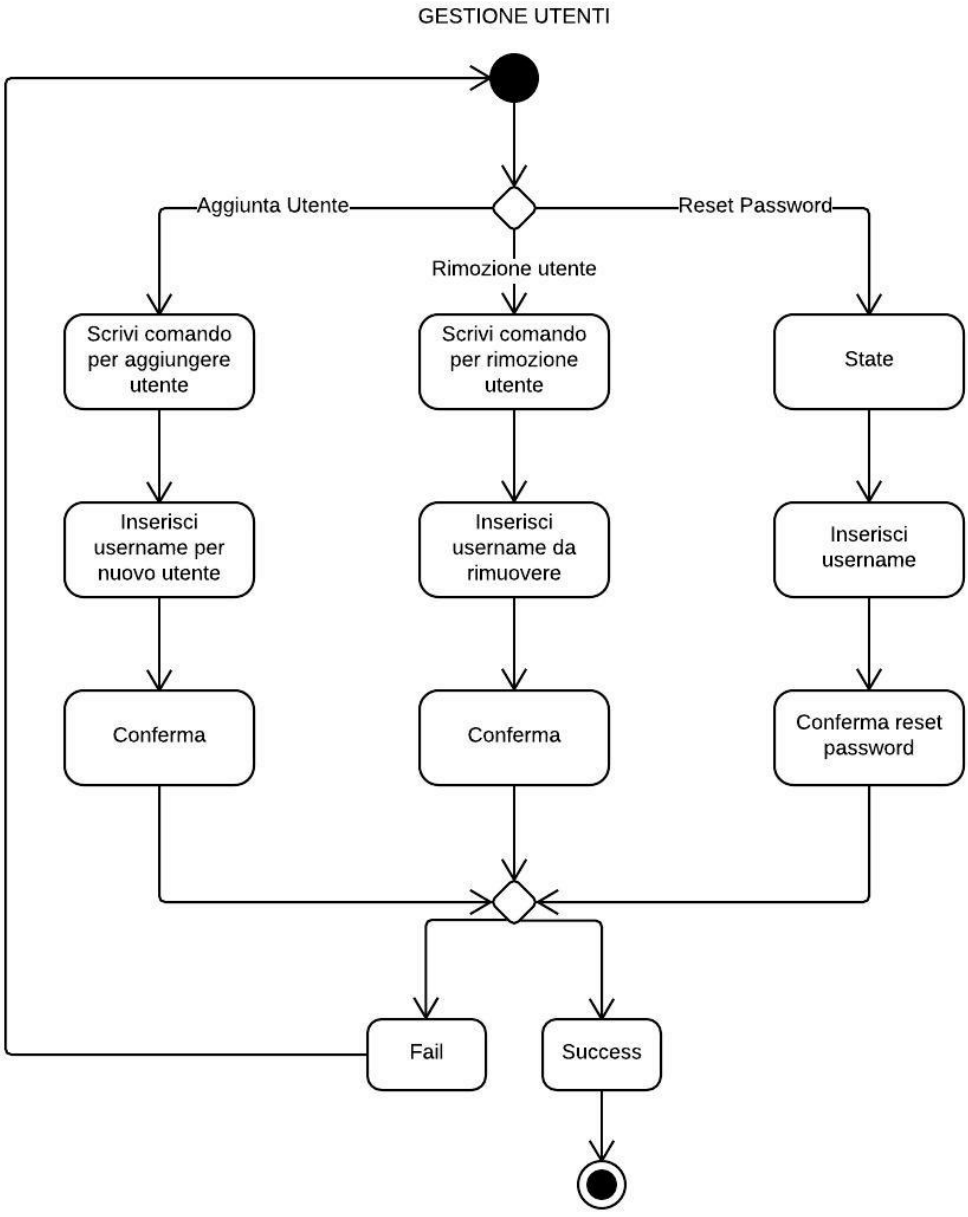

Figura 33: Diagrammi attività - Gestione utenti

6 Diagrammi di Sequenza

In questa sezione, vengono illustrati i diagrammi di sequenza che descrivono le interazioni tra le varie componenti del sistema e in particolare lo scambio di messaggi che avviene tra gli attori nella componente server di actorbase.
Sono stati creati dai *Progettisti* diversi diagrammi con diversi livelli di astrazione.
Per semplificare i diagrammi, evitare ripetitività e migliorare la leggibilità di questi, nella maggior parte dei seguenti l'input sarà dato dalla componente Driver$_G$. In un utilizzo reale sarà un utente che utilizza il Driver$_G$ o la CLI$_G$ che darà l'input iniziale.

6.1 Interazioni generali tra componenti

Questo diagramma rappresenta l'interazione generale tra le tre componenti che compongono il sistema actorbase.
L'utente è connesso alla CLI$_G$ e inserisce un comando. A questo punto la componente CLI$_G$ (se non è instanziato un Driver$_G$ lo crea per poi utilizzarlo) utilizza il Driver$_G$ per richiamare il metodo relativo al comando inserito dall'utente.
Il Driver$_G$ provvederà a tradurre il comando coi relativi parametri immessi dall'utente in una stringa RESP e la invierà al server attraverso un socket tcp$_G$. Il server provvederà a fare le elaborazioni richieste attraverso l'uso degli attori e quando queste saranno finite risponderà al Driver$_G$ mandando via socket tcp$_G$ una stringa RESP che a sua volta provvederà a trasformarla in un actorbase object e lo restituirà al suo utilizzatore.

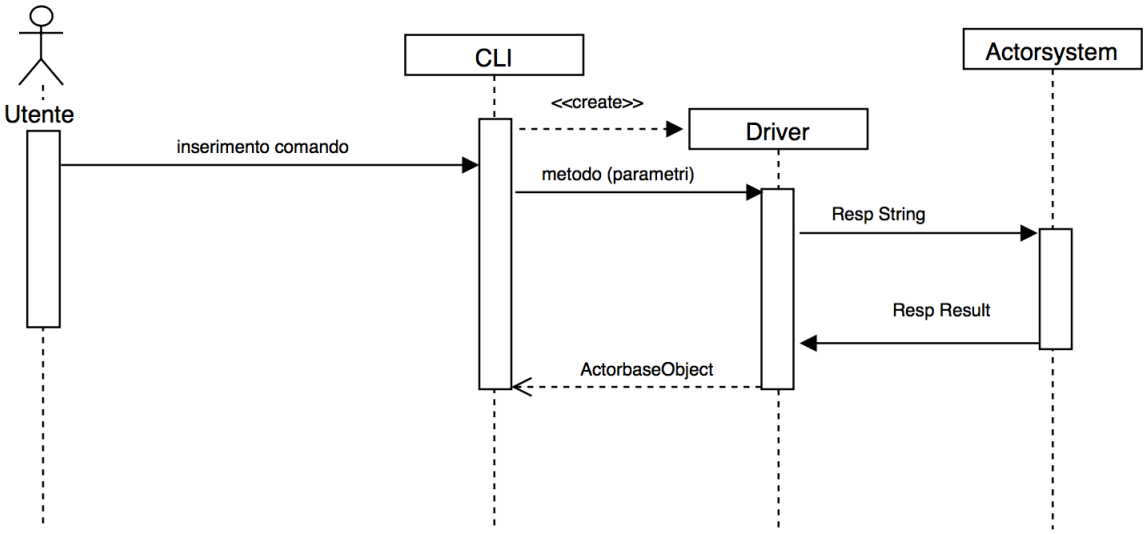

Figura 34: Diagramma di sequenza - Interazioni generali tra componenti

6.2 Inserimento di un Item

Questo diagramma rappresenta l'interazione tra le componenti nel caso di una richiesta di inserimento di un item$_G$.
La componente actorbase::driver::client::Connection manda tramite socket tcp$_G$ al actorbase::actorsystem::clientactor::Clien una stringa RESP contenente il comando di inserimento item$_G$ con i parametri.
Da questo componente parte il messaggio di Insert(key, value) ad un actorbase::actorsystem::clientactor::MainActor scelto in base ad una politica di routing$_G$. Il MainActor manderà un messaggio allo actorbase::actorsystem::clientactor::Store relativo il quale manderà un messaggio allo actorbase::actorsystem::clientactor::StoreKeeper che dovrà salvarsi l'informazione sulla propria mappa.
A questo punto lo Storekeeper manderà un messaggio al ClientActor$_G$ con la risposta (positiva o negativa che sia) il quale risponderà sempre tramite socket tcp$_G$ al Driver$_G$ con una stringa RESP.

