
SGAD



protech.unipd@gmail.com

SPECIFICA TECNICA V3.00

Nome del documento	Specifiche Tecniche
Versione del documento	3.00
Data redazione	2014/01/14
Redazione	Nessi Alberto
Verifica	Biancucci Maurizio
Approvazione	Segantin Fabio
Uso	Esterno
Distribuzione	Prof. Vardanega Tullio Prof. Cardin Riccardo FunGo Studios S.r.l.

Sommario

Questo documento vuol definire l'architettura generale del prodotto SGAD.

Diario delle modifiche

Modifica	Autore	Ruolo	Data	Versione
<i>Approvazione del documento</i>	Segantin Fabio	Project Manager	2014/03/13	v 3.00
<i>Verifica del documento</i>	Biancucci Maurizio	Verificatore	2014/03/13	v 2.06
<i>Approfondita trattazione sull'utilizzo del Design Pattern DAO</i>	Nessi Alberto	Progettista	2014/03/12	v 2.05
<i>Spostamento della trattazione del framework per il design pattern MVC nella sezione "MVC" della sezione "Design Pattern"</i>	Nessi Alberto	Progettista	2014/03/12	v 2.04
<i>Indicazione di un esempio di un messaggio concreto nella sezione "Protocollo di comunicazione"</i>	Nessi Alberto	Progettista	2014/03/12	v 2.03
<i>Aggiunta classe: InternalRequester al pacchetto usermanager. Aggiunte classi ToInternalRequester-Request e ToPublisherAndUser-InternalRequest al pacchetto messages</i>	Nessi Alberto	Progettista	2014/03/08	v 2.02
<i>Aggiornata firma del metodo execute in tutte le classi del pacchetto operation che posseggono il metodo. Rimosse le classi LoadAnotherUser e ReceiveLoadAnotherUser.</i>	Nessi Alberto	Progettista	2014/03/08	v 2.01
<i>Approvazione del documento</i>	Nessi Alberto	Project Manager	2014/02/24	v 2.00
<i>Verifica del documento</i>	Gallo Francesco	Verificatore	2014/02/24	v 1.16

<i>Aggiunto YUI Compressor alla sezione tecnologie</i>	Biancucci Maurizio	Progettista	2014/02/24	v 1.15
<i>Aggiunte nuove classi ai pacchetti <code>userdata</code> e <code>httpresponder</code></i>	Biancucci Maurizio	Progettista	2014/02/24	v 1.14
<i>Approfondito tracciamento Componenti-Requisiti e Requisiti-Componenti</i>	Biancucci Maurizio	Progettista	2014/02/24	v 1.13
<i>Sono stati approfonditi i pacchetti <code>userdata</code>, <code>shareddata</code>, <code>personaldata</code> e <code>generaldata</code> sistemando alcune relazioni o aggiungendo attributi mancanti. È stato aggiornato di conseguenza lo schema del database</i>	Biancucci Maurizio	Progettista	2014/02/23	v 1.12
<i>Nel pacchetto <code>operation</code> sono ora presenti tutte e sole le classi necessarie a soddisfare i requisiti</i>	Biancucci Maurizio	Progettista	2014/02/23	v 1.11
<i>Adeguamento di molti schemi e relazioni in base alle precedenti modifiche</i>	Biancucci Maurizio	Progettista	2014/02/23	v 1.10
<i>Aggiunti i pacchetti <code>cluster</code>, <code>messages</code> e <code>timeout</code></i>	Biancucci Maurizio	Progettista	2014/02/23	v 1.09
<i>Aggiunto pacchetto <code>databaseaccess</code> in cui è stato spostato in pacchetto <code>databasemanager</code> ed inseriti i nuovi pacchetti <code>shareddatadao</code> e <code>userdatadao</code></i>	Biancucci Maurizio	Progettista	2014/02/22	v 1.08
<i>Ristrutturazione profonda del pacchetto <code>presentation</code> permettendo la parallelizzazione della gestione delle richieste di login e registrazione</i>	Biancucci Maurizio	Progettista	2014/02/22	v 1.07

<i>Approfondimento nell'utilizzo del design pattern MVC</i>	Segantin Fabio	Progettista	2014/02/22	v 1.06
<i>Sostituzione di diagrammi non conformi a UML con diagrammi conformi</i>	Segantin Fabio	Progettista	2014/02/20	v 1.05
<i>Rimosso svantaggio nell'utilizzo della tecnologia JavaScript</i>	Segantin Fabio	Progettista	2014/02/18	v 1.04
<i>Approfondimento del protocollo di comunicazione</i>	Segantin Fabio	Progettista	2014/02/18	v 1.03
<i>Aggiunta delle componenti BackupManager e UserDataManager</i>	Segantin Fabio	Progettista	2014/02/18	v 1.02
<i>Eliminazione delle classi AuthenticationString e Resize</i>	Segantin Fabio	Progettista	2014/02/16	v 1.01
<i>Approvazione del documento</i>	Biancucci Maurizio	Project Manager	2014/01/30	v 1.00
<i>Verifica del documento</i>	Gatto Francesco	Verificatore	2014/01/29	v 0.15
<i>Correzione degli errori individuati</i>	Battistella Stefano	Progettista	2014/01/28	v 0.14
<i>Verifica del documento</i>	Gatto Francesco	Verificatore	2014/01/28	v 0.13
<i>Stesura sezione "Tracciamento"</i>	Gallo Francesco	Progettista	2014/01/26	v 0.12
<i>Stesura sezione "Descrizione generale design pattern"</i>	Gallo Francesco	Progettista	2014/01/25	v 0.11
<i>Stesura sezione "Design pattern"</i>	Gallo Francesco	Progettista	2014/01/23	v 0.10
<i>Stesura sezione "Stime di fattibilità e bisogno di risorse"</i>	Gallo Francesco	Progettista	2014/01/21	v 0.09
<i>Completata stesura sezione "Componenti e classi"</i>	Battistella Stefano	Progettista	2014/01/21	v 0.08
<i>Continuata stesura sezione "Componenti e classi"</i>	Battistella Stefano	Progettista	2014/01/17	v 0.07
<i>Stesura sezione "Schema della base di dati"</i>	Gallo Francesco	Progettista	2014/01/16	v 0.06

<i>Inizio stesura sezione "Componenti e classi"</i>	Battistella Stefano	Progettista	2014/01/16	v 0.05
<i>Stesura sezione "Descrizione architettura"</i>	Battistella Stefano	Progettista	2014/01/15	v 0.04
<i>Stesura sezione "Tecnologie utilizzate"</i>	Gallo Francesco	Progettista	2014/01/14	v 0.03
<i>Stesa sezione "Introduzione"</i>	Battistella Stefano	Progettista	2014/01/15	v 0.02
<i>Inizio stesura documento con impostazione generale dello scheletro del documento</i>	Battistella Stefano	Progettista	2014/01/14	v 0.01

Indice

1 Introduzione	15
1.1 Scopo del documento	15
1.2 Scopo del prodotto	15
1.3 Glossario	15
1.4 Riferimenti	15
1.4.1 Normativi	15
1.4.2 Informativi	15
2 Tecnologie utilizzate	17
2.1 Scala	17
2.2 Akka	17
2.3 spray	17
2.4 Casbah	18
2.5 HTML5	18
2.6 CSS	18
2.7 JavaScript	18
2.8 jQuery	19
2.9 AJAX	19
2.10 JSON	19
2.11 MongoDB	20
2.12 YUI Compressor	20
3 Descrizione architettura	21
3.1 Metodo e formalismo di specifica	21
3.2 Architettura generale	21
3.2.1 Client Tier	22
3.2.2 Server Tier	23
3.2.3 Db Tier	24
3.2.4 Protocollo di comunicazione client-server	24
3.2.5 Protocollo di comunicazione server-database	26
4 Componenti e classi	27
4.1 sgad	29
4.1.1 Informazioni sul package	29
4.2 sgad::clienttier	30
4.2.1 Informazioni sul package	30
4.3 sgad::clienttier::controller	31
4.3.1 Informazioni sul package	31
4.4 sgad::clienttier::controller::actions	32
4.4.1 Informazioni sul package	32
4.4.2 Classi	34
4.4.2.1 Action	34
4.4.2.2 ActionListener	36
4.4.2.3 BackToVillage	37

4.4.2.4	BuildConstruction	37
4.4.2.5	ChangePasswordAction	38
4.4.2.6	CloseContextualMenu	39
4.4.2.7	DeleteAccountAction	40
4.4.2.8	DemolishBuilding	41
4.4.2.9	DismissUnits	41
4.4.2.10	EnableRightClick	42
4.4.2.11	HarvestResources	43
4.4.2.12	LoadAnotherUser	43
4.4.2.13	LoadGeneralData	44
4.4.2.14	LoadPersonalData	45
4.4.2.15	Logout	46
4.4.2.16	ReceiveStealResources	46
4.4.2.17	ReloadPageAction	47
4.4.2.18	RemoveGraphicObjectAction	47
4.4.2.19	RequestAttackUser	48
4.4.2.20	RequestBuildConstruction	48
4.4.2.21	RequestChangePassword	49
4.4.2.22	RequestDeleteAccount	49
4.4.2.23	RequestDemolishBuilding	50
4.4.2.24	RequestDismissUnits	50
4.4.2.25	RequestHarvestResources	51
4.4.2.26	RequestLoadGeneralData	51
4.4.2.27	RequestLoadPersonalData	52
4.4.2.28	RequestStoleResources	53
4.4.2.29	RequestTrainUnits	53
4.4.2.30	RequestUpgradeBuilding	54
4.4.2.31	SetAllVoidFilter	54
4.4.2.32	SetBuildModeFilter	55
4.4.2.33	ShowAccountDeletedMenu	55
4.4.2.34	ShowAccountManagerMenu	56
4.4.2.35	ShowAnotherUserMenu	56
4.4.2.36	ShowAttackMenu	57
4.4.2.37	ShowAttackResultMenu	58
4.4.2.38	ShowBuildConstructionMenu	59
4.4.2.39	ShowBuildingContextualMenu	59
4.4.2.40	ShowDemolishMenu	60
4.4.2.41	ShowDismissUnitMenu	61
4.4.2.42	ShowInteractionMenu	61
4.4.2.43	ShowOperationFailureMenu	62
4.4.2.44	ShowPasswordChangedMenu	64
4.4.2.45	ShowResourceMenu	64
4.4.2.46	ShowTileContextualMenu	65
4.4.2.47	ShowTrainUnitMenu	66
4.4.2.48	ShowUnitSelectionMenu	67
4.4.2.49	ShowUpgradeMenu	67

4.4.2.50	ShowUserListMenu	68
4.4.2.51	StealResources	68
4.4.2.52	TrainUnit	69
4.4.2.53	UpgradeBuilding	70
4.4.2.54	RequestUserList	71
4.4.2.55	ShowLogMenu	71
4.5	sgad::clienttier::controller::backupmanager	72
4.5.1	Informazioni sul package	72
4.5.2	Classi	72
4.5.2.1	BackupManager	72
4.6	sgad::clienttier::controller::menufactory	74
4.6.1	Informazioni sul package	74
4.6.2	Classi	75
4.6.2.1	MenuFactory	75
4.6.2.2	AccountManagerMenuFactory	75
4.6.2.3	AnotherUserMenuFactory	76
4.6.2.4	AttackResultMenuFactory	77
4.6.2.5	BuildingContextualMenuFactory	78
4.6.2.6	ConfirmMenuFactory	79
4.6.2.7	InteractionMenuFactory	80
4.6.2.8	LogMenuFactory	81
4.6.2.9	NotifyMenuFactory	82
4.6.2.10	ResourceMenuFactory	83
4.6.2.11	TileContextualMenuFactory	84
4.6.2.12	UnitSelectionMenu	85
4.6.2.13	UserListMenuFactory	86
4.7	sgad::clienttier::controller::messageinterpreter	88
4.7.1	Informazioni sul package	88
4.7.2	Classi	88
4.7.2.1	MessageInterpreter	88
4.8	sgad::clienttier::controller::requester	89
4.8.1	Informazioni sul package	89
4.8.2	Classi	89
4.8.2.1	AJAXRequester	89
4.9	sgad::clienttier::model	91
4.9.1	Informazioni sul package	91
4.10	sgad::clienttier::model::generaldata	92
4.10.1	Informazioni sul package	92
4.10.2	Classi	92
4.10.2.1	Bonus	92
4.10.2.2	BuildingWithLevel	93
4.10.2.3	Cost	94
4.10.2.4	DataFactory	95
4.10.2.5	ProducedResource	95
4.10.2.6	QuantityResource	96
4.10.2.7	Resource	97

4.10.2.8	Unit	97
4.11	sgad::clienttier::model::observer	98
4.11.1	Informazioni sul package	98
4.11.2	Classi	99
4.11.2.1	Observable	99
4.11.2.2	Observer	99
4.12	sgad::clienttier::model::personaldata	101
4.12.1	Informazioni sul package	101
4.12.2	Classi	102
4.12.2.1	AuthenticationData	102
4.12.2.2	BuildingPossession	102
4.12.2.3	OwnedResource	103
4.12.2.4	Position	104
4.12.2.5	UnitPossession	105
4.12.2.6	UnitInProgress	106
4.12.2.7	UserData	107
4.13	sgad::clienttier::model::userdatamanager	108
4.13.1	Informazioni sul package	108
4.13.2	Classi	108
4.13.2.1	UserDataManager	108
4.14	sgad::clienttier::view	110
4.14.1	Informazioni sul package	110
4.15	sgad::clienttier::view::commands	111
4.15.1	Informazioni sul package	111
4.15.2	Classi	111
4.15.2.1	Command	111
4.15.2.2	Click	112
4.15.2.3	DragAndDrop	112
4.15.2.4	RightClick	113
4.16	sgad::clienttier::view::context	114
4.16.1	Informazioni sul package	114
4.16.2	Classi	114
4.16.2.1	Context	114
4.17	sgad::clienttier::view::graphicobjects	117
4.17.1	Informazioni sul package	117
4.18	sgad::clienttier::view::graphicobjects::bound	118
4.18.1	Informazioni sul package	118
4.18.2	Classi	119
4.18.2.1	Bound	119
4.19	sgad::clienttier::view::graphicobjects::collection	120
4.19.1	Informazioni sul package	120
4.19.2	Classi	121
4.19.2.1	Collection	121
4.19.2.2	GraphicObjectCollection	121
4.19.2.3	Iterator	122
4.19.2.4	GraphicObjectIterator	122

4.20 sgad::clienttier::view::graphicobjects::components	123
4.20.1 Informazioni sul package	123
4.21 sgad::clienttier::view::graphicobjects::components::filter	124
4.21.1 Informazioni sul package	124
4.21.2 Classi	124
4.21.2.1 GraphicFilter	124
4.21.2.2 BuildModeFilter	125
4.21.2.3 VoidFilter	126
4.22 sgad::clienttier::view::graphicobjects::components::worldcomponent	127
4.22.1 Informazioni sul package	127
4.22.2 Classi	127
4.22.2.1 WorldComponent	127
4.22.2.2 BuildingComponent	128
4.22.2.3 TileComponent	129
4.23 sgad::clienttier::view::graphicobjects::components::worldcomponentshape	131
4.23.1 Informazioni sul package	131
4.23.2 Classi	132
4.23.2.1 WorldComponentShapeFactory	132
4.23.2.2 WorldComponentShapeImg	133
4.23.2.3 BarracksL1Shape	134
4.23.2.4 BarracksL2Shape	135
4.23.2.5 BarracksL3Shape	135
4.23.2.6 InConstructionShape	135
4.23.2.7 MineL1Shape	136
4.23.2.8 MineL2Shape	136
4.23.2.9 MineL3Shape	137
4.23.2.10 SchoolOfMagicL1Shape	137
4.23.2.11 SchoolOfMagicL2Shape	137
4.23.2.12 SchoolOfMagicL3Shape	138
4.23.2.13 StableL1Shape	138
4.23.2.14 StableL2Shape	139
4.23.2.15 StableL3Shape	139
4.23.2.16 TileShape	139
4.23.2.17 WizardTowerL1Shape	140
4.23.2.18 WizardTowerL2Shape	140
4.23.2.19 WizardTowerL3Shape	141
4.24 sgad::clienttier::view::graphicobjects::graphicobject	142
4.24.1 Informazioni sul package	142
4.24.2 Classi	142
4.24.2.1 GraphicObject	142
4.25 sgad::clienttier::view::graphicobjects::point	144
4.25.1 Informazioni sul package	144
4.25.2 Classi	144
4.25.2.1 Point2D	144
4.26 sgad::clienttier::view::graphicobjects::shape	146
4.26.1 Informazioni sul package	146

4.26.2 Classi	147
4.26.2.1 Shape	147
4.27 sgad::clienttier::view::graphicobjects::widget	149
4.27.1 Informazioni sul package	149
4.27.2 Classi	150
4.27.2.1 Widget	150
4.27.2.2 ButtonWidget	152
4.27.2.3 FrameWidget	154
4.27.2.4 ImageWidget	157
4.27.2.5 TextWidget	158
4.28 sgad::servertier	159
4.28.1 Informazioni sul package	159
4.29 sgad::servertier::application	160
4.29.1 Informazioni sul package	160
4.29.2 Classi	160
4.29.2.1 Application	160
4.30 sgad::servertier::businesslogic	161
4.30.1 Informazioni sul package	161
4.31 sgad::servertier::businesslogic::logic	162
4.31.1 Informazioni sul package	162
4.31.2 Classi	162
4.31.2.1 Logic	162
4.32 sgad::servertier::businesslogic::operations	164
4.32.1 Informazioni sul package	164
4.32.2 Classi	166
4.32.2.1 Operation	166
4.32.2.2 Attack	167
4.32.2.3 BuildConstruction	167
4.32.2.4 ChangeAccountData	168
4.32.2.5 DeleteAccount	168
4.32.2.6 DemolishBuilding	169
4.32.2.7 DismissUnit	169
4.32.2.8 DonateResources	169
4.32.2.9 DonateUnits	170
4.32.2.10 GetAllServerData	170
4.32.2.11 GetServerData	171
4.32.2.12 HarvestResource	171
4.32.2.13 InternalLogin	171
4.32.2.14 LoadGlobalData	172
4.32.2.15 LoadVillage	172
4.32.2.16 LoadUserList	173
4.32.2.17 Login	173
4.32.2.18 Logout	174
4.32.2.19 ReceiveAttack	174
4.32.2.20 ReceiveStealResource	175
4.32.2.21 Registration	175

4.32.2.22 SaveUser	176
4.32.2.23 StealResource	176
4.32.2.24 TrainUnit	177
4.32.2.25 UpdateUserData	177
4.32.2.26 UpgradeBuilding	178
4.32.2.27 OperationFactory	178
4.33 sgad::servertier::dataaccess	180
4.33.1 Informazioni sul package	180
4.34 sgad::servertier::dataaccess::data	181
4.34.1 Informazioni sul package	181
4.35 sgad::servertier::dataaccess::data::shareddata	182
4.35.1 Informazioni sul package	182
4.35.2 Classi	183
4.35.2.1 Bonus	183
4.35.2.2 BuildingWithLevel	183
4.35.2.3 Cost	184
4.35.2.4 DataFactory	184
4.35.2.5 ProductedResource	185
4.35.2.6 QuantityResource	185
4.35.2.7 Resource	186
4.35.2.8 Unit	186
4.36 sgad::servertier::dataaccess::data::userdata	187
4.36.1 Informazioni sul package	187
4.36.2 Classi	187
4.36.2.1 AuthenticationData	187
4.36.2.2 BuildingPossession	188
4.36.2.3 OwnedResource	189
4.36.2.4 Position	189
4.36.2.5 UnitPossession	189
4.36.2.6 UnitInProgress	190
4.36.2.7 UserData	190
4.37 sgad::servertier::dataaccess::databaseaccess	192
4.37.1 Informazioni sul package	192
4.38 sgad::servertier::dataaccess::databaseaccess::databasemanager	193
4.38.1 Informazioni sul package	193
4.38.2 Classi	194
4.38.2.1 DataBaseManager	194
4.39 sgad::servertier::dataaccess::databaseaccess::shareddatadao	196
4.39.1 Informazioni sul package	196
4.39.2 Classi	197
4.39.2.1 BonusDAO	197
4.39.2.2 BuildingWithLevelDAO	197
4.39.2.3 CostDAO	198
4.39.2.4 ProductedResourceDAO	198
4.39.2.5 QuantityResourceDAO	198
4.39.2.6 ResourceDAO	199

4.39.2.7	SharedDataDAO	199
4.39.2.8	UnitDAO	200
4.40	sgad::servertier::dataaccess::databaseaccess::userdatadao	201
4.40.1	Informazioni sul package	201
4.40.2	Classi	202
4.40.2.1	AuthenticationDataDAO	202
4.40.2.2	BuildingPossessionDAO	202
4.40.2.3	OwnedResourceDAO	203
4.40.2.4	PositionDAO	203
4.40.2.5	UnitInProgressDAO	203
4.40.2.6	UnitPossessionDAO	204
4.40.2.7	UserDataAdapter	204
4.41	sgad::servertier::presentation	206
4.41.1	Informazioni sul package	206
4.42	sgad::servertier::presentation::cluster	207
4.42.1	Informazioni sul package	207
4.42.2	Classi	207
4.42.2.1	ClusterListener	207
4.43	sgad::servertier::presentation::httpresponder	208
4.43.1	Informazioni sul package	208
4.43.2	Classi	209
4.43.2.1	ResponderActor	209
4.43.2.2	WorkerActor	209
4.44	sgad::servertier::presentation::messages	211
4.44.1	Informazioni sul package	211
4.44.2	Classi	212
4.44.2.1	ToInternalRequesterRequest	212
4.44.2.2	ToLoginActorRequest	212
4.44.2.3	ToPublisherAndUserInternalRequest	212
4.44.2.4	ToPublisherAndUserIsAliveRequest	213
4.44.2.5	ToPublisherAndUserRequest	213
4.44.2.6	ToRegistrationActorRequest	213
4.44.2.7	ToRequesterIsAliveRequest	214
4.44.2.8	ToWorkerLoginRequest	214
4.44.2.9	ToWorkerRegistrationRequest	214
4.44.2.10	ToWorkerUserRequest	215
4.44.2.11	UserActorTimeoutMessage	215
4.44.2.12	WorkerActorTimeoutMessage	215
4.45	sgad::servertier::presentation::pagemanager	216
4.45.1	Informazioni sul package	216
4.45.2	Classi	216
4.45.2.1	PageFactory	216
4.46	sgad::servertier::presentation::timeout	218
4.46.1	Informazioni sul package	218
4.46.2	Classi	218
4.46.2.1	STimeout	218

4.47 sgad::servertier::presentation::usermanager	220
4.47.1 Informazioni sul package	220
4.47.2 Classi	221
4.47.2.1 InternalRequester	221
4.47.2.2 IsUserActorAliveRequester	222
4.47.2.3 IsUserActorAliveResponder	222
4.47.2.4 LoginActor	222
4.47.2.5 PublisherActor	223
4.47.2.6 RegistrationActor	224
4.47.2.7 UserActor	224
5 Design Pattern	226
5.1 Design pattern architetturali	227
5.1.1 DAO	227
5.1.2 MVC	229
5.1.3 Three-Tier	230
5.2 Design pattern creazionali	231
5.2.1 Factory Method	231
5.2.2 Singleton	233
5.3 Design pattern strutturali	236
5.3.1 Composite	236
5.3.2 Flyweight	239
5.4 Design pattern comportamentali	242
5.4.1 Command	242
5.4.2 Iterator	243
5.4.3 Observer	244
5.4.4 State	245
5.4.5 Strategy	246
6 Diagrammi delle attività	248
6.1 Attività principali dell’utente giocatore	248
6.2 Visualizzazione delle informazioni utente	249
6.3 Modifica dei dati dell’utente	250
6.4 Visualizzazione del manuale utente giocatore	251
6.5 Interazione con il mondo di gioco	252
6.6 Interazione con un altro utente	252
6.7 Saccheggio del villaggio di un altro utente	254
6.8 Regalo di risorse ad un altro utente	255
6.9 Regalo di unità ad un altro utente	257
6.10 Gestione del villaggio	258
6.11 Raccolta di risorse	259
6.12 Demolizione di un edificio	259
6.13 Miglioramento di un edificio	260
6.14 Produzione di unità	261
6.15 Congedo di unità	262
6.16 Costruzione di un edificio	263
6.17 Attività principali dell’amministratore	264

6.18 Visualizzazione delle informazioni di un server del cluster	265
6.19 Aggiunta di un nuovo server al cluster	266
6.20 Visualizzazione del manuale utente amministratore	267
6.21 Rimozione di un server dal cluster	268
6.22 Visualizzazione delle informazioni generali del cluster di server	269
7 Stime di fattibilità e di bisogno di risorse	270
8 Tracciamento	271
8.1 Tracciamento Componenti-Requisiti	271
8.2 Tracciamento Requisiti-Componenti	287
A Descrizione generale design pattern	326
A.1 Design pattern architetturali	326
A.1.1 MVC, Model-View-Controller	326
A.1.2 Three-Tier	328
A.1.3 Data Access Object	329
A.2 Design pattern creazionali	329
A.2.1 Factory Method	329
A.2.2 Singleton	330
A.3 Design pattern strutturali	331
A.3.1 Composite	331
A.3.2 Flyweight	332
A.4 Design pattern comportamentali	333
A.4.1 Command	333
A.4.2 Iterator	334
A.4.3 Observer	335
A.4.4 State	336
A.4.5 Strategy	337

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello del *prodotto_[g]* SGAD.

Vengono quindi presentate le componenti, le classi e i vari *design pattern_[g]* utilizzati per la realizzazione del prodotto. Inoltre, viene presentato il tracciamento tra le componenti e i requisiti individuati.

1.2 Scopo del prodotto

Il prodotto denominato SGAD si propone di fornire un sistema per il gioco online avente architettura distribuita. Il *sistema_[g]* permette all'utente, tramite una semplice interfaccia web, di utilizzare un *social game_[g]* di tipo strategico. Esso intende offrire all'utente un'esperienza di gioco nella quale si possano collezionare risorse, costruire edifici, ampliare la propria squadra ed eventualmente combattere con altri utenti.

1.3 Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio e ai termini utilizzati nei documenti formali, viene allegato il “*Glossario v2.00*”. In tale documento vengono definiti e descritti tutti i termini evidenziati con l'apposita marcatura.

1.4 Riferimenti

1.4.1 Normativi

- “*Norme di Progetto v5.00*”
- “*Analisi dei Requisiti v6.00*”

1.4.2 Informativi

- Scala for the Impatient (Cay Horstmann, Addison-Wesley, 2012)
- Effective Akka - Patterns and Best Practices (Jamie Allen, O'Reilly Media, 2013)
- Documentazione Akka-actors: <http://doc.akka.io/docs/akka/2.2.3/scala/actors.htm>
- Documentazione Akka-cluster: <http://doc.akka.io/docs/akka/2.2.3/common/cluster.htm>
- HTML5 Candidate Recommendation: <http://www.w3.org/TR/html5/>
- CSS, direttive W3C: <http://www.w3.org/TR/CSS/>
- Standard ECMA-262 di JavaScript: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf>

- Standard ECMA-404 di JSON: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>
- Documentazione API jQuery: <http://api.jquery.com/>
- Documentazione spray: <http://spray.io/documentation/1.2.0/>
- Documentazione MongoDB: <http://docs.mongodb.org/manual/>
- Documentazione Casbah: <http://mongodb.github.io/casbah/>
- SWEBOK - Chapter 3: Software Design: <http://www.computer.org/portal/web/swebok/html/ch3>
- SWEBOK - Chapter 11: Software Quality: <http://www.computer.org/portal/web/swebok/html/ch11>
- Design Pattern - Elementi per il riuso di software ad oggetti (Gamma, Helm, Johnson, Vlissides, editore Pearson, 2002)
- Software Engineering - Chapter 6: Architectural design - Ian Sommerville - 9th ed. (2010)

2 Tecnologie utilizzate

In questa sezione vengono descritte le tecnologie su cui si basa lo sviluppo del progetto. Per ognuna di esse, verrà indicato l'ambito di utilizzo della tecnologia ed i vantaggi che ne derivano.

2.1 Scala

L'utilizzo di *Scala*_{|g|} è vincolato dalla richiesta del *proponente*_{|g|} (requisito: R0V4). Tale linguaggio verrà utilizzato per lo sviluppo della parte server. Si è preferito tale linguaggio a *Ruby*_{|g|}, per poter beneficiare del modello ad attori e di distribuzione del *framework*_{|g|} *Akka*_{|g|}, ritenuto più maturo e performante rispetto a *DCell*_{|g|}.

Vantaggi:

- prestazioni superiori rispetto a Ruby;
- linguaggio orientato agli oggetti comprendente elementi funzionali;
- tecnica di gestione della concorrenza basata sul modello ad attori¹.

2.2 Akka

L'utilizzo del framework Akka è vincolato dalla richiesta del proponente (requisito: R0V4). Si è scelto di utilizzare tale framework al posto di DCell, in quanto si è ritenuto il sistema distribuito ad attori Akka, più performante e maturo. Tale scelta ha tenuto in considerazione anche l'opinione del creatore di *Celluloid*_{|g|} e DCell.

Vantaggi:

- implementa il modello ad attori²;
- la concorrenza si basa sullo scambio di messaggi ed è asincrona;
- normalmente non sono usate primitive per la sincronizzazione ed è proibita la condivisione di dati mutabili.

2.3 spray

Si è deciso di utilizzare *spray*_{|g|} come *libreria*_{|g|} *open-source*_{|g|} per costruire livelli di integrazione basati su *REST*_{|g|} *HTTP*_{|g|} su Akka e Scala.

Vantaggi:

- gli attori Akka ed i *future*_{|g|} sono costrutti chiave delle sue *API*_{|g|};
- alte prestazioni in ambienti con elevato carico.

¹Dalla versione 2.0 gli attori di Scala sono stati dichiarati obsoleti e sostituiti con l'implementazione fornita da Akka, per approfondimenti si veda sezione 2.2 Akka

²Dalla versione 2.0 gli attori di Akka sono diventati gli attori di riferimento anche per Scala.

2.4 Casbah

Si è deciso di utilizzare *Casbah_{|g|}* come *toolkit_{|g|}* di Scala per la comunicazione con *MongoDB_{|g|}*.

Vantaggi:

- rende più facile l'interfacciamento con *database_{|g|}* MongoDB rispetto ai driver standard di *Java_{|g|}*.

2.5 HTML5

Si è deciso di utilizzare *HTML5_{|g|}* come *linguaggio di markup_{|g|}* per la strutturazione delle pagine web.

Vantaggi:

- insieme a *CSS_{|g|}* permette di definire semplicemente pagine web che si adattino al *browser_{|g|}*;
- supporta l'elemento *canvas_{|g|}* che permette di utilizzare *JavaScript_{|g|}* per creare animazioni e grafica.

Svantaggi:

- tecnologia disponibile in *candidate recommendation_{|g|}*, non pienamente supportata da tutti i browser.

2.6 CSS

Si è deciso di utilizzare CSS come linguaggio per la formattazione dei documenti *HTML_{|g|}* che compongono la parte client.

Vantaggi:

- consente di avere il pieno controllo degli aspetti visivi delle pagine web separando la presentazione dalla struttura;
- controllo più preciso e completo dell'aspetto grafico;
- la manutenzione grafica del sito risulta molto più facile.

2.7 JavaScript

Si è deciso di utilizzare JavaScript come linguaggio di *scripting_{|g|}* lato client per la creazione dell'interfaccia utente.

Vantaggi:

- dà dinamicità alle pagine web;
- permette di reagire ad eventi generati dall'interazione tra l'utente e la pagina;
- possibilità di validare in prima istanza i dati inseriti dall'utente;
- possibilità di interagire con il $DOM_{|g|}$;
- possibilità di gestire i canvas.

2.8 jQuery

Si è deciso di utilizzare $jQuery_{|g|}$ come framework JavaScript per lo sviluppo della parte client.

Vantaggi:

- la gestione delle chiamate asincrone è notevolmente semplificata;
- semplifica la programmazione lato client delle pagine web;
- manipolazione del DOM scrivendo codice più compatto.

2.9 AJAX

Si è deciso di utilizzare $AJAX_{|g|}$ come tecnologia per lo sviluppo della comunicazione dei client verso il server.

Vantaggi:

- consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente;
- è una tecnica multi-piattaforma utilizzabile cioè su molti *sistemi operativi* $_{|g|}$, architetture informatiche e browser web;
- esistono numerose implementazioni open-source di *librerie* $_{|g|}$ e framework.

2.10 JSON

Si è deciso di utilizzare $JSON_{|g|}$ come formato per lo scambio dati tra client e server.

Vantaggi:

- semplice utilizzo attraverso JavaScript;
- parsing molto facile nei linguaggi più conosciuti;
- formato utilizzato anche da MongoDB per la gestione dei dati.

2.11 MongoDB

Si è deciso di utilizzare MongoDB come sistema di gestione della base di dati documentale. Si è preferito tale database rispetto a *Couchbase_{|g|}*, in quanto in ambito *NoSQL_{|g|}* è l'attuale riferimento di fatto. Tale sistema di gestione della base di dati è inoltre utilizzato anche in grandi prodotti software quali ad esempio *Facebook_{|g|}* o *Twitter_{|g|}*.

Vantaggi:

- è un database di tipo documentale, i documenti sono manipolati in JSON e salvati in *BSON_{|g|}*;
- scalabilità (*sharding_{|g|}*) molto facile e prestante;
- superamento delle limitazioni dettate dallo schema della base di dati e quindi più manutenibilità della stessa;
- migliori performance in lettura/scrittura;
- tipi di dato più flessibili;
- eliminazione totale delle operazioni di *join_{|g|}* tra *collection_{|g|}*.

Svantaggi:

- rientrando nella famiglia dei database NoSQL, non sono disponibili caratteristiche tipiche dei database relazionali, quali ad esempio il supporto alle transazioni e a vincoli di integrità.

2.12 YUI Compressor

Si è deciso di utilizzare la libreria YUI Compressor per minimizzare il codice JavaScript che verrà inviato al client.

Vantaggi:

- permette di offuscare il codice inviato al client;
- permette di ridurre il tempo di caricamento della pagina e di ridurre la banda utilizzata.

3 Descrizione architettura

3.1 Metodo e formalismo di specifica

L'esposizione dell'architettura dell'applicazione procede con un approccio di tipo top-down. Si descrive quindi l'architettura partendo dal generale, decomponendo le componenti fino a raggiungere il particolare. Si inizia quindi con la descrizione delle componenti. Per ogni componente si analizzeranno in dettaglio le singole classi, specificando per ognuna il tipo, l'obiettivo, la funzione e le relazioni in ingresso e in uscita. Le relazioni in uscita sono rappresentate da ' \rightarrow ', quelle in entrata da ' \leftarrow '. Tutte le componenti e le classi che compaiono solo all'interno degli elenchi, vengono nominate per esteso.

Dopo aver illustrato l'architettura ad alto livello, si procede con l'indicazione di esempi relativi all'utilizzo dei design pattern. È inoltre presente un approfondimento, che non dipende dall'implementazione scelta, in Appendice nella sottosezione A.

Per i diagrammi delle componenti, di classe e di attività si utilizza il formalismo $UML_{[g]}$ 2.0. Per permettere di distinguere facilmente le classi e le componenti presenti in librerie e framework esterni utilizzati dall'applicazione, questi verranno rappresentati con un colore verde. Le classi e le componenti dell'applicazione sono invece raffigurati con un colore azzurro.

L'architettura generale non specifica l'intero insieme delle sottoclassi del sistema. Di conseguenza viene specificato lo scopo di una gerarchia e vengono individuate le relazioni con le varie componenti del sistema. L'individuazione di tutti gli attributi, metodi e sottoclassi viene rimandata alla **Progettazione di Dettaglio**. Tale condizione viene supportata dalla numerosità delle varie operazioni e funzionalità che il sistema si propone di soddisfare. Poiché l'individuazione di tutte le classi specifiche per fornire tali funzionalità non aggiunge variazioni nell'architettura ad alto livello, vengono fornite classi di esempio per indicare la funzione generale di una gerarchia.