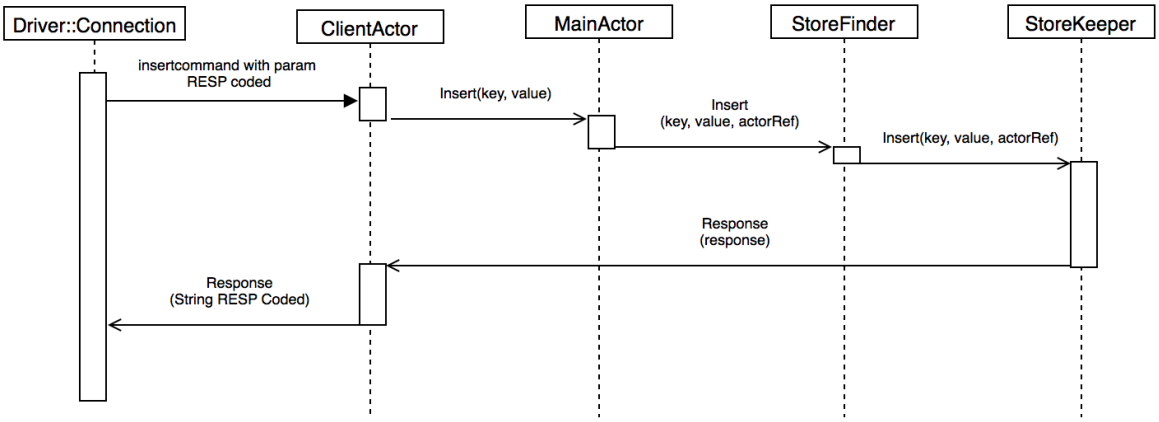

Figura 35: Diagramma di sequenza - Inserimento di un item

6.3 Inserimento di un Item con Storekeeper pieno

Questo diagramma rappresenta l'interazione tra le componenti nel caso di una richiesta di inserimento di un item$_G$ in uno Storekeeper con una mappa piena.
La componente actorbase::driver::client::Connection manda tramite socket tcp$_G$ al actorbase::actorsystem::clientactor::Clien una stringa RESP contenente il comando di inserimento item$_G$ con i parametri.
Da questo componente parte il messaggio di Insert(key, value) ad un actorbase::actorsystem::clientactor::MainActor scelto in base ad una politica di routing$_G$. Il MainActor manderà un messaggio allo actorbase::actorsystem::clientactor::Store relativo il quale manderà un messaggio allo actorbase::actorsystem::clientactor::StoreKeeper che essendo pieno manda un messaggio DuplicateRequestSK contenente metà della propria mappa e una ActorRef del proprio parent ad un actorbase::actorsystem::clientactor::Manager. Il Manager inoltra il messaggio DuplicateRequestSK al parent dello Storekeeper precedentemente coinvolto. Questo Storefinder infine creerà un nuovo Storekeeper con il pezzo di mappa ricevuto.

A questo punto il nuovo Storekeeper manderà un messaggio al ClientActor$_G$ con la risposta (positiva o negativa che sia) il quale risponderà sempre tramite socket tcp$_G$ al Driver$_G$ con una stringa RESP.

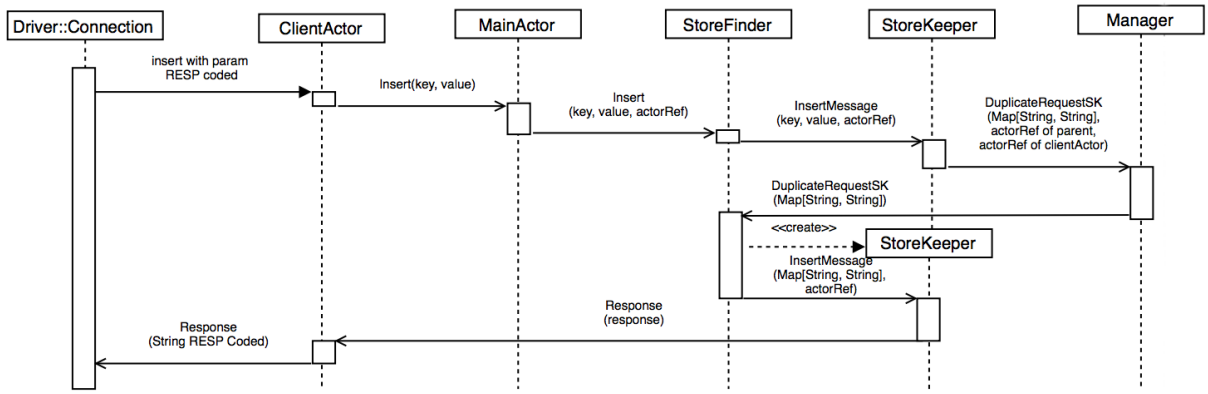

Figura 36: Diagramma di sequenza - Inserimento di un item

6.4 Autenticazione

Questo diagramma rappresenta l'interazione tra le componenti nel caso di una richiesta di autenticazione di un utente.
La componente actorbase::driver::client::Connection manda tramite socket tcp$_G$ al actorbase::actorsystem::clientactor::Clien una stringa RESP contenente il comando di inserimento item$_G$ con i parametri.
Da questo componente parte il messaggio di GetItemFrom(UsersCollection, Username) ad un actorbase::actorsystem::clienta scelto in base ad una politica di routing$_G$. Il MainActor manderà un messaggio allo actorbase::actorsystem::clientactor::Store relativo il quale manderà un messaggio getPassword(username) allo actorbase::actorsystem::clientactor::UserKeeper.
Quest'ultimo risponderà al ClientActor$_G$ con un messaggio loginResponse contenente la password relativa allo username richiesto.
Il ClientActor$_G$ eseguirà un confronto tra la password immessa dall'utente e la password ritornata dallo Userkeeper. In caso positivo cambierà il proprio stato e risponderà al Driver$_G$ con un messaggio che indica un esito positivo. In caso negativo resterà nello stato attuale e manderà un messaggio al Driver$_G$ che indica un esito negativo.

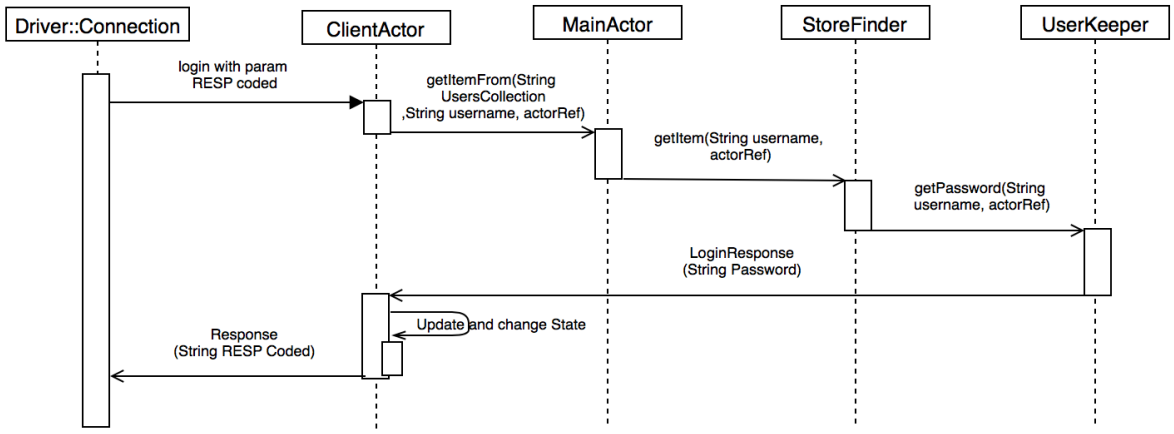

Figura 37: Diagramma di sequenza - Autenticazione

6.5 Salvataggio su filesystem

Questo diagramma rappresenta l'interazione tra le componenti nel caso venga effettuato un salvataggio dei dati su filesystem.
La componente actorbase::driver::client::Connection manda tramite socket tcp_G al actorbase::actorsystem::clientactor::Clien una stringa RESP contenente il comando di inserimento $item_G$ con i parametri.
Da questo componente parte il messaggio di Insert(key, value) ad un actorbase::actorsystem::clientactor::MainActor scelto in base ad una politica di $routing_G$. Il MainActor manderà un messaggio allo actorbase::actorsystem::clientactor::Store relativo il quale manderà un messaggio allo actorbase::actorsystem::clientactor::StoreKeeper che dovrà salvarsi l'informazione sulla propria mappa.
A questo punto lo Storekeeper manderà un messaggio al Warehouseman di tipo Save e successivamente manderà un messaggio al $ClientActor_G$ con la risposta (positiva o negativa che sia) il quale risponderà sempre tramite socket tcp_G al $Driver_G$ con una stringa RESP. Nel frattempo l'$attore_G$ di tipo Warehouseman eseguirà un salvataggio su disco dei propri dati previa $serializzazione_G$.
Si noti che la frequenza con cui gli attori di tipo Warehousename eseguiranno l'aggiornamento delle proprie strutture dati e il salvataggio sarà specificata tramite un parametro di configurazione del sistema.

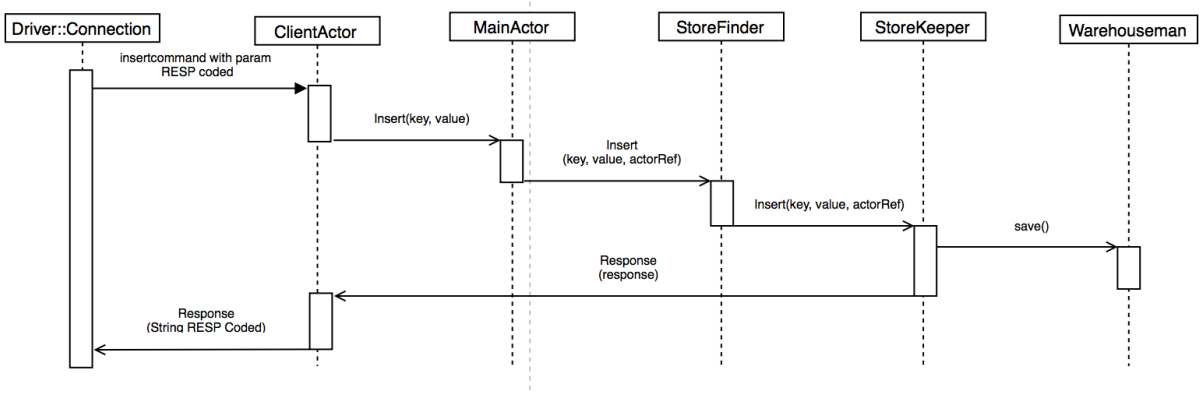

Figura 38: Diagramma di sequenza - Inserimento di un item

6.6 Aggiornamento di un attore ninja

Questo diagramma rappresenta l'interazione tra le componenti nel caso venga effettuato un aggiornamento dei dati su un attore$_G$ Ninja.
La componente actorbase::driver::client::Connection manda tramite socket tcp$_G$ al actorbase::actorsystem::clientactor::Clien una stringa RESP contenente il comando di inserimento item$_G$ con i parametri.
Da questo componente parte il messaggio di Insert(key, value) ad un actorbase::actorsystem::clientactor::MainActor scelto in base ad una politica di routing$_G$. Il MainActor manderà un messaggio allo actorbase::actorsystem::clientactor::Store relativo il quale manderà un messaggio allo actorbase::actorsystem::clientactor::StoreKeeper che dovrà salvarsi l'informazione sulla propria mappa.
A questo punto lo Storekeeper manderà un messaggio al Ninja di tipo Update contenente la struttura dati aggiornata e successivamente manderà un messaggio al ClientActor$_G$ con la risposta (positiva o negativa che sia) il quale risponderà sempre tramite socket tcp$_G$ al Driver$_G$ con una stringa RESP. Nel frattempo l'attore$_G$ di tipo Ninja eseguirà un aggiornamento della propria struttura dati.
Si noti che la frequenza di aggiornamento degli attori di tipo ninja sarà specificata tramite un parametro di configurazione del sistema.

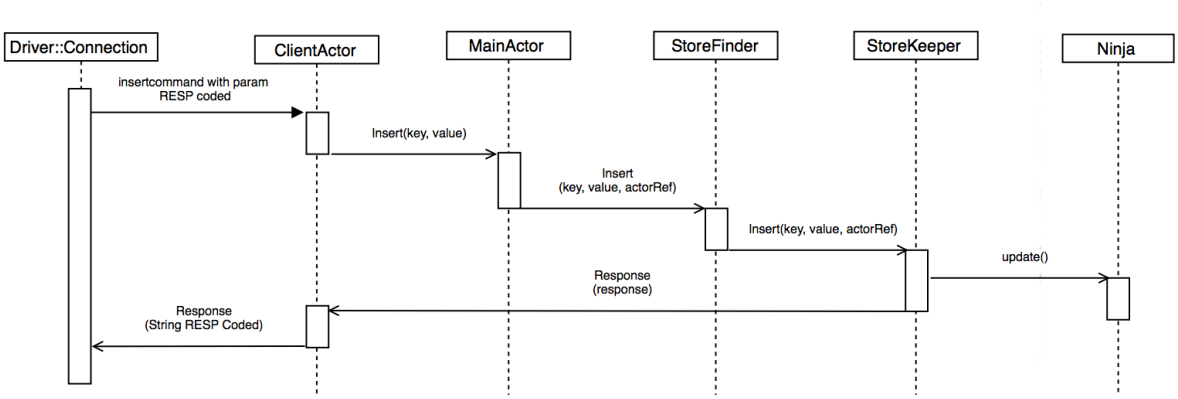

Figura 39: Diagramma di sequenza - Aggiornamento di un attore ninja

6.7 Aggiunta collaboratore (in sola lettura)

Questo diagramma rappresenta l'interazione tra le componenti nel caso di una richiesta di autenticazione di un utente.
La componente actorbase::driver::client::Connection manda tramite socket tcp_G al actorbase::actorsystem::clientactor::Clien una stringa RESP contenente il comando di aggiunta collaboratore $item_G$ con i parametri.
Da questo componente parte il messaggio di addCollaborator(collection, permission, username) ad un actorbase::actorsystem::clientactor::MainActor scelto in base ad una politica di $routing_G$. Il MainActor manderà un messaggio allo actorbase::actorsystem::clientactor::StoreFinder relativo il quale manderà un messaggio addReadCollection(collection) allo actorbase::actorsystem::clientactor::UserKeeper. Quest'ultimo dovrà aggiungere alla mappa delle collezioni di cui l'utente relativo ha accesso in sola lettura la collezione inviatagli. Poi manderà al $ClientActor_G$ un messaggio UpdateReadCollections(collection) contenente la collezione da aggiungere.
Il $ClientActor_G$ effettuerà l'aggiornamento e risponderà al $Driver_G$ con un messaggio che indica un esito positivo. In caso negativo resterà nello stato attuale e manderà un messaggio al $Driver_G$ che indica un esito negativo.
Nel caso di aggiunta di un collaboratore con permessi di lettura e scrittura l'interazione tra gli attori è la stessa con differenza di qualche messaggio specifico.