3.2 Architettura generale

L'architettura del software segue il design pattern Three-Tier. Viene quindi suddivisa in tre livelli:

1. Client Tier;
2. Server Tier;
3. Db Tier.

Di seguito viene proposto un diagramma rappresentante le relazioni tra i vari livelli. Vengono individuate le componenti che permettono ai vari livelli di interagire senza dover esporre la struttura interna degli stessi.

3 Descrizione architettura

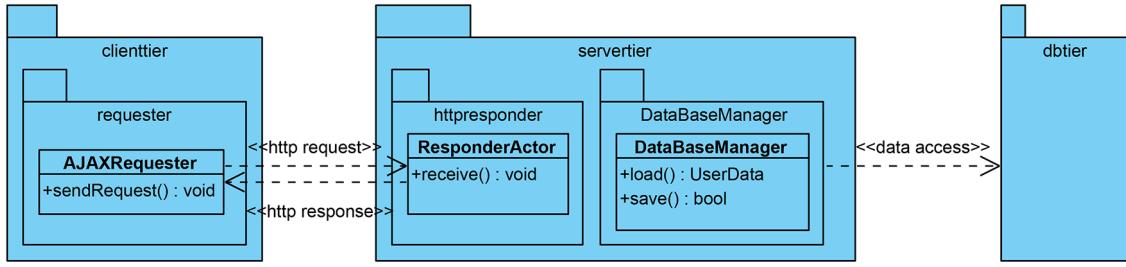


Figura 1: Diagramma delle interazioni tra i tre livelli dell'architettura

Come evidenziato nel diagramma, che non rispetta il formalismo UML 2.0, il livello del client interagisce con il server tramite una determinata classe **AJAXRequester** per l'invio di una richiesta al server. Il server riceverà e analizzerà la richiesta mediante la classe **ResponderActor** che si occuperà di trasferirla alla componente **businesslogic**. Restituisce infine una risposta al client. Il client, una volta ricevuta, si occuperà di analizzare la risposta per determinare se la richiesta ha avuto un esito positivo o negativo. Successivamente delegherà la gestione alla componente **controller** del client.

3.2.1 Client Tier

Per il livello client è stato adottato il design pattern MVC. L'architettura viene quindi suddivisa in **model**, **view** e **controller**. Di seguito si propone un diagramma rappresentante le relazioni tra le varie componenti.

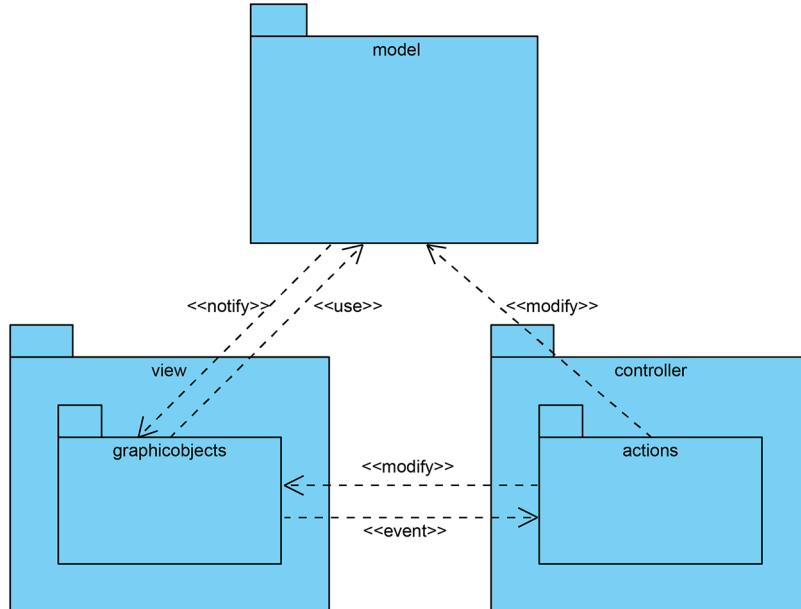


Figura 2: Diagramma delle interazioni tra le tre componenti del MVC

Come indicato in figura, gli oggetti grafici costituiranno l’interfaccia utente che verrà usata per interagire con il mondo di gioco. Essi invocheranno determinate azioni al verificarsi di eventi scaturiti dall’interazione con l’utente. Una volta che le azioni determinano come il sistema deve comportarsi al verificarsi degli eventi, agiscono sull’interfaccia utente per modificare gli oggetti grafici. Inoltre, le stesse modificano il modello dei dati per mantenerlo aggiornato al corrispettivo presente sul server. Quando il modello è aggiornato, esso si occuperà di notificare tutti gli oggetti dell’interfaccia grafica che necessitano di sapere che il modello ha cambiato il suo stato. Di conseguenza ogni oggetto grafico saprà se la sua rappresentazione non è più consistente e si occuperà di ottenere lo stato del **model** per potersi aggiornare.

3.2.2 Server Tier

Per il livello server è stato adottato il design pattern Three-Tier. L’architettura viene quindi suddivisa in **presentation**, **businesslogic** e **dataaccess**. Di seguito si propone un diagramma che non rispetta il formalismo UML 2.0, rappresentate le relazioni tra i vari livelli.

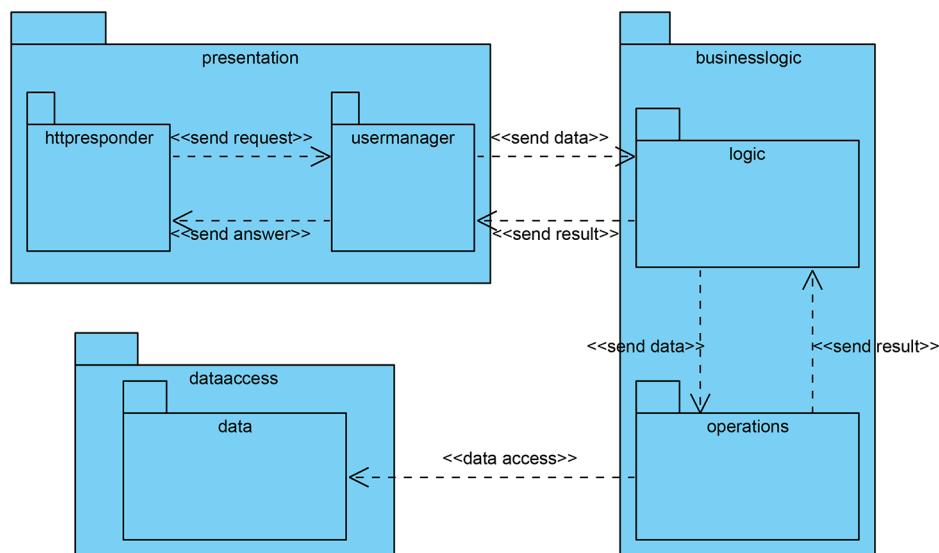


Figura 3: Diagramma delle interazioni tra i tre livelli del server

La rappresentazione evidenzia il passaggio di dati e richieste tra i vari livelli dell’architettura. Una volta che il server riceve una richiesta HTTP, si occuperà di analizzarla. Viene di conseguenza individuato l’oggetto (attore) che sta gestendo la sessione dell’utente e si trasferisce la richiesta. Vengono quindi estratti i dati effettivi della richiesta e trasferiti alla **businesslogic**. Essa individua la giusta operazione da eseguire, la quale analizzerà i dati associati alla richiesta ed eventualmente comunicherà al **model** come aggiornarsi. Viene quindi fornita una risposta che risalirà i livelli per poter essere ritrasmessa al client. La gestione del **model** e la comunicazione con il **tier_{1g}** sottostante è stata progettata seguendo la linea guida del design pattern DAO (Data Access Object). Esisterà una componente che

si occuperà di interagire con la base di dati, che sarà l'unico punto di comunicazione con il livello database.

3.2.3 Db Tier

Il livello database è gestito da sistema esterno che fornisce la funzionalità di persistenza dei dati. Slegare il database in un livello a parte rende più debole l'accoppiamento con il resto del sistema. Il database verrà implementato con MongoDB come da requisito R0V1.

3.2.4 Protocollo di comunicazione client-server

È stato progettato un protocollo per la comunicazione tra client e server. Lo scopo di esso è quello di risolvere una problematica relativa alle tecnologie utilizzate. Non avendo preferito la gestione della comunicazione mediante websocket, il server non è in grado di comunicare con il client a meno di una specifica interazione proveniente dal client verso il server. Poiché i dati di un utente possono essere modificati anche da altri utenti, in merito ad operazioni tra giocatori, si necessita di notificare al client che il suo modello dei dati non è più consistente con il corrispettivo del server. Di conseguenza, è stato preferito modellare la comunicazione mediante un protocollo adatto alla particolare situazione. Tale protocollo si basa su l'utilizzo del *piggybacking*_[g] per notificare al client le modifiche da apportare al modello per mantenerlo consistente. Le modifiche vengono notificate solamente quando l'utente effettua un'azione che richiede un'interazione con il server. La scelta di adottare questo protocollo a sfavore dell'utilizzo dei websocket non compromette l'esperienza dell'utente.

Poiché questo è un protocollo non basato su una particolare tecnologia, è possibile implementarlo secondo le proprie esigenze. Tuttavia, è stato preferito l'utilizzo del formato JSON per la costruzione dei messaggi.

Un'eventuale futura applicazione, che utilizzi la medesima architettura server, dovrà rispettare tale protocollo per comunicare correttamente con il *back end*_[g]. Di seguito viene presentata la struttura di un messaggio standard inviato dal client:

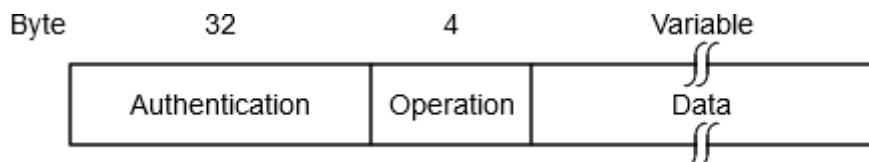


Figura 4: Diagramma della struttura di un messaggio inviato dal client al server

1. **Authentication:** è una stringa di dimensione pari a 32 byte che identifica univocamente il client durante la sessione. Essa è necessaria al server per identificare il client che ha trasmesso il messaggio. Tale stringa viene generata automaticamente dal server al momento dell'autenticazione da parte del client. Se l'autenticazione ha successo, il server comunicherà tale dato al client che la utilizzerà ogni volta che effettuerà una richiesta.

2. **Operation:** è un codice identificativo di dimensione pari a 4 byte dell'operazione da eseguire. In base a tale campo il server determinerà l'operazione che è stata richiesta. Sarà poi l'operazione a determinare se è stata invocata correttamente o se i dati che sono stati trasmessi non sono quelli necessari per completarla.
3. **Data:** contiene i dati che verranno analizzati dall'operazione. Non c'è un limite per la quantità di dati che possono essere trasmessi.

Un esempio di tale protocollo implementato con il formato previsto è il seguente:

```
{authentication: d1f5da4f5da4f6d5a4f6a4ffe156daf, operation: 1250, data:
    {key: MinieraL1X4Y3}}
```

Di seguito viene rappresentato un tipico messaggio inviato dal server in risposta ad una richiesta del client.

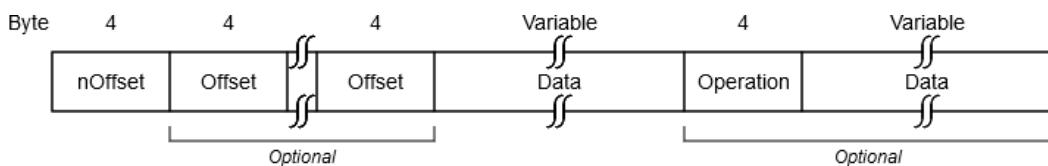


Figura 5: Diagramma della struttura di un messaggio inviato dal server al client

1. **nOffset:** è una valore numerico, di dimensione pari a 4 byte, che indica il numero di dati aggiuntivi che sono stati inseriti nel messaggio. Il client, leggendo questo valore, determina quanti offset saranno indicati per determinare l'inizio di ogni stringa di dati.
2. **Offset:** è un valore numerico, di dimensione pari a 4 byte, indicante la posizione, all'interno del messaggio, di una determinata sequenza di dati. Nel caso in cui il campo "nOffset" sia pari a zero, non sarà presente alcun campo di tale tipo.
3. **Data:** contiene i dati ritornati dal server in seguito alla richiesta precedentemente inviata.
4. **Operation:** è un codice identificativo, di dimensione pari a 4 byte, dell'operazione che deve occuparsi di leggere i dati inviati in piggybacking. Tale campo non è presente se non sono stati inseriti nel messaggio dei dati aggiuntivi.
5. **Data:** contiene i dati che verranno analizzati dall'operazione. Non c'è un limite per la quantità di dati che possono essere trasmessi. Tale campo non è presente se non sono stati inseriti nel messaggio dei dati aggiuntivi.

Considerando che il formato JSON permette di memorizzare un array di coppie chiavi valore, i primi due campi vengono omessi sfruttando le caratteristiche del formato. Un esempio di tale protocollo implementato con il formato previsto è il seguente:

```
{data: {result: true}, [{operation: 1202, data: {key: MinieraL1}}, {operation: 2302, data: {key: Cavaliere}}]}
```

3.2.5 Protocollo di comunicazione server-database

Verrà utilizzato lo schema di comunicazione offerto dal $DBMS_{|g|}$ adottato: il formato JSON. L'accesso al DBMS esterno avverrà grazie al toolkit Casbah.

4 Componenti e classi

I seguenti diagrammi rappresentano la struttura gerarchica delle componenti del prodotto SGAD.

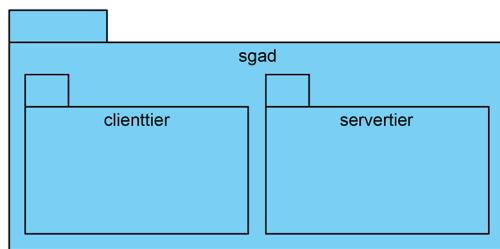


Figura 6: Albero delle componenti relativo alla componente **sgad**

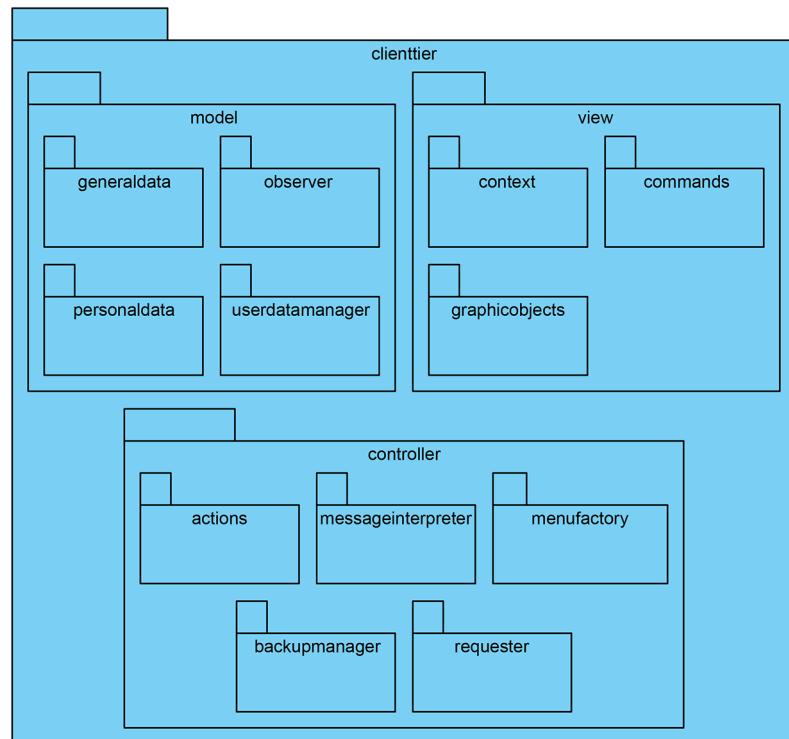


Figura 7: Albero delle componenti relativo alla componente **clienttier**

4 Componenti e classi

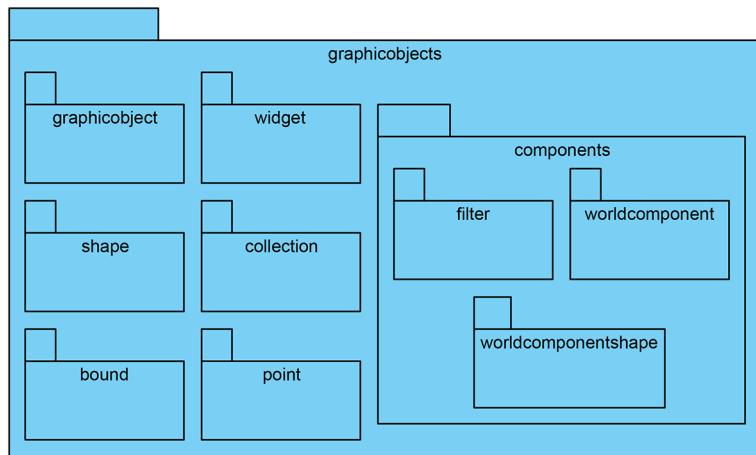


Figura 8: Albero delle componenti relativo alla componente **graphicobjects**

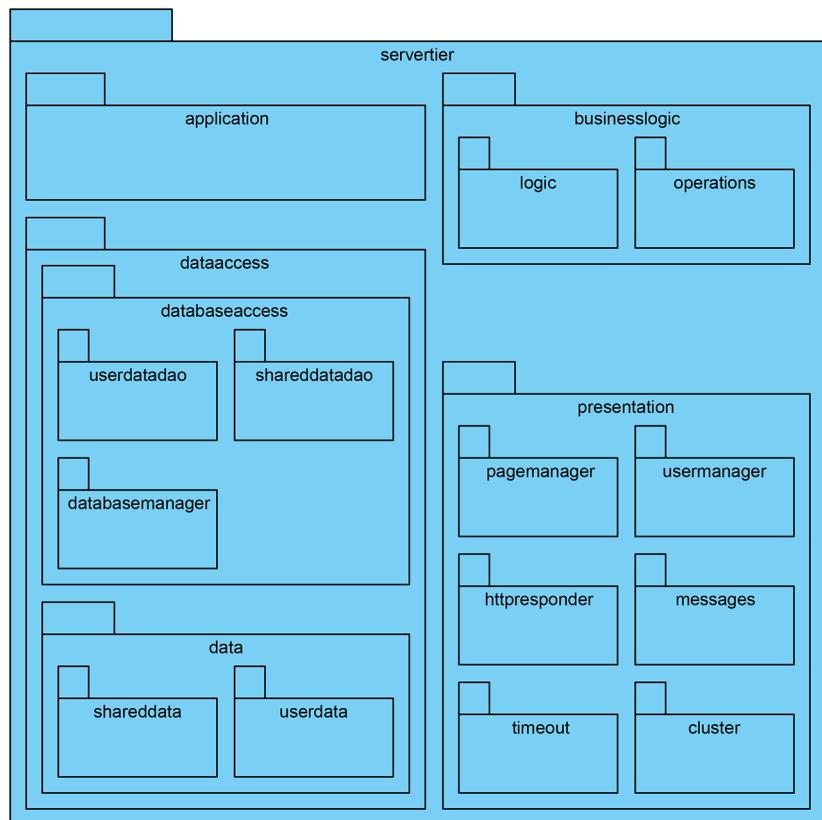


Figura 9: Albero delle componenti relativo alla componente **servtier**

4.1 Componente sgad

4.1.1 Informazioni sul package

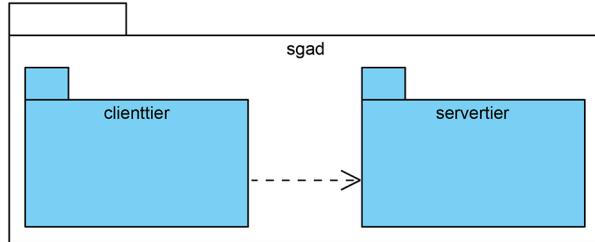


Figura 10: Diagramma della componente sgad

- **Descrizione:** componente che contiene il prodotto SGAD.

- **Package contenuti**

- **clienttier:** componente globale per il *front end_g* del prodotto. Nel *Three-Tier_g* del sistema rappresenta il livello Client Tier. Le relazioni tra le sottocomponenti **model**, **view** e **controller** rappresentano le interazioni tipiche che intercorrono tra le componenti del design pattern MVC. Esso si occupa di visualizzare i dati all'utente e di inoltrare le richieste al back end.
- **servertier:** componente globale per il back end del prodotto. Le relazioni tra le componenti **presentation**, **businesslogic** e **dataaccess** rappresentano delle interazioni tipiche. Esse intercorrono tra le componenti del design pattern Three-Tier.

4.2 Componente sgad::clienttier

4.2.1 Informazioni sul package

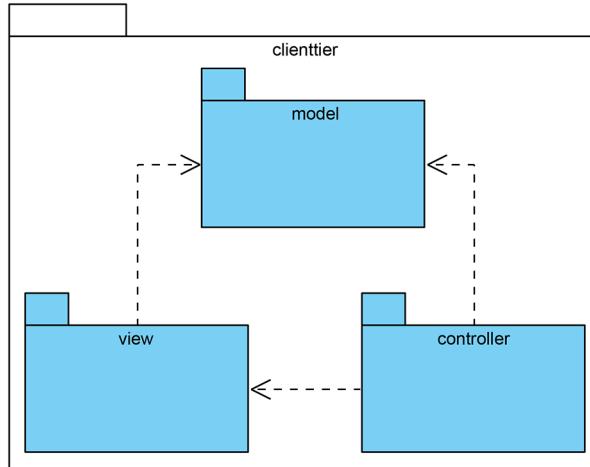


Figura 11: Diagramma della componente `sgad::clienttier`

- **Descrizione:** componente globale per il front end del prodotto. Nel Three-Tier del sistema rappresenta il livello Client Tier. Le relazioni tra le sottocomponenti `model`, `view` e `controller` rappresentano le interazioni tipiche che intercorrono tra le componenti del design pattern MVC. Esso si occupa di visualizzare i dati all'utente e di inoltrare le richieste al back end.
- **Padre:** `sgad`
- **Interazioni con altri componenti**
 - `servertier`: componente globale per il back end del prodotto. Le relazioni tra le componenti `presentation`, `businesslogic` e `dataaccess` rappresentano delle interazioni tipiche. Esse intercorrono tra le componenti del design pattern Three-Tier.
- **Package contenuti**
 - `controller`: componente Controller dell'architettura $MVC_{|g|}$. Essa esegue le operazioni che l'utente ha richiesto tramite la view agendo se necessario sul $model_{|g|}$ data.
 - `model`: componente Model dell'architettura MVC. Essa memorizza tutti i dati di gioco dell'utente su cui la view si basa per disegnarsi.
 - `view`: componente View dell'architettura MVC. Essa gestisce la visualizzazione del gioco e l'input utente all'interno del canvas HTML5.

4.3 Componente sgad::clienttier::controller

4.3.1 Informazioni sul package

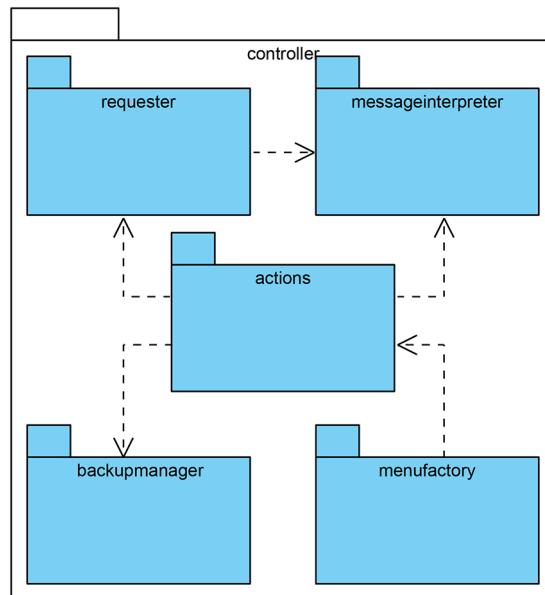


Figura 12: Diagramma della componente `sgad::clienttier::controller`

- **Descrizione:** componente Controller dell'architettura MVC. Essa esegue le operazioni che l'utente ha richiesto tramite la view agendo se necessario sul model data.
- **Padre:** `clienttier`
- **Interazioni con altri componenti**
 - **model**: componente Model dell'architettura MVC. Essa memorizza tutti i dati di gioco dell'utente su cui la view si basa per disegnarsi.
 - **view**: componente View dell'architettura MVC. Essa gestisce la visualizzazione del gioco e l'input utente all'interno del canvas HTML5.
- **Package contenuti**
 - **actions**: componente per la gestione delle possibili azioni che un utente può intraprendere durante l'interazione con il mondo di gioco.
 - **backupmanager**: componente che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
 - **menufactory**: componente per la gestione dei vari menu che possono essere visualizzati durante le interazioni con il mondo di gioco.
 - **messageinterpreter**: componente per l'interpretazione delle reply ricevute dal server.
 - **requester**: componente per la gestione dell'invio di richieste al server.

4.4 Componente sgad::clienttier::controller::actions

4.4.1 Informazioni sul package

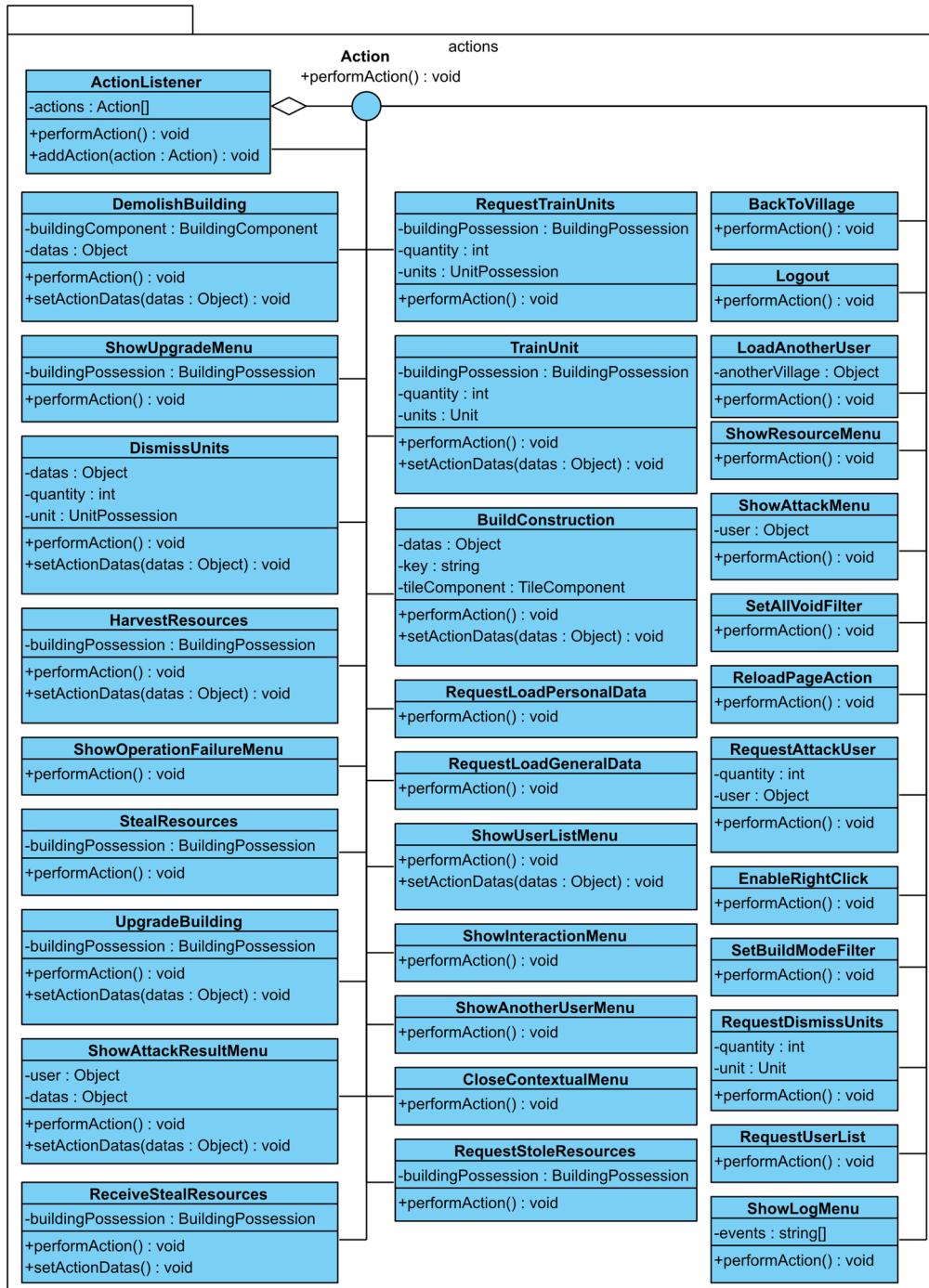
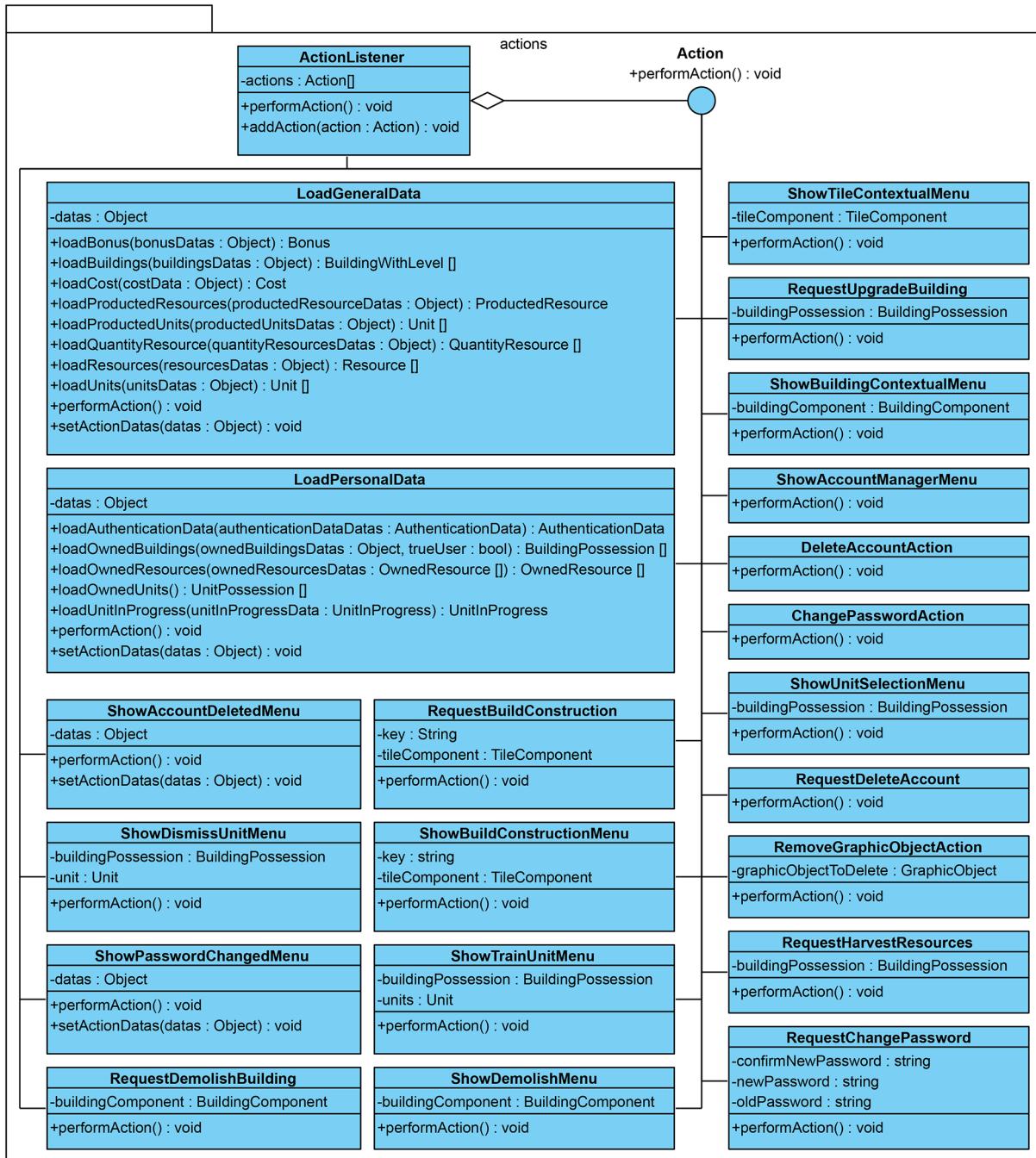


Figura 13: Diagramma della componente `sgad::clienttier::controller::actions`


 Figura 14: Diagramma della componente `sgad::clienttier::controller::actions`

- **Descrizione:** componente per la gestione delle possibili azioni che un utente può intraprendere durante l'interazione con il mondo di gioco.
- **Padre:** `controller`

- **Interazioni con altri componenti**

- **messageinterpreter**: componente per l'interpretazione delle reply ricevute dal server.
- **requester**: componente per la gestione dell'invio di richieste al server.
- **graphicobjects**: componente per la gestione degli oggetti grafici che vengono rappresentati nel mondo di gioco quali widget e immagini rappresentanti il mondo di gioco (esempio: edificio).

4.4.2 Classi

4.4.2.1 sgad::clienttier::controller::actions::Action

- **Descrizione**: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.
- **Utilizzo**: viene utilizzata per fornire una firma di metodo standard per qualsiasi tipo di azione che è possibile eseguire. Le classi che realizzano l'interfaccia si occuperanno di ridefinire il metodo per eseguire la particolare azione. Permette di stabilire un punto di comunicazione tra la view e il model. Inoltre, facilita l'associazione di un'azione ad un elemento della view.

- **Sottoclassi**

- ActionListener
- BackToVillage
- BuildConstruction
- ChangePasswordAction
- CloseContextualMenu
- DeleteAccountAction
- DemolishBuilding
- DismissUnits
- EnableRightClick
- HarvestResources
- LoadAnotherUser
- LoadGeneralData
- LoadPersonalData
- Logout
- ReceiveStealResources
- ReloadPageAction
- RemoveGraphicObjectAction

- RequestAttackUser
- RequestBuildConstruction
- RequestChangePassword
- RequestDeleteAccount
- RequestDemolishBuilding
- RequestDismissUnits
- RequestHarvestResources
- RequestLoadGeneralData
- RequestLoadPersonalData
- RequestStoleResources
- RequestTrainUnits
- RequestUpgradeBuilding
- SetAllVoidFilter
- SetBuildModeFilter
- ShowAccountDeletedMenu
- ShowAccountManagerMenu
- ShowAnotherUserMenu
- ShowAttackMenu
- ShowAttackResultMenu
- ShowBuildConstructionMenu
- ShowBuildingContextualMenu
- ShowDemolishMenu
- ShowDismissUnitMenu
- ShowInteractionMenu
- ShowOperationFailureMenu
- ShowPasswordChangedMenu
- ShowResourceMenu
- ShowTileContextualMenu
- ShowTrainUnitMenu
- ShowUnitSelectionMenu
- ShowUpgradeMenu
- ShowUserListMenu
- StealResources
- TrainUnit
- UpgradeBuilding

- Relazioni con altre classi

- ← **ActionListener**: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
- ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
- ← **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
- ← **AJAXRequester**: classe per la gestione dell'invio delle richieste al server da parte del client.
- ← **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- ← **TileComponent**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- ← **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.2 sgad::clienttier::controller::actions::ActionListener

- **Descrizione**: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
- **Utilizzo**: viene utilizzata per mantenere una lista di azioni. Ciò permette di comporre una sequenza di azioni senza dover definire una nuova classe che raggruppa tali azioni esplicitamente. Fornisce quindi la possibilità di associare un insieme di azioni ad un elemento dell'interfaccia grafica.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **LoadPersonalData**: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← **RequestLoadGeneralData**: classe per la gestione della richiesta per il caricamento dei dati globali.
 - ← **RequestLoadPersonalData**: classe per la gestione della richiesta per il caricamento dei dati dell'utente.
 - ← **AccountManagerMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
 - ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.

- ← ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
- ← LogMenuFactory: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← NotifyMenuFactory: classe per la costruzione di un menu per la visualizzazione di una notifica.
- ← TileContextualMenuFactory: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- ← UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- → Action: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.

4.4.2.3 sgad::clienttier::controller::actions::BackToVillage

- **Descrizione:** classe per permettere di ritornare al proprio villaggio nella mappa.
- **Utilizzo:** viene utilizzata per permettere all'utente di ritornare al proprio villaggio una volta che ha saccheggiato il villaggio di un altro utente.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← AnotherUserMenuFactory: classe per la costruzione di un menu per il ritorno al villaggio dell'utente.
 - → BackupManager: classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.

4.4.2.4 sgad::clienttier::controller::actions::BuildConstruction

- **Descrizione:** classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
- **Utilizzo:** viene utilizzata per permettere all'utente di costruire un nuovo edificio nel villaggio.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - → RequestHarvestResources: classe per la richiesta di raccolta di risorse.
 - → SetAllVoidFilter: classe per la gestione di un filtro standard.

- → `ShowBuildingContextualMenu`: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
- → `ShowOperationFailureMenu`: classe per la visualizzazione di un menu al fallimento di un'operazione.
- → `NotifyMenuFactory`: classe per la costruzione di un menu per la visualizzazione di una notifica.
- → `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
- → `Cost`: classe per la gestione di un costo in termini di risorse e tempo.
- → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
- → `QuantityResource`: classe per la gestione della quantità di una particolare risorsa.
- → `Resource`: classe che rappresenta una risorsa.
- → `BuildingPossession`: classe per la gestione di un edificio posseduto da un utente.
- → `OwnedResource`: classe che rappresenta una risorsa posseduta da un utente.
- → `Position`: classe per la gestione di una posizione sulla griglia del villaggio.
- → `UserData`: classe per la gestione dei dati di un utente.
- → `UserDataManager`: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- → `Context`: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → `VoidFilter`: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
- → `BuildingComponent`: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → `TileComponent`: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- → `FrameWidget`: classe per la gestione di un generico frame nell'interfaccia grafica.
- → `Widget`: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.5 sgad::clienttier::controller::actions::ChangePasswordAction

- **Descrizione:** classe per permettere di modificare la password di un account.
- **Utilizzo:** viene utilizzata per modificare la password associata all'account.
- **Classi ereditate**

– Action

- Relazioni con altre classi

- → RequestChangePassword: classe per la gestione della richiesta di cambiamento password.
- → ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
- → NotifyMenuFactory: classe per la costruzione di un menu per la visualizzazione di una notifica.
- → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
- → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.6 sgad::clienttier::controller::actions::CloseContextualMenu

- **Descrizione:** classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
- **Utilizzo:** viene utilizzata per gestire la chiusura del menu contestuale correntemente visualizzato.

- Classi ereditate

– Action

- Relazioni con altre classi

- ← LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.
- ← ShowAccountManagerMenu: classe per la creazione di un menu per la gestione dell'account.
- ← ShowAttackMenu: classe per la visualizzazione del menu per l'attacco.
- ← ShowBuildConstructionMenu: classe per la richiesta della conferma della costruzione.
- ← ShowBuildingContextualMenu: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
- ← ShowDemolishMenu: classe che visualizza il menu per la demolizione di un edificio.
- ← ShowDismissUnitMenu: classe per la gestione della visualizzazione di un menu per il congedo di unità.

- ← `ShowTileContextualMenu`: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
- ← `ShowUpgradeMenu`: classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
- ← `AccountManagerMenuFactory`: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← `BuildingContextualMenuFactory`: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← `TileContextualMenuFactory`: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- → `Context`: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → `FrameWidget`: classe per la gestione di un generico frame nell'interfaccia grafica.
- → `Widget`: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.7 sgad::clienttier::controller::actions::DeleteAccountAction

- **Descrizione:** la classe viene utilizzata per l'eliminazione di un account utente.
- **Utilizzo:** viene utilizzata per eliminare i dati dell'account.
- **Classi ereditate**
 - `Action`
- **Relazioni con altre classi**
 - → `RequestDeleteAccount`: classe per la gestione della richiesta di eliminazione dell'account.
 - → `ConfirmMenuFactory`: classe per la creazione di un menu che permetta di confermare una generica azione.
 - → `Context`: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → `FrameWidget`: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → `Widget`: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.8 sgad::clienttier::controller::actions::DemolishBuilding

- **Descrizione:** classe per la gestione della demolizione di un edificio già presente nel villaggio.
- **Utilizzo:** viene utilizzata per permettere all'utente di demolire un edificio che ha precedentemente costruito.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← RequestDemolishBuilding: classe per la richiesta di demolizione di un edificio.
 - → RemoveGraphicObjectAction: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → ShowTileContextualMenu: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
 - → Cost: classe per la gestione di un costo in termini di risorse e tempo.
 - → QuantityResource: classe per la gestione della quantità di una particolare risorsa.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → OwnedResource: classe che rappresenta una risorsa posseduta da un utente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → UserDataManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
 - → BuildingComponent: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
 - → TileComponent: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.

4.4.2.9 sgad::clienttier::controller::actions::DismissUnits

- **Descrizione:** classe per la gestione del congedo di unità arruolate in precedenza dall'utente.
- **Utilizzo:** viene utilizzata per permettere all'utente di congedare un determinato numero di unità di un certo tipo.
- **Classi ereditate**

– Action

- Relazioni con altre classi

- ← RequestDismissUnits: classe per la richiesta di dimissione di unità possedute.
- → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
- → UnitPossession: classe per la rappresentazione di un'unità con associata una quantità.
- → UserData: classe per la gestione dei dati di un utente.
- → UserManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.

4.4.2.10 sgad::clienttier::controller::actions::EnableRightClick

- **Descrizione:** classe per permettere di abilitare la gestione dell'evento click destro del mouse.
- **Utilizzo:** viene utilizzata dai vari menu per riabilitare la gestione dell'evento rightclick. Ciò serve per evitare di continuare ad interagire con il mondo di gioco senza aver chiuso il menu visualizzato.
- **Classi ereditate**

– Action

- Relazioni con altre classi

- ← ShowAttackMenu: classe per la visualizzazione del menu per l'attacco.
- ← ShowTrainUnitMenu: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← AttackResultMenuFactory: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
- ← ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
- ← LogMenuFactory: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.

4.4.2.11 sgad::clienttier::controller::actions::HarvestResources

- **Descrizione:** classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
- **Utilizzo:** viene utilizzata per permettere all'utente di raccogliere le risorse prodotte dagli edifici di produzione.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← RequestHarvestResources: classe per la richiesta di raccolta di risorse.
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → BuildingWithLevel: classe per la gestione di un edificio ad un particolare livello.
 - → ProductedResource: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - → Resource: classe che rappresenta una risorsa.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → OwnedResource: classe che rappresenta una risorsa posseduta da un utente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → UserDataManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.

4.4.2.12 sgad::clienttier::controller::actions::LoadAnotherUser

- **Descrizione:** classe per il caricamento di un altro utente.
- **Utilizzo:** viene utilizzata per caricare i dati di un altro utente. Si occupa quindi di salvare i dati dell'utente attualmente visualizzato per poterli ripristinare una volta che si ritorna al proprio villaggio.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← AttackResultMenuFactory: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
 - → LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.

- → **ShowAnotherUserMenu**: classe per la visualizzazione del menu per ritornare al proprio villaggio una volta che si è nel villaggio di un altro utente.
- → **BackupManager**: classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **GraphicObjectCollection**: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.

4.4.2.13 sgad::clienttier::controller::actions::LoadGeneralData

- **Descrizione**: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
- **Utilizzo**: viene utilizzata per caricare i dati globali all'avvio di una nuova sessione. Il caricamento di tali dati è quindi unico. Viene effettuato ad ogni accesso al mondo di gioco.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **RequestLoadGeneralData**: classe per la gestione della richiesta per il caricamento dei dati globali.
 - → **RequestLoadPersonalData**: classe per la gestione della richiesta per il caricamento dei dati dell'utente.
 - → **Bonus**: classe per la gestione dei bonus eventualmente disponibili per certi edifici.
 - → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
 - → **Cost**: classe per la gestione di un costo in termini di risorse e tempo.
 - → **DataFactory**: classe che ritorna un'istanza delle componenti di gioco.
 - → **ProductedResource**: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - → **QuantityResource**: classe per la gestione della quantità di una particolare risorsa.
 - → **Resource**: classe che rappresenta una risorsa.
 - → **Unit**: classe per la gestione delle informazioni di una unità.

4.4.2.14 sgad::clienttier::controller::actions::LoadPersonalData

- **Descrizione:** classe per la gestione del caricamento dei dati personali dell'utente.
- **Utilizzo:** viene utilizzata per caricare i dati dell'utente all'avvio di una nuova sessione. Il caricamento viene effettuato ad ogni accesso al mondo di gioco. Si occupa inoltre di creare gli elementi di interfaccia grafica che rispecchieranno i dati del mondo di gioco.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← LoadAnotherUser: classe per il caricamento di un altro utente.
 - ← RequestLoadPersonalData: classe per la gestione della richiesta per il caricamento dei dati dell'utente.
 - → ActionListener: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
 - → CloseContextualMenu: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → ShowTileContextualMenu: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
 - → StealResources: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
 - → DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - → ProductedResource: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - → AuthenticationData: classe per la gestione delle informazioni personali e di accesso di un utente.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → OwnedResource: classe che rappresenta una risorsa posseduta da un utente.
 - → Position: classe per la gestione di una posizione sulla griglia del villaggio.
 - → UnitInProgress: classe per la gestione della coda di costruzione di unità di un particolare edificio.
 - → UnitPossession: classe per la rappresentazione di un'unità con associata una quantità.
 - → UserData: classe per la gestione dei dati di un utente.
 - → UserManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.

- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **VoidFilter**: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
- → **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → **TileComponent**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- → **Resource**: classe che rappresenta una risorsa.

4.4.2.15 sgad::clienttier::controller::actions::Logout

- **Descrizione**: classe per la gestione del logout dell'utente.
- **Utilizzo**: viene utilizzata per eseguire il logout una volta che l'utente ha terminato di interagire con il mondo di gioco.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - → **AJAXRequester**: classe per la gestione dell'invio delle richieste al server da parte del client.

4.4.2.16 sgad::clienttier::controller::actions::ReceiveStealResources

- **Descrizione**: classe per la ricezione delle risorse rubate
- **Utilizzo**: gestisce le risorse rubate
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - → **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → **BackupManager**: classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
 - → **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.

- → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- → **ProducedResource**: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
- → **Resource**: classe che rappresenta una risorsa.
- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.17 sgad::clienttier::controller::actions::ReloadPageAction

- **Descrizione**: azione per il ricaricamento della pagina.
- **Utilizzo**: viene utilizzata per richiedere il ricaricamento di una pagina in seguito ad un fallimento di una richiesta al server.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.

4.4.2.18 sgad::clienttier::controller::actions::RemoveGraphicObjectAction

- **Descrizione**: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
- **Utilizzo**: viene utilizzata per rimuovere un oggetto grafico che rappresenta un elemento dell'interfaccia grafica. Tale azione non conosce il tipo effettivo dell'oggetto che andrà a rimuovere. Tuttavia richiede che l'oggetto da rimuovere sia un sottotipo di **GraphicObject**.
- **Classi ereditate**
 - Action

- **Relazioni con altre classi**

- ← **DemolishBuilding**: classe per la gestione della demolizione di un edificio già presente nel villaggio.
- ← **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
- ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
- ← **LogMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
- ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **GraphicObject**: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.

4.4.2.19 sgad::clienttier::controller::actions::RequestAttackUser

- **Descrizione**: classe per la gestione della richiesta di attacco di un utente desiderato.

- **Utilizzo**: viene utilizzata per richiedere al server di attaccare l'utente desiderato.

- **Classi ereditate**

- Action

- **Relazioni con altre classi**

- ← **ShowAttackMenu**: classe per la visualizzazione del menu per l'attacco.
- → **ShowAttackResultMenu**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
- → **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.

4.4.2.20 sgad::clienttier::controller::actions::RequestBuildConstruction

- **Descrizione**: classe per la richiesta di costruzione di un edificio.

- **Utilizzo**: viene utilizzata per inviare al server una richiesta per la costruzione dell'edificio desiderato.

- **Classi ereditate**

- Action
- Relazioni con altre classi
 - ← ShowBuildConstructionMenu: classe per la richiesta della conferma della costruzione.
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → AJAXRequester: classe per la gestione dell'invio delle richieste al server da parte del client.
 - → Position: classe per la gestione di una posizione sulla griglia del villaggio.
 - → TileComponent: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.

4.4.2.21 sgad::clienttier::controller::actions::RequestChangePassword

- Descrizione: classe per la gestione della richiesta di cambiamento password.
- Utilizzo: viene utilizzata per richiedere al server la modifica della password dell'utente.
- Classi ereditate
 - Action
- Relazioni con altre classi
 - ← ChangePasswordAction: classe per permettere di modificare la password di un account.
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → ShowPasswordChangedMenu: classe per la visualizzazione di un menu di conferma del cambiamento della password.
 - → AJAXRequester: classe per la gestione dell'invio delle richieste al server da parte del client.

4.4.2.22 sgad::clienttier::controller::actions::RequestDeleteAccount

- Descrizione: classe per la gestione della richiesta di eliminazione dell'account.
- Utilizzo: viene utilizzata per richiedere al server di eliminare l'account dell'utente.
- Classi ereditate
 - Action
- Relazioni con altre classi

- ← **DeleteAccountAction**: la classe viene utilizzata per l'eliminazione di un account utente.
- → **ShowAccountDeletedMenu**: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.
- → **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
- → **AJAXRequester**: classe per la gestione dell'invio delle richieste al server da parte del client.

4.4.2.23 sgad::clienttier::controller::actions::RequestDemolishBuilding

- **Descrizione**: classe per la richiesta di demolizione di un edificio.
- **Utilizzo**: viene utilizzata per inviare al server una richiesta per demolire l'edificio desiderato.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **ShowDemolishMenu**: classe che visualizza il menu per la demolizione di un edificio.
 - → **DemolishBuilding**: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - → **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → **AJAXRequester**: classe per la gestione dell'invio delle richieste al server da parte del client.
 - → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
 - → **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.

4.4.2.24 sgad::clienttier::controller::actions::RequestDismissUnits

- **Descrizione**: classe per la richiesta di dimissione di unità possedute.
- **Utilizzo**: viene utilizzata per inviare una richiesta al server per congedare le unità desiderate.
- **Classi ereditate**
 - Action

- Relazioni con altre classi

- ← `ShowDismissUnitMenu`: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- → `DismissUnits`: classe per la gestione del congedo di unità arruolate in precedenza dall'utente.
- → `ShowOperationFailureMenu`: classe per la visualizzazione di un menu al fallimento di un'operazione.
- → `AJAXRequester`: classe per la gestione dell'invio delle richieste al server da parte del client.
- → `Unit`: classe per la gestione delle informazioni di una unità.

4.4.2.25 sgad::clienttier::controller::actions::RequestHarvestResources

- **Descrizione:** classe per la richiesta di raccolta di risorse.

- **Utilizzo:** viene utilizzata per inviare al server una richiesta per raccogliere le risorse dell'edificio selezionato.

- Classi ereditate

- `Action`

- Relazioni con altre classi

- ← `BuildConstruction`: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
- → `HarvestResources`: classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
- → `ShowOperationFailureMenu`: classe per la visualizzazione di un menu al fallimento di un'operazione.
- → `AJAXRequester`: classe per la gestione dell'invio delle richieste al server da parte del client.
- → `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
- → `ProductedResource`: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
- → `BuildingPossession`: classe per la gestione di un edificio posseduto da un utente.

4.4.2.26 sgad::clienttier::controller::actions::RequestLoadGeneralData

- **Descrizione:** classe per la gestione della richiesta per il caricamento dei dati globali.

- **Utilizzo:** viene utilizzata per generare la richiesta che avvierà il caricamento dei dati globali. Viene quindi imposto, come azione al successo della richiesta per il caricamento effettivo dei dati globali, un oggetto di tipo `LoadGeneralData`. Viene poi eseguita la richiesta per il caricamento dei dati personali dell'utente.
- **Classi ereditate**
 - `Action`
- **Relazioni con altre classi**
 - → `ActionListener`: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
 - → `LoadGeneralData`: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - → `ShowOperationFailureMenu`: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → `AJAXRequester`: classe per la gestione dell'invio delle richieste al server da parte del client.

4.4.2.27 `sgad::clienttier::controller::actions::RequestLoadPersonalData`

- **Descrizione:** classe per la gestione della richiesta per il caricamento dei dati dell'utente.
- **Utilizzo:** viene utilizzata per generare la richiesta che avvierà il caricamento dei dati dell'utente. Viene quindi imposto, come azione al successo della richiesta per il caricamento effettivo dei dati personali, un oggetto di tipo `LoadPersonalData`
- **Classi ereditate**
 - `Action`
- **Relazioni con altre classi**
 - ← `LoadGeneralData`: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - → `ActionListener`: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
 - → `LoadPersonalData`: classe per la gestione del caricamento dei dati personali dell'utente.
 - → `ShowOperationFailureMenu`: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → `ShowResourceMenu`: classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
 - → `AJAXRequester`: classe per la gestione dell'invio delle richieste al server da parte del client.

4.4.2.28 sgad::clienttier::controller::actions::RequestStoleResources

- **Descrizione:** classe per la richiesta di furto di risorse
- **Utilizzo:** gestisce l'azione del furto di risorse
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → StealResources: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
 - → AJAXRequester: classe per la gestione dell'invio delle richieste al server da parte del client.
 - → AuthenticationData: classe per la gestione delle informazioni personali e di accesso di un utente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → UserDataManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.

4.4.2.29 sgad::clienttier::controller::actions::RequestTrainUnits

- **Descrizione:** classe per la gestione della richiesta per il caricamento dei dati globali.
- **Utilizzo:** viene utilizzata per inviare al server la richiesta di addestramento delle unità selezionate nell'edificio desiderato.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → TrainUnit: classe per la gestione dell'addestramento di unità in un edificio.
 - → AJAXRequester: classe per la gestione dell'invio delle richieste al server da parte del client.
 - → Unit: classe per la gestione delle informazioni di una unità.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.

4.4.2.30 sgad::clienttier::controller::actions::RequestUpgradeBuilding

- **Descrizione:** classe per la gestione della richiesta di miglioramento di un edificio.
- **Utilizzo:** viene utilizzata per inviare al server la richiesta di miglioramento dell'edificio desiderato.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← ShowUpgradeMenu: classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → UpgradeBuilding: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
 - → AJAXRequester: classe per la gestione dell'invio delle richieste al server da parte del client.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.

4.4.2.31 sgad::clienttier::controller::actions::SetAllVoidFilter

- **Descrizione:** classe per la gestione di un filtro standard.
- **Utilizzo:** viene utilizzata per permettere all'utente di ritornare ad una visualizzazione normale. Viene quindi impostato il filtro standard agli elementi di interfaccia grafica del villaggio dell'utente.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← BuildConstruction: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← InteractionMenuFactory: classe per la creazione di un menu per interagire con l'intero mondo di gioco.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → GraphicObjectCollection: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.

- → **GraphicObjectIterator**: classe per la gestione di un iteratore per accedere ad una collezione di oggetti grafici.
- → **Iterator**: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
- → **VoidFilter**: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.

4.4.2.32 sgad::clienttier::controller::actions::SetBuildModeFilter

- **Descrizione**: classe per la gestione dei filtri per la modalità costruzione.
- **Utilizzo**: viene utilizzata per impostare come visualizzazione la modalità di costruzione.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → **GraphicObjectCollection**: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.
 - → **GraphicObjectIterator**: classe per la gestione di un iteratore per accedere ad una collezione di oggetti grafici.
 - → **Iterator**: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
 - → **BuildModeFilter**: classe per la gestione del filtro quando viene selezionata la modalità costruzione.

4.4.2.33 sgad::clienttier::controller::actions::ShowAccountDeletedMenu

- **Descrizione**: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.
- **Utilizzo**: visualizza un menu per indicare l'esito della cancellazione dell'account dell'utente.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**

- ← `RequestDeleteAccount`: classe per la gestione della richiesta di eliminazione dell'account.
- → `ShowOperationFailureMenu`: classe per la visualizzazione di un menu al fallimento di un'operazione.
- → `NotifyMenuFactory`: classe per la costruzione di un menu per la visualizzazione di una notifica.
- → `Context`: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → `FrameWidget`: classe per la gestione di un generico frame nell'interfaccia grafica.
- → `Widget`: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.
- → `Logout`: classe che rappresenta l'operazione di logout.

4.4.2.34 `sgad::clienttier::controller::actions::ShowAccountManagerMenu`

- **Descrizione:** classe per la creazione di un menu per la gestione dell'account.
- **Utilizzo:** viene utilizzata per visualizzare il menu per modificare i dati dell'account utente.
- **Classi ereditate**
 - `Action`
- **Relazioni con altre classi**
 - → `CloseContextualMenu`: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → `AccountManagerMenuFactory`: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - → `Context`: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → `FrameWidget`: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → `Widget`: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.35 `sgad::clienttier::controller::actions::ShowAnotherUserMenu`

- **Descrizione:** classe per la visualizzazione del menu per ritornare al proprio villaggio una volta che si è nel villaggio di un altro utente.

- **Utilizzo:** viene utilizzata per visualizzare il menu una volta che si sta visualizzando il villaggio di un altro utente.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← LoadAnotherUser: classe per il caricamento di un altro utente.
 - → AnotherUserMenuFactory: classe per la costruzione di un menu per il ritorno al villaggio dell'utente.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.36 sgad::clienttier::controller::actions::ShowAttackMenu

- **Descrizione:** classe per la visualizzazione del menu per l'attacco.
- **Utilizzo:** viene utilizzata per visualizzare il menu per richiedere quali truppe si desidera inviare per l'attacco.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← UserListMenuFactory: classe per la visualizzazione della lista di altri giocatori connessi.
 - → CloseContextualMenu: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → EnableRightClick: classe per permettere di abilitare la gestione dell'evento click destro del mouse.
 - → RequestAttackUser: classe per la gestione della richiesta di attacco di un utente desiderato.
 - → ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
 - → NotifyMenuFactory: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - → DataFactory: classe che ritorna un'istanza delle componenti di gioco.

- → **Unit**: classe per la gestione delle informazioni di una unità.
- → **UnitPossession**: classe per la rappresentazione di un'unità con associata una quantità.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.37 sgad::clienttier::controller::actions::ShowAttackResultMenu

- **Descrizione**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
- **Utilizzo**: viene utilizzata per visualizzare l'esito dello scontro con l'utente attaccato una volta inviato l'attacco.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **RequestAttackUser**: classe per la gestione della richiesta di attacco di un utente desiderato.
 - → **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
 - → **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - → **UnitPossession**: classe per la rappresentazione di un'unità con associata una quantità.
 - → **UserData**: classe per la gestione dei dati di un utente.
 - → **UserDataManager**: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
 - → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.38 sgad::clienttier::controller::actions::ShowBuildConstructionMenu

- **Descrizione:** classe per la richiesta della conferma della costruzione.
- **Utilizzo:** viene utilizzata per richiedere all'utente la conferma della costruzione.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← TileContextualMenuFactory: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
 - → CloseContextualMenu: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → RequestBuildConstruction: classe per la richiesta di costruzione di un edificio.
 - → ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → TileComponent: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.39 sgad::clienttier::controller::actions::ShowBuildingContextualMenu

- **Descrizione:** classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
- **Utilizzo:** viene utilizzata per permettere all'utente di visualizzare un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← BuildConstruction: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.

- ← **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → **CloseContextualMenu**: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
- → **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.40 sgad::clienttier::controller::actions::ShowDemolishMenu

- **Descrizione**: classe che visualizza il menu per la demolizione di un edificio.
- **Utilizzo**: viene utilizzata per richiedere la conferma dell'azione di demolizione dell'edificio richiesto.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - → **CloseContextualMenu**: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → **RequestDemolishBuilding**: classe per la richiesta di demolizione di un edificio.
 - → **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
 - → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
 - → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.41 sgad::clienttier::controller::actions::ShowDismissUnitMenu

- **Descrizione:** classe per la gestione della visualizzazione di un menu per il congedo di unità.
- **Utilizzo:** viene utilizzata per richiedere la conferma dell'azione di congedo di unità.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - → CloseContextualMenu: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → RequestDismissUnits: classe per la richiesta di dimissione di unità possedute.
 - → ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
 - → NotifyMenuFactory: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - → Unit: classe per la gestione delle informazioni di una unità.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → UnitPossession: classe per la rappresentazione di un'unità con associata una quantità.
 - → UserDataManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.42 sgad::clienttier::controller::actions::ShowInteractionMenu

- **Descrizione:** classe per la gestione della visualizzazione di un menu per entrare in modalità costruzione o per la visualizzazione della lista degli utenti.
- **Utilizzo:** viene utilizzata per visualizzare un menu che l'utente può utilizzare per entrare in modalità costruzione o per visualizzare la lista degli utenti.

- **Classi ereditate**

- Action

- **Relazioni con altre classi**

- ← Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → InteractionMenuFactory: classe per la creazione di un menu per interagire con l'intero mondo di gioco.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.43 sgad::clienttier::controller::actions::ShowOperationFailureMenu

- **Descrizione:** classe per la visualizzazione di un menu al fallimento di un'operazione.
- **Utilizzo:** viene utilizzata dai vari gestori delle richieste per notificare che la connessione con il server ha comportato il fallimento dell'operazione.

- **Classi ereditate**

- Action

- **Relazioni con altre classi**

- ← BuildConstruction: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← DemolishBuilding: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← DismissUnits: classe per la gestione del congedo di unità arruolate in precedenza dall'utente.
 - ← HarvestResources: classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
 - ← ReceiveStealResources: classe per la ricezione delle risorse rubate
 - ← RequestAttackUser: classe per la gestione della richiesta di attacco di un utente desiderato.
 - ← RequestBuildConstruction: classe per la richiesta di costruzione di un edificio.

- ← **RequestChangePassword**: classe per la gestione della richiesta di cambiamento password.
- ← **RequestDeleteAccount**: classe per la gestione della richiesta di eliminazione dell'account.
- ← **RequestDemolishBuilding**: classe per la richiesta di demolizione di un edificio.
- ← **RequestDismissUnits**: classe per la richiesta di dimissione di unità possedute.
- ← **RequestHarvestResources**: classe per la richiesta di raccolta di risorse.
- ← **RequestLoadGeneralData**: classe per la gestione della richiesta per il caricamento dei dati globali.
- ← **RequestLoadPersonalData**: classe per la gestione della richiesta per il caricamento dei dati dell'utente.
- ← **RequestStoleResources**: classe per la richiesta di furto di risorse
- ← **RequestTrainUnits**: classe per la gestione della richiesta per il caricamento dei dati globali.
- ← **RequestUpgradeBuilding**: classe per la gestione della richiesta di miglioramento di un edificio.
- ← **RequestUserList**: classe per la richiesta della lista di utenti presenti
- ← **ShowAccountDeletedMenu**: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.
- ← **ShowPasswordChangedMenu**: classe per la visualizzazione di un menu di conferma del cambiamento della password.
- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- ← **TrainUnit**: classe per la gestione dell'addestramento di unità in un edificio.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- → **ReloadPageAction**: azione per il ricaricamento della pagina.
- → **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.44 sgad::clienttier::controller::actions::ShowPasswordChangedMenu

- **Descrizione:** classe per la visualizzazione di un menu di conferma del cambiamento della password.
- **Utilizzo:** viene utilizzata per visualizzare l'esito del cambiamento della password.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← RequestChangePassword: classe per la gestione della richiesta di cambiamento password.
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → NotifyMenuFactory: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.45 sgad::clienttier::controller::actions::ShowResourceMenu

- **Descrizione:** classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
- **Utilizzo:** viene utilizzata per visualizzare il menu per le risorse da selezionare per essere donate all'utente.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← RequestLoadPersonalData: classe per la gestione della richiesta per il caricamento dei dati dell'utente.
 - → ResourceMenuFactory: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - → OwnedResource: classe che rappresenta una risorsa posseduta da un utente.

- → **UserData**: classe per la gestione dei dati di un utente.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.46 sgad::clienttier::controller::actions::ShowTileContextualMenu

- **Descrizione**: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
- **Utilizzo**: viene utilizzata per visualizzare il menu riepilogativo delle risorse e delle unità possedute dall'utente.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← **DemolishBuilding**: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← **LoadPersonalData**: classe per la gestione del caricamento dei dati personali dell'utente.
 - → **CloseContextualMenu**: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → **TileContextualMenuFactory**: classe per la creazione di un menu che visualizza gli edifici costruibili in una casella.
 - → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → **TileComponent**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
 - → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
 - → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.47 sgad::clienttier::controller::actions::ShowTrainUnitMenu

- **Descrizione:** classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- **Utilizzo:** viene utilizzata per la visualizzazione di un menu. Esso permette all'utente di inserire i dati necessari per eseguire poi l'azione di addestramento delle unità desiderate.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← BuildingContextualMenuFactory: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - → EnableRightClick: classe per permettere di abilitare la gestione dell'evento click destro del mouse.
 - → TrainUnit: classe per la gestione dell'addestramento di unità in un edificio.
 - → ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
 - → NotifyMenuFactory: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - → Cost: classe per la gestione di un costo in termini di risorse e tempo.
 - → QuantityResource: classe per la gestione della quantità di una particolare risorsa.
 - → Resource: classe che rappresenta una risorsa.
 - → Unit: classe per la gestione delle informazioni di una unità.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → OwnedResource: classe che rappresenta una risorsa posseduta da un utente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → UserManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.48 sgad::clienttier::controller::actions::ShowUnitSelectionMenu

- **Descrizione:** classe per la visualizzazione di un menu per le unità addestrabili.
- **Utilizzo:** viene utilizzata per la visualizzazione di un menu per la selezione delle unità da mandare in battaglia.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - → UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.49 sgad::clienttier::controller::actions::ShowUpgradeMenu

- **Descrizione:** classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
- **Utilizzo:** viene utilizzata per la visualizzazione di un menu. Esso permette all'utente di confermare, o annullare, la richiesta di miglioramento dell'edificio desiderato.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← BuildingContextualMenuFactory: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - → CloseContextualMenu: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → RequestUpgradeBuilding: classe per la gestione della richiesta di miglioramento di un edificio.
 - → ConfirmBuilderFactory: classe per la creazione di un menu che permetta di confermare una generica azione.

- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.50 sgad::clienttier::controller::actions::ShowUserListMenu

- **Descrizione**: classe per la gestione della visualizzazione di un menu che permette all'utente di visualizzare la lista degli utenti con i quali può interagire.
- **Utilizzo**: viene utilizzata per la visualizzazione di un menu. Esso permette all'utente di selezionare un utente con il quale interagire.
- **Classi ereditate**
 - **Action**
- **Relazioni con altre classi**
 - ← **RequestUserList**: classe per la richiesta della lista di utenti presenti
 - ← **InteractionMenuFactory**: classe per la creazione di un menu per interagire con l'intero mondo di gioco.
 - → **UserListMenuFactory**: classe per la visualizzazione della lista di altri giocatori connessi.
 - → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.51 sgad::clienttier::controller::actions::StealResources

- **Descrizione**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- **Utilizzo**: viene usata per eseguire l'effettivo saccheggio di un edificio di produzione desiderato dall'utente vincitore dell'attacco.

- **Classi ereditate**
 - Action
- **Relazioni con altre classi**
 - ← LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← RequestStoleResources: classe per la richiesta di furto di risorse
 - → ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → BackupManager: classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
 - → NotifyMenuFactory: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - → BuildingWithLevel: classe per la gestione di un edificio ad un particolare livello.
 - → ProductedResource: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - → Resource: classe che rappresenta una risorsa.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → OwnedResource: classe che rappresenta una risorsa posseduta da un utente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → Widget: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.52 sgad::clienttier::controller::actions::TrainUnit

- **Descrizione:** classe per la gestione dell'addestramento di unità in un edificio.
- **Utilizzo:** viene utilizzata per eseguire l'effettivo addestramento delle unità desiderate dall'utente.
- **Classi ereditate**
 - Action
- **Relazioni con altre classi**

- ← **RequestTrainUnits**: classe per la gestione della richiesta per il caricamento dei dati globali.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- → **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
- → **Cost**: classe per la gestione di un costo in termini di risorse e tempo.
- → **QuantityResource**: classe per la gestione della quantità di una particolare risorsa.
- → **Resource**: classe che rappresenta una risorsa.
- → **Unit**: classe per la gestione delle informazioni di una unità.
- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
- → **UnitInProgress**: classe per la gestione della coda di costruzione di unità di un particolare edificio.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **UserDataManager**: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.

4.4.2.53 sgad::clienttier::controller::actions::UpgradeBuilding

- **Descrizione**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- **Utilizzo**: viene utilizzata per eseguire l'effettivo miglioramento di un edificio desiderato dall'utente.
- **Classi ereditate**
 - **Action**
- **Relazioni con altre classi**
 - ← **RequestUpgradeBuilding**: classe per la gestione della richiesta di miglioramento di un edificio.
 - → **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
 - → **Cost**: classe per la gestione di un costo in termini di risorse e tempo.

- → **QuantityResource**: classe per la gestione della quantità di una particolare risorsa.
- → **Resource**: classe che rappresenta una risorsa.
- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **UserDataManager**: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.4.2.54 sgad::clienttier::controller::actions::RequestUserList

- **Descrizione**: classe per la richiesta della lista di utenti presenti
- **Utilizzo**: gestisce la richiesta della lista di utenti presenti
- **Relazioni con altre classi**
 - → **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - → **ShowUserListMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di visualizzare la lista degli utenti con i quali può interagire.
 - → **AJAXRequester**: classe per la gestione dell'invio delle richieste al server da parte del client.

4.4.2.55 sgad::clienttier::controller::actions::ShowLogMenu

- **Descrizione**: classe per la visualizzazione di un menu di gestione account
- **Utilizzo**: permette la visualizzazione di un menu di gestione account
- **Relazioni con altre classi**
 - → **LogMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.

- → **FrameWidget**: classe per la gestione di un generico frame nell’interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.5 Componente sgad::clienttier::controller::backupmanager

4.5.1 Informazioni sul package

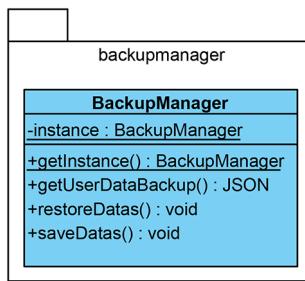


Figura 15: Diagramma della componente sgad::clienttier::controller::-backupmanager

- **Descrizione**: componente che si occupa di gestire il mantenimento dei dati e delle viste dell’utente proprietario dell’account.
- **Padre**: controller

4.5.2 Classi

4.5.2.1 sgad::clienttier::controller::backupmanager::BackupManager

- **Descrizione**: classe che si occupa di gestire il mantenimento dei dati e delle viste dell’utente proprietario dell’account.
- **Utilizzo**: viene utilizzata per memorizzare i dati e le viste dell’utente senza necessariamente richiederle, e quindi ricostruire.
- **Relazioni con altre classi**

- ← **BackToVillage**: classe per permettere di ritornare al proprio villaggio nella mappa.
- ← **LoadAnotherUser**: classe per il caricamento di un altro utente.
- ← **ReceiveStealResources**: classe per la ricezione delle risorse rubate
- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- → **UserData**: classe per la gestione dei dati di un utente.

- → **UserDataManager**: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **GraphicObjectCollection**: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.