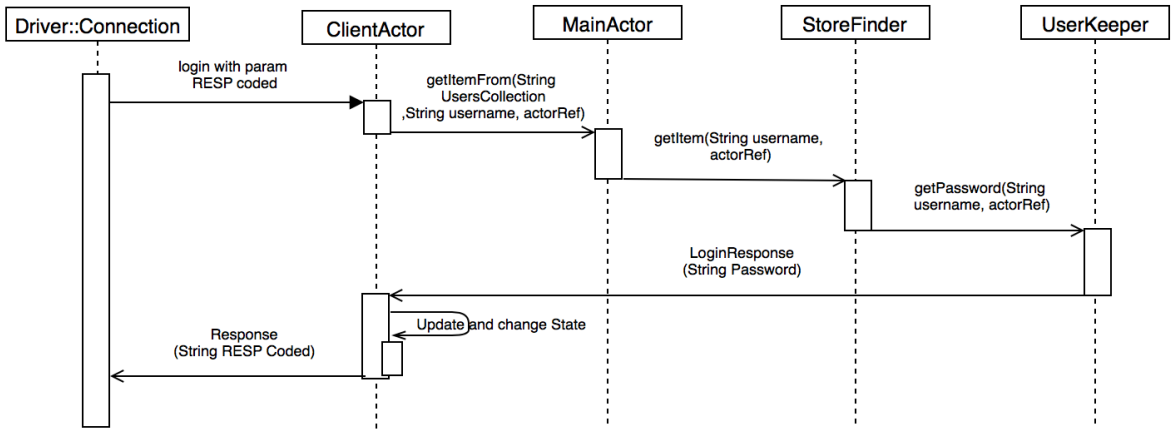

Figura 40: Diagramma di sequenza - Autenticazione

6.8 Connessione al server tramite Driver

Questo diagramma rappresenta l'interazione tra le componenti nel caso di una richiesta di connessione al server di un utente tramite uso del Driver$_G$.
Si noti che tramite utilizzo della CLI$_G$ il diagramma sarebbe lo stesso con l'aggiunta della componente CLI$_G$.
L'utente crea un oggetto di tipo actorbase::driver::client::ActorbaseClient il quale crea a sua volta un oggetto di tipo actorbase::driver::client::Connection. actorbase::driver::client::Connection instaura una connessione tcp$_G$ con la componente actorbase::actorsystem::tcpserver::TCPServer la quale crea un attore$_G$ di tipo actorbase::actorsystem::clientactor::ClientActor per gestire l'interazione tra Driver$_G$ e server. A creazione avvenuta risponde al Driver$_G$ il quale a cascata informa l'utente dell'avvenuta connessione.
Dal diagramma possiamo vedere che le richieste successivamente inserite dall'utente verranno gestite dal proprio ClientActor$_G$.

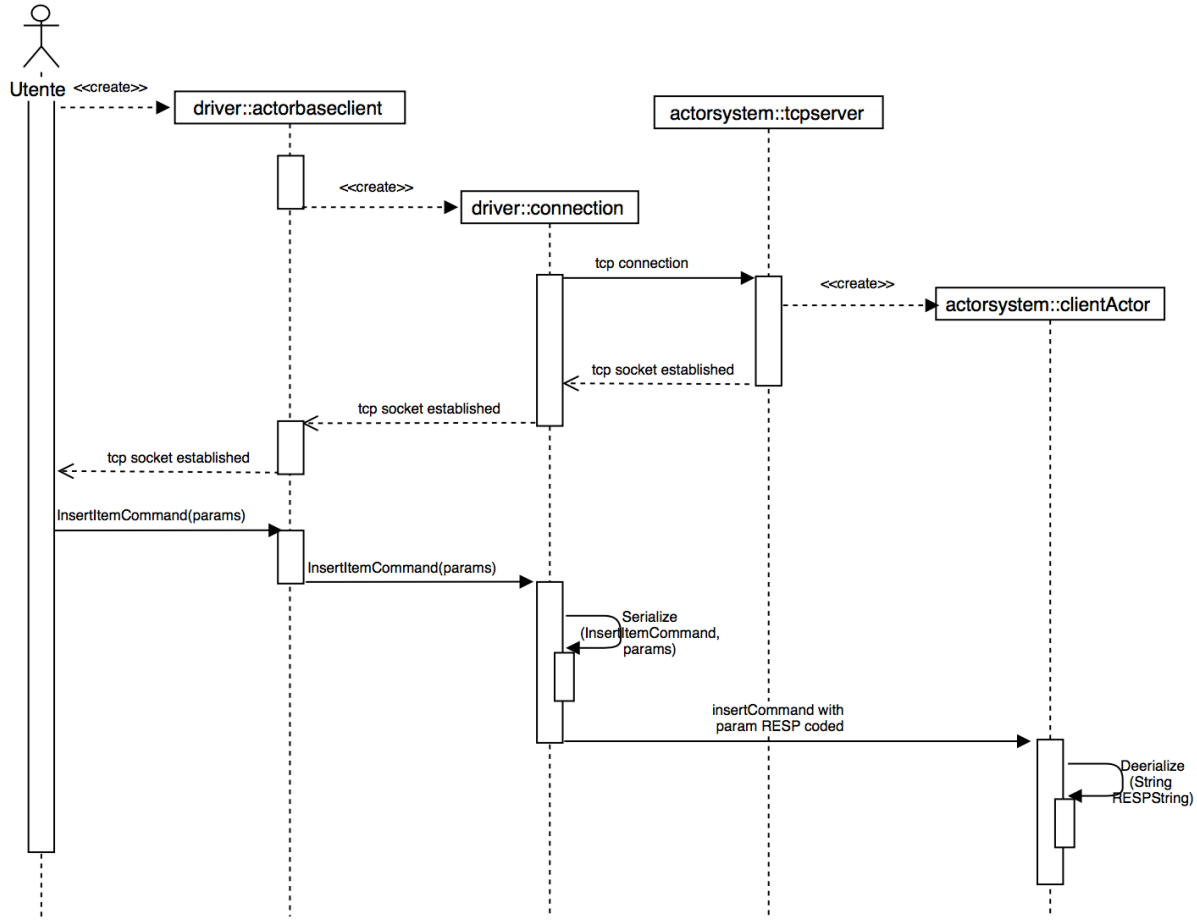

Figura 41: Diagramma di sequenza - Connessione al server

7 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 **Actorbase**.

7.1 Design Pattern Architetturali

7.1.1 MVC

- **Scopo:** È stato scelto l'utilizzo del pattern Model$_G$ View$_G$ 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 estrapolato 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$_G$.

7.2 Design Pattern Creazionali

7.2.1 Singleton

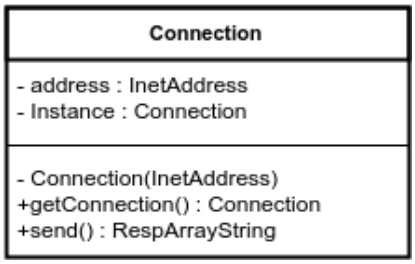

Figura 42: Connection Singleton

- **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$_G$.

7.3 Design Pattern Strutturali

7.3.1 Facade

- **Scopo:** Il pattern Facade$_G$ viene usato per fornire un'interfaccia unica e semplificata a più classi;
- **Utilizzo:** Viene utilizzato all'interno della componente driver$_G$ per l'esecuzione e l'invio al server$_G$ dei comandi mediante la classe CommandRunner, la quale mediante l'istanza Singleton$_G$ di Connection invia e riceve i comandi al server$_G$.

7.4 Design Pattern Comportamentali

7.4.1 Command Pattern

- **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`
- `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$_G$.

7.4.2 Iterator Pattern

- **Scopo:** Fornisce un sistema di navigazione attraverso gli elementi di una generica struttura dati contenitrice, senza esporre i dettagli dell'implementazione e della struttura interna del contenitore.
- **Utilizzo:** Viene utilizzato nella componente driver$_G$, più specificamente nel package$_G$ `actorbaseobject` per permettere l'iterazione degli oggetti restituiti dalla componente `actorsystem`.
  Le classi che implementano l'interfaccia `cursor` sono:
  - `CollectionCursor`
  - `ItemCursor`

7.4.3 Observer Pattern

- **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.

7.4.4 Strategy

- **Scopo:** Viene usato per isolare più algoritmi all'interno di oggetti e ne permette la modifica dinamica.
- **Utilizzo:** Nella componente `actorsystem` è stato utilizzato per offrire più possibilità di serializzazione$_G$ e deserializzazione$_G$ sia in ambito di persistenza su disco che in ambito comunicazioni tra server$_G$ ed esterno. Nella componente driver$_G$ viene offerta una singola strategia, dedicata esclusivamente alla comunicazione con la componente `actorsystem`, con la scelta di mantenere il Design Pattern$_G$ in vista di eventuali estensioni. Contesti di utilizzo:
  - **driver:**
    - Dopo che la classe `Connection` riceve i parametri da inviare è possibile selezionare mediante `SerializationContext`, una delle strategie di serializzazione$_G$ tra:
      - `RESPSerialize`
    - In ricezione dei dati all'interno della classe `Connection`, utilizzando `DeserializationContext` è possibile selezionare una delle strategie di deserializzazione$_G$ tra:
      - `RESPDeserialize`
  - **actorsystem:**
    - All'interno del package$_G$ `clientactor`, la classe utilizzata per la gestione dei messaggi in ingresso `RESPInput` è possibile selezionare mediante `DeserializationContext`, una delle strategie di deserializzazione$_G$ tra:
      - `RESPDeserialization`
      - `PickleDeserialization`
    - All'interno del package$_G$ `clientactor`, la classe utilizzata per la gestione dei messaggi in uscita `Response` è possibile selezionare mediante `SerializationContext`, una delle strategie di serializzazione$_G$ tra:
      - `RESPSerialization`
      - `PickleSerialization`
    - All'interno del package$_G$ `warehouseman`, la classe utilizzata per la gestione del salvataggio in persistenza dei dati `Save` è possibile selezionare mediante `SerializationContext`, una delle strategie di serializzazione$_G$ tra:
      - `PickleSerialization`

  - `RESPSerialization`

* All'interno del package$_G$ `warehouseman`, la classe utilizzata per la gestione della lettura da persistenza dei dati `Init` è possibile selezionare mediante `DeserializationContext`, una delle strategie di serializzazione$_G$ tra:
  - `PickleDeserialization`
  - `RESPDeserialization`

8 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.

9 Tracciamento

9.1 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 OBF1.1.10.2.3 OBF1.1.10.2.3.1 OBF1.1.10.2.3.2 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 OBF1.1.2.3 OBF1.1.2.4 OBF1.1.2.5 OBF1.1.2.6
actorbase::actorsystem::main	OBF1.1.10.1 OBF1.1.10.2 OBF1.1.10.2.1 OBF1.1.10.2.1.1 OBF1.1.10.2.3 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 OBF1.1.3.7 OBF1.1.3.8 OBF1.1.3.9
actorbase::actorsystem::manager	DEF1.1.8 OBF1.1.10.3 OBF1.1.10.3.1
actorbase::actorsystem::manager::messages	DEF1.1.8.1 OBF1.1.8.2
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::serialization	OBF1.1.2.1 OBF1.1.2.3 OBF1.1.6 OBF1.1.6.1 OBF1.1.6.2
actorbase::actorsystem::storefinder	OBF1.1.10.1 OBF1.1.10.2 OBF1.1.10.2.1 OBF1.1.10.2.3 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
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
actorbase::actorsystem::tcpserver	OBF1.1 OBF1.1.1 OBF1.1.1.1 OBF1.1.10
actorbase::actorsystem::tcpserver::messages	OBF1.1.1.1.1 OBF1.1.1.1.2 OBF1.1.1.1.3

actorbase::actorsystem::userkeeper	OBF1.1.10.1 OBF1.1.10.1.1 OBF1.1.10.2 OBF1.1.10.2.1 OBF1.1.10.2.3 OBF1.1.10.2.3.1 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 OBF1.1.9.7 OBF1.1.9.8 OBF1.1.9.9
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
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 OBF2.1.2.4 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 OBF2.1.2.4 OBF2.1.2.4.1 OBF2.1.2.4.2 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 OBF2.1.6 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 DEF2.1.3.1.2 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 OBF2.1.2.10 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 OBF2.1.2.4 OBF2.1.2.4.1 OBF2.1.2.4.2 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 OBF2.1.2.9 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 OBF2.1.3.5 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::driver	DEF3
actorbase::driver::actorbasedata	DEF3.8 DEF3.8.1 DEF3.8.2 DEF3.8.3

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 DEF3.2.4 DEF3.2.4.1 DEF3.2.4.2 DEF3.2.4.3 DEF3.2.4.4 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 DEF3.3 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
actorbase::driver::exceptions	DEF3.1.3 DEF3.2.1.2 DEF3.2.3.2 DEF3.2.4.3 DEF3.2.4.4 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
actorbase::driver::serializer	DEF3.1 DEF3.2 DEF3.2.1 DEF3.2.2 DEF3.2.3 DEF3.2.4 DEF3.2.5 DEF3.2.6 DEF3.2.7 DEF3.3 DEF3.3.1 DEF3.3.2 DEF3.4 DEF3.5 DEF3.6 DEF3.6.1 DEF3.6.2 DEF3.6.3 DEF3.7

9.2 Tracciamento Requisiti-Componenti

Requisito	**Componenti**
DEF1.1.7	actorbase::actorsystem::ninja
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
DEF2.1.2.7	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
DEF2.1.2.7.1	actorbase::cli::models 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::serializer
DEF3.1.1	actorbase::driver::client
DEF3.1.2	actorbase::driver::client
DEF3.1.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.2	actorbase::driver::client actorbase::driver::serializer
DEF3.2.1	actorbase::driver::client actorbase::driver::serializer
DEF3.2.1.1	actorbase::driver::client
DEF3.2.1.2	actorbase::driver::client actorbase::driver::exceptions
DEF3.2.2	actorbase::driver::client actorbase::driver::serializer
DEF3.2.3	actorbase::driver::client actorbase::driver::serializer
DEF3.2.3.1	actorbase::driver::client
DEF3.2.3.2	actorbase::driver::client actorbase::driver::exceptions
DEF3.2.4	actorbase::driver::client actorbase::driver::serializer
DEF3.2.4.1	actorbase::driver::client
DEF3.2.4.2	actorbase::driver::client
DEF3.2.4.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.2.4.4	actorbase::driver::client actorbase::driver::exceptions
DEF3.2.5	actorbase::driver::client actorbase::driver::serializer
DEF3.2.5.1	actorbase::driver::client
DEF3.2.5.2	actorbase::driver::client