4.6 Componente sgad::clienttier::controller::menufactory

4.6.1 Informazioni sul package

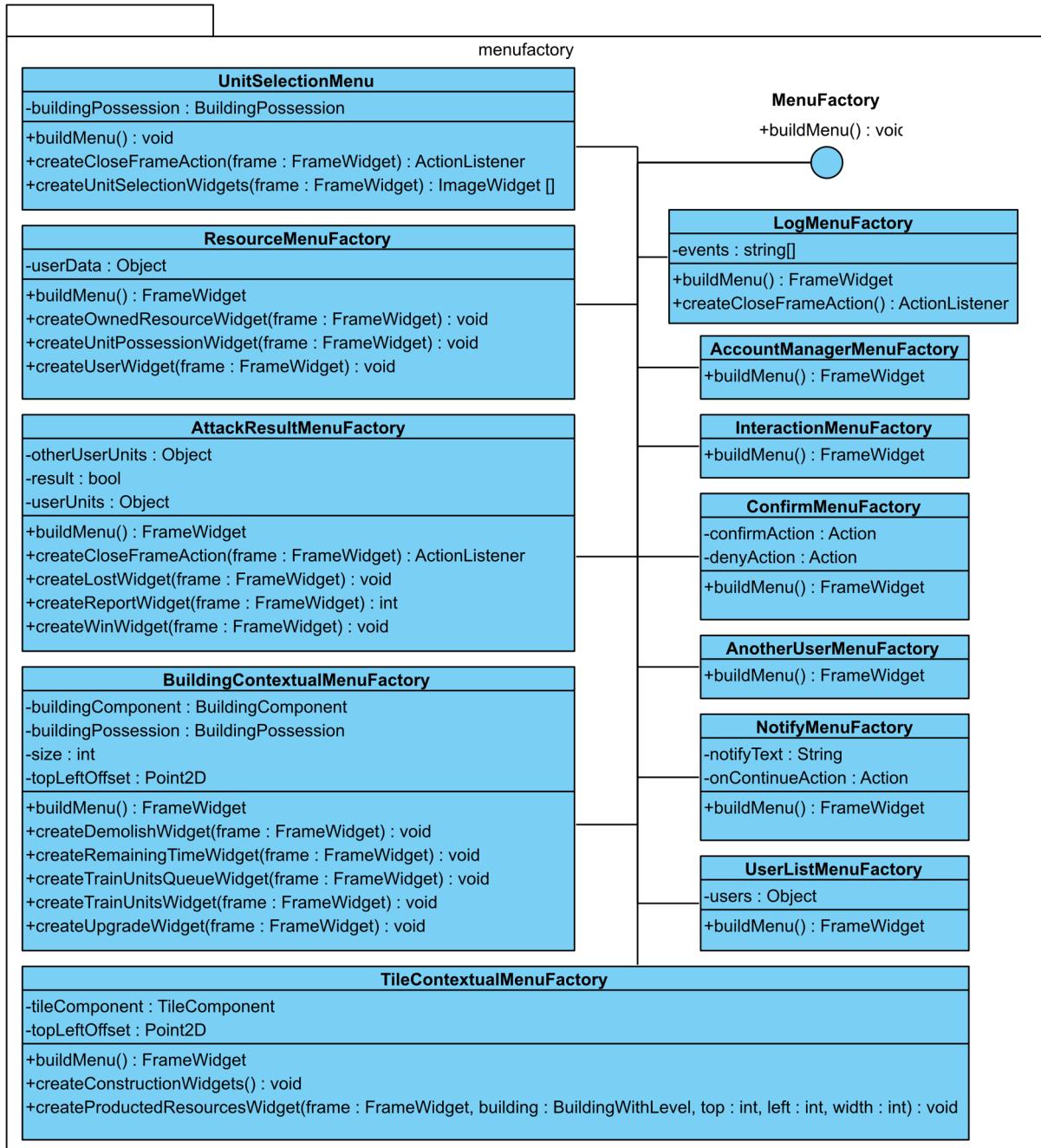


Figura 16: Diagramma della componente `sgad::clienttier::controller::menufactory`

- **Descrizione:** componente per la gestione dei vari menu che possono essere visualizzati durante le interazioni con il mondo di gioco.
- **Padre:** controller
- **Interazioni con altri componenti**
 - components: componente per la gestione degli oggetti del mondo di gioco e della relativa visualizzazione.
 - widget: componente per la gestione dei widget per permettere all'utente di interagire con il mondo di gioco attraverso menu e finestre.

4.6.2 Classi

4.6.2.1 sgad::clienttier::controller::menufactory::MenuFactory

- **Descrizione:** interfaccia per i possibili menu che possono essere visualizzati durante l'interazione dell'utente con il mondo di gioco.
- **Utilizzo:** viene utilizzata per fornire una firma di metodo standard per qualsiasi tipo di menu che è possibile costruire. Le classi che realizzano l'interfaccia si occuperanno di ridefinire il metodo per costruire il menu effettivo. Permette di disaccoppiare l'effettivo menu che viene creato dall'oggetto che richiede la costruzione di un menu.
- **Sottoclassi**
 - AccountManagerMenuFactory
 - AnotherUserMenuFactory
 - AttackResultMenuFactory
 - BuildingContextualMenuFactory
 - ConfirmMenuFactory
 - InteractionMenuFactory
 - LogMenuFactory
 - NotifyMenuFactory
 - ResourceMenuFactory
 - TileContextualMenuFactory
 - UnitSelectionMenu
 - UserListMenuFactory

4.6.2.2 sgad::clienttier::controller::menufactory::AccountManagerMenuFactory

- **Descrizione:** classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.

- **Utilizzo:** viene utilizzata per costruire un menu che permetta all'utente di modificare il proprio account.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← ShowAccountManagerMenu: classe per la creazione di un menu per la gestione dell'account.
 - → ActionListener: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
 - → CloseContextualMenu: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → AuthenticationData: classe per la gestione delle informazioni personali e di accesso di un utente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → UserManager: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
 - → Bound: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
 - → Point2D: classe per la rappresentazione di un punto avente due coordinate.
 - → ButtonWidget: classe per la gestione di un generico bottone nell'interfaccia grafica.
 - → TextWidget: classe per la gestione di un generico testo nell'interfaccia grafica.

4.6.2.3 sgad::clienttier::controller::menufactory::AnotherUserMenuFactory

- **Descrizione:** classe per la costruzione di un menu per il ritorno al villaggio dell'utente.
- **Utilizzo:** viene utilizzata per creare un menu che richieda l'inserimento di un dato numerico da parte dell'utente. Si occupa quindi di istanziare oggetti di tipo Widget ai quali assocerà oggetti di tipo Action indicando a che evento associare tali azioni. Richiede di conoscere eventuali limiti da impostare al numero inserito.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← ShowAnotherUserMenu: classe per la visualizzazione del menu per ritornare al proprio villaggio una volta che si è nel villaggio di un altro utente.

- → **BackToVillage**: classe per permettere di ritornare al proprio villaggio nella mappa.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **ButtonWidget**: classe per la gestione di un generico bottone nell'interfaccia grafica.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.

4.6.2.4 sgad::clienttier::controller::menufactory::AttackResultMenuFactory

- **Descrizione**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
- **Utilizzo**: viene utilizzata per creare un menu che permetta all'utente di visualizzare l'esito dello scontro. Permette inoltre di passare al villaggio dell'utente se l'esito della battaglia è positivo.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← **ShowAttackResultMenu**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
 - → **ActionListener**: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
 - → **EnableRightClick**: classe per permettere di abilitare la gestione dell'evento click destro del mouse.
 - → **LoadAnotherUser**: classe per il caricamento di un altro utente.
 - → **RemoveGraphicObjectAction**: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
 - → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
 - → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
 - → **ButtonWidget**: classe per la gestione di un generico bottone nell'interfaccia grafica.
 - → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → **ImageWidget**: classe per la gestione di una generica immagine nell'interfaccia grafica.

- → **TextWidget**: classe per la gestione di un generico testo nell’interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.6.2.5 sgad::clienttier::controller::menufactory::BuildingContextualMenuFactory

- **Descrizione**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- **Utilizzo**: viene utilizzata per creare dinamicamente un menu adatto secondo le caratteristiche dell’edificio richiesto. Si occupa quindi di istanziare oggetti di tipo **Widget** ai quali assocerà oggetti di tipo **Action** indicando a che evento associare tali azioni.
- **Classi ereditate**
 - **MenuFactory**
- **Relazioni con altre classi**
 - ← **ShowBuildingContextualMenu**: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
 - → **ActionListener**: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l’azione su tale classe.
 - → **CloseContextualMenu**: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → **ShowDemolishMenu**: classe che visualizza il menu per la demolizione di un edificio.
 - → **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all’utente di addestrare le unità.
 - → **ShowUpgradeMenu**: classe per la gestione della visualizzazione di un menu che permette all’utente di confermare il miglioramento di un edificio.
 - → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
 - → **Cost**: classe per la gestione di un costo in termini di risorse e tempo.
 - → **DataFactory**: classe che ritorna un’istanza delle componenti di gioco.
 - → **ProductedResource**: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - → **QuantityResource**: classe per la gestione della quantità di una particolare risorsa.
 - → **Resource**: classe che rappresenta una risorsa.

- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
- → **UnitInProgress**: classe per la gestione della coda di costruzione di unità di un particolare edificio.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **UserDataManager**: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **ButtonWidget**: classe per la gestione di un generico bottone nell'interfaccia grafica.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **ImageWidget**: classe per la gestione di una generica immagine nell'interfaccia grafica.
- → **TextWidget**: classe per la gestione di un generico testo nell'interfaccia grafica.

4.6.2.6 sgad::clienttier::controller::menufactory::ConfirmMenuFactory

- **Descrizione**: classe per la creazione di un menu che permetta di confermare una generica azione.
- **Utilizzo**: viene utilizzata per creare un menu standard che permetta all'utente di confermare o meno un'operazione. Si occupa quindi di istanziare oggetti di tipo **Widget** ai quali assocerà oggetti di tipo **Action** indicando a che evento associare tali azioni. Richiede che vengano specificate quali siano le azioni da associare alla conferma e all'annullamento dell'operazione al momento della sua creazione.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← **ChangePasswordAction**: classe per permettere di modificare la password di un account.

- ← **DeleteAccountAction**: la classe viene utilizzata per l'eliminazione di un account utente.
- ← **ShowAttackMenu**: classe per la visualizzazione del menu per l'attacco.
- ← **ShowBuildConstructionMenu**: classe per la richiesta della conferma della costruzione.
- ← **ShowDemolishMenu**: classe che visualizza il menu per la demolizione di un edificio.
- ← **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← **ShowUpgradeMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
- → **Action**: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.
- → **ActionListener**: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
- → **EnableRightClick**: classe per permettere di abilitare la gestione dell'evento click destro del mouse.
- → **RemoveGraphicObjectAction**: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
- → **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **ButtonWidget**: classe per la gestione di un generico bottone nell'interfaccia grafica.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **ImageWidget**: classe per la gestione di una generica immagine nell'interfaccia grafica.
- → **TextWidget**: classe per la gestione di un generico testo nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.6.2.7 sgad::clienttier::controller::menufactory::InteractionMenuFactory

- **Descrizione:** classe per la creazione di un menu per interagire con l'intero mondo di gioco.
- **Utilizzo:** viene utilizzata per costruire un menu che permetta all'utente di eseguire interazione legate all'intero mondo di gioco.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← ShowInteractionMenu: classe per la gestione della visualizzazione di un menu per entrare in modalità costruzione o per la visualizzazione della lista degli utenti.
 - → SetAllVoidFilter: classe per la gestione di un filtro standard.
 - → ShowUserListMenu: classe per la gestione della visualizzazione di un menu che permette all'utente di visualizzare la lista degli utenti con i quali può interagire.
 - → Bound: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
 - → Point2D: classe per la rappresentazione di un punto avente due coordinate.
 - → ButtonWidget: classe per la gestione di un generico bottone nell'interfaccia grafica.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.

4.6.2.8 sgad::clienttier::controller::menufactory::LogMenuFactory

- **Descrizione:** classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- **Utilizzo:** gestisce la costruzione di un menu che permetta all'utente di modificare il proprio account.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← ShowLogMenu: classe per la visualizzazione di un menu di gestione account
 - → ActionListener: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
 - → EnableRightClick: classe per permettere di abilitare la gestione dell'evento click destro del mouse.

- → RemoveGraphicObjectAction: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
- → Bound: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.
- → ButtonWidget: classe per la gestione di un generico bottone nell'interfaccia grafica.
- → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
- → TextWidget: classe per la gestione di un generico testo nell'interfaccia grafica.

4.6.2.9 sgad::clienttier::controller::menufactory::NotifyMenuFactory

- **Descrizione:** classe per la costruzione di un menu per la visualizzazione di una notifica.
- **Utilizzo:** viene utilizzata dalle varie azioni per notificare un'evento che impedisce il normale proseguimento dell'azione.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← BuildConstruction: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← ChangePasswordAction: classe per permettere di modificare la password di un account.
 - ← ReceiveStealResources: classe per la ricezione delle risorse rubate
 - ← ShowAccountDeletedMenu: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.
 - ← ShowAttackMenu: classe per la visualizzazione del menu per l'attacco.
 - ← ShowAttackResultMenu: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
 - ← ShowDismissUnitMenu: classe per la gestione della visualizzazione di un menu per il congedo di unità.
 - ← ShowOperationFailureMenu: classe per la visualizzazione di un menu al fallimento di un'operazione.
 - ← ShowPasswordChangedMenu: classe per la visualizzazione di un menu di conferma del cambiamento della password.
 - ← ShowTrainUnitMenu: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.

- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- → **Action**: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.
- → **ActionListener**: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
- → **RemoveGraphicObjectAction**: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **ButtonWidget**: classe per la gestione di un generico bottone nell'interfaccia grafica.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **TextWidget**: classe per la gestione di un generico testo nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.6.2.10 sgad::clienttier::controller::menufactory::ResourceMenuFactory

- **Descrizione**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
- **Utilizzo**: viene utilizzata per costruire il menu delle risorse possedute dall'utente e delle unità.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← **ShowResourceMenu**: classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
 - → **Logout**: classe per la gestione del logout dell'utente.
 - → **DataFactory**: classe che ritorna un'istanza delle componenti di gioco.
 - → **Resource**: classe che rappresenta una risorsa.
 - → **Unit**: classe per la gestione delle informazioni di una unità.
 - → **AuthenticationData**: classe per la gestione delle informazioni personali e di accesso di un utente.

- → `OwnedResource`: classe che rappresenta una risorsa posseduta da un utente.
- → `UnitPossession`: classe per la rappresentazione di un'unità con associata una quantità.
- → `UserData`: classe per la gestione dei dati di un utente.
- → `UserDataManager`: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- → `Bound`: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → `Point2D`: classe per la rappresentazione di un punto avente due coordinate.
- → `ButtonWidget`: classe per la gestione di un generico bottone nell'interfaccia grafica.
- → `FrameWidget`: classe per la gestione di un generico frame nell'interfaccia grafica.
- → `ImageWidget`: classe per la gestione di una generica immagine nell'interfaccia grafica.
- → `TextWidget`: classe per la gestione di un generico testo nell'interfaccia grafica.

4.6.2.11 `sgad::clienttier::controller::menufactory::TileContextualMenuFactory`

- **Descrizione:** classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- **Utilizzo:** viene utilizzata per costruire un menu contestuale che permetta all'utente di costruire un edificio nella casella selezionata.
- **Classi ereditate**
 - `MenuFactory`
- **Relazioni con altre classi**
 - ← `ShowTileContextualMenu`: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
 - → `ActionListener`: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
 - → `CloseContextualMenu`: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - → `ShowBuildConstructionMenu`: classe per la richiesta della conferma della costruzione.
 - → `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
 - → `Cost`: classe per la gestione di un costo in termini di risorse e tempo.

- → **DataFactory**: classe che ritorna un’istanza delle componenti di gioco.
- → **ProducedResource**: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
- → **QuantityResource**: classe per la gestione della quantità di una particolare risorsa.
- → **Resource**: classe che rappresenta una risorsa.
- → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **UserDataManager**: classe che si occupa di mantenere l’accesso globale ai dati dell’utente di cui si sta attualmente visualizzando il villaggio.
- → **Bound**: classe per la gestione dei limiti della porzione dell’area di gioco effettivamente visualizzata.
- → **TileComponent**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- → **WorldComponentShapeFactory**: classe per la gestione del controllo ai vari oggetti di tipo **WorldComponentShapeImg**.
- → **WorldComponentShapeImg**: classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all’interno del mondo di gioco.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **ButtonWidget**: classe per la gestione di un generico bottone nell’interfaccia grafica.
- → **FrameWidget**: classe per la gestione di un generico frame nell’interfaccia grafica.
- → **ImageWidget**: classe per la gestione di una generica immagine nell’interfaccia grafica.
- → **TextWidget**: classe per la gestione di un generico testo nell’interfaccia grafica.

4.6.2.12 sgad::clienttier::controller::menufactory::UnitSelectionMenu

- **Descrizione**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- **Utilizzo**: viene utilizzata per costruire un menu che permetta all’utente di selezionare quali unità desidera addestrare o congedare.
- **Classi ereditate**
 - MenuFactory
- **Relazioni con altre classi**
 - ← **ShowUnitSelectionMenu**: classe per la visualizzazione di un menu per le unità addestrabili.

- → **ActionListener**: classe per permettere di memorizzare una coda di azioni da eseguire in successione. Queste verranno invocate quando viene eseguito il metodo per eseguire l'azione su tale classe.
- → **EnableRightClick**: classe per permettere di abilitare la gestione dell'evento click destro del mouse.
- → **RemoveGraphicObjectAction**: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
- → **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- → **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- → **QuantityResource**: classe per la gestione della quantità di una particolare risorsa.
- → **Resource**: classe che rappresenta una risorsa.
- → **Unit**: classe per la gestione delle informazioni di una unità.
- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
- → **UnitInProgress**: classe per la gestione della coda di costruzione di unità di un particolare edificio.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **UserDataManager**: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **ButtonWidget**: classe per la gestione di un generico bottone nell'interfaccia grafica.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **ImageWidget**: classe per la gestione di una generica immagine nell'interfaccia grafica.
- → **TextWidget**: classe per la gestione di un generico testo nell'interfaccia grafica.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.6.2.13 sgad::clienttier::controller::menufactory::UserListMenuFactory

- **Descrizione**: classe per la visualizzazione della lista di altri giocatori connessi.
- **Utilizzo**: viene utilizzata per costruire un menu che visualizzi i vari utenti con i quali l'utente può interagire.

- Classi ereditate
 - MenuFactory
- Relazioni con altre classi
 - ← ShowUserListMenu: classe per la gestione della visualizzazione di un menu che permette all'utente di visualizzare la lista degli utenti con i quali può interagire.
 - → ShowAttackMenu: classe per la visualizzazione del menu per l'attacco.
 - → Bound: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
 - → Point2D: classe per la rappresentazione di un punto avente due coordinate.
 - → ButtonWidget: classe per la gestione di un generico bottone nell'interfaccia grafica.
 - → FrameWidget: classe per la gestione di un generico frame nell'interfaccia grafica.
 - → ImageWidget: classe per la gestione di una generica immagine nell'interfaccia grafica.

4.7 Componente sgad::clienttier::controller::messageinterpreter

4.7.1 Informazioni sul package

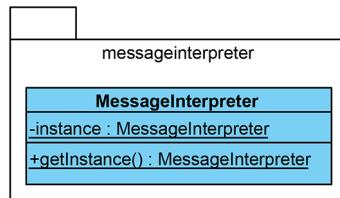


Figura 17: Diagramma della componente sgad::clienttier::controller::messageinterpreter

- **Descrizione:** componente per l'interpretazione delle reply ricevute dal server.
- **Padre:** controller
- **Interazioni con altri componenti**
 - **actions:** componente per la gestione delle possibili azioni che un utente può intraprendere durante l'interazione con il mondo di gioco.
 - **requester:** componente per la gestione dell'invio di richieste al server.

4.7.2 Classi

4.7.2.1 sgad::clienttier::controller::messageinterpreter::MessageInterpreter

- **Descrizione:** classe per l'interpretazione di una risposta che è stata ricevuta dal server in seguito ad una determinata richiesta.
- **Utilizzo:** viene utilizzata per analizzare un messaggio ricevuto dal server dopo che è stata inviata una richiesta. La classe analizza se il messaggio ricevuto ha al suo interno un ulteriore messaggio in piggybacking. Ciò può verificarsi quando sono state apportate modifiche ai dati che risiedono sul server non in merito ad operazioni eseguite dall'utente stesso (esempio: interazione con un altro utente). La classe invocherà determinate azioni per riportare le modifiche anche sul client.
- **Relazioni con altre classi**
 - ← **AJAXRequester:** classe per la gestione dell'invio delle richieste al server da parte del client.

4.8 Componente sgad::clienttier::controller::requester

4.8.1 Informazioni sul package

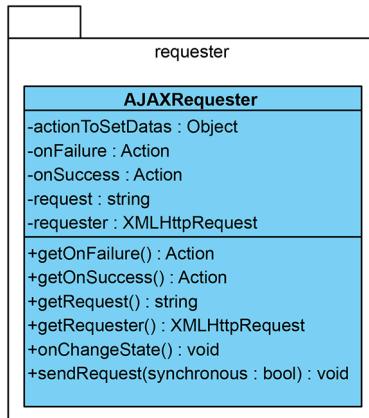


Figura 18: Diagramma della componente `sgad::clienttier::controller::requester`

- **Descrizione:** componente per la gestione dell'invio di richieste al server.
- **Padre:** controller
- **Interazioni con altri componenti**
 - **actions:** componente per la gestione delle possibili azioni che un utente può intraprendere durante l'interazione con il mondo di gioco.
 - **messageinterpreter:** componente per l'interpretazione delle reply ricevute dal server.
 - **httpresponder:** componente per la gestione delle richieste HTTP ricevute dai vari client.

4.8.2 Classi

4.8.2.1 sgad::clienttier::controller::requester::AJAXRequester

- **Descrizione:** classe per la gestione dell'invio delle richieste al server da parte del client.
- **Utilizzo:** viene utilizzata per inviare una richiesta al server. Tale classe si occuperà di eseguire determinate azioni in base a determinati errori nella comunicazione. Essa implementerà le richieste tramite la tecnologia AJAX. Necessita di conoscere le azioni da eseguire nel caso in cui la richiesta venga soddisfatta correttamente o meno.
- **Relazioni con altre classi**

- ← **Logout**: classe per la gestione del logout dell'utente.
- ← **RequestBuildConstruction**: classe per la richiesta di costruzione di un edificio.
- ← **RequestChangePassword**: classe per la gestione della richiesta di cambiamento password.
- ← **RequestDeleteAccount**: classe per la gestione della richiesta di eliminazione dell'account.
- ← **RequestDemolishBuilding**: classe per la richiesta di demolizione di un edificio.
- ← **RequestDismissUnits**: classe per la richiesta di dimissione di unità possedute.
- ← **RequestHarvestResources**: classe per la richiesta di raccolta di risorse.
- ← **RequestLoadGeneralData**: classe per la gestione della richiesta per il caricamento dei dati globali.
- ← **RequestLoadPersonalData**: classe per la gestione della richiesta per il caricamento dei dati dell'utente.
- ← **RequestStoleResources**: classe per la richiesta di furto di risorse
- ← **RequestTrainUnits**: classe per la gestione della richiesta per il caricamento dei dati globali.
- ← **RequestUpgradeBuilding**: classe per la gestione della richiesta di miglioramento di un edificio.
- ← **RequestUserList**: classe per la richiesta della lista di utenti presenti
- → **Action**: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.
- → **MessageInterpreter**: classe per l'interpretazione di una risposta che è stata ricevuta dal server in seguito ad una determinata richiesta.

4.9 Componente sgad::clienttier::model

4.9.1 Informazioni sul package

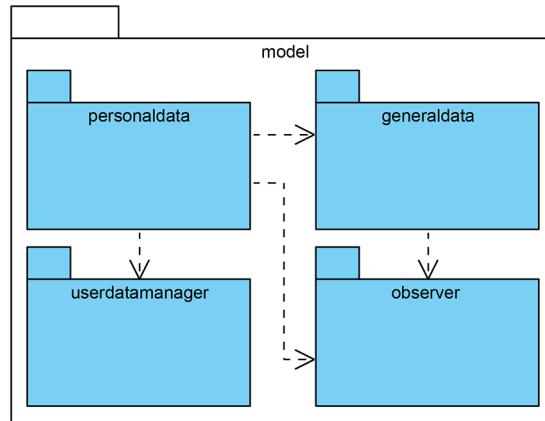


Figura 19: Diagramma della componente `sgad::clienttier::model`

- **Descrizione:** componente Model dell'architettura MVC. Essa memorizza tutti i dati di gioco dell'utente su cui la view si basa per disegnarsi.
- **Padre:** `clienttier`
- **Interazioni con altri componenti**
 - **controller:** componente Controller dell'architettura MVC. Essa esegue le operazioni che l'utente ha richiesto tramite la view agendo se necessario sul model data.
 - **view:** componente View dell'architettura MVC. Essa gestisce la visualizzazione del gioco e l'input utente all'interno del canvas HTML5.
- **Package contenuti**
 - **generaldata:** componente del Model per la raccolta dei dati non specifici dell'utente ma relativi all'intero gioco.
 - **observer:** componente per raccogliere le classi che realizzeranno il pattern Observer tipico dell'architettura MVC.
 - **personaldata:** componente del model che gestisce le classi per la memorizzazione dei dati legati all'utente.
 - **userdatamanager:** componente per il mantenimento di un accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.

4.10 Componente sgad::clienttier::model::generaldata

4.10.1 Informazioni sul package

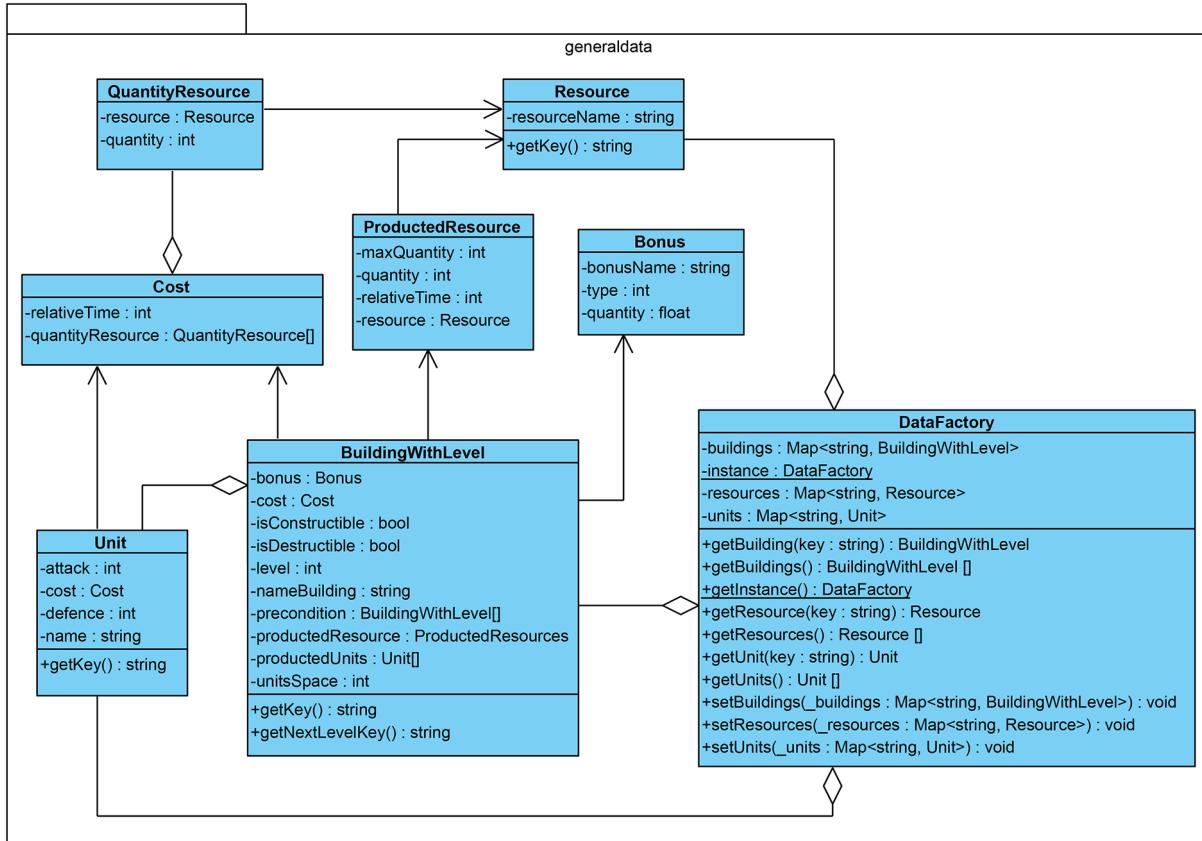


Figura 20: Diagramma della componente `sgad::clienttier::model::generaldata`

- **Descrizione:** componente del Model per la raccolta dei dati non specifici dell'utente ma relativi all'intero gioco.
- **Padre:** `model`
- **Interazioni con altri componenti**
 - `observer`: componente per raccogliere le classi che realizzeranno il pattern Observer tipico dell'architettura MVC.
 - `personaldata`: componente del model che gestisce le classi per la memorizzazione dei dati legati all'utente.

4.10.2 Classi

4.10.2.1 `sgad::clienttier::model::generaldata::Bonus`

- **Descrizione:** classe per la gestione dei bonus eventualmente disponibili per certi edifici.
- **Utilizzo:** viene utilizzata per mantenere l'informazione sul nome del bonus, sul tipo e sulla quantità dello stesso.
- **Relazioni con altre classi**
 - ← LoadGeneralData: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - ← BuildingWithLevel: classe per la gestione di un edificio ad un particolare livello.
 - ← BuildingPossession: classe per la gestione di un edificio posseduto da un utente.

4.10.2.2 sgad::clienttier::model::generaldata::BuildingWithLevel

- **Descrizione:** classe per la gestione di un edificio ad un particolare livello.
- **Utilizzo:** viene utilizzata per memorizzare tutte le informazioni che descrivono un edificio ad un particolare livello.
- **Relazioni con altre classi**
 - ← BuildConstruction: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← HarvestResources: classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
 - ← LoadGeneralData: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - ← ReceiveStealResources: classe per la ricezione delle risorse rubate
 - ← RequestHarvestResources: classe per la richiesta di raccolta di risorse.
 - ← StealResources: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
 - ← UpgradeBuilding: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
 - ← BuildingContextualMenuFactory: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← TileContextualMenuFactory: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
 - ← UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - ← DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - ← BuildingPossession: classe per la gestione di un edificio posseduto da un utente.

- ← **UserData**: classe per la gestione dei dati di un utente.
- ← **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → **Bonus**: classe per la gestione dei bonus eventualmente disponibili per certi edifici.
- → **Cost**: classe per la gestione di un costo in termini di risorse e tempo.
- → **ProductedResource**: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
- → **Unit**: classe per la gestione delle informazioni di una unità.

4.10.2.3 sgad::clienttier::model::generaldata::Cost

- **Descrizione**: classe per la gestione di un costo in termini di risorse e tempo.
- **Utilizzo**: viene utilizzata per la gestione di un costo in termini di quantità di risorse e tempo.
- **Relazioni con altre classi**

- ← **BuildConstruction**: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
- ← **DemolishBuilding**: classe per la gestione della demolizione di un edificio già presente nel villaggio.
- ← **LoadGeneralData**: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← **TrainUnit**: classe per la gestione dell'addestramento di unità in un edificio.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizza gli edifici costruibili in una casella.
- ← **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- ← **Unit**: classe per la gestione delle informazioni di una unità.
- ← **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **QuantityResource**: classe per la gestione della quantità di una particolare risorsa.

4.10.2.4 sgad::clienttier::model::generaldata::DataFactory

- **Descrizione:** classe che ritorna un'istanza delle componenti di gioco.
- **Utilizzo:** viene utilizzata per ottenere riferimenti agli oggetti che compongono la sua stessa componente.
- **Relazioni con altre classi**
 - ← **BuildConstruction:** classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← **LoadGeneralData:** classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - ← **LoadPersonalData:** classe per la gestione del caricamento dei dati personali dell'utente.
 - ← **ShowAttackMenu:** classe per la visualizzazione del menu per l'attacco.
 - ← **BuildingContextualMenuFactory:** classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← **ResourceMenuFactory:** classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← **TileContextualMenuFactory:** classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
 - → **BuildingWithLevel:** classe per la gestione di un edificio ad un particolare livello.
 - → **Resource:** classe che rappresenta una risorsa.
 - → **Unit:** classe per la gestione delle informazioni di una unità.

4.10.2.5 sgad::clienttier::model::generaldata::ProducedResource

- **Descrizione:** classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
- **Utilizzo:** viene utilizzata per la memorizzazione delle informazioni relative alla risorsa prodotta da un edificio.
- **Relazioni con altre classi**
 - ← **HarvestResources:** classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
 - ← **LoadGeneralData:** classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - ← **LoadPersonalData:** classe per la gestione del caricamento dei dati personali dell'utente.
 - ← **ReceiveStealResources:** classe per la ricezione delle risorse rubate

- ← **RequestHarvestResources**: classe per la richiesta di raccolta di risorse.
- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizza gli edifici costruibili in una casella.
- ← **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- ← **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **Resource**: classe che rappresenta una risorsa.

4.10.2.6 sgad::clienttier::model::generaldata::QuantityResource

- **Descrizione**: classe per la gestione della quantità di una particolare risorsa.
- **Utilizzo**: viene utilizzata per associare una quantità ad una risorsa.
- **Relazioni con altre classi**
 - ← **BuildConstruction**: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← **DemolishBuilding**: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← **LoadGeneralData**: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
 - ← **TrainUnit**: classe per la gestione dell'addestramento di unità in un edificio.
 - ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
 - ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizza gli edifici costruibili in una casella.
 - ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - ← **Cost**: classe per la gestione di un costo in termini di risorse e tempo.
 - → **Resource**: classe che rappresenta una risorsa.

4.10.2.7 sgad::clienttier::model::generaldata::Resource

- **Descrizione:** classe che rappresenta una risorsa.
- **Utilizzo:** viene utilizzata per memorizzare le informazioni su una risorsa.
- **Relazioni con altre classi**
 - ← **BuildConstruction:** classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← **HarvestResources:** classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
 - ← **LoadGeneralData:** classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
 - ← **ReceiveStealResources:** classe per la ricezione delle risorse rubate
 - ← **ShowTrainUnitMenu:** classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
 - ← **StealResources:** classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
 - ← **TrainUnit:** classe per la gestione dell'addestramento di unità in un edificio.
 - ← **UpgradeBuilding:** classe per la gestione del miglioramento di un edificio desiderato dall'utente.
 - ← **BuildingContextualMenuFactory:** classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← **ResourceMenuFactory:** classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← **TileContextualMenuFactory:** classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
 - ← **UnitSelectionMenu:** classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - ← **DataFactory:** classe che ritorna un'istanza delle componenti di gioco.
 - ← **ProducedResource:** classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - ← **QuantityResource:** classe per la gestione della quantità di una particolare risorsa.
 - ← **OwnedResource:** classe che rappresenta una risorsa posseduta da un utente.

4.10.2.8 sgad::clienttier::model::generaldata::Unit

- **Descrizione:** classe per la gestione delle informazioni di una unità.
- **Utilizzo:** viene utilizzata per la memorizzazione delle informazioni sulle unità.