DEF3.2.5.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.2.5.4	actorbase::driver::client actorbase::driver::exceptions
DEF3.2.5.5	actorbase::driver::client
DEF3.2.6	actorbase::driver::client actorbase::driver::serializer
DEF3.2.6.1	actorbase::driver::client
DEF3.2.6.2	actorbase::driver::client
DEF3.2.6.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.2.7	actorbase::driver::client actorbase::driver::serializer
DEF3.2.7.1	actorbase::driver::client
DEF3.2.7.2	actorbase::driver::client
DEF3.3	actorbase::driver::client actorbase::driver::serializer
DEF3.3.1	actorbase::driver::client actorbase::driver::serializer
DEF3.3.1.1	actorbase::driver::client
DEF3.3.1.1.1	actorbase::driver::client
DEF3.3.1.1.2	actorbase::driver::client
DEF3.3.1.1.3	actorbase::driver::client
DEF3.3.1.1.4	actorbase::driver::client
DEF3.3.1.2	actorbase::driver::client
DEF3.3.1.2.1	actorbase::driver::client
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::exceptions
DEF3.3.2	actorbase::driver::client actorbase::driver::serializer
DEF3.3.2.1	actorbase::driver::client
DEF3.3.2.2	actorbase::driver::client
DEF3.3.2.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.4	actorbase::driver::client actorbase::driver::serializer
DEF3.4.1	actorbase::driver::client
DEF3.4.2	actorbase::driver::client

DEF3.5	actorbase::driver::client actorbase::driver::serializer
DEF3.6	actorbase::driver::client actorbase::driver::serializer
DEF3.6.1	actorbase::driver::client actorbase::driver::serializer
DEF3.6.1.1	actorbase::driver::client
DEF3.6.2	actorbase::driver::client actorbase::driver::serializer
DEF3.6.2.1	actorbase::driver::client
DEF3.6.3	actorbase::driver::client actorbase::driver::serializer
DEF3.6.3.1	actorbase::driver::client
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::serializer
DEF3.7.1	actorbase::driver::client
DEF3.7.2	actorbase::driver::client
DEF3.7.3	actorbase::driver::client actorbase::driver::exceptions
DEF3.7.4	actorbase::driver::client actorbase::driver::exceptions
DEF3.8	actorbase::driver::actorbasedata
DEF3.8.1	actorbase::driver::actorbasedata
DEF3.8.2	actorbase::driver::actorbasedata
DEF3.8.3	actorbase::driver::actorbasedata
DEV8	actorbase
DEV9	actorbase
OBF1	actorbase actorbase::actorsystem
OBF1.1	actorbase::actorsystem::tcpserver
OBF1.1.1	actorbase::actorsystem::tcpserver
OBF1.1.1.1	actorbase::actorsystem::tcpserver
OBF1.1.1.1.1	actorbase::actorsystem::tcpserver::messages
OBF1.1.1.1.2	actorbase::actorsystem::tcpserver::messages
OBF1.1.1.1.3	actorbase::actorsystem::tcpserver::messages
OBF1.1.10	actorbase::actorsystem::clientactor actorbase::actorsystem::tcpserver

OBF1.1.10.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userkeeper
OBF1.1.10.1.1	actorbase::actorsystem::clientactor actorbase::actorsystem::userkeeper
OBF1.1.10.2	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::ninja actorbase::actorsystem::storefinder actorbase::actorsystem::storekeeper 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::userkeeper
OBF1.1.10.2.1.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main
OBF1.1.10.2.2	actorbase::actorsystem::clientactor
OBF1.1.10.2.3	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder actorbase::actorsystem::userkeeper
OBF1.1.10.2.3.1	actorbase::actorsystem::clientactor actorbase::actorsystem::userkeeper
OBF1.1.10.2.3.2	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::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::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::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::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::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::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::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::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::userkeeper
OBF1.1.10.5.1	actorbase::actorsystem::clientactor 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::userkeeper
OBF1.1.10.7.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder 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::userkeeper
OBF1.1.10.7.3.1	actorbase::actorsystem::clientactor actorbase::actorsystem::main actorbase::actorsystem::storefinder
OBF1.1.2	actorbase::actorsystem::clientactor
OBF1.1.2.1	actorbase::actorsystem::clientactor::messages actorbase::actorsystem::serialization
OBF1.1.2.2	actorbase::actorsystem::clientactor::messages
OBF1.1.2.3	actorbase::actorsystem::clientactor::messages actorbase::actorsystem::serialization
OBF1.1.2.4	actorbase::actorsystem::clientactor::messages
OBF1.1.2.5	actorbase::actorsystem::clientactor::messages
OBF1.1.2.6	actorbase::actorsystem::clientactor::messages
OBF1.1.3	actorbase::actorsystem::main
OBF1.1.3.1	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
OBF1.1.3.7	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
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
OBF1.1.5	actorbase::actorsystem::storekeeper
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

OBF1.1.6	actorbase::actorsystem::serialization actorbase::actorsystem::warehouseman
OBF1.1.6.1	actorbase::actorsystem::serialization actorbase::actorsystem::warehouseman::messages
OBF1.1.6.2	actorbase::actorsystem::serialization actorbase::actorsystem::warehouseman::messages
OBF1.1.8.2	actorbase::actorsystem::manager::messages
OBF1.1.9	actorbase::actorsystem::userkeeper
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
OBF1.1.9.7	actorbase::actorsystem::userkeeper::messages
OBF1.1.9.8	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
OBF2.1.2.10	actorbase::cli::views
OBF2.1.2.11	actorbase::cli::views
OBF2.1.2.12	actorbase::cli::views
OBF2.1.2.13	actorbase::cli::views
OBF2.1.2.14	actorbase::cli::views

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
OBF2.1.2.4	actorbase::cli::controllers actorbase::cli::models actorbase::cli::views
OBF2.1.2.4.1	actorbase::cli::models actorbase::cli::views
OBF2.1.2.4.2	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
OBF2.1.2.9	actorbase::cli::views
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
OBF2.1.3.5	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
OBF2.1.7.5	actorbase::cli::views
OBF2.1.7.6	actorbase::cli::views
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
OBF2.1.9.5	actorbase::cli::views
OBF2.1.9.6	actorbase::cli::views
OBQ12	actorbase
OBV4	actorbase
OBV5	actorbase
OBV6	actorbase
OBV7	actorbase

A Descrizione Design Pattern

A.1 Design Pattern Architetturali

A.1.1 MVC

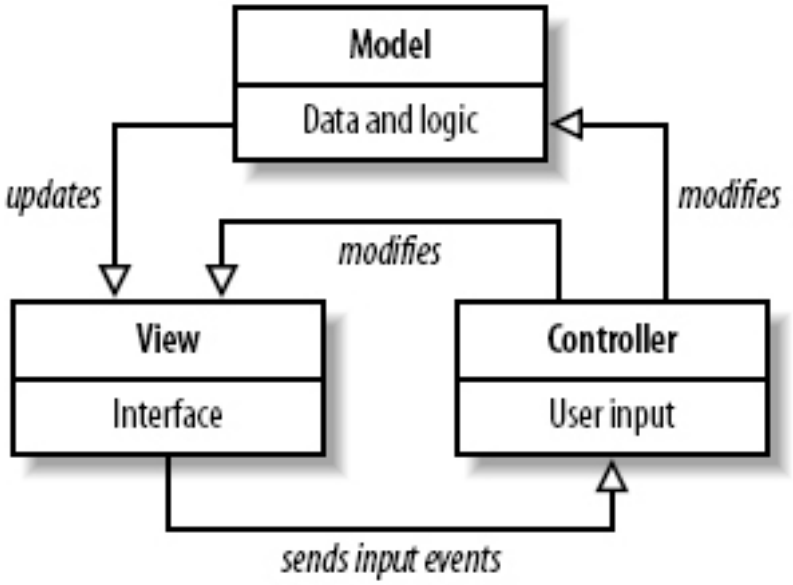

Figura 43: Esempio di applicazione del design pattern MVC

- **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 pattern$_G$ 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.2 Design Pattern Creazionali

A.2.1 Singleton

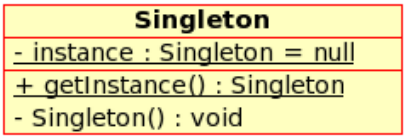

Figura 44: Esempio di applicazione del design pattern singleton

- **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.3 Design Pattern Strutturali

A.3.1 Façade

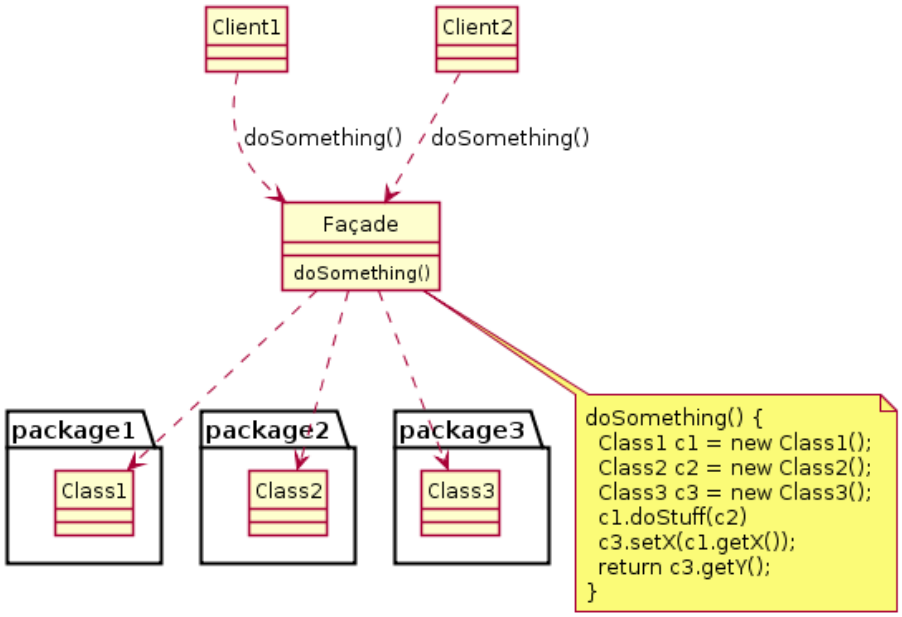

Figura 45: Esempio di applicazione del design pattern façade

- **Scopo:** Semplificare l'interfaccia visibile all'utente di un sistema complesso. Per fare ciò viene creata un'unica interfaccia che racchiude più classi, nascondendole all'esterno;
- **Motivazione:** Raggruppare più sottosistemi in un sistema unico aiuta a ridurne la complessità. L'utilizzo di una classe façade riduce le dipendenze tra i vari sottosistemi;
- **Applicabilità:** Questo design pattern$_G$ può essere applicato nei seguenti casi:
  - Quando c'è la necessità di avere una singola interfaccia semplice che raggruppi più sottosistemi;
  - Quando c'è un alto livello di accoppiamento tra sottosistemi e client. La classe façade diminuisce queste dipendenze offrendo un'interfaccia singola;
  - Quando si vogliono stratificare i sottosistemi con una struttura a livelli.

A.4 Design Pattern Comportamentali

A.4.1 Command

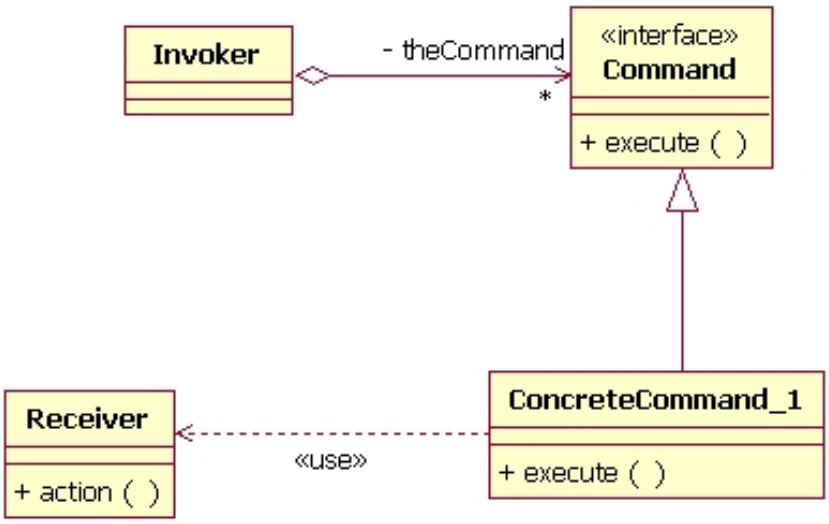

Figura 46: Esempio di applicazione del design pattern command

- **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 incapsulare un'azione in una operazione atomica.

A.4.2 Iterator

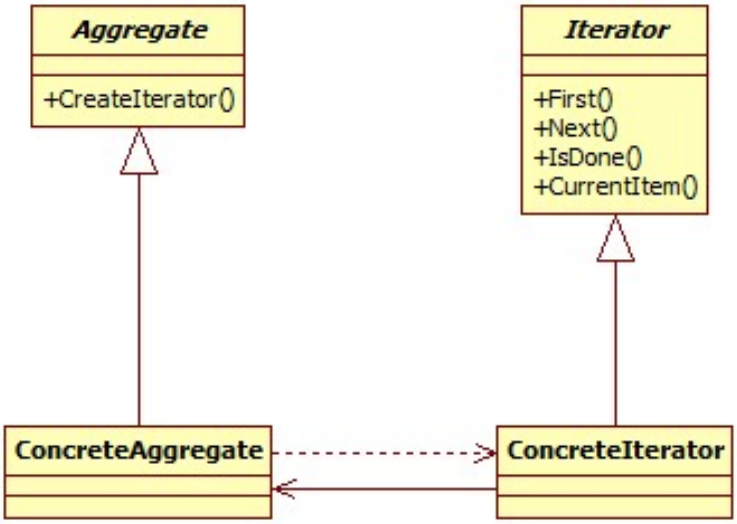

Figura 47: Esempio di applicazione del design pattern iterator

- **Scopo:** In un aggregato delegare la responsabilità dell'accesso da una classe separata dal contenitore stesso senza darne l'effettiva implementazione: si usa quindi l'iteratore, che fornisce metodi per navigare in maniera sequenziale attraverso il contenitore.
- **Motivazione:** Permette l'accesso ai dati nella medesima maniera dall'esterno e evita di appesantire il contenitore che altrimenti andrebbe arricchito di operazioni, rendendo il codice più complesso e difficilmente manutenibile. Inoltre permette di slegare il contenitore dalla maniera in cui si accede ad esso. Attraverso un iteratore si può inoltre ad esempio tenere traccia dell'elemento corrente, del precedente e del successivo.
- **Applicabilità:** Fornire l'accesso ad un aggregato senza fornire dettagli implementativi quali la struttura interna e la sua implementazione. Poiché rende il contenitore indipendente dalla maniera in cui si percorre, esso è dunque aperto a diverse implementazioni per percorrerlo: è sufficiente reimplementare l'iteratore, senza toccare il contenitore. Permette inoltre la presenza di più iteratori su uno stesso aggregato.