- Relazioni con altre classi

- ← LoadGeneralData: classe per la gestione del caricamento dei dati che non dipendono dai particolari dati dell'utente.
- ← RequestDismissUnits: classe per la richiesta di dimissione di unità possedute.
- ← RequestTrainUnits: classe per la gestione della richiesta per il caricamento dei dati globali.
- ← ShowAttackMenu: classe per la visualizzazione del menu per l'attacco.
- ← ShowDismissUnitMenu: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- ← ShowTrainUnitMenu: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← TrainUnit: classe per la gestione dell'addestramento di unità in un edificio.
- ← ResourceMenuFactory: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
- ← UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- ← BuildingWithLevel: classe per la gestione di un edificio ad un particolare livello.
- ← DataFactory: classe che ritorna un'istanza delle componenti di gioco.
- ← UnitInProgress: classe per la gestione della coda di costruzione di unità di un particolare edificio.
- ← UnitPossession: classe per la rappresentazione di un'unità con associata una quantità.
- → Cost: classe per la gestione di un costo in termini di risorse e tempo.

4.11 Componente sgad::clienttier::model::observer

4.11.1 Informazioni sul package

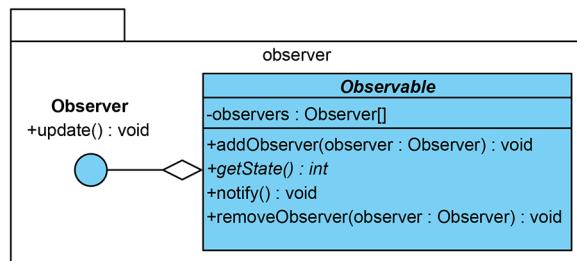


Figura 21: Diagramma della componente sgad::clienttier::model::observer

- **Descrizione:** componente per raccogliere le classi che realizzeranno il pattern Observer tipico dell'architettura MVC.

- **Padre:** `model`
- **Interazioni con altri componenti**
 - `generaldata`: componente del Model per la raccolta dei dati non specifici dell'utente ma relativi all'intero gioco.
 - `personaldata`: componente del model che gestisce le classi per la memorizzazione dei dati legati all'utente.
 - `context`: componente che raccoglie la classe che gestisce l'oggetto che rappresenterà l'ambiente generale di gioco.
 - `graphicobjects`: componente per la gestione degli oggetti grafici che vengono rappresentati nel mondo di gioco quali widget e immagini rappresentanti il mondo di gioco (esempio: edificio).

4.11.2 Classi

4.11.2.1 `sgad::clienttier::model::observer::Observable`

- **Descrizione:** classe astratta per la gestione di oggetti osservabili. Essi, al cambiamento di stato, notificheranno tutti gli oggetti osservatori che richiedono di conoscere tali cambiamenti.
- **Utilizzo:** viene utilizzata da alcune delle classi della componente `model` per notificare un cambiamento dello stato del model a cui gli oggetti della componente `view` sono interessati per potersi rappresentare correttamente.
- **Sottoclassi**
 - `BuildingPossession`
 - `OwnedResource`
 - `UnitInProgress`
 - `UnitPossession`
- **Relazioni con altre classi**
 - ← `TextWidget`: classe per la gestione di un generico testo nell'interfaccia grafica.
 - → `Observer`: interfaccia per la gestione di oggetti che osservano altri oggetti.

4.11.2.2 `sgad::clienttier::model::observer::Observer`

- **Descrizione:** interfaccia per la gestione di oggetti che osservano altri oggetti.
- **Utilizzo:** viene utilizzata ereditando questa classe e definendo il metodo esposto quando si vuole che questo oggetto sia aggiornato alla modifica di un oggetto di tipo `Observable`.
- **Sottoclassi**

- BuildingComponent
 - TextWidget
- Relazioni con altre classi
 - ← Observable: classe astratta per la gestione di oggetti osservabili. Essi, al cambiamento di stato, notificheranno tutti gli oggetti osservatori che richiedono di conoscere tali cambiamenti.

4.12 Componente sgad::clienttier::model::personaldata

4.12.1 Informazioni sul package

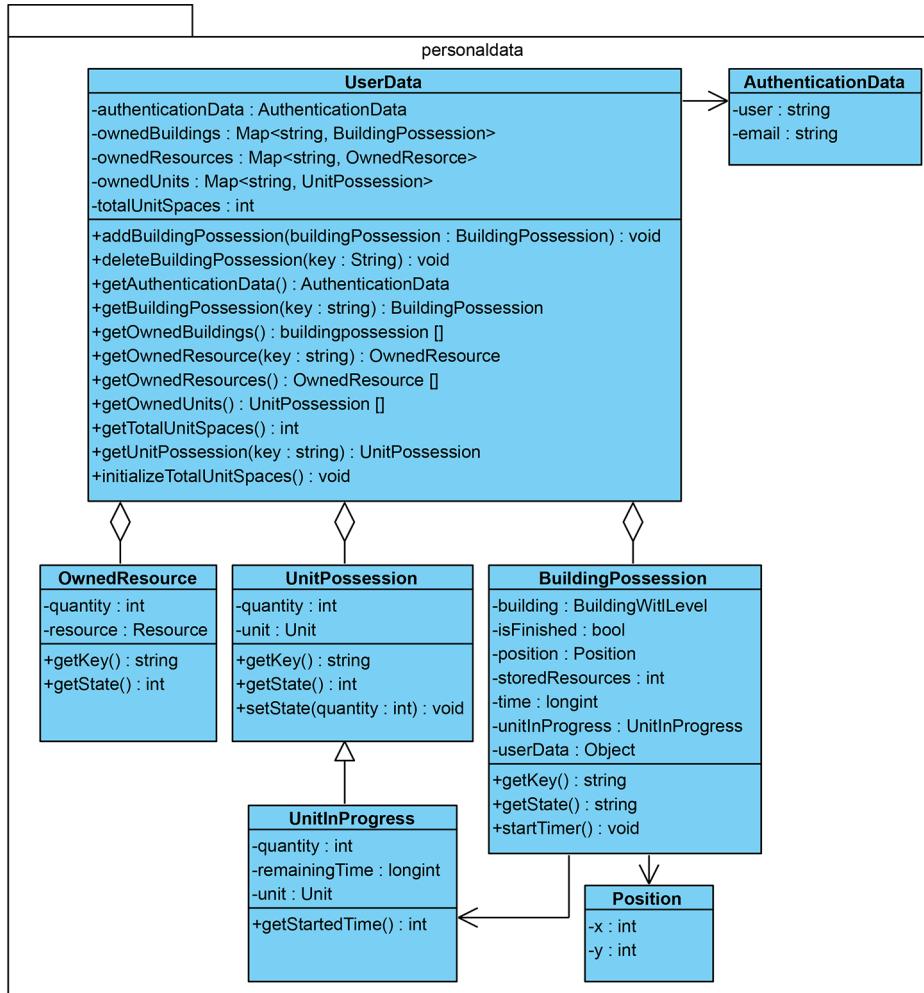


Figura 22: Diagramma della componente `sgad::clienttier::model::personaldata`

- **Descrizione:** componente del model che gestisce le classi per la memorizzazione dei dati legati all'utente.
- **Padre:** `model`
- **Interazioni con altri componenti**
 - `generaldata`: componente del Model per la raccolta dei dati non specifici dell'utente ma relativi all'intero gioco.
 - `observer`: componente per raccogliere le classi che realizzeranno il pattern Observer tipico dell'architettura MVC.

4.12.2 Classi

4.12.2.1 sgad::clienttier::model::personaldata::AuthenticationData

- **Descrizione:** classe per la gestione delle informazioni personali e di accesso di un utente.
- **Utilizzo:** viene utilizzata istanziando un oggetto che memorizza le informazioni.
- **Relazioni con altre classi**
 - ← LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← RequestStoleResources: classe per la richiesta di furto di risorse
 - ← AccountManagerMenuFactory: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← ResourceMenuFactory: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← UserData: classe per la gestione dei dati di un utente.

4.12.2.2 sgad::clienttier::model::personaldata::BuildingPossession

- **Descrizione:** classe per la gestione di un edificio posseduto da un utente.
- **Utilizzo:** viene utilizzata per la memorizzazione delle informazioni riguardanti un edificio posseduto da un utente.
- **Classi ereditate**
 - Observable
- **Relazioni con altre classi**
 - ← BuildConstruction: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← DemolishBuilding: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← HarvestResources: classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
 - ← LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← ReceiveStealResources: classe per la ricezione delle risorse rubate
 - ← RequestDemolishBuilding: classe per la richiesta di demolizione di un edificio.
 - ← RequestHarvestResources: classe per la richiesta di raccolta di risorse.

- ← **RequestTrainUnits**: classe per la gestione della richiesta per il caricamento dei dati globali.
- ← **RequestUpgradeBuilding**: classe per la gestione della richiesta di miglioramento di un edificio.
- ← **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← **ShowUnitSelectionMenu**: classe per la visualizzazione di un menu per le unità addestrabili.
- ← **ShowUpgradeMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- ← **TrainUnit**: classe per la gestione dell'addestramento di unità in un edificio.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- ← **UserData**: classe per la gestione dei dati di un utente.
- ← **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → **Bonus**: classe per la gestione dei bonus eventualmente disponibili per certi edifici.
- → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- → **Cost**: classe per la gestione di un costo in termini di risorse e tempo.
- → **ProducedResource**: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
- → **Position**: classe per la gestione di una posizione sulla griglia del villaggio.
- → **UnitInProgress**: classe per la gestione della coda di costruzione di unità di un particolare edificio.

4.12.2.3 sgad::clienttier::model::personaldata::OwnedResource

- **Descrizione**: classe che rappresenta una risorsa posseduta da un utente.
- **Utilizzo**: viene utilizzata per la memorizzazione di una risorsa posseduta da un utente.

- **Classi ereditate**

- `Observable`

- **Relazioni con altre classi**

- ← `BuildConstruction`: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← `DemolishBuilding`: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← `HarvestResources`: classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
 - ← `LoadPersonalData`: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← `ReceiveStealResources`: classe per la ricezione delle risorse rubate
 - ← `ShowResourceMenu`: classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
 - ← `ShowTrainUnitMenu`: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
 - ← `StealResources`: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
 - ← `TrainUnit`: classe per la gestione dell'addestramento di unità in un edificio.
 - ← `UpgradeBuilding`: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
 - ← `BuildingContextualMenuFactory`: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← `ResourceMenuFactory`: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← `TileContextualMenuFactory`: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
 - ← `UnitSelectionMenu`: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - ← `UserData`: classe per la gestione dei dati di un utente.
 - → `Resource`: classe che rappresenta una risorsa.

4.12.2.4 `sgad::clienttier::model::personaldata::Position`

- **Descrizione:** classe per la gestione di una posizione sulla griglia del villaggio.
- **Utilizzo:** viene utilizzata per memorizzare la posizione di un edificio posseduto da un utente.
- **Relazioni con altre classi**

- ← **BuildConstruction**: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
- ← **LoadPersonalData**: classe per la gestione del caricamento dei dati personali dell'utente.
- ← **RequestBuildConstruction**: classe per la richiesta di costruzione di un edificio.
- ← **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- ← **BuildModeFilter**: classe per la gestione del filtro quando viene selezionata la modalità costruzione.
- ← **VoidFilter**: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
- ← **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- ← **TileComponent**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.

4.12.2.5 sgad::clienttier::model::personaldata::UnitPossession

- **Descrizione**: classe per la rappresentazione di un'unità con associata una quantità.
- **Utilizzo**: viene utilizzata per memorizzare per ogni tipo di unità la quantità di queste possedute da un utente.
- **Classi ereditate**
 - **Observable**
- **Sottoclassi**
 - **UnitInProgress**
- **Relazioni con altre classi**
 - ← **DismissUnits**: classe per la gestione del congedo di unità arruolate in precedenza dall'utente.
 - ← **LoadPersonalData**: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← **ShowAttackMenu**: classe per la visualizzazione del menu per l'attacco.
 - ← **ShowAttackResultMenu**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
 - ← **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
 - ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← **UserData**: classe per la gestione dei dati di un utente.
 - → **Unit**: classe per la gestione delle informazioni di una unità.

4.12.2.6 sgad::clienttier::model::personaldata::UnitInProgress

- **Descrizione:** classe per la gestione della coda di costruzione di unità di un particolare edificio.
- **Utilizzo:** viene utilizzata per gestire le code di costruzione degli edifici completi posseduti da un particolare utente.
- **Classi ereditate**
 - Observable
 - UnitPossession
- **Relazioni con altre classi**
 - ← LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← TrainUnit: classe per la gestione dell'addestramento di unità in un edificio.
 - ← BuildingContextualMenuFactory: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← UnitSelectionMenu: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - ← BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → Unit: classe per la gestione delle informazioni di una unità.

4.12.2.7 sgad::clienttier::model::personaldata::UserData

- **Descrizione:** classe per la gestione dei dati di un utente.
- **Utilizzo:** viene utilizzata per la gestione dei dati di un utente.
- **Relazioni con altre classi**
 - ← **BuildConstruction:** classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← **DemolishBuilding:** classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← **DismissUnits:** classe per la gestione del congedo di unità arruolate in precedenza dall'utente.
 - ← **HarvestResources:** classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
 - ← **LoadPersonalData:** classe per la gestione del caricamento dei dati personali dell'utente.
 - ← **ReceiveStealResources:** classe per la ricezione delle risorse rubate
 - ← **RequestStoleResources:** classe per la richiesta di furto di risorse
 - ← **ShowAttackMenu:** classe per la visualizzazione del menu per l'attacco.
 - ← **ShowAttackResultMenu:** classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
 - ← **ShowResourceMenu:** classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
 - ← **ShowTrainUnitMenu:** classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
 - ← **StealResources:** classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
 - ← **TrainUnit:** classe per la gestione dell'addestramento di unità in un edificio.
 - ← **UpgradeBuilding:** classe per la gestione del miglioramento di un edificio desiderato dall'utente.
 - ← **BackupManager:** classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
 - ← **AccountManagerMenuFactory:** classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← **BuildingContextualMenuFactory:** classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← **ResourceMenuFactory:** classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← **TileContextualMenuFactory:** classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.

- ← `UnitSelectionMenu`: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- ← `UserDataManager`: classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- → `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
- → `AuthenticationData`: classe per la gestione delle informazioni personali e di accesso di un utente.
- → `BuildingPossession`: classe per la gestione di un edificio posseduto da un utente.
- → `OwnedResource`: classe che rappresenta una risorsa posseduta da un utente.
- → `UnitPossession`: classe per la rappresentazione di un'unità con associata una quantità.

4.13 Componente `sgad::clienttier::model::userdatamanager`

4.13.1 Informazioni sul package

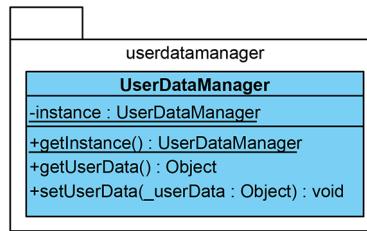


Figura 23: Diagramma della componente `sgad::clienttier::model::userdatamanager`

- **Descrizione:** componente per il mantenimento di un accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- **Padre:** `model`

4.13.2 Classi

4.13.2.1 `sgad::clienttier::model::userdatamanager::UserDataManager`

- **Descrizione:** classe che si occupa di mantenere l'accesso globale ai dati dell'utente di cui si sta attualmente visualizzando il villaggio.
- **Utilizzo:** viene utilizzata per accedere globalmente ai dati dell'utente. Permette inoltre di sostituire i dati dell'utente con i dati di un altro utente per modificare velocemente la rappresentazione del mondo di gioco.
- **Relazioni con altre classi**

- ← **BuildConstruction**: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
- ← **DemolishBuilding**: classe per la gestione della demolizione di un edificio già presente nel villaggio.
- ← **DismissUnits**: classe per la gestione del congedo di unità arruolate in precedenza dall'utente.
- ← **HarvestResources**: classe per la gestione della raccolta di risorse dagli edifici di produzione costruiti nel villaggio.
- ← **LoadPersonalData**: classe per la gestione del caricamento dei dati personali dell'utente.
- ← **RequestStoleResources**: classe per la richiesta di furto di risorse
- ← **ShowAttackResultMenu**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
- ← **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← **TrainUnit**: classe per la gestione dell'addestramento di unità in un edificio.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- ← **BackupManager**: classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
- ← **AccountManagerMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
- ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- → **UserData**: classe per la gestione dei dati di un utente.

4.14 Componente sgad::clienttier::view

4.14.1 Informazioni sul package

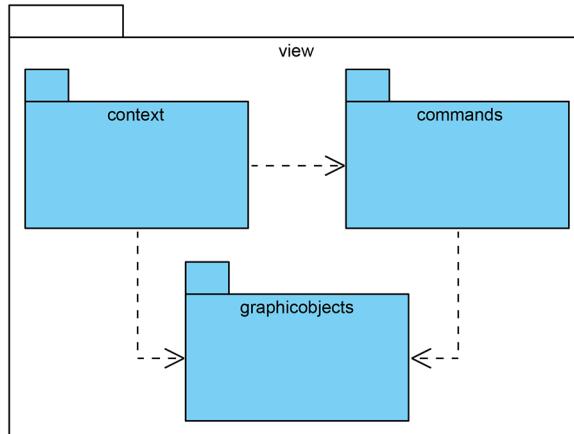


Figura 24: Diagramma della componente sgad::clienttier::view

- **Descrizione:** componente View dell'architettura MVC. Essa gestisce la visualizzazione del gioco e l'input utente all'interno del canvas HTML5.
- **Padre:** clienttier
- **Interazioni con altri componenti**
 - **controller:** componente Controller dell'architettura MVC. Essa esegue le operazioni che l'utente ha richiesto tramite la view agendo se necessario sul model data.
 - **model:** componente Model dell'architettura MVC. Essa memorizza tutti i dati di gioco dell'utente su cui la view si basa per disegnarsi.
- **Package contenuti**
 - **commands:** componente per la gestione dei comandi che vengono ricevuti dall'oggetto in cui viene rappresentato il mondo di gioco.
 - **context:** componente che raccoglie la classe che gestisce l'oggetto che rappresenterà l'ambiente generale di gioco.
 - **graphicobjects:** componente per la gestione degli oggetti grafici che vengono rappresentati nel mondo di gioco quali widget e immagini rappresentanti il mondo di gioco (esempio: edificio).

4.15 Componente sgad::clienttier::view::commands

4.15.1 Informazioni sul package

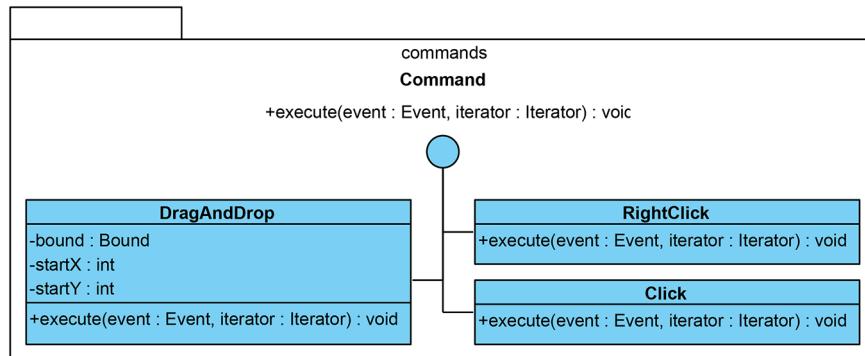


Figura 25: Diagramma della componente `sgad::clienttier::view::commands`

- **Descrizione:** componente per la gestione dei comandi che vengono ricevuti dall'oggetto in cui viene rappresentato il mondo di gioco.
- **Padre:** `view`
- **Interazioni con altri componenti**
 - `context`: componente che raccoglie la classe che gestisce l'oggetto che rappresenterà l'ambiente generale di gioco.
 - `graphicobjects`: componente per la gestione degli oggetti grafici che vengono rappresentati nel mondo di gioco quali widget e immagini rappresentanti il mondo di gioco (esempio: edificio).

4.15.2 Classi

4.15.2.1 sgad::clienttier::view::commands::Command

- **Descrizione:** interfaccia per la gestione di un evento nell'area di gioco. Le sottoclassi determineranno quale evento gestire.
- **Utilizzo:** viene utilizzata per fornire un metodo comune a qualsiasi tipo di eventi che possono verificarsi nell'interazione tra utente e mondo di gioco. Le classi che realizzano l'interfaccia si occuperanno di ridefinire il metodo per gestire l'evento.
- **Sottoclassi**
 - `Click`
 - `DragAndDrop`
 - `RightClick`

- **Relazioni con altre classi**

- ← **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.

4.15.2.2 sgad::clienttier::view::commands::Click

- **Descrizione**: classe per la gestione di un evento, di tipo click sinistro del mouse, nell'area di gioco.
- **Utilizzo**: viene utilizzata per encapsulare la gestione di un evento di tipo click sinistro del mouse. Tale evento deve essere delegato all'effettivo oggetto grafico sul quale è avvenuto. Una volta individuato l'oggetto, verrà invocato su di esso il metodo appropriato.

- **Classi ereditate**

- **Command**

- **Relazioni con altre classi**

- ← **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → **Iterator**: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
 - → **GraphicObject**: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.
 - → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.

4.15.2.3 sgad::clienttier::view::commands::DragAndDrop

- **Descrizione**: classe per la gestione di un evento, di tipo drag and drop, nell'area di gioco.
- **Utilizzo**: viene utilizzata per encapsulare la gestione di un evento di tipo drag and drop. Tale evento prevede di spostare la telecamera sul mondo di gioco sfruttando la relativa azione.
- **Classi ereditate**
- **Command**
- **Relazioni con altre classi**

- ← **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **Iterator**: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.

4.15.2.4 sgad::clienttier::view::commands::RightClick

- **Descrizione**: classe per la gestione di un evento, di tipo click destro del mouse, nell'area di gioco.
- **Utilizzo**: viene utilizzata per encapsulare la gestione di un evento di tipo click destro del mouse. Tale evento deve essere delegato all'effettivo oggetto grafico sul quale è avvenuto. Una volta individuato l'oggetto, verrà invocato su di esso il metodo appropriato.
- **Classi ereditate**
 - **Command**
- **Relazioni con altre classi**
 - ← **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - → **Iterator**: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
 - → **GraphicObject**: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.
 - → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.

4.16 Componente sgad::clienttier::view::context

4.16.1 Informazioni sul package

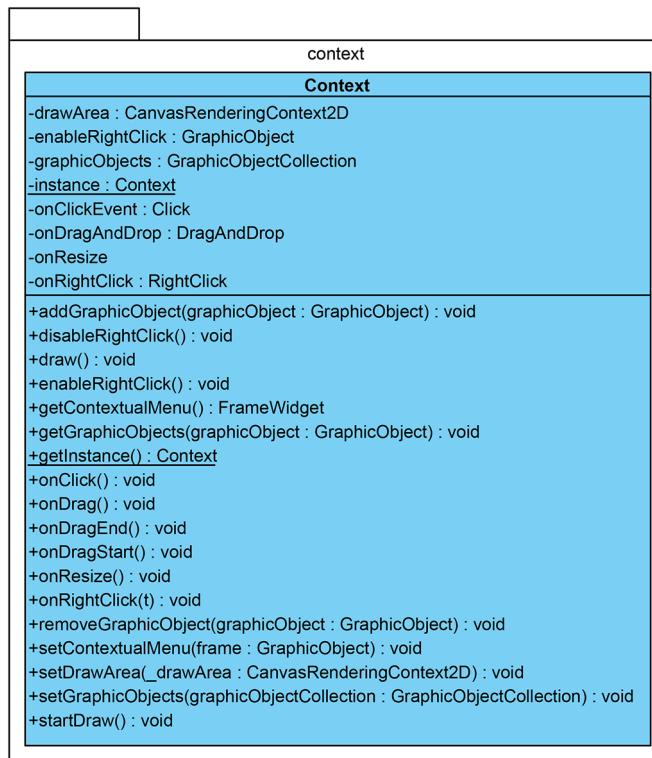


Figura 26: Diagramma della componente `sgad::clienttier::view::context`

- **Descrizione:** componente che raccoglie la classe che gestisce l'oggetto che rappresenterà l'ambiente generale di gioco.
- **Padre:** `view`
- **Interazioni con altri componenti**
 - `observer`: componente per raccogliere le classi che realizzeranno il pattern Observer tipico dell'architettura MVC.
 - `commands`: componente per la gestione dei comandi che vengono ricevuti dall'oggetto in cui viene rappresentato il mondo di gioco.
 - `graphicobject`: componente per la gestione dell'interfaccia per qualsiasi oggetto che verrà visualizzato nell'area di gioco.

4.16.2 Classi

4.16.2.1 `sgad::clienttier::view::context::Context`

- **Descrizione:** classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- **Utilizzo:** viene utilizzata come pannello sul quale viene rappresentato il mondo di gioco. Quando si verifica un evento su tale oggetto, l'evento verrà trasferito ad un gestore opportuno. La classe prevede un metodo opportuno per mantenere aggiornata costantemente la rappresentazione del mondo di gioco.

- **Relazioni con altre classi**

- ← **BuildConstruction**: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
- ← **ChangePasswordAction**: classe per permettere di modificare la password di un account.
- ← **CloseContextualMenu**: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
- ← **DeleteAccountAction**: la classe viene utilizzata per l'eliminazione di un account utente.
- ← **EnableRightClick**: classe per permettere di abilitare la gestione dell'evento click destro del mouse.
- ← **LoadAnotherUser**: classe per il caricamento di un altro utente.
- ← **LoadPersonalData**: classe per la gestione del caricamento dei dati personali dell'utente.
- ← **ReceiveStealResources**: classe per la ricezione delle risorse rubate
- ← **RemoveGraphicObjectAction**: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
- ← **SetAllVoidFilter**: classe per la gestione di un filtro standard.
- ← **SetBuildModeFilter**: classe per la gestione dei filtri per la modalità costruzione.
- ← **ShowAccountDeletedMenu**: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.
- ← **ShowAccountManagerMenu**: classe per la creazione di un menu per la gestione dell'account.
- ← **ShowAnotherUserMenu**: classe per la visualizzazione del menu per ritornare al proprio villaggio una volta che si è nel villaggio di un altro utente.
- ← **ShowAttackMenu**: classe per la visualizzazione del menu per l'attacco.
- ← **ShowAttackResultMenu**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
- ← **ShowBuildConstructionMenu**: classe per la richiesta della conferma della costruzione.

- ← **ShowBuildingContextualMenu**: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
- ← **ShowDemolishMenu**: classe che visualizza il menu per la demolizione di un edificio.
- ← **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- ← **ShowInteractionMenu**: classe per la gestione della visualizzazione di un menu per entrare in modalità costruzione o per la visualizzazione della lista degli utenti.
- ← **ShowLogMenu**: classe per la visualizzazione di un menu di gestione account
- ← **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
- ← **ShowPasswordChangedMenu**: classe per la visualizzazione di un menu di conferma del cambiamento della password.
- ← **ShowResourceMenu**: classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
- ← **ShowTileContextualMenu**: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← **ShowUnitSelectionMenu**: classe per la visualizzazione di un menu per le unità addestrabili.
- ← **ShowUpgradeMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
- ← **ShowUserListMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di visualizzare la lista degli utenti con i quali può interagire.
- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- ← **BackupManager**: classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
- ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
- → **ShowInteractionMenu**: classe per la gestione della visualizzazione di un menu per entrare in modalità costruzione o per la visualizzazione della lista degli utenti.
- → **Click**: classe per la gestione di un evento, di tipo click sinistro del mouse, nell'area di gioco.

- → **Command**: interfaccia per la gestione di un evento nell'area di gioco. Le sottoclassi determineranno quale evento gestire.
- → **DragAndDrop**: classe per la gestione di un evento, di tipo drag and drop, nell'area di gioco.
- → **RightClick**: classe per la gestione di un evento, di tipo click destro del mouse, nell'area di gioco.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **GraphicObjectCollection**: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.
- → **Iterator**: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
- → **GraphicObject**: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.

4.17 Componente sgad::clienttier::view::graphicobjects

4.17.1 Informazioni sul package

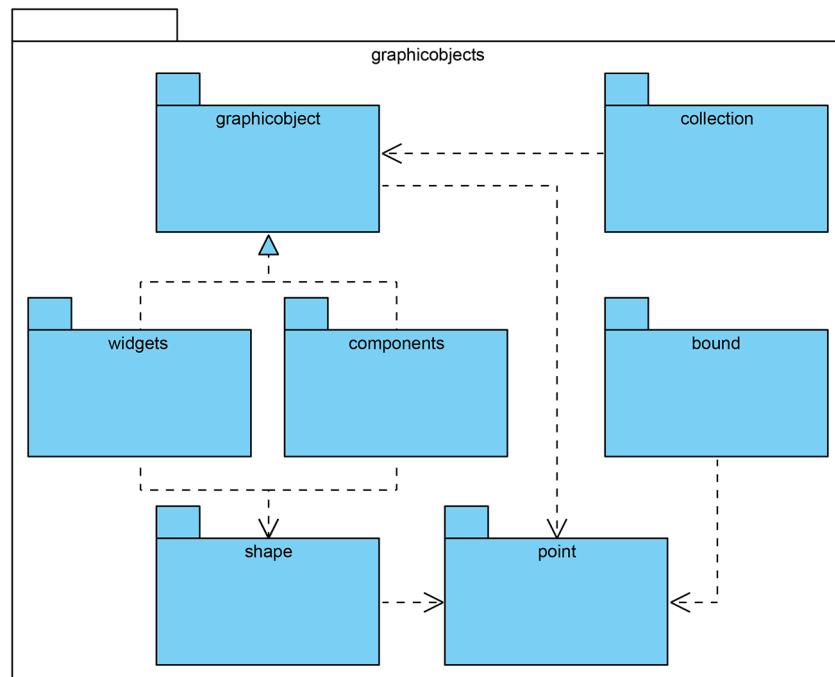


Figura 27: Diagramma della componente `sgad::clienttier::view::graphicobjects`

- **Descrizione:** componente per la gestione degli oggetti grafici che vengono rappresentati nel mondo di gioco quali widget e immagini rappresentanti il mondo di gioco (esempio: edificio).
- **Padre:** view
- **Interazioni con altri componenti**
 - actions: componente per la gestione delle possibili azioni che un utente può intraprendere durante l'interazione con il mondo di gioco.
 - observer: componente per raccogliere le classi che realizzeranno il pattern Observer tipico dell'architettura MVC.
 - commands: componente per la gestione dei comandi che vengono ricevuti dall'oggetto in cui viene rappresentato il mondo di gioco.
- **Package contenuti**
 - bound: componente per la gestione dell'area visualizzata dall'utente rispetto all'intero mondo di gioco.
 - collection: componente per la raccolta delle classi che gestiranno le collezioni di oggetti grafici del mondo di gioco.
 - components: componente per la gestione degli oggetti del mondo di gioco e della relativa visualizzazione.
 - graphicobject: componente per la gestione dell'interfaccia per qualsiasi oggetto che verrà visualizzato nell'area di gioco.
 - point: componente per la gestione dei punti rappresentabili con due coordinate.
 - shape: componente per la gestione delle forme degli oggetti grafici.
 - widget: componente per la gestione dei widget per permettere all'utente di interagire con il mondo di gioco attraverso menu e finestre.

4.18 Componente sgad::clienttier::view::graphicobjects::bound

4.18.1 Informazioni sul package

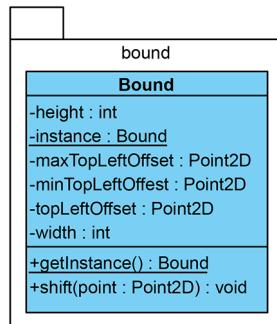


Figura 28: Diagramma della componente sgad::clienttier::view::graphicobjects::-bound

- **Descrizione:** componente per la gestione dell'area visualizzata dall'utente rispetto all'intero mondo di gioco.
- **Padre:** `graphicobjects`

4.18.2 Classi

4.18.2.1 `sgad::clienttier::view::graphicobjects::bound::Bound`

- **Descrizione:** classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- **Utilizzo:** viene utilizzata gestire i limiti entro i quali devono essere rappresentati gli oggetti grafici del mondo di gioco.

• Relazioni con altre classi

- ← `AccountManagerMenuFactory`: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← `AnotherUserMenuFactory`: classe per la costruzione di un menu per il ritorno al villaggio dell'utente.
- ← `AttackResultMenuFactory`: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
- ← `BuildingContextualMenuFactory`: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← `ConfirmMenuFactory`: classe per la creazione di un menu che permetta di confermare una generica azione.
- ← `InteractionMenuFactory`: classe per la creazione di un menu per interagire con l'intero mondo di gioco.
- ← `LogMenuFactory`: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← `NotifyMenuFactory`: classe per la costruzione di un menu per la visualizzazione di una notifica.
- ← `ResourceMenuFactory`: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
- ← `TileContextualMenuFactory`: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- ← `UserListMenuFactory`: classe per la visualizzazione della lista di altri giocatori connessi.
- ← `DragAndDrop`: classe per la gestione di un evento, di tipo drag and drop, nell'area di gioco.
- ← `Context`: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.

- ← **BuildModeFilter**: classe per la gestione del filtro quando viene selezionata la modalità costruzione.
- ← **VoidFilter**: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
- ← **TileComponent**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- ← **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.

4.19 Componente sgad::clienttier::view::graphicobjects::collection

4.19.1 Informazioni sul package

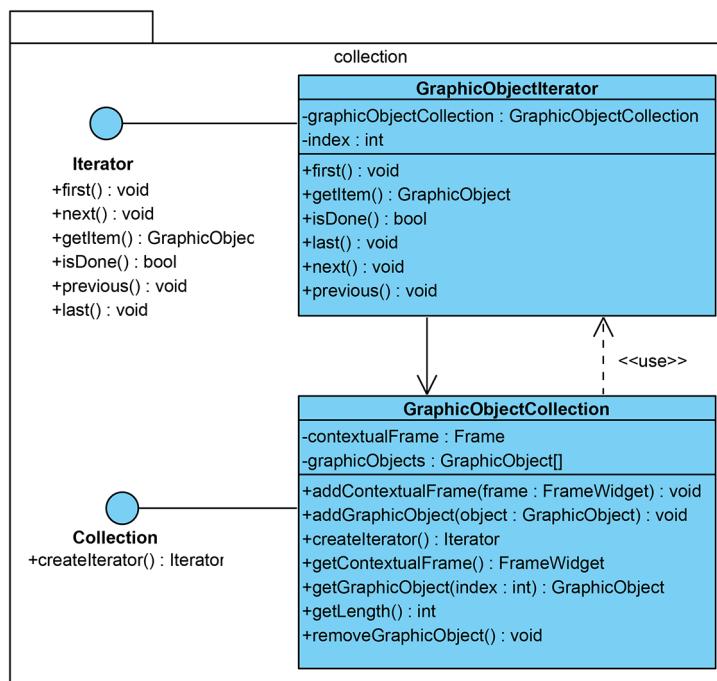


Figura 29: Diagramma della componente sgad::clienttier::view::graphicobjects::collection

- **Descrizione:** componente per la raccolta delle classi che gestiranno le collezioni di oggetti grafici del mondo di gioco.
- **Padre:** `graphicobjects`
- **Interazioni con altri componenti**
 - `graphicobject`: componente per la gestione dell'interfaccia per qualsiasi oggetto che verrà visualizzato nell'area di gioco.

4.19.2 Classi

4.19.2.1 sgad::clienttier::view::graphicobjects::collection::Collection

- **Descrizione:** interfaccia per le classi che rappresentano collezioni di oggetti
- **Utilizzo:** viene utilizzata per fornire dei metodi standard per qualsiasi classe che rappresenta una collezione di oggetti. Le classi che realizzano l'interfaccia si occuperanno di ridefinire i metodi per accedere alla collezione.
- **Sottoclassi**
 - GraphicObjectCollection
- **Relazioni con altre classi**
 - → Iterator: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.

4.19.2.2 sgad::clienttier::view::graphicobjects::collection::GraphicObjectCollection

- **Descrizione:** classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.
- **Utilizzo:** viene utilizzata per organizzare la struttura dati che raccoglierà l'insieme degli oggetti grafici da rappresentare. Ciò permette di nascondere l'effettiva struttura dati alle classi che utilizzano questa classe. Tale astrazione è necessaria per permettere di gestire facilmente l'ordine di rappresentazione degli oggetti grafici. La collezione prevede di essere ordinata secondo la profondità decrescente dell'oggetto.
- **Classi ereditate**
 - Collection
- **Relazioni con altre classi**
 - ← LoadAnotherUser: classe per il caricamento di un altro utente.
 - ← SetAllVoidFilter: classe per la gestione di un filtro standard.
 - ← SetBuildModeFilter: classe per la gestione dei filtri per la modalità costruzione.
 - ← BackupManager: classe che si occupa di gestire il mantenimento dei dati e delle viste dell'utente proprietario dell'account.
 - ← Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.

- ← **GraphicObjectIterator**: classe per la gestione di un iteratore per accedere ad una collezione di oggetti grafici.
- → **GraphicObjectIterator**: classe per la gestione di un iteratore per accedere ad una collezione di oggetti grafici.
- → **GraphicObject**: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.
- → **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.

4.19.2.3 sgad::clienttier::view::graphicobjects::collection::Iterator

- **Descrizione**: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
- **Utilizzo**: viene utilizzata per fornire dei metodi standard per qualsiasi tipo di iteratore. Le classi che realizzano l'interfaccia si occuperanno di ridefinire i metodi per utilizzare l'iteratore.
- **Sottoclassi**
 - **GraphicObjectIterator**
- **Relazioni con altre classi**
 - ← **SetAllVoidFilter**: classe per la gestione di un filtro standard.
 - ← **SetBuildModeFilter**: classe per la gestione dei filtri per la modalità costruzione.
 - ← **Click**: classe per la gestione di un evento, di tipo click sinistro del mouse, nell'area di gioco.
 - ← **DragAndDrop**: classe per la gestione di un evento, di tipo drag and drop, nell'area di gioco.
 - ← **RightClick**: classe per la gestione di un evento, di tipo click destro del mouse, nell'area di gioco.
 - ← **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
 - ← **Collection**: interfaccia per le classi che rappresentano collezioni di oggetti
 - → **GraphicObject**: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.

4.19.2.4 sgad::clienttier::view::graphicobjects::collection::GraphicObjectIterator

- **Descrizione**: classe per la gestione di un iteratore per accedere ad una collezione di oggetti grafici.

- **Utilizzo:** viene utilizzata per permettere di accedere ad una collezione di elementi grafici senza conoscere l'effettiva struttura di tale collezione.

- **Classi ereditate**
 - `Iterator`

- **Relazioni con altre classi**
 - \leftarrow `SetAllVoidFilter`: classe per la gestione di un filtro standard.
 - \leftarrow `SetBuildModeFilter`: classe per la gestione dei filtri per la modalità costruzione.
 - \leftarrow `GraphicObjectCollection`: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.
 - \rightarrow `GraphicObjectCollection`: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.
 - \rightarrow `GraphicObject`: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.

4.20 Componente sgad::clienttier::view::graphicobjects::components

4.20.1 Informazioni sul package

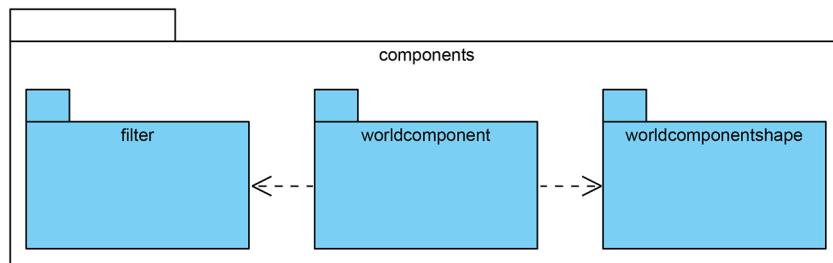


Figura 30: Diagramma della componente `sgad::clienttier::view::graphicobjects::components`

- **Descrizione:** componente per la gestione degli oggetti del mondo di gioco e della relativa visualizzazione.

- **Padre:** `graphicobjects`

- **Interazioni con altri componenti**
 - `menufactory`: componente per la gestione dei vari menu che possono essere visualizzati durante le interazioni con il mondo di gioco.
 - `graphicobject`: componente per la gestione dell'interfaccia per qualsiasi oggetto che verrà visualizzato nell'area di gioco.

- **shape**: componente per la gestione delle forme degli oggetti grafici.

- **Package contenuti**

- **filter**: componente per la gestione dei vari filtri con i quali vengono visualizzati gli oggetti del mondo di gioco.
- **worldcomponent**: componente per la gestione dei vari oggetti, quali edifici e unità, del mondo di gioco.
- **worldcomponentshape**: componente per la gestione delle forme e delle immagini per gli oggetti del mondo di gioco.

4.21 Componente sgad::clienttier::view::graphicobjects::components::filter

4.21.1 Informazioni sul package

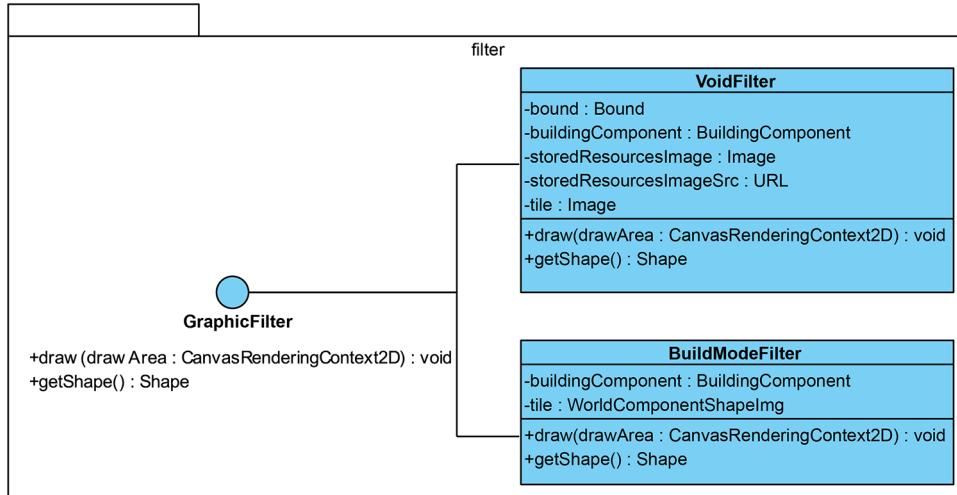


Figura 31: Diagramma della componente `sgad::clienttier::view::graphicobjects::components::filter`

- **Descrizione**: componente per la gestione dei vari filtri con i quali vengono visualizzati gli oggetti del mondo di gioco.
- **Padre**: `components`

4.21.2 Classi

4.21.2.1 sgad::clienttier::view::graphicobjects::components::filter::GraphicFilter

- **Descrizione**: interfaccia per i possibili filtri che possono essere applicati ad un edificio.

- **Utilizzo:** viene utilizzata per fornire un metodo comune a qualsiasi tipo di filtro che può essere utilizzato. Le classi che realizzano l'interfaccia si occuperanno di ridefinire il metodo per gestire il filtro.

- **Sottoclassi**

- `BuildModeFilter`
- `VoidFilter`

- **Relazioni con altre classi**

- ← `BuildingComponent`: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → `Shape`: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.21.2.2 `sgad::clienttier::view::graphicobjects::components::filter::BuildModeFilter`

- **Descrizione:** classe per la gestione del filtro quando viene selezionata la modalità costruzione.

- **Utilizzo:** viene utilizzata per incapsulare l'algoritmo per la rappresentazione di una costruzione quando l'utente entra in modalità costruzione.

- **Classi ereditate**

- `GraphicFilter`

- **Relazioni con altre classi**

- ← `SetBuildModeFilter`: classe per la gestione dei filtri per la modalità costruzione.
- → `Position`: classe per la gestione di una posizione sulla griglia del villaggio.
- → `Bound`: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → `BuildingComponent`: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- → `WorldComponentShapeFactory`: classe per la gestione del controllo ai vari oggetti di tipo `WorldComponentShapeImg`.
- → `WorldComponentShapeImg`: classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all'interno del mondo di gioco.
- → `Point2D`: classe per la rappresentazione di un punto avente due coordinate.
- → `Shape`: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.21.2.3 sgad::clienttier::view::graphicobjects::components::filter::VoidFilter

- **Descrizione:** classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
- **Utilizzo:** viene utilizzata per gestire la rappresentazione standard di un edificio nell'area di gioco.
- **Classi ereditate**
 - GraphicFilter
- **Relazioni con altre classi**
 - ← BuildConstruction: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← SetAllVoidFilter: classe per la gestione di un filtro standard.
 - → Position: classe per la gestione di una posizione sulla griglia del villaggio.
 - → Bound: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
 - → BuildingComponent: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
 - → WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
 - → WorldComponentShapeImg: classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all'interno del mondo di gioco.
 - → Point2D: classe per la rappresentazione di un punto avente due coordinate.
 - → Shape: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.22 Componente sgad::clienttier::view::graphicobjects::components::worldcomponent

4.22.1 Informazioni sul package

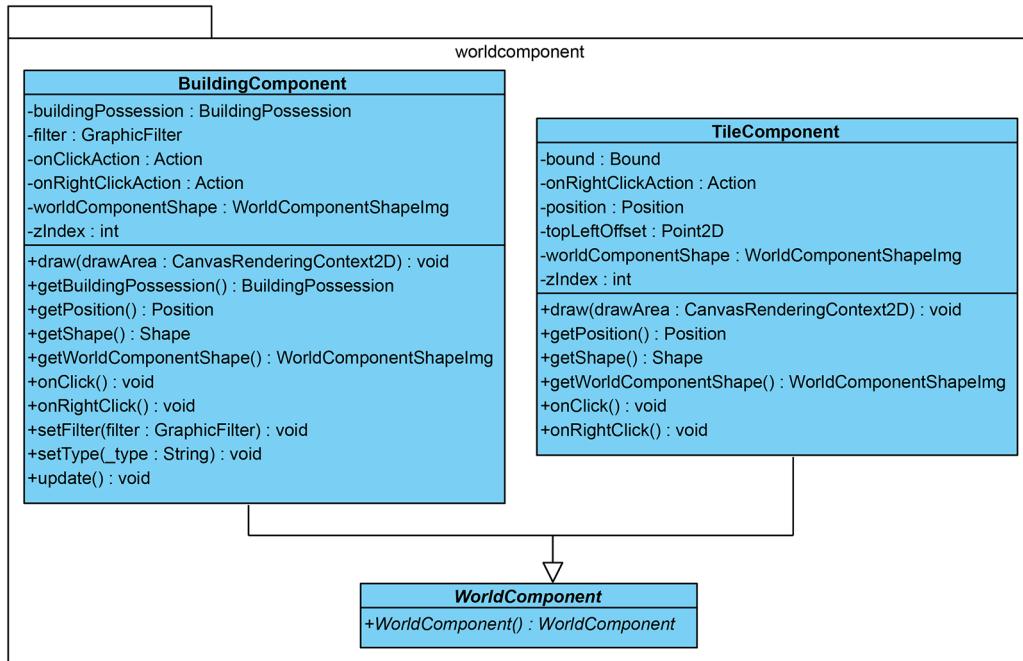


Figura 32: Diagramma della componente `sgad::clienttier::view::graphicobjects::components::worldcomponent`

- **Descrizione:** componente per la gestione dei vari oggetti, quali edifici e unità, del mondo di gioco.
- **Padre:** `components`
- **Interazioni con altri componenti**
 - `worldcomponentshape`: componente per la gestione delle forme e delle immagini per gli oggetti del mondo di gioco.

4.22.2 Classi

4.22.2.1 sgad::clienttier::view::graphicobjects::components::worldcomponent::WorldComponent

- **Descrizione:** classe astratta per la gestione di un qualsiasi edificio o casella all'interno del mondo di gioco.
- **Utilizzo:** viene utilizzata per fornire dei metodi comuni a tutti gli edifici e caselle che verranno creati all'interno del mondo di gioco. Tale classe contiene il riferimento ad

un oggetto di tipo `WorldComponentShapeFactory` per ottenere degli oggetti di tipo `WorldComponentShapeImg`.

- **Classi ereditate**

- `GraphicObject`

- **Sottoclassi**

- `BuildingComponent`
 - `TileComponent`

- **Relazioni con altre classi**

- → `GraphicObject`: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.

4.22.2.2 `sgad::clienttier::view::graphicobjects::components::worldcomponent::BuildingComponent`

- **Descrizione:** classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.

- **Utilizzo:** viene utilizzata per fornire un comportamento comune ai possibili edifici del mondo di gioco. Utilizza il modello dei dati per determinare quale sia il tipo di edificio da rappresentare e il comportamento associato.

- **Classi ereditate**

- `Observer`
 - `WorldComponent`

- **Relazioni con altre classi**

- ← `BuildConstruction`: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← `DemolishBuilding`: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← `LoadPersonalData`: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← `RequestDemolishBuilding`: classe per la richiesta di demolizione di un edificio.
 - ← `ShowDemolishMenu`: classe che visualizza il menu per la demolizione di un edificio.
 - ← `BuildingContextualMenuFactory`: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← `BuildModeFilter`: classe per la gestione del filtro quando viene selezionata la modalità costruzione.

- ← **VoidFilter**: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
- → **Action**: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.
- → **ShowBuildingContextualMenu**: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
- → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **Position**: classe per la gestione di una posizione sulla griglia del villaggio.
- → **GraphicFilter**: interfaccia per i possibili filtri che possono essere applicati ad un edificio.
- → **WorldComponentShapeFactory**: classe per la gestione del controllo ai vari oggetti di tipo **WorldComponentShapeImg**.
- → **WorldComponentShapeImg**: classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all'interno del mondo di gioco.
- → **Shape**: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.22.2.3 sgad::clienttier::view::graphicobjects::components::worldcomponent::TileComponent

- **Descrizione**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- **Utilizzo**: viene utilizzata per rappresentare una casella libera nel mondo di gioco.
- **Classi ereditate**
 - **WorldComponent**
- **Relazioni con altre classi**
 - ← **BuildConstruction**: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← **DemolishBuilding**: classe per la gestione della demolizione di un edificio già presente nel villaggio.
 - ← **LoadPersonalData**: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← **RequestBuildConstruction**: classe per la richiesta di costruzione di un edificio.
 - ← **ShowBuildConstructionMenu**: classe per la richiesta della conferma della costruzione.

- ← **ShowTileContextualMenu**: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
- ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizza gli edifici costruibili in una casella.
- → **Action**: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.
- → **Position**: classe per la gestione di una posizione sulla griglia del villaggio.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **WorldComponentShapeImg**: classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all'interno del mondo di gioco.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **Shape**: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.23 Componente sgad::clienttier::view::graphicobjects::components::worldcomponentshape

4.23.1 Informazioni sul package

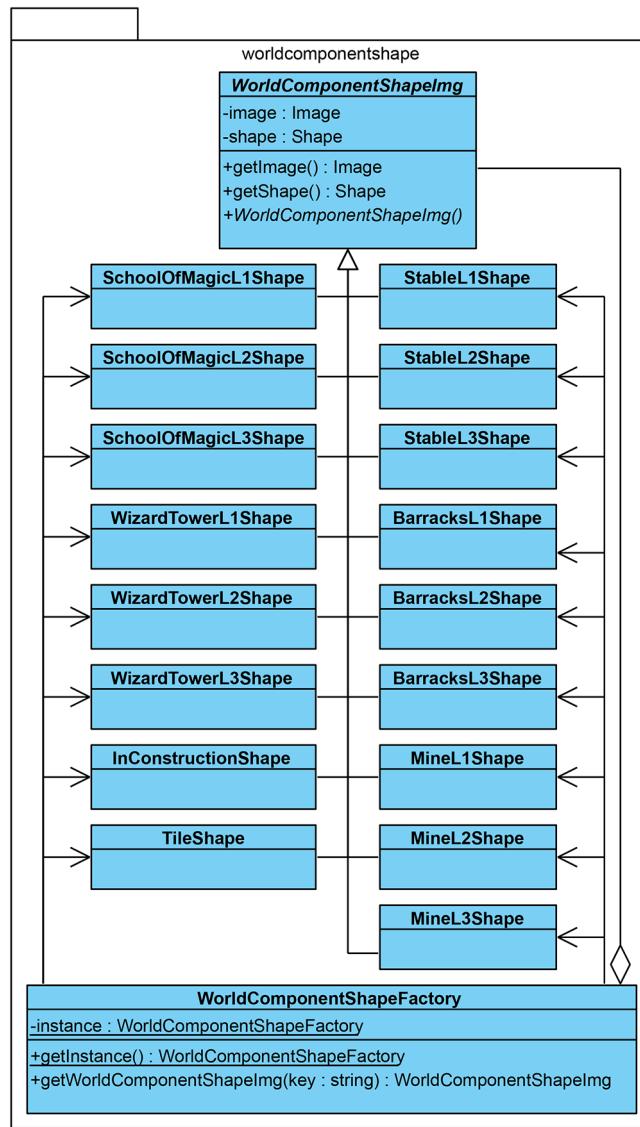


Figura 33: Diagramma della componente `sgad::clienttier::view::graphicobjects::components::worldcomponentshape`

- **Descrizione:** componente per la gestione delle forme e delle immagini per gli oggetti del mondo di gioco.
- **Padre:** `components`
- **Interazioni con altri componenti**

- **worldcomponent**: componente per la gestione dei vari oggetti, quali edifici e unità, del mondo di gioco.

4.23.2 Classi

4.23.2.1 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WorldComponentShapeFactory

- **Descrizione**: classe per la gestione del controllo ai vari oggetti di tipo **WorldComponentShapeImg**.
- **Utilizzo**: viene utilizzata per fornire un determinato oggetto di tipo **WorldComponentShapeImg** restituendo un riferimento a tale oggetto se già esistente oppure creandolo al momento.
- **Relazioni con altre classi**
 - ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizza gli edifici costruibili in una casella.
 - ← **BuildModeFilter**: classe per la gestione del filtro quando viene selezionata la modalità costruzione.
 - ← **VoidFilter**: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
 - ← **BuildingComponent**: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
 - → **BarracksL1Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una caserma di livello 1.
 - → **BarracksL2Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una caserma di livello 2.
 - → **BarracksL3Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una caserma di livello 3.
 - → **InConstructionShape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di un edificio in costruzione.
 - → **MineL1Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una miniera di livello 1.
 - → **MineL2Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una miniera di livello 2.
 - → **MineL3Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una miniera di livello 3.
 - → **SchoolOfMagicL1Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una scuola di magia di livello 1.
 - → **SchoolOfMagicL2Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una scuola di magia di livello 2.
 - → **SchoolOfMagicL3Shape**: classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una scuola di magia di livello 3.

- → `StableL1Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una stalla di livello 1.
- → `StableL2Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una stalla di livello 2.
- → `StableL3Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una stalla di livello 3.
- → `TileShape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una casella.
- → `WizardTowerL1Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una torre dello stregone di livello 1.
- → `WizardTowerL2Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una torre dello stregone di livello 2.
- → `WizardTowerL3Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una torre dello stregone di livello 3.
- → `WorldComponentShapeImg`: classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all’interno del mondo di gioco.

4.23.2.2 `sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WorldComponentShapeImg`

- **Descrizione:** classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all’interno del mondo di gioco.
- **Utilizzo:** viene utilizzata per fornire un metodo comune a tutti i tipi di forme che possono essere rappresentate. Le classi che ereditano dalla classe si occuperanno di ridefinire il metodo per essere rappresentate.
- **Sottoclassi**
 - `BarracksL1Shape`
 - `BarracksL2Shape`
 - `BarracksL3Shape`
 - `InConstructionShape`
 - `MineL1Shape`
 - `MineL2Shape`
 - `MineL3Shape`
 - `SchoolOfMagicL1Shape`
 - `SchoolOfMagicL2Shape`
 - `SchoolOfMagicL3Shape`
 - `StableL1Shape`
 - `StableL2Shape`

- StableL3Shape
- TileShape
- WizardTowerL1Shape
- WizardTowerL2Shape
- WizardTowerL3Shape

- Relazioni con altre classi

- ← TileContextualMenuFactory: classe per la creazione di un menu che visualizza gli edifici costruibili in una casella.
- ← BuildModeFilter: classe per la gestione del filtro quando viene selezionata la modalità costruzione.
- ← VoidFilter: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
- ← BuildingComponent: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
- ← TileComponent: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Shape: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.23.2.3 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::BarracksL1Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una caserma di livello 1.
- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una caserma di livello 1.

- Classi ereditate

- WorldComponentShapeImg

- Relazioni con altre classi

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.4 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::BarracksL2Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una caserma di livello 2.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una caserma di livello 2.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.

- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.5 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::BarracksL3Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una caserma di livello 3.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una caserma di livello 3.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.

- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.6 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::InConstructionShape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di un edificio in costruzione.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare un edificio in costruzione.

- **Classi ereditate**

- WorldComponentShapeImg

- Relazioni con altre classi

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.7 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::MineL1Shape

- **Descrizione:** classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una miniera di livello 1.
- **Utilizzo:** viene utilizzata per mantenere un’unica volta in tutto il sistema i dati comuni per rappresentare una miniera di livello 1.
- **Classi ereditate**

- WorldComponentShapeImg

- Relazioni con altre classi

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.8 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::MineL2Shape

- **Descrizione:** classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una miniera di livello 2.
- **Utilizzo:** viene utilizzata per mantenere un’unica volta in tutto il sistema i dati comuni per rappresentare una miniera di livello 2.
- **Classi ereditate**

- WorldComponentShapeImg

- Relazioni con altre classi

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.9 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::MineL3Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una miniera di livello 3.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una miniera di livello 3.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.

- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.10 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::SchoolOfMagicL1Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una scuola di magia di livello 1.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una scuola di magia di livello 1.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.

- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.11 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::SchoolOfMagicL2Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una scuola di magia di livello 2.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una scuola di magia di livello 2.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.12 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::SchoolOfMagicL3Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una scuola di magia di livello 3.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una scuola di magia di livello 3.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.13 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::StableL1Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una stalla di livello 1.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una stalla di livello 1.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.14 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::StableL2Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una stalla di livello 2.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una stalla di livello 2.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.

- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.15 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::StableL3Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una stalla di livello 3.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una stalla di livello 3.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.

- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.16 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::TileShape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una casella.

- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una casella.

- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.17 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WizardTowerL1Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una torre dello stregone di livello 1.
- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una torre dello stregone di livello 1.
- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.18 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WizardTowerL2Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una torre dello stregone di livello 2.
- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una torre dello stregone di livello 2.
- **Classi ereditate**

- WorldComponentShapeImg

- **Relazioni con altre classi**

- ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.23.2.19 sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WizardTowerL3Shape

- **Descrizione:** classe per la memorizzazione dell'immagine e del poligono per la rappresentazione di una torre dello stregone di livello 3.
- **Utilizzo:** viene utilizzata per mantenere un'unica volta in tutto il sistema i dati comuni per rappresentare una torre dello stregone di livello 3.
- **Classi ereditate**
 - WorldComponentShapeImg
- **Relazioni con altre classi**
 - ← WorldComponentShapeFactory: classe per la gestione del controllo ai vari oggetti di tipo WorldComponentShapeImg.
 - → Point2D: classe per la rappresentazione di un punto avente due coordinate.

4.24 Componente sgad::clienttier::view::graphicobjects::graphicobject

4.24.1 Informazioni sul package

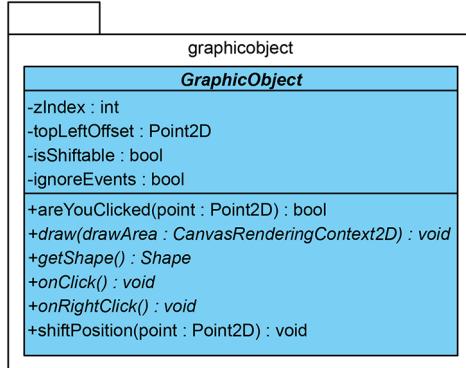


Figura 34: Diagramma della componente sgad::clienttier::view::graphicobjects::-graphicobject

- **Descrizione:** componente per la gestione dell’interfaccia per qualsiasi oggetto che verrà visualizzato nell’area di gioco.
- **Padre:** `graphicobjects`
- **Interazioni con altri componenti**
 - `context`: componente che raccoglie la classe che gestisce l’oggetto che rappresenterà l’ambiente generale di gioco.
 - `collection`: componente per la raccolta delle classi che gestiranno le collezioni di oggetti grafici del mondo di gioco.
 - `components`: componente per la gestione degli oggetti del mondo di gioco e della relativa visualizzazione.
 - `widget`: componente per la gestione dei widget per permettere all’utente di interagire con il mondo di gioco attraverso menu e finestre.

4.24.2 Classi

4.24.2.1 sgad::clienttier::view::graphicobjects::graphicobject::GraphicObject

- **Descrizione:** classe astratta per la rappresentazione di un qualsiasi oggetto grafico all’interno dell’area di gioco.
- **Utilizzo:** viene utilizzata per fornire dei metodi comuni per interagire con un oggetto grafico. Le varie sottoclassi si occuperanno di realizzare tali metodi.
- **Sottoclassi**

- WorldComponent
- Widget

- Relazioni con altre classi

- ← RemoveGraphicObjectAction: classe per la rimozione di un oggetto grafico visualizzato nell'area di gioco.
- ← Click: classe per la gestione di un evento, di tipo click sinistro del mouse, nell'area di gioco.
- ← RightClick: classe per la gestione di un evento, di tipo click destro del mouse, nell'area di gioco.
- ← Context: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- ← GraphicObjectCollection: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.
- ← GraphicObjectIterator: classe per la gestione di un iteratore per accedere ad una collezione di oggetti grafici.
- ← Iterator: interfaccia per i possibili iteratori che possono essere realizzati per accedere a classi per la gestione di aggregati di dati.
- ← WorldComponent: classe astratta per la gestione di un qualsiasi edificio o casella all'interno del mondo di gioco.
- → Point2D: classe per la rappresentazione di un punto avente due coordinate.
- → Shape: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.25 Componente sgad::clienttier::view::graphicobjects::point

4.25.1 Informazioni sul package

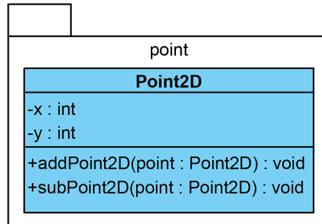


Figura 35: Diagramma della componente sgad::clienttier::view::graphicobjects::-point

- **Descrizione:** componente per la gestione dei punti rappresentabili con due coordinate.
- **Padre:** graphicobjects

4.25.2 Classi

4.25.2.1 sgad::clienttier::view::graphicobjects::point::Point2D

- **Descrizione:** classe per la rappresentazione di un punto avente due coordinate.
- **Utilizzo:** viene utilizzata per rappresentare un punto indicato dalle coordinate x e y.
- **Relazioni con altre classi**
 - ← ShowBuildingContextualMenu: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
 - ← ShowTileContextualMenu: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
 - ← AccountManagerMenuFactory: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← AnotherUserMenuFactory: classe per la costruzione di un menu per il ritorno al villaggio dell'utente.
 - ← AttackResultMenuFactory: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
 - ← BuildingContextualMenuFactory: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← ConfirmMenuFactory: classe per la creazione di un menu che permetta di confermare una generica azione.
 - ← InteractionMenuFactory: classe per la creazione di un menu per interagire con l'intero mondo di gioco.

- ← **LogMenuFactory**: classe per la costruzione di un menu che permetta all’utente di modificare il proprio account.
- ← **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
- ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall’utente.
- ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- ← **UserListMenuFactory**: classe per la visualizzazione della lista di altri giocatori connessi.
- ← **Click**: classe per la gestione di un evento, di tipo click sinistro del mouse, nell’area di gioco.
- ← **DragAndDrop**: classe per la gestione di un evento, di tipo drag and drop, nell’area di gioco.
- ← **RightClick**: classe per la gestione di un evento, di tipo click destro del mouse, nell’area di gioco.
- ← **Bound**: classe per la gestione dei limiti della porzione dell’area di gioco effettivamente visualizzata.
- ← **BuildModeFilter**: classe per la gestione del filtro quando viene selezionata la modalità costruzione.
- ← **VoidFilter**: classe per la gestione di un filtro che non deve modificare la rappresentazione dell’edificio.
- ← **TileComponent**: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
- ← **BarracksL1Shape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una caserma di livello 1.
- ← **BarracksL2Shape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una caserma di livello 2.
- ← **BarracksL3Shape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una caserma di livello 3.
- ← **InConstructionShape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di un edificio in costruzione.
- ← **MineL1Shape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una miniera di livello 1.
- ← **MineL2Shape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una miniera di livello 2.
- ← **MineL3Shape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una miniera di livello 3.
- ← **SchoolOfMagicL1Shape**: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una scuola di magia di livello 1.

- ← `SchoolOfMagicL2Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una scuola di magia di livello 2.
- ← `SchoolOfMagicL3Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una scuola di magia di livello 3.
- ← `StableL1Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una stalla di livello 1.
- ← `StableL2Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una stalla di livello 2.
- ← `StableL3Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una stalla di livello 3.
- ← `TileShape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una casella.
- ← `WizardTowerL1Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una torre dello stregone di livello 1.
- ← `WizardTowerL2Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una torre dello stregone di livello 2.
- ← `WizardTowerL3Shape`: classe per la memorizzazione dell’immagine e del poligono per la rappresentazione di una torre dello stregone di livello 3.
- ← `GraphicObject`: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all’interno dell’area di gioco.
- ← `Shape`: classe per la gestione di un insieme di punti rappresentanti un poligono.
- ← `ButtonWidget`: classe per la gestione di un generico bottone nell’interfaccia grafica.
- ← `FrameWidget`: classe per la gestione di un generico frame nell’interfaccia grafica.
- ← `ImageWidget`: classe per la gestione di una generica immagine nell’interfaccia grafica.
- ← `TextWidget`: classe per la gestione di un generico testo nell’interfaccia grafica.

4.26 Componente `sgad::clienttier::view::graphicobjects::shape`

4.26.1 Informazioni sul package

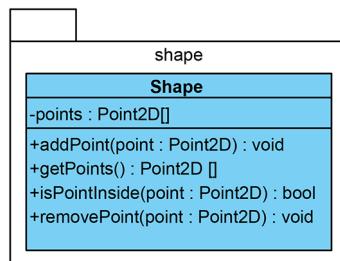


Figura 36: Diagramma della componente `sgad::clienttier::view::graphicobjects::-shape`

- **Descrizione:** componente per la gestione delle forme degli oggetti grafici.
- **Padre:** `graphicobjects`
- **Interazioni con altri componenti**
 - `components`: componente per la gestione degli oggetti del mondo di gioco e della relativa visualizzazione.
 - `widget`: componente per la gestione dei widget per permettere all'utente di interagire con il mondo di gioco attraverso menu e finestre.

4.26.2 Classi

4.26.2.1 `sgad::clienttier::view::graphicobjects::shape::Shape`

- **Descrizione:** classe per la gestione di un insieme di punti rappresentanti un poligono.
- **Utilizzo:** viene utilizzata per rappresentare un insieme di punti. Le sottoclassi di `GraphicObject` usano oggetti di tale classe per avere un'indicazione su quale sia il poligono che le rappresentano.
- **Relazioni con altre classi**
 - ← `BuildModeFilter`: classe per la gestione del filtro quando viene selezionata la modalità costruzione.
 - ← `GraphicFilter`: interfaccia per i possibili filtri che possono essere applicati ad un edificio.
 - ← `VoidFilter`: classe per la gestione di un filtro che non deve modificare la rappresentazione dell'edificio.
 - ← `BuildingComponent`: classe per la rappresentazione di un edificio generico all'interno dell'area di gioco.
 - ← `TileComponent`: classe per la rappresentazione di una casella del mondo di gioco sulla quale non è stato costruito alcun edificio.
 - ← `WorldComponentShapeImg`: classe astratta per le possibili forme degli edifici e delle unità che possono essere rappresentate all'interno del mondo di gioco.
 - ← `GraphicObject`: classe astratta per la rappresentazione di un qualsiasi oggetto grafico all'interno dell'area di gioco.
 - ← `ButtonWidget`: classe per la gestione di un generico bottone nell'interfaccia grafica.
 - ← `FrameWidget`: classe per la gestione di un generico frame nell'interfaccia grafica.
 - ← `ImageWidget`: classe per la gestione di una generica immagine nell'interfaccia grafica.
 - ← `TextWidget`: classe per la gestione di un generico testo nell'interfaccia grafica.

- ← **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.

4.27 Componente sgad::clienttier::view::graphicobjects::widget

4.27.1 Informazioni sul package

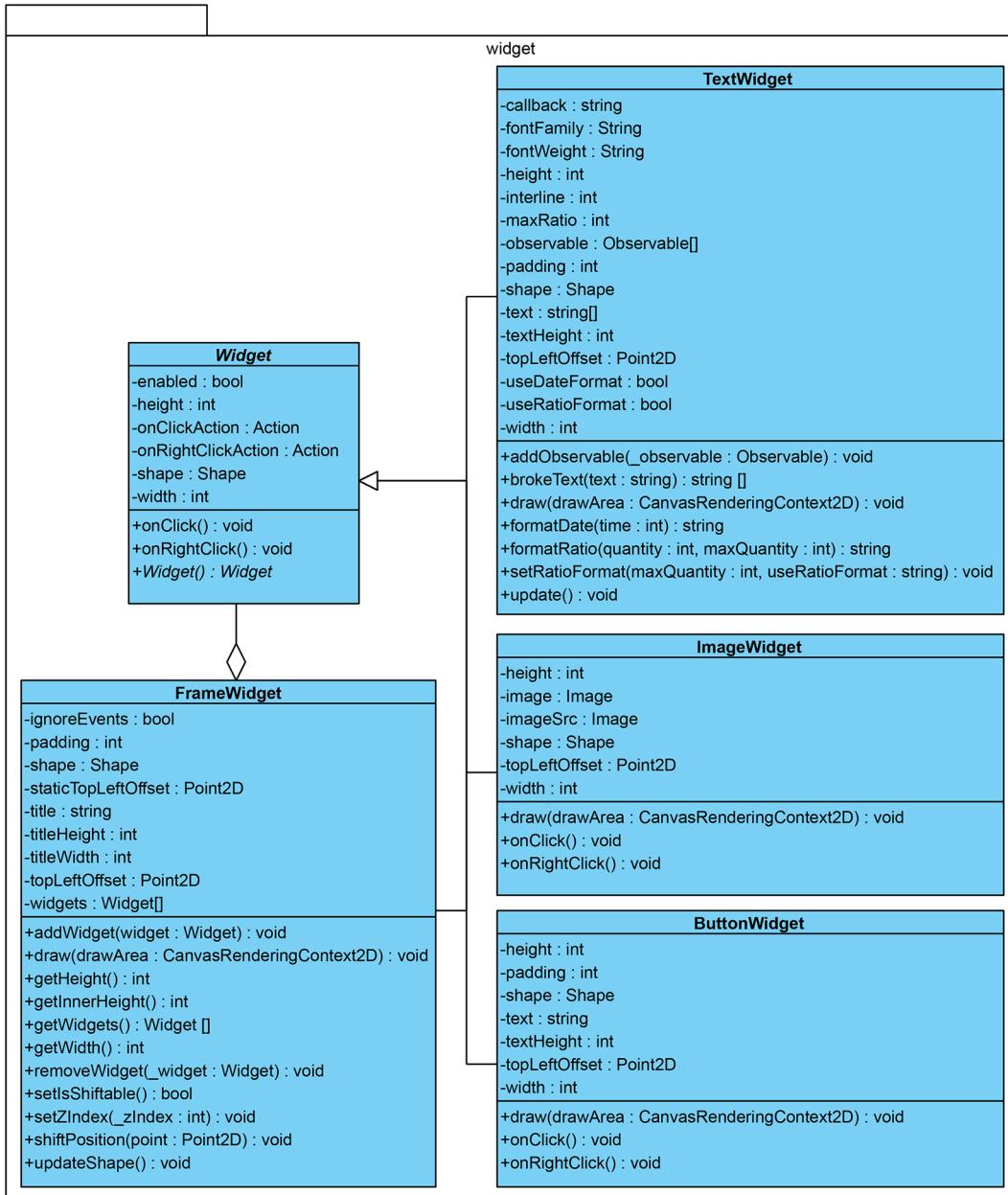


Figura 37: Diagramma della componente `sgad::clienttier::view::graphicobjects::widget`

- **Descrizione:** componente per la gestione dei widget per permettere all'utente di interagire con il mondo di gioco attraverso menu e finestre.