A.4.3 Observer

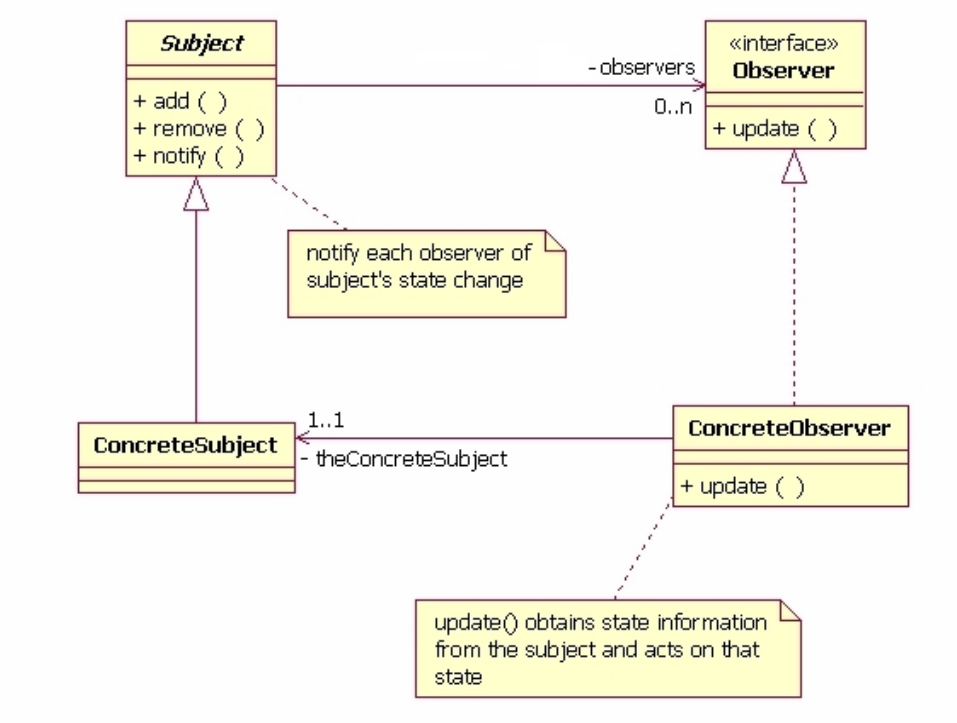

Figura 48: Esempio di applicazione del design pattern observer

- **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”.

A.4.4 Strategy

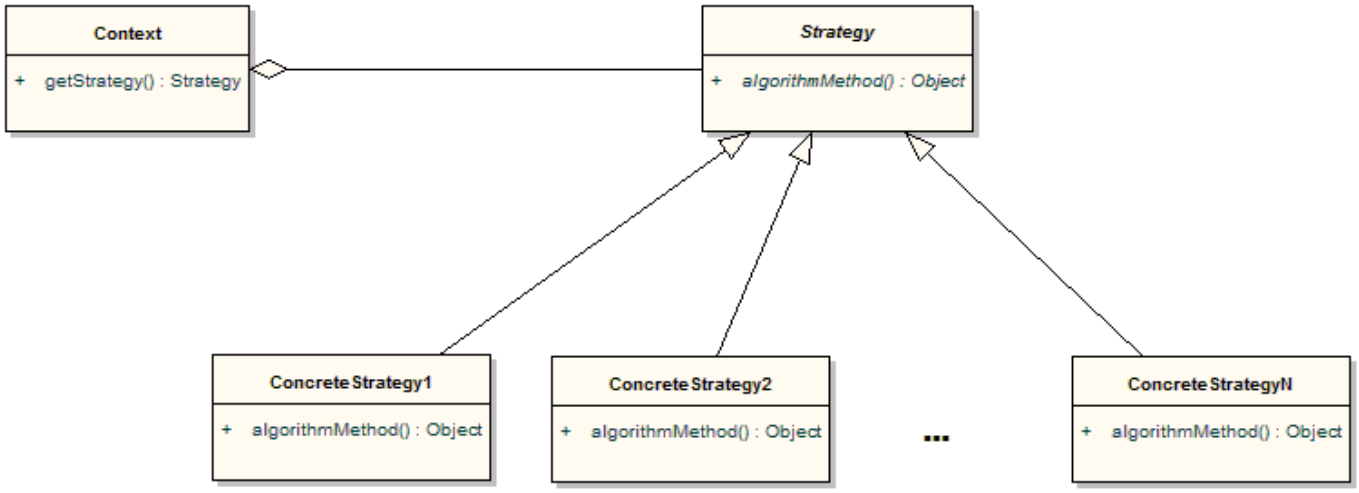

Figura 49: Esempio di applicazione del design pattern strategy

- **Scopo:** Isolare un algoritmo all'interno di un oggetto, allo scopo di definire un gruppo di algoritmi, rendendoli interscambiabili e indipendenti dal client$_G$ che li utilizza. A questo proposito è necessario che la famiglia di algoritmi che implementa una stessa funzionalità esporti la medesima interfaccia in ogni caso, in questo modo il client$_G$ che applica l'algoritmo non deve fare nessuna assunzioni sulla strategia applicata.
- **Motivazione:** Esistono differenti algoritmi (strategie) che non possono essere inserite direttamente nel client. Infatti i client$_G$, senza l'utilizzo di questo pattern rischiano di divenire troppo complessi: differenti strategie sono appropriate in casi differenti e dover scrivere codice ogni volta è troppo gravoso. Inoltre, se non si utilizza questo design pattern$_G$, è difficile aggiungere nuovi algoritmi e modificare gli esistenti.
- **Applicabilità:** questo design pattern$_G$ si usa quando delle classi differiscono solo per il loro comportamento e si necessita di diverse varianti dello stesso algoritmo oppure quando una classe implementa diverse strategie attraverso molti statement condizionali, che si possono eliminare con questo design pattern$_G$. Inoltre l'iterator pattern si basa proprio su questo pattern.

Elenco delle tabelle

Elenco delle figure

1 Diagramma architettura concettuale . . . 5
2 Diagramma componenti principali . . . 8
3 Actorsystem, visione generale . . . 10
4 Clientactor e interazioni con main e package serialization . . . 12
5 tcpserver, visione generale del package . . . 15
6 Serialization e interazioni con main e clientactor . . . 18
7 Warehouseman e interazioni con main e serialization . . . 22
8 Main e interazioni con storefinder . . . 24
9 Userkeeper, visione generale del package . . . 30
10 Storefinder e interazioni con main . . . 34
11 Storekeeper e interazioni con ninja . . . 38
12 Ninja e interazioni con storekeeper . . . 41
13 Manager, visione generale del package . . . 43
14 CLI:architettura MVC variante push model . . . 45
15 CLI, architettura MVC variante push model . . . 46
16 Driver, package serializer e interazioni con package client e package actorbasedata . . . 49
17 Driver, package actorbasedata e interazioni con package serializer e package client . . . 52
18 Driver:Package exceptions . . . 58
19 CLI, architettura MVC variante push model . . . 63
20 CLI, views package, interazioni con Controllers e Models . . . 64
21 Diagrammi attività - Visione generale . . . 78
22 Diagrammi attività - Creazione collezione . . . 79
23 Diagrammi attività - Cancellazione collezione . . . 80
24 Diagrammi attività - Visualizzazione collezioni . . . 81
25 Diagrammi attività - Modifica nome collezione . . . 82
26 Diagrammi attività - Inserimento item . . . 83
27 Diagrammi attività - Rimozione item . . . 84
28 Diagrammi attività - Aggiunta collaboratore . . . 85
29 Diagrammi attività - Rimozione collaboratore . . . 86
30 Diagrammi attività - Import . . . 87
31 Diagrammi attività - Interrogazione del database$_G$ . . . 88
32 Diagrammi attività - Modifica password . . . 89
33 Diagrammi attività - Gestione utenti . . . 90
34 Diagramma di sequenza - Interazioni generali tra componenti . . . 91
35 Diagramma di sequenza - Inserimento di un item . . . 92
36 Diagramma di sequenza - Inserimento di un item . . . 93
37 Diagramma di sequenza - Autenticazione . . . 94
38 Diagramma di sequenza - Inserimento di un item . . . 95
39 Diagramma di sequenza - Aggiornamento di un attore ninja . . . 96
40 Diagramma di sequenza - Autenticazione . . . 97
41 Diagramma di sequenza - Connessione al server . . . 98
42 Connection Singleton . . . 100
43 Esempio di applicazione del design pattern MVC . . . 119

44 Esempio di applicazione del design pattern singleton 120
45 Esempio di applicazione del design pattern façade 121
46 Esempio di applicazione del design pattern command 122
47 Esempio di applicazione del design pattern iterator 123
48 Esempio di applicazione del design pattern observer 124
49 Esempio di applicazione del design pattern strategy 125