- **Padre:** `graphicobjects`
- **Interazioni con altri componenti**
 - `menufactory`: componente per la gestione dei vari menu che possono essere visualizzati durante le interazioni con il mondo di gioco.
 - `graphicobject`: componente per la gestione dell'interfaccia per qualsiasi oggetto che verrà visualizzato nell'area di gioco.
 - `shape`: componente per la gestione delle forme degli oggetti grafici.

4.27.2 Classi

4.27.2.1 `sgad::clienttier::view::graphicobjects::widget::Widget`

- **Descrizione:** classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.
- **Utilizzo:** viene utilizzata per fornire dei metodi comuni a qualsiasi tipo di widget che possono essere rappresentati. Le classi che ereditano da tale classe si occuperanno di ridefinire i metodi astratti. Mantiene un riferimento ad un oggetto di tipo `Shape` per rappresentare la forma del widget.
- **Classi ereditate**
 - `GraphicObject`
- **Sottoclassi**
 - `ButtonWidget`
 - `FrameWidget`
 - `ImageWidget`
 - `TextWidget`
- **Relazioni con altre classi**
 - ← `BuildConstruction`: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← `ChangePasswordAction`: classe per permettere di modificare la password di un account.
 - ← `CloseContextualMenu`: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - ← `DeleteAccountAction`: la classe viene utilizzata per l'eliminazione di un account utente.
 - ← `ReceiveStealResources`: classe per la ricezione delle risorse rubate
 - ← `ShowAccountDeletedMenu`: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.

- ← **ShowAccountManagerMenu**: classe per la creazione di un menu per la gestione dell'account.
- ← **ShowAnotherUserMenu**: classe per la visualizzazione del menu per ritornare al proprio villaggio una volta che si è nel villaggio di un altro utente.
- ← **ShowAttackMenu**: classe per la visualizzazione del menu per l'attacco.
- ← **ShowAttackResultMenu**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
- ← **ShowBuildConstructionMenu**: classe per la richiesta della conferma della costruzione.
- ← **ShowBuildingContextualMenu**: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
- ← **ShowDemolishMenu**: classe che visualizza il menu per la demolizione di un edificio.
- ← **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
- ← **ShowInteractionMenu**: classe per la gestione della visualizzazione di un menu per entrare in modalità costruzione o per la visualizzazione della lista degli utenti.
- ← **ShowLogMenu**: classe per la visualizzazione di un menu di gestione account
- ← **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
- ← **ShowPasswordChangedMenu**: classe per la visualizzazione di un menu di conferma del cambiamento della password.
- ← **ShowResourceMenu**: classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
- ← **ShowTileContextualMenu**: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← **ShowUnitSelectionMenu**: classe per la visualizzazione di un menu per le unità addestrabili.
- ← **ShowUpgradeMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
- ← **ShowUserListMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di visualizzare la lista degli utenti con i quali può interagire.
- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- ← **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.

- ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
- ← **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
- ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- ← **FrameWidget**: classe per la gestione di un generico frame nell'interfaccia grafica.
- → **Action**: interfaccia per le possibili azioni che possono essere eseguite sull'interfaccia grafica o sul modello dei dati.
- → **Shape**: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.27.2.2 sgad::clienttier::view::graphicobjects::widget::ButtonWidget

- **Descrizione**: classe per la gestione di un generico bottone nell'interfaccia grafica.
- **Utilizzo**: viene utilizzata per rappresentare un bottone fornendo metodi comuni a qualsiasi possibile bottone che viene costruito. Le sottoclassi di **MenuFactory** determineranno quali saranno le azioni da eseguire al verificarsi di determinati eventi.
- **Classi ereditate**
 - **Widget**
- **Relazioni con altre classi**
 - ← **AccountManagerMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← **AnotherUserMenuFactory**: classe per la costruzione di un menu per il ritorno al villaggio dell'utente.
 - ← **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
 - ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
 - ← **InteractionMenuFactory**: classe per la creazione di un menu per interagire con l'intero mondo di gioco.
 - ← **LogMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.

- ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
- ← **UserListMenuFactory**: classe per la visualizzazione della lista di altri giocatori connessi.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **Shape**: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.27.2.3 sgad::clienttier::view::graphicobjects::widget::FrameWidget

- **Descrizione:** classe per la gestione di un generico frame nell'interfaccia grafica.
- **Utilizzo:** viene utilizzata per rappresentare un frame fornendo metodi comuni a qualsiasi possibile frame che viene costruito. Le sottoclassi di **MenuFactory** determineranno quali saranno le azioni da eseguire al verificarsi di determinati eventi. Stabiliranno inoltre quali altri oggetti di tipo **Widget** il frame dovrà contenere.
- **Classi ereditate**
 - **Widget**
- **Relazioni con altre classi**
 - ← **BuildConstruction**: classe per la gestione della costruzione di un nuovo edificio nel villaggio dell'utente.
 - ← **ChangePasswordAction**: classe per permettere di modificare la password di un account.
 - ← **CloseContextualMenu**: classe per la gestione della chiusura del menu contestuale visualizzato alla pressione del pulsante destro del mouse.
 - ← **DeleteAccountAction**: la classe viene utilizzata per l'eliminazione di un account utente.
 - ← **ReceiveStealResources**: classe per la ricezione delle risorse rubate
 - ← **ShowAccountDeletedMenu**: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.
 - ← **ShowAccountManagerMenu**: classe per la creazione di un menu per la gestione dell'account.
 - ← **ShowAnotherUserMenu**: classe per la visualizzazione del menu per ritornare al proprio villaggio una volta che si è nel villaggio di un altro utente.
 - ← **ShowAttackMenu**: classe per la visualizzazione del menu per l'attacco.
 - ← **ShowAttackResultMenu**: classe per la visualizzazione del menu che visualizza il risultato dell'attacco.
 - ← **ShowBuildConstructionMenu**: classe per la richiesta della conferma della costruzione.
 - ← **ShowBuildingContextualMenu**: classe per la gestione della visualizzazione di un menu contestuale per le possibili azioni che si possono eseguire su un edificio.
 - ← **ShowDemolishMenu**: classe che visualizza il menu per la demolizione di un edificio.
 - ← **ShowDismissUnitMenu**: classe per la gestione della visualizzazione di un menu per il congedo di unità.
 - ← **ShowInteractionMenu**: classe per la gestione della visualizzazione di un menu per entrare in modalità costruzione o per la visualizzazione della lista degli utenti.

- ← **ShowLogMenu**: classe per la visualizzazione di un menu di gestione account
- ← **ShowOperationFailureMenu**: classe per la visualizzazione di un menu al fallimento di un'operazione.
- ← **ShowPasswordChangedMenu**: classe per la visualizzazione di un menu di conferma del cambiamento della password.
- ← **ShowResourceMenu**: classe per la gestione della visualizzazione di un menu per selezionare le risorse da donare ad un altro utente.
- ← **ShowTileContextualMenu**: classe per la gestione della visualizzazione della barra riepilogativa delle risorse e delle unità possedute dall'utente.
- ← **ShowTrainUnitMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di addestrare le unità.
- ← **ShowUnitSelectionMenu**: classe per la visualizzazione di un menu per le unità addestrabili.
- ← **ShowUpgradeMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di confermare il miglioramento di un edificio.
- ← **ShowUserListMenu**: classe per la gestione della visualizzazione di un menu che permette all'utente di visualizzare la lista degli utenti con i quali può interagire.
- ← **StealResources**: classe per la gestione del saccheggio di un edificio produttivo di un altro utente.
- ← **UpgradeBuilding**: classe per la gestione del miglioramento di un edificio desiderato dall'utente.
- ← **AnotherUserMenuFactory**: classe per la costruzione di un menu per il ritorno al villaggio dell'utente.
- ← **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
- ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
- ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
- ← **InteractionMenuFactory**: classe per la creazione di un menu per interagire con l'intero mondo di gioco.
- ← **LogMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
- ← **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
- ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
- ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
- ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.

- ← **UserListMenuFactory**: classe per la visualizzazione della lista di altri giocatori connessi.
- ← **Context**: classe per la gestione dell'area del mondo di gioco. La classe permetterà di trasferire gli eventi verificatisi durante l'interazione con il mondo di gioco ai vari oggetti grafici.
- ← **GraphicObjectCollection**: classe per organizzare l'insieme degli oggetti grafici da rappresentare nel mondo di gioco.
- → **Bound**: classe per la gestione dei limiti della porzione dell'area di gioco effettivamente visualizzata.
- → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
- → **Shape**: classe per la gestione di un insieme di punti rappresentanti un poligono.
- → **Widget**: classe astratta per i possibili widget che possono essere rappresentati per interagire con il mondo di gioco.

4.27.2.4 sgad::clienttier::view::graphicobjects::widget::ImageWidget

- **Descrizione:** classe per la gestione di una generica immagine nell'interfaccia grafica.
- **Utilizzo:** viene utilizzata per rappresentare un'immagine fornendo metodi comuni a qualsiasi possibile immagine che viene costruita. Le sottoclassi di **MenuFactory** determineranno quali saranno le azioni da eseguire al verificarsi di determinati eventi e quale effettiva immagine dovrà essere visualizzata.
- **Classi ereditate**
 - **Widget**
- **Relazioni con altre classi**
 - ← **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
 - ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
 - ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
 - ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - ← **UserListMenuFactory**: classe per la visualizzazione della lista di altri giocatori connessi.
 - → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
 - → **Shape**: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.27.2.5 sgad::clienttier::view::graphicobjects::widget::TextWidget

- **Descrizione:** classe per la gestione di un generico testo nell'interfaccia grafica.
- **Utilizzo:** viene utilizzata per rappresentare un testo fornendo metodi comuni a qualsiasi possibile testo che viene costruito. Le sottoclassi di **MenuFactory** determineranno quale sarà l'effettivo testo da visualizzare. Stabiliranno inoltre se tale testo dovrà essere collegato ad oggetto di tipo **Observable** per mantenere aggiornato il testo.
- **Classi ereditate**
 - **Observer**
 - **Widget**
- **Relazioni con altre classi**
 - ← **AccountManagerMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← **AttackResultMenuFactory**: classe per la creazione di un menu che permetta all'utente di visualizzare l'esito dello scontro.
 - ← **BuildingContextualMenuFactory**: classe per la creazione di un menu di tipo contestuale relativo ad una costruzione.
 - ← **ConfirmMenuFactory**: classe per la creazione di un menu che permetta di confermare una generica azione.
 - ← **LogMenuFactory**: classe per la costruzione di un menu che permetta all'utente di modificare il proprio account.
 - ← **NotifyMenuFactory**: classe per la costruzione di un menu per la visualizzazione di una notifica.
 - ← **ResourceMenuFactory**: classe per la creazione di un menu che visualizzi le risorse e le unità possedute dall'utente.
 - ← **TileContextualMenuFactory**: classe per la creazione di un menu che visualizzi gli edifici costruibili in una casella.
 - ← **UnitSelectionMenu**: classe per la creazione di un menu che visualizzi le unità producibili in un edificio.
 - → **Observable**: classe astratta per la gestione di oggetti osservabili. Essi, al cambiamento di stato, notificheranno tutti gli oggetti osservatori che richiedono di conoscere tali cambiamenti.
 - → **Point2D**: classe per la rappresentazione di un punto avente due coordinate.
 - → **Shape**: classe per la gestione di un insieme di punti rappresentanti un poligono.

4.28 Componente sgad::servertier

4.28.1 Informazioni sul package

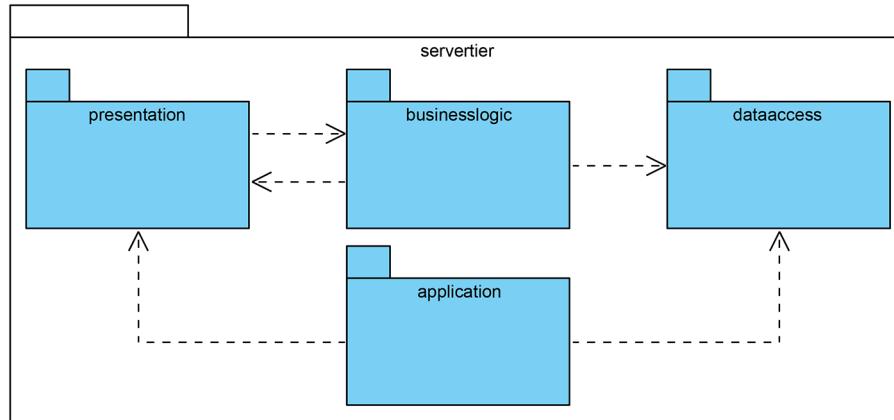


Figura 38: Diagramma della componente `sgad::servertier`

- **Descrizione:** componente globale per il back end del prodotto. Le relazioni tra le componenti `presentation`, `businesslogic` e `dataaccess` rappresentano delle interazioni tipiche. Esse intercorrono tra le componenti del design pattern Three-Tier.
- **Padre:** `sgad`
- **Interazioni con altri componenti**
 - `clienttier`: componente globale per il front end del prodotto. Nel Three-Tier del sistema rappresenta il livello Client Tier. Le relazioni tra le sottocomponenti `model`, `view` e `controller` rappresentano le interazioni tipiche che intercorrono tra le componenti del design pattern MVC. Esso si occupa di visualizzare i dati all'utente e di inoltrare le richieste al back end.
- **Package contenuti**
 - `application`: componente che si occupa del `bootstrap[g]` dell'applicazione.
 - `businesslogic`: componente corrispondente al livello business logic dell'architettura Three-Tier. Essa si occupa di effettuare le operazioni richieste dagli utenti. Nel caso le operazioni siano relative a interazioni tra più utenti, comunicherà con la componente `presentation` per instradare correttamente la richiesta verso gli altri client coinvolti.
 - `dataaccess`: componente per la gestione dell'accesso ad database e della creazione degli oggetti delle componenti `shareddata` e `userdata`.
 - `presentation`: componente per la gestione della comunicazione con i client. Rappresenta il livello superiore e visibile dall'esterno dell'architettura Three-Tier. Si occupa di smistare le richieste in arrivo alle corrette componenti della logica.

4.29 Componente sgad::servertier::application

4.29.1 Informazioni sul package

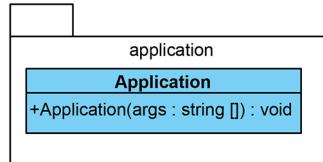


Figura 39: Diagramma della componente `sgad::servertier::application`

- **Descrizione:** componente che si occupa del bootstrap dell'applicazione.
- **Padre:** `servertier`
- **Interazioni con altri componenti**
 - `databaseaccess`: componente corrispondente al livello data access dell'architettura Three-Tier. Essa si occupa del mantenimento dei dati e della comunicazione con il database.
 - `presentation`: componente per la gestione della comunicazione con i client. Rappresenta il livello superiore e visibile dall'esterno dell'architettura Three-Tier. Si occupa di smistare le richieste in arrivo alle corrette componenti della logica.
 - `cluster`: componente per la gestione e la partecipazione del singolo server a $cluster_{|g|}$ di server.

4.29.2 Classi

4.29.2.1 `sgad::servertier::application::Application`

- **Descrizione:** classe per la gestione del bootstrap dell'applicativo lato server.
- **Utilizzo:** viene utilizzata per la creazione, l'avvio e il caricamento dei vari moduli che compongono l'applicativo di back end.
- **Relazioni con altre classi**
 - → `DataBaseManager`: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
 - → `ClusterListener`: classe che gestisce l'interazione del server con gli altri server del cluster.
 - → `ResponderActor`: classe per la gestione degli $end\ point_{|g|}$ per le richieste HTTP inviate dal client.
 - → `PageFactory`: classe che gestisce le pagine HTML da inviare ai client.

- → `STimeout`: classe che rappresenta i timeout delle richieste all'interno dell'applicazione.
- → `IsUserActorAliveRequester`: attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.
- → `PublisherActor`: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto `usermanager`.
- → `UserActor`: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.30 Componente `sgad::servertier::businesslogic`

4.30.1 Informazioni sul package

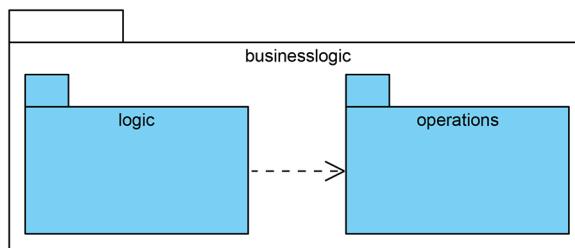


Figura 40: Diagramma della componente `sgad::servertier::businesslogic`

- **Descrizione:** componente corrispondente al livello business logic dell'architettura Three-Tier. Essa si occupa di effettuare le operazioni richieste dagli utenti. Nel caso le operazioni siano relative a interazioni tra più utenti, comunicherà con la componente `presentation` per instradare correttamente la richiesta verso gli altri client coinvolti.
- **Padre:** `servertier`
- **Interazioni con altri componenti**
 - `databaseaccess`: componente corrispondente al livello data access dell'architettura Three-Tier. Essa si occupa del mantenimento dei dati e della comunicazione con il database.
 - `presentation`: componente per la gestione della comunicazione con i client. Rappresenta il livello superiore e visibile dall'esterno dell'architettura Three-Tier. Si occupa di smistare le richieste in arrivo alle corrette componenti della logica.
- **Package contenuti**
 - `logic`: componente che gestisce il controllo delle operazioni richieste dal client.
 - `operations`: componente per la gestione dell'insieme delle possibili operazioni che un utente può effettuare.

4.31 Componente sgad::servertier::businesslogic::logic

4.31.1 Informazioni sul package

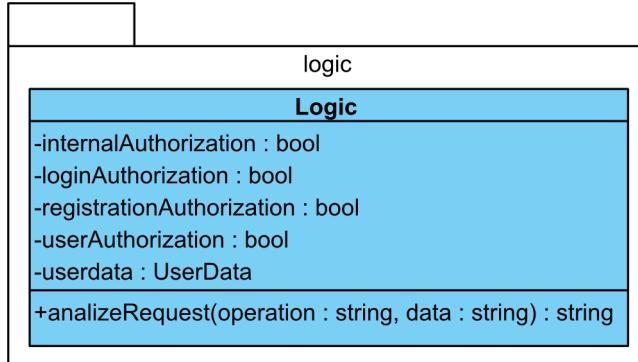


Figura 41: Diagramma della componente sgad::servertier::businesslogic::logic

- **Descrizione:** componente che gestisce il controllo delle operazioni richieste dal client.
- **Padre:** businesslogic
- **Interazioni con altri componenti**
 - **operations:** componente per la gestione dell'insieme delle possibili operazioni che un utente può effettuare.
 - **databasemanager:** componente per la gestione dell'interazione tra il server e il database, esegue azioni come il salvataggio e il caricamento dei dati. Esso è l'unico punto di accesso con il database.
 - **userManager:** componente per la gestione degli attori relativi alla gestione delle richieste generate dai client. Si occupa di ricercare in tutto il cluster di server l'attore associato alla richiesta.

4.31.2 Classi

4.31.2.1 sgad::servertier::businesslogic::logic::Logic

- **Descrizione:** classe per la gestione dell'operazione richiesta dall'utente. Essa si occupa di instradare la richiesta tramite l'invocazione dell'operazione giusta.
- **Utilizzo:** viene utilizzata per la scelta della giusta operazione da eseguire come risposta alla richiesta del client.
- **Relazioni con altre classi**
 - ← LoginActor: classe per la gestione delle richieste di login.

- ← **RegistrationActor**: classe per la gestione delle richieste di registrazione.
- ← **UserActor**: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.
- → **Operation**: classe astratta per l'esecuzione di un'operazione.
- → **OperationFactory**: classe per la gestione delle classi che realizzano **Operation** concretamente.
- → **UserData**: classe per la gestione dei dati di un utente.

4.32 Componente sgad::servertier::businesslogic::operations

4.32.1 Informazioni sul package

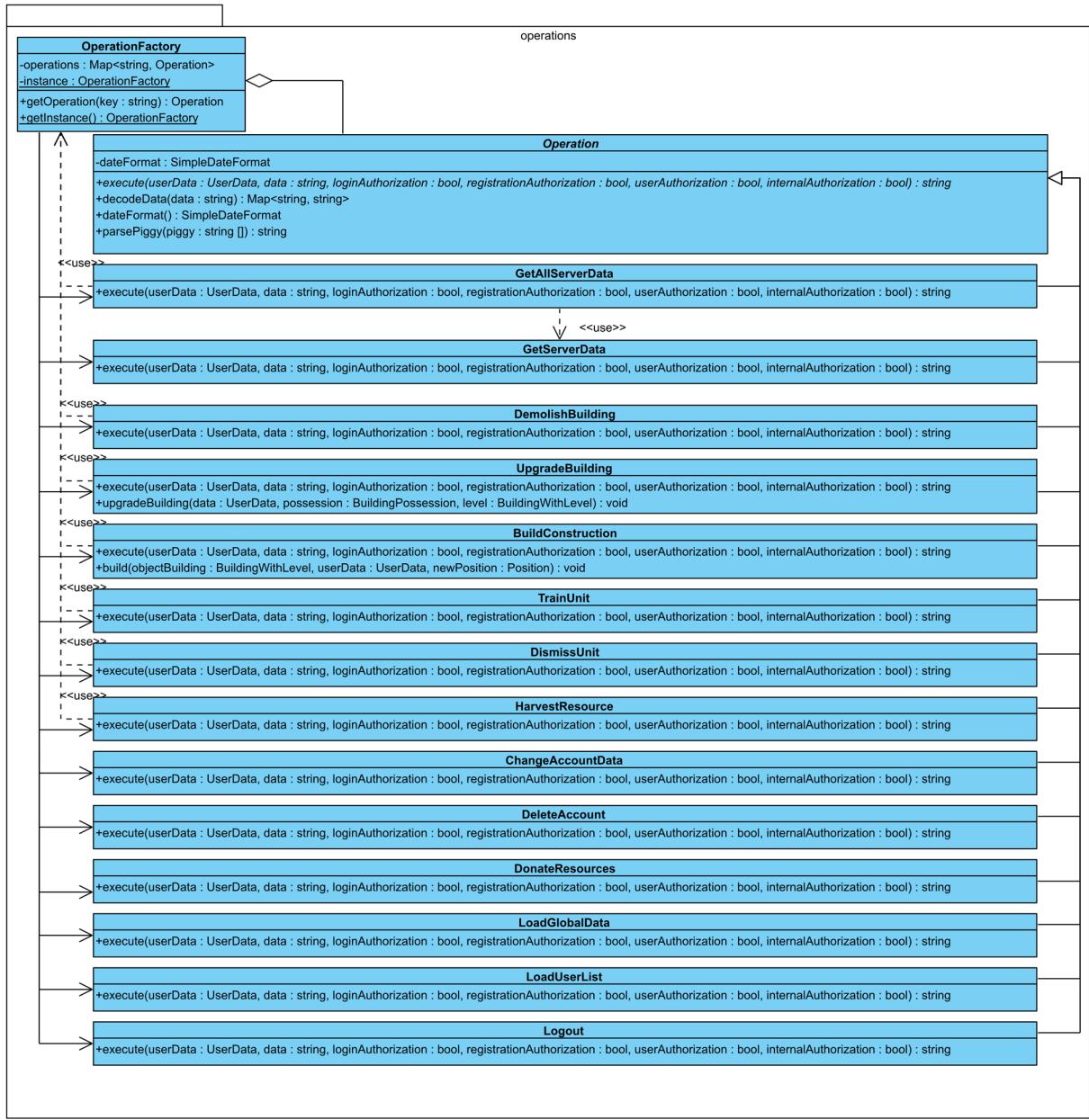


Figura 42: Diagramma della componente `sgad::servertier::businesslogic::operations`

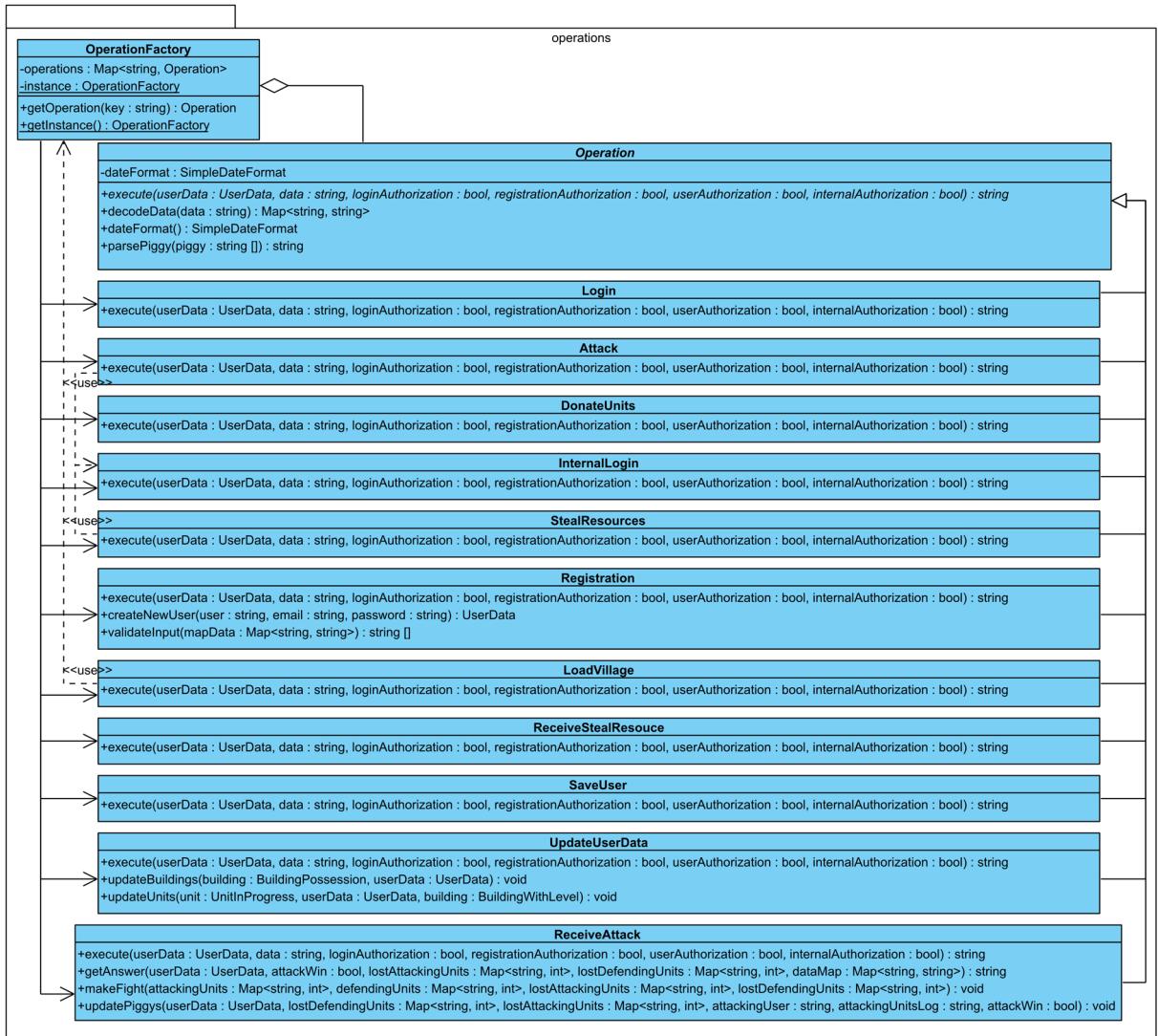


Figura 43: Diagramma della componente `sgad::servertier::businesslogic::operations`

- **Descrizione:** componente per la gestione dell'insieme delle possibili operazioni che un utente può effettuare.
- **Padre:** `businesslogic`
- **Interazioni con altri componenti**
 - `logic`: componente che gestisce il controllo delle operazioni richieste dal client.
 - `data`: componente che gestisce l'insieme dei dati necessari per mantenere memorizzata, sul server, la sessione dell'utente.

4.32.2 Classi

4.32.2.1 sgad::servertier::businesslogic::operations::Operation

- **Descrizione:** classe astratta per l'esecuzione di un'operazione.
- **Utilizzo:** viene utilizzata mediante un riferimento dalla classe Logic per l'esecuzione di un'operazione sul model.
- **Sottoclassi**
 - Attack
 - BuildConstruction
 - ChangeAccountData
 - DeleteAccount
 - DemolishBuilding
 - DismissUnit
 - DonateResources
 - DonateUnits
 - GetAllServerData
 - GetServerData
 - HarvestResource
 - InternalLogin
 - LoadGlobalData
 - LoadVillage
 - LoadUserList
 - Login
 - Logout
 - ReceiveAttack
 - ReceiveStealResource
 - Registration
 - SaveUser
 - StealResource
 - TrainUnit
 - UpdateUserData
 - UpgradeBuilding
- **Relazioni con altre classi**
 - ← Logic: classe per la gestione dell'operazione richiesta dall'utente. Essa si occupa di instradare la richiesta tramite l'invocazione dell'operazione giusta.
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → UserData: classe per la gestione dei dati di un utente.

4.32.2.2 sgad::servertier::businesslogic::operations::Attack

- **Descrizione:** classe che rappresenta l'operazione di lancio attacco ad un altro utente.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di attacco. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite OperationFactory.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → InternalLogin: classe che rappresenta l'operazione di login.
 - → OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - → Unit: classe per la gestione delle informazioni di una unità.
 - → UnitPossession: classe per la rappresentazione di un'unità con associata una quantità.
 - → UserData: classe per la gestione dei dati di un utente.
 - → InternalRequester: classe di gestione della richiesta di user ad autorizzazione interna incapsulandone l'algoritmo.

4.32.2.3 sgad::servertier::businesslogic::operations::BuildConstruction

- **Descrizione:** classe che rappresenta l'operazione di costruzione di un edificio.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di costruzione di un edificio. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite OperationFactory.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.

- → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- → **DataFactory**: classe che ritorna un’istanza delle componenti di gioco.
- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **Position**: classe per la gestione di una posizione sulla griglia del villaggio.
- → **UserData**: classe per la gestione dei dati di un utente.

4.32.2.4 sgad::servertier::businesslogic::operations::ChangeAccountData

- **Descrizione**: classe che rappresenta l’operazione di modifica dei dati dell’account utente.
- **Utilizzo**: la classe viene utilizzata per la modifica dei dati dell’account di un utente.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← **OperationFactory**: classe per la gestione delle classi che realizzano Operation concretamente.
 - → **AuthenticationData**: classe per la gestione delle informazioni personali e di accesso di un utente.
 - → **UserData**: classe per la gestione dei dati di un utente.
 - → **DataBaseManager**: classe per la gestione del database. Essa astrae l’utilizzo di un particolare database al resto dell’applicativo.

4.32.2.5 sgad::servertier::businesslogic::operations::DeleteAccount

- **Descrizione**: classe che rappresenta l’operazione di eliminazione di un account utente.
- **Utilizzo**: la classe viene utilizzata per l’eliminazione di un account utente.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← **OperationFactory**: classe per la gestione delle classi che realizzano Operation concretamente.
 - → **UserData**: classe per la gestione dei dati di un utente.
 - → **DataBaseManager**: classe per la gestione del database. Essa astrae l’utilizzo di un particolare database al resto dell’applicativo.

4.32.2.6 sgad::servertier::businesslogic::operations::DemolishBuilding

- **Descrizione:** classe che rappresenta la demolizione di un edificio.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di demolizione di un edificio. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite OperationFactory.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → UserData: classe per la gestione dei dati di un utente.

4.32.2.7 sgad::servertier::businesslogic::operations::DismissUnit

- **Descrizione:** classe che rappresenta l'operazione di congedo di unità.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di congedo di unità. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite OperationFactory.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → UserData: classe per la gestione dei dati di un utente.

4.32.2.8 sgad::servertier::businesslogic::operations::DonateResources

- **Descrizione:** classe che rappresenta l'operazione di dono di risorse ad un altro utente.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di dono di risorse ad un altro utente.

- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.

4.32.2.9 sgad::servertier::businesslogic::operations::DonateUnits

- **Descrizione:** classe che rappresenta il dono di unità ad un altro utente.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di dono di unità ad un altro utente.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.

4.32.2.10 sgad::servertier::businesslogic::operations::GetAllServerData

- **Descrizione:** classe per la fornitura delle informazioni sullo stato di tutto il cluster.
- **Utilizzo:** viene utilizzata per fornire al client tutte le informazioni di stato sul cluster.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → GetServerData: classe per la fornitura dei dati di stato sul server.
 - → OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → PublisherActor: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto usermanager.

4.32.2.11 sgad::servtier::businesslogic::operations::GetServerData

- **Descrizione:** classe per la fornitura dei dati di stato sul server.
- **Utilizzo:** viene utilizzata per fornire i dati di stato sul server.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← GetAllServerData: classe per la fornitura delle informazioni sullo stato di tutto il cluster.
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.

4.32.2.12 sgad::servtier::businesslogic::operations::HarvestResource

- **Descrizione:** classe per la gestione dell'operazione di raccolta risorse da un edificio.
- **Utilizzo:** viene utilizzata per la raccolta delle risorse prodotte da un edificio. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite OperationFactory.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → UserData: classe per la gestione dei dati di un utente.

4.32.2.13 sgad::servtier::businesslogic::operations::InternalLogin

- **Descrizione:** classe che rappresenta l'operazione di login.
- **Utilizzo:** la classe viene utilizzata per effettuare login interni allo scopo di permettere ad altri utente di comunicare con utenti non loggati in quel momento.
- **Classi ereditate**
 - Operation

- **Relazioni con altre classi**

- ← **Attack**: classe che rappresenta l'operazione di lancio attacco ad un altro utente.
- ← **OperationFactory**: classe per la gestione delle classi che realizzano **Operation** concretamente.
- ← **StealResource**: classe che rappresenta l'operazione di saccheggio di risorse da un edificio di un altro utente.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **DataBaseManager**: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
- → **IsUserActorAliveRequester**: attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.
- → **UserActor**: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.32.2.14 sgad::servertier::businesslogic::operations::LoadGlobalData

- **Descrizione**: classe per la gestione del caricamento da parte del client dei dati globali di gioco.
- **Utilizzo**: viene utilizzata per dare al client le informazioni sui dati generali di gioco.
- **Classi ereditate**

- **Operation**

- **Relazioni con altre classi**

- ← **OperationFactory**: classe per la gestione delle classi che realizzano **Operation** concretamente.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **SharedDataDAO**: classe per la trasformazione dei dati di gioco generali contenuti nel DataFactory in formato JSON.

4.32.2.15 sgad::servertier::businesslogic::operations::LoadVillage

- **Descrizione**: classe per la gestione dell'operazione di caricamento del villaggio richiesta del client.
- **Utilizzo**: viene utilizzata per dare al client i dati del villaggio dell'utente. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite OperationFactory.
- **Classi ereditate**

- Operation
- Relazioni con altre classi
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → UserDataDAO: classe per l'interazione con il database di una classe di tipo UserData.

4.32.2.16 sgad::servtier::businesslogic::operations::LoadUserList

- Descrizione: classe per la gestione dell'operazione di scelta degli utenti con i quali un utente può interagire.
- Utilizzo: viene utilizzata per dire al client quali sono gli utenti con i quali può interagire.
- Classi ereditate
 - Operation
- Relazioni con altre classi
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.
 - → UserData: classe per la gestione dei dati di un utente.
 - → DataBaseManager: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.

4.32.2.17 sgad::servtier::businesslogic::operations::Login

- Descrizione: classe che rappresenta l'operazione di login.
- Utilizzo: viene utilizzata per l'esecuzione dell'operazione di login richiesta da un client.
- Classi ereditate
 - Operation
- Relazioni con altre classi
 - ← OperationFactory: classe per la gestione delle classi che realizzano Operation concretamente.

- → **AuthenticationData**: classe per la gestione delle informazioni personali e di accesso di un utente.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **DataBaseManager**: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
- → **PageFactory**: classe che gestisce le pagine HTML da inviare ai client.
- → **IsUserActorAliveRequester**: attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.
- → **UserActor**: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.32.2.18 sgad::servertier::businesslogic::operations::Logout

- **Descrizione**: classe che rappresenta l'operazione di logout.
- **Utilizzo**: viene utilizzata per l'esecuzione dell'operazione di logout.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**
 - ← **ShowAccountDeletedMenu**: classe per la visualizzazione di un menu indicante l'esito dell'operazione di eliminazione dell'account.
 - ← **OperationFactory**: classe per la gestione delle classi che realizzano Operation concretamente.
 - → **UserData**: classe per la gestione dei dati di un utente.
 - → **UserActor**: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.32.2.19 sgad::servertier::businesslogic::operations::ReceiveAttack

- **Descrizione**: classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglie.
- **Utilizzo**: classe utilizzata per la gestione della ricezione dell'attacco e per la computazione del risultato delle battaglie. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite OperationFactory.
- **Classi ereditate**
 - Operation
- **Relazioni con altre classi**

- ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
- → `Unit`: classe per la gestione delle informazioni di una unità.
- → `BuildingPossession`: classe per la gestione di un edificio posseduto da un utente.
- → `UnitPossession`: classe per la rappresentazione di un'unità con associata una quantità.
- → `UserData`: classe per la gestione dei dati di un utente.
- → `BuildingPossessionDAO`: classe per l'interazione con il database di una classe di tipo `BuildingPossession`.

4.32.2.20 sgad::servtier::businesslogic::operations::ReceiveStealResource

- **Descrizione:** classe che rappresenta l'operazione interna di ricezione di un attacco e computo dell'esito dello scontro.
- **Utilizzo:** classe utilizzata per rappresentare l'operazione interna di ricezione di un attacco e computazione dell'esito dello scontro. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite `OperationFactory`.
- **Classi ereditate**
 - `Operation`
- **Relazioni con altre classi**
 - ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
 - → `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
 - → `UserData`: classe per la gestione dei dati di un utente.

4.32.2.21 sgad::servtier::businesslogic::operations::Registration

- **Descrizione:** classe per la gestione dell'operazione di registrazione.
- **Utilizzo:** viene utilizzata per registrare nel database un nuovo utente.
- **Classi ereditate**
 - `Operation`

- **Relazioni con altre classi**

- ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `DataFactory`: classe che ritorna un’istanza delle componenti di gioco.
- → `AuthenticationData`: classe per la gestione delle informazioni personali e di accesso di un utente.
- → `BuildingPossession`: classe per la gestione di un edificio posseduto da un utente.
- → `Position`: classe per la gestione di una posizione sulla griglia del villaggio.
- → `DataBaseManager`: classe per la gestione del database. Essa astrae l’utilizzo di un particolare database al resto dell’applicativo.
- → `PageFactory`: classe che gestisce le pagine HTML da inviare ai client.

4.32.2.22 sgad::servertier::businesslogic::operations::SaveUser

- **Descrizione:** classe che rappresenta l’operazione di salvataggio sul database i dati aggiornati dell’utente in input.
- **Utilizzo:** viene utilizzata per salvare i dati dell’utente aggiornati nel database.
- **Classi ereditate**
 - `Operation`
- **Relazioni con altre classi**
 - ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
 - → `UserData`: classe per la gestione dei dati di un utente.
 - → `DataBaseManager`: classe per la gestione del database. Essa astrae l’utilizzo di un particolare database al resto dell’applicativo.

4.32.2.23 sgad::servertier::businesslogic::operations::StealResource

- **Descrizione:** classe che rappresenta l’operazione di saccheggio di risorse da un edificio di un altro utente.
- **Utilizzo:** viene utilizzata per l’esecuzione dell’operazione di saccheggio di risorse da un edificio di un altro utente. Questa operazione, per poter portare a termine il proprio compito, utilizza un’altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite `OperationFactory`.
- **Classi ereditate**
 - `Operation`

- **Relazioni con altre classi**

- ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `InternalLogin`: classe che rappresenta l'operazione di login.
- → `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `UserData`: classe per la gestione dei dati di un utente.
- → `InternalRequester`: classe di gestione della richiesta di user ad autorizzazione interna incapsulandone l'algoritmo.

4.32.2.24 sgad::servertier::businesslogic::operations::TrainUnit

- **Descrizione:** classe che rappresenta l'operazione di arruolamento di unità.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di arruolamento di unità. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite `OperationFactory`.

- **Classi ereditate**

- `Operation`

- **Relazioni con altre classi**

- ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
- → `UnitInProgress`: classe per la gestione della coda di costruzione di unità di un particolare edificio.
- → `UserData`: classe per la gestione dei dati di un utente.

4.32.2.25 sgad::servertier::businesslogic::operations::UpdateUserData

- **Descrizione:** mantiene aggiornati i dati dell'utente in merito alle costruzioni e alle unità in coda di produzione.
- **Utilizzo:** classe utilizzata per aggiornare i dati dell'utente riguardanti le costruzioni e le unità in coda di produzione.

- **Classi ereditate**

- `Operation`

- **Relazioni con altre classi**

- ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `BuildingPossession`: classe per la gestione di un edificio posseduto da un utente.
- → `UnitInProgress`: classe per la gestione della coda di costruzione di unità di un particolare edificio.
- → `UserData`: classe per la gestione dei dati di un utente.

4.32.2.26 sgad::servtier::businesslogic::operations::UpgradeBuilding

- **Descrizione:** classe che rappresenta l'operazione di avanzamento di livello di un edificio posseduto dall'utente.
- **Utilizzo:** viene utilizzata per l'esecuzione dell'operazione di avanzamento di livello di un edificio posseduto dall'utente. Questa operazione, per poter portare a termine il proprio compito, utilizza un'altra operazione ad autorizzazione interna. Questa viene recuperata ed eseguita tramite `OperationFactory`.

- **Classi ereditate**

- `Operation`

- **Relazioni con altre classi**

- ← `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `OperationFactory`: classe per la gestione delle classi che realizzano `Operation` concretamente.
- → `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
- → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
- → `BuildingPossession`: classe per la gestione di un edificio posseduto da un utente.
- → `UserData`: classe per la gestione dei dati di un utente.

4.32.2.27 sgad::servtier::businesslogic::operations::OperationFactory

- **Descrizione:** classe per la gestione delle classi che realizzano `Operation` concretamente.
 - **Utilizzo:** viene utilizzato per ottenere un riferimento all'operazione concreta.
-
- **Relazioni con altre classi**

- ← **Logic**: classe per la gestione dell'operazione richiesta dall'utente. Essa si occupa di instradare la richiesta tramite l'invocazione dell'operazione giusta.
- ← **Attack**: classe che rappresenta l'operazione di lancio attacco ad un altro utente.
- ← **BuildConstruction**: classe che rappresenta l'operazione di costruzione di un edificio.
- ← **DemolishBuilding**: classe che rappresenta la demolizione di un edificio.
- ← **DismissUnit**: classe che rappresenta l'operazione di congedo di unità.
- ← **GetAllServerData**: classe per la fornitura delle informazioni sullo stato di tutto il cluster.
- ← **HarvestResource**: classe per la gestione dell'operazione di raccolta risorse da un edificio.
- ← **LoadVillage**: classe per la gestione dell'operazione di caricamento del villaggio richiesta del client.
- ← **ReceiveAttack**: classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglie.
- ← **ReceiveStealResource**: classe che rappresenta l'operazione interna di ricezione di un attacco e computo dell'esito dello scontro.
- ← **StealResource**: classe che rappresenta l'operazione di saccheggio di risorse da un edificio di un altro utente.
- ← **TrainUnit**: classe che rappresenta l'operazione di arruolamento di unità.
- ← **UpgradeBuilding**: classe che rappresenta l'operazione di avanzamento di livello di un edificio posseduto dall'utente.
- → **Attack**: classe che rappresenta l'operazione di lancio attacco ad un altro utente.
- → **BuildConstruction**: classe che rappresenta l'operazione di costruzione di un edificio.
- → **ChangeAccountData**: classe che rappresenta l'operazione di modifica dei dati dell'account utente.
- → **DeleteAccount**: classe che rappresenta l'operazione di eliminazione di un account utente.
- → **DemolishBuilding**: classe che rappresenta la demolizione di un edificio.
- → **DismissUnit**: classe che rappresenta l'operazione di congedo di unità.
- → **DonateResources**: classe che rappresenta l'operazione di dono di risorse ad un altro utente.
- → **DonateUnits**: classe che rappresenta il dono di unità ad un altro utente.
- → **GetAllServerData**: classe per la fornitura delle informazioni sullo stato di tutto il cluster.
- → **GetServerData**: classe per la fornitura dei dati di stato sul server.
- → **HarvestResource**: classe per la gestione dell'operazione di raccolta risorse da un edificio.

- → `InternalLogin`: classe che rappresenta l'operazione di login.
- → `LoadGlobalData`: classe per la gestione del caricamento da parte del client dei dati globali di gioco.
- → `LoadVillage`: classe per la gestione dell'operazione di caricamento del villaggio richiesta del client.
- → `LoadUserList`: classe per la gestione dell'operazione di scelta degli utenti con i quali un utente può interagire.
- → `Login`: classe che rappresenta l'operazione di login.
- → `Logout`: classe che rappresenta l'operazione di logout.
- → `Operation`: classe astratta per l'esecuzione di un'operazione.
- → `ReceiveAttack`: classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglia.
- → `ReceiveStealResource`: classe che rappresenta l'operazione interna di ricezione di un attacco e computo dell'esito dello scontro.
- → `Registration`: classe per la gestione dell'operazione di registrazione.
- → `SaveUser`: classe che rappresenta l'operazione di salvataggio sul database i dati aggiornati dell'utente in input.
- → `StealResource`: classe che rappresenta l'operazione di saccheggio di risorse da un edificio di un altro utente.
- → `TrainUnit`: classe che rappresenta l'operazione di arruolamento di unità.
- → `UpdateUserData`: mantiene aggiornati i dati dell'utente in merito alle costruzioni e alle unità in coda di produzione.
- → `UpgradeBuilding`: classe che rappresenta l'operazione di avanzamento di livello di un edificio posseduto dall'utente.

4.33 Componente `sgad::servertier::dataaccess`

4.33.1 Informazioni sul package

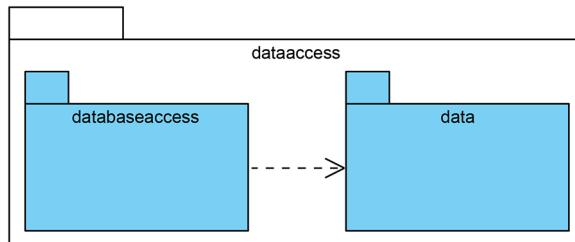


Figura 44: Diagramma della componente `sgad::servertier::dataaccess`

- **Descrizione:** componente per la gestione dell'accesso ad database e della creazione degli oggetti delle componenti shareddata e userdata.

- Padre: **servertier**
- Package contenuti
 - **data**: componente che gestisce l'insieme dei dati necessari per mantenere memorizzata, sul server, la sessione dell'utente.
 - **databaseaccess**: componente corrispondente al livello data access dell'architettura Three-Tier. Essa si occupa del mantenimento dei dati e della comunicazione con il database.

4.34 Componente sgad::servertier::dataaccess::data

4.34.1 Informazioni sul package

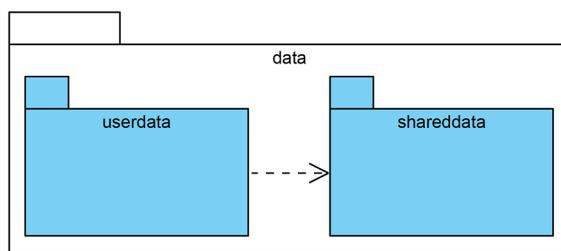


Figura 45: Diagramma della componente `sgad::servertier::dataaccess::data`

- **Descrizione**: componente che gestisce l'insieme dei dati necessari per mantenere memorizzata, sul server, la sessione dell'utente.
- Padre: **dataaccess**
- Interazioni con altri componenti
 - **operations**: componente per la gestione dell'insieme delle possibili operazioni che un utente può effettuare.
 - **databasemanager**: componente per la gestione dell'interazione tra il server e il database, esegue azioni come il salvataggio e il caricamento dei dati. Esso è l'unico punto di accesso con il database.
- Package contenuti
 - **shareddata**: componente per la gestione dei dati che non dipendono dall'utente ma dall'implementazione del mondo di gioco.
 - **userdata**: componente per la gestione dei dati relativi alla sessione dell'utente mantenuta dal server.

4.35 Componente sgad::servtier::dataaccess::data::shareddata

4.35.1 Informazioni sul package

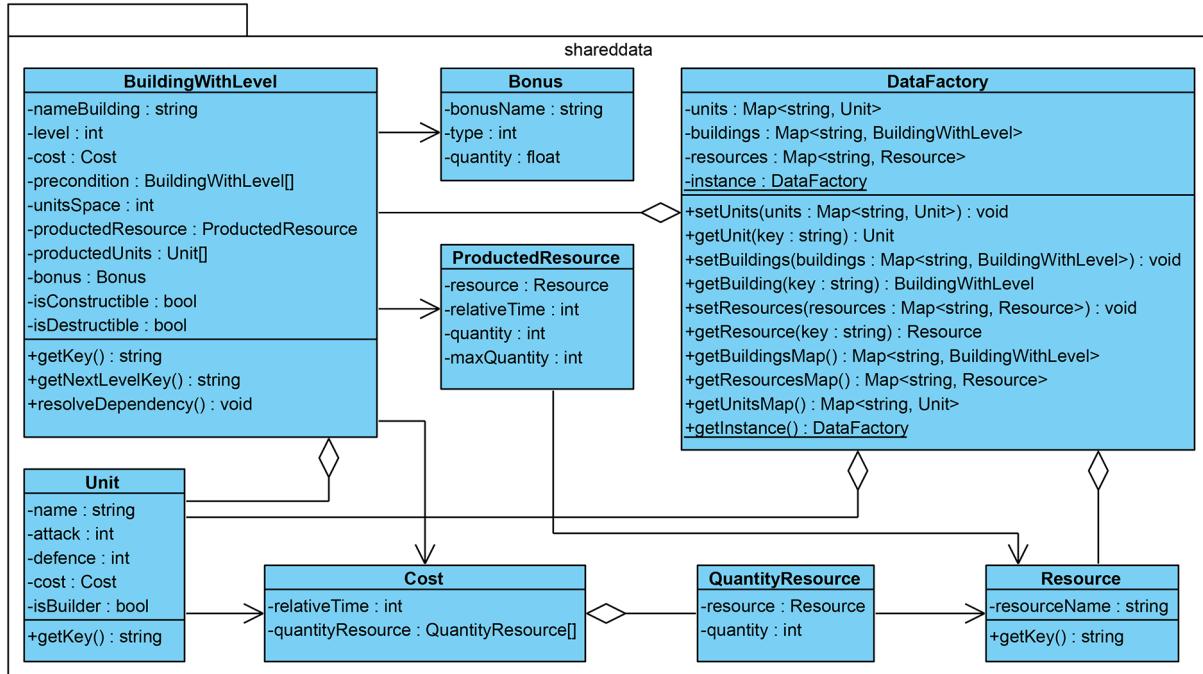


Figura 46: Diagramma della componente `sgad::servtier::dataaccess::data::-`
`shareddata`

- **Descrizione:** componente per la gestione dei dati che non dipendono dall'utente ma dall'implementazione del mondo di gioco.

- **Padre:** `data`

- **Interazioni con altri componenti**

- **userdata:** componente per la gestione dei dati relativi alla sessione dell'utente mantenuta dal server.
- **databasemanager:** componente per la gestione dell'interazione tra il server e il database, esegue azioni come il salvataggio e il caricamento dei dati. Esso è l'unico punto di accesso con il database.
- **shareddatadao:** componente per l'istanziazione delle classi del pacchetto `shareddata` a partire dai dati letti dal database.
- **userdatadao:** componente per l'istanziazione delle classi del pacchetto `userdata` a partire dai dati letti dal database.

4.35.2 Classi

4.35.2.1 sgad::servertier::dataaccess::data::shareddata::Bonus

- **Descrizione:** classe per la gestione dei bonus eventualmente disponibili per certi edifici.
- **Utilizzo:** viene utilizzata per mantenere l'informazione sul nome del bonus, sul tipo e sulla quantità dello stesso.
- **Relazioni con altre classi**

- ← BuildingWithLevel: classe per la gestione di un edificio ad un particolare livello.
- ← BonusDAO: classe per l'interazione con il database di una classe di tipo Bonus.

4.35.2.2 sgad::servertier::dataaccess::data::shareddata::BuildingWithLevel

- **Descrizione:** classe per la gestione di un edificio ad un particolare livello.
 - **Utilizzo:** viene utilizzata per memorizzare tutte le informazioni che descrivono un edificio ad un particolare livello.
 - **Relazioni con altre classi**
- ← BuildConstruction: classe che rappresenta l'operazione di costruzione di un edificio.
 - ← UpgradeBuilding: classe che rappresenta l'operazione di avanzamento di livello di un edificio posseduto dall'utente.
 - ← DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - ← BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - ← DataBaseManager: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
 - ← BuildingWithLevelDAO: classe per l'interazione con il database di una classe di tipo BuildingWithLevel.
 - → Bonus: classe per la gestione dei bonus eventualmente disponibili per certi edifici.
 - → Cost: classe per la gestione di un costo in termini di risorse e tempo.
 - → ProductedResource: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - → Unit: classe per la gestione delle informazioni di una unità.

4.35.2.3 sgad::servertier::dataaccess::data::shareddata::Cost

- **Descrizione:** classe per la gestione di un costo in termini di risorse e tempo.
- **Utilizzo:** viene utilizzata per la gestione di un costo in termini di quantità di risorse.
- **Relazioni con altre classi**
 - ← **BuildingWithLevel:** classe per la gestione di un edificio ad un particolare livello.
 - ← **Unit:** classe per la gestione delle informazioni di una unità.
 - ← **CostDAO:** classe per l'interazione con il database di una classe di tipo Cost.
 - → **QuantityResource:** classe per la gestione della quantità di una particolare risorsa.

4.35.2.4 sgad::servertier::dataaccess::data::shareddata::DataFactory

- **Descrizione:** classe che ritorna un'istanza delle componenti di gioco.
- **Utilizzo:** viene utilizzata per ottenere riferimenti agli oggetti che compongono la sua stessa componente.
- **Relazioni con altre classi**
 - ← **Attack:** classe che rappresenta l'operazione di lancio attacco ad un altro utente.
 - ← **BuildConstruction:** classe che rappresenta l'operazione di costruzione di un edificio.
 - ← **ReceiveAttack:** classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglia.
 - ← **Registration:** classe per la gestione dell'operazione di registrazione.
 - ← **TrainUnit:** classe che rappresenta l'operazione di arruolamento di unità.
 - ← **UpgradeBuilding:** classe che rappresenta l'operazione di avanzamento di livello di un edificio posseduto dall'utente.
 - ← **UserData:** classe per la gestione dei dati di un utente.
 - ← **DataBaseManager:** classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
 - ← **BuildingWithLevelDAO:** classe per l'interazione con il database di una classe di tipo BuildingWithLevel.
 - ← **ProductedResourceDAO:** classe per l'interazione con il database di una classe di tipo ProductedResource.
 - ← **QuantityResourceDAO:** classe per l'interazione con il database di una classe di tipo QuantityResource.

- ← `SharedDataDAO`: classe per la trasformazione dei dati di gioco generali contenuti nel DataFactory in formato JSON.
- ← `BuildingPossessionDAO`: classe per l'interazione con il database di una classe di tipo BuildingPossession.
- ← `OwnedResourceDAO`: classe per l'interazione con il database di una classe di tipo OwnedResource.
- ← `UnitInProgressDAO`: classe per l'interazione con il database di una classe di tipo UnitInProgress.
- ← `UnitPossessionDAO`: classe per l'interazione con il database di una classe di tipo UnitPossession.
- → `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
- → `Resource`: classe che rappresenta una risorsa.
- → `Unit`: classe per la gestione delle informazioni di una unità.

4.35.2.5 `sgad::servertier::dataaccess::data::shareddata::ProducedResource`

- **Descrizione:** classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
- **Utilizzo:** viene utilizzata per la memorizzazione delle informazioni relative alla risorsa prodotta da un edificio.
- **Relazioni con altre classi**

- ← `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
- ← `ProducedResourceDAO`: classe per l'interazione con il database di una classe di tipo ProducedResource.
- → `Resource`: classe che rappresenta una risorsa.

4.35.2.6 `sgad::servertier::dataaccess::data::shareddata::QuantityResource`

- **Descrizione:** classe per la gestione della quantità di una particolare risorsa.
 - **Utilizzo:** viene utilizzata per associare una quantità ad una risorsa.
 - **Relazioni con altre classi**
- ← `Cost`: classe per la gestione di un costo in termini di risorse e tempo.
 - ← `QuantityResourceDAO`: classe per l'interazione con il database di una classe di tipo QuantityResource.
 - → `Resource`: classe che rappresenta una risorsa.

4.35.2.7 sgad::servertier::dataaccess::data::shareddata::Resource

- **Descrizione:** classe che rappresenta una risorsa.
- **Utilizzo:** viene utilizzata per memorizzare le informazioni su una risorsa.
- **Relazioni con altre classi**
 - ← LoadPersonalData: classe per la gestione del caricamento dei dati personali dell'utente.
 - ← DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - ← ProductedResource: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - ← QuantityResource: classe per la gestione della quantità di una particolare risorsa.
 - ← OwnedResource: classe che rappresenta una risorsa posseduta da un utente.
 - ← DataBaseManager: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
 - ← ProductedResourceDAO: classe per l'interazione con il database di una classe di tipo ProductedResource.
 - ← ResourceDAO: classe per l'interazione con il database di una classe di tipo Resource.

4.35.2.8 sgad::servertier::dataaccess::data::shareddata::Unit

- **Descrizione:** classe per la gestione delle informazioni di una unità.
- **Utilizzo:** viene utilizzata per la memorizzazione delle informazioni sulle unità.
- **Relazioni con altre classi**
 - ← Attack: classe che rappresenta l'operazione di lancio attacco ad un altro utente.
 - ← ReceiveAttack: classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglia.
 - ← BuildingWithLevel: classe per la gestione di un edificio ad un particolare livello.
 - ← DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - ← UnitInProgress: classe per la gestione della coda di costruzione di unità di un particolare edificio.
 - ← UnitPossession: classe per la rappresentazione di un'unità con associata una quantità.
 - ← DataBaseManager: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
 - ← UnitDAO: classe per l'interazione con il database di una classe di tipo Unit.
 - → Cost: classe per la gestione di un costo in termini di risorse e tempo.

4.36 Componente sgad::servertier::dataaccess::data::userdata

4.36.1 Informazioni sul package

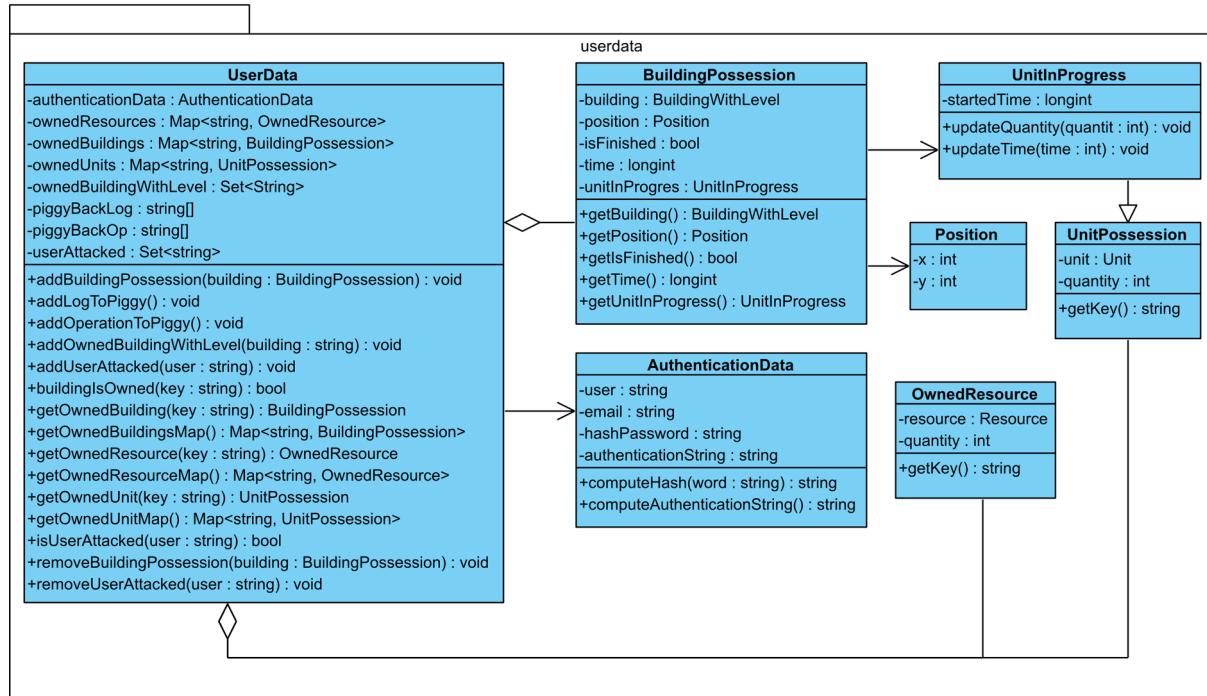


Figura 47: Diagramma della componente `sgad::servertier::dataaccess::data::userdata`

- **Descrizione:** componente per la gestione dei dati relativi alla sessione dell'utente mantenuta dal server.
- **Padre:** data
- **Interazioni con altri componenti**
 - `shareddata`: componente per la gestione dei dati che non dipendono dall'utente ma dall'implementazione del mondo di gioco.
 - `userdatadao`: componente per l'istanziazione delle classi del pacchetto `userdata` a partire dai dati letti dal database.

4.36.2 Classi

4.36.2.1 sgad::servertier::dataaccess::data::userdata::AuthenticationData

- **Descrizione:** classe per la gestione delle informazioni personali e di accesso di un utente.

- **Utilizzo:** viene utilizzata istanziando un oggetto che memorizza le informazioni.

- **Relazioni con altre classi**

- ← **ChangeAccountData:** classe che rappresenta l'operazione di modifica dei dati dell'account utente.
- ← **Login:** classe che rappresenta l'operazione di login.
- ← **Registration:** classe per la gestione dell'operazione di registrazione.
- ← **UserData:** classe per la gestione dei dati di un utente.
- ← **AuthenticationDataDAO:** classe per l'interazione con il database di una classe di tipo AuthenticationData.

4.36.2.2 sgad::servertier::dataaccess::data::userdata::BuildingPossession

- **Descrizione:** classe per la gestione di un edificio posseduto da un utente.

- **Utilizzo:** viene utilizzata per la memorizzazione delle informazioni riguardanti un edificio posseduto da un utente.

- **Relazioni con altre classi**

- ← **BuildConstruction:** classe che rappresenta l'operazione di costruzione di un edificio.
- ← **ReceiveAttack:** classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglia.
- ← **Registration:** classe per la gestione dell'operazione di registrazione.
- ← **UpdateUserData:** mantiene aggiornati i dati dell'utente in merito alle costruzioni e alle unità in coda di produzione.
- ← **UpgradeBuilding:** classe che rappresenta l'operazione di avanzamento di livello di un edificio posseduto dall'utente.
- ← **UserData:** classe per la gestione dei dati di un utente.
- ← **BuildingPossessionDAO:** classe per l'interazione con il database di una classe di tipo BuildingPossession.
- ← **UserDataDAO:** classe per l'interazione con il database di una classe di tipo UserData.
- → **BuildingWithLevel:** classe per la gestione di un edificio ad un particolare livello.
- → **Position:** classe per la gestione di una posizione sulla griglia del villaggio.
- → **UnitInProgress:** classe per la gestione della coda di costruzione di unità di un particolare edificio.

4.36.2.3 sgad::servertier::dataaccess::data::userdata::OwnedResource

- **Descrizione:** classe che rappresenta una risorsa posseduta da un utente.
- **Utilizzo:** viene utilizzata per la memorizzazione di una risorsa posseduta da un utente.
- **Relazioni con altre classi**
 - ← **UserData:** classe per la gestione dei dati di un utente.
 - ← **OwnedResourceDAO:** classe per l'interazione con il database di una classe di tipo OwnedResource.
 - ← **UserDataDAO:** classe per l'interazione con il database di una classe di tipo UserData.
 - → **Resource:** classe che rappresenta una risorsa.

4.36.2.4 sgad::servertier::dataaccess::data::userdata::Position

- **Descrizione:** classe per la gestione di una posizione sulla griglia del villaggio.
- **Utilizzo:** viene utilizzata per memorizzare la posizione di un edificio posseduto da un utente.
- **Relazioni con altre classi**
 - ← **BuildConstruction:** classe che rappresenta l'operazione di costruzione di un edificio.
 - ← **Registration:** classe per la gestione dell'operazione di registrazione.
 - ← **BuildingPossession:** classe per la gestione di un edificio posseduto da un utente.
 - ← **PositionDAO:** classe per l'interazione con il database di una classe di tipo Position.

4.36.2.5 sgad::servertier::dataaccess::data::userdata::UnitPossession

- **Descrizione:** classe per la rappresentazione di un'unità con associata una quantità.
- **Utilizzo:** viene utilizzata per memorizzare per ogni tipo di unità la quantità di queste possedute da un utente.
- **Sottoclassi**
 - **UnitInProgress**
- **Relazioni con altre classi**

- ← **Attack**: classe che rappresenta l'operazione di lancio attacco ad un altro utente.
- ← **ReceiveAttack**: classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglie.
- ← **UserData**: classe per la gestione dei dati di un utente.
- ← **UnitPossessionDAO**: classe per l'interazione con il database di una classe di tipo UnitPossession.
- ← **UserDataDAO**: classe per l'interazione con il database di una classe di tipo UserData.
- → **Unit**: classe per la gestione delle informazioni di una unità.

4.36.2.6 sgad::servertier::dataaccess::data::userdata::UnitInProgress

- **Descrizione**: classe per la gestione della coda di costruzione di unità di un particolare edificio.
- **Utilizzo**: viene utilizzata per gestire le code di costruzione degli edifici completi posseduti da un particolare utente.
- **Classi ereditate**
 - UnitPossession
- **Relazioni con altre classi**
 - ← **TrainUnit**: classe che rappresenta l'operazione di arruolamento di unità.
 - ← **UpdateUserData**: mantiene aggiornati i dati dell'utente in merito alle costruzioni e alle unità in coda di produzione.
 - ← **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
 - ← **UnitInProgressDAO**: classe per l'interazione con il database di una classe di tipo UnitInProgress.
 - → **Unit**: classe per la gestione delle informazioni di una unità.

4.36.2.7 sgad::servertier::dataaccess::data::userdata::UserData

- **Descrizione**: classe per la gestione dei dati di un utente.
- **Utilizzo**: viene utilizzata per la gestione dei dati di un utente.
- **Relazioni con altre classi**
 - ← **Logic**: classe per la gestione dell'operazione richiesta dall'utente. Essa si occupa di instradare la richiesta tramite l'invocazione dell'operazione giusta.

- ← **Attack**: classe che rappresenta l'operazione di lancio attacco ad un altro utente.
- ← **BuildConstruction**: classe che rappresenta l'operazione di costruzione di un edificio.
- ← **ChangeAccountData**: classe che rappresenta l'operazione di modifica dei dati dell'account utente.
- ← **DeleteAccount**: classe che rappresenta l'operazione di eliminazione di un account utente.
- ← **DemolishBuilding**: classe che rappresenta la demolizione di un edificio.
- ← **DismissUnit**: classe che rappresenta l'operazione di congedo di unità.
- ← **HarvestResource**: classe per la gestione dell'operazione di raccolta risorse da un edificio.
- ← **InternalLogin**: classe che rappresenta l'operazione di login.
- ← **LoadGlobalData**: classe per la gestione del caricamento da parte del client dei dati globali di gioco.
- ← **LoadVillage**: classe per la gestione dell'operazione di caricamento del villaggio richiesta del client.
- ← **LoadUserList**: classe per la gestione dell'operazione di scelta degli utenti con i quali un utente può interagire.
- ← **Login**: classe che rappresenta l'operazione di login.
- ← **Logout**: classe che rappresenta l'operazione di logout.
- ← **Operation**: classe astratta per l'esecuzione di un'operazione.
- ← **ReceiveAttack**: classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglia.
- ← **ReceiveStealResource**: classe che rappresenta l'operazione interna di ricezione di un attacco e computo dell'esito dello scontro.
- ← **SaveUser**: classe che rappresenta l'operazione di salvataggio sul database i dati aggiornati dell'utente in input.
- ← **StealResource**: classe che rappresenta l'operazione di saccheggio di risorse da un edificio di un altro utente.
- ← **TrainUnit**: classe che rappresenta l'operazione di arruolamento di unità.
- ← **UpdateUserData**: mantiene aggiornati i dati dell'utente in merito alle costruzioni e alle unità in coda di produzione.
- ← **UpgradeBuilding**: classe che rappresenta l'operazione di avanzamento di livello di un edificio posseduto dall'utente.
- ← **DataBaseManager**: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
- ← **UserDataDAO**: classe per l'interazione con il database di una classe di tipo **UserData**.
- → **DataFactory**: classe che ritorna un'istanza delle componenti di gioco.

- → **AuthenticationData**: classe per la gestione delle informazioni personali e di accesso di un utente.
- → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
- → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
- → **UnitPossession**: classe per la rappresentazione di un'unità con associata una quantità.

4.37 Componente sgad::servtier::dataaccess::databaseaccess

4.37.1 Informazioni sul package

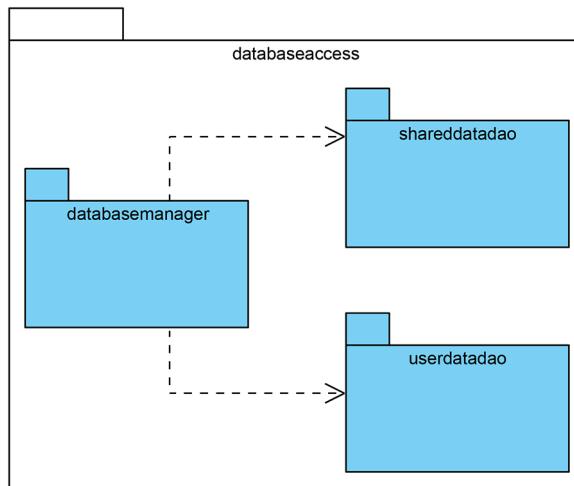


Figura 48: Diagramma della componente `sgad::servtier::dataaccess::databaseaccess`

- **Descrizione:** componente corrispondente al livello data access dell'architettura Three-Tier. Essa si occupa del mantenimento dei dati e della comunicazione con il database.
- **Padre:** `dataaccess`
- **Interazioni con altri componenti**
 - `casbah`.
 - `application`: componente che si occupa del bootstrap dell'applicazione.
 - `businesslogic`: componente corrispondente al livello business logic dell'architettura Three-Tier. Essa si occupa di effettuare le operazioni richieste dagli utenti. Nel caso le operazioni siano relative a interazioni tra più utenti, comunicherà con la componente `presentation` per instradare correttamente la richiesta verso gli altri client coinvolti.

- Package contenuti

- **databasemanager**: componente per la gestione dell’interazione tra il server e il database, esegue azioni come il salvataggio e il caricamento dei dati. Esso è l’unico punto di accesso con il database.
- **shareddatadao**: componente per l’istanziazione delle classi del pacchetto shareddata a partire dai dati letti dal database.
- **userdatadao**: componente per l’istanziazione delle classi del pacchetto userdata a partire dai dati letti dal database.

4.38 Componente sgad::servtier::dataaccess::databaseaccess::databasemanager

4.38.1 Informazioni sul package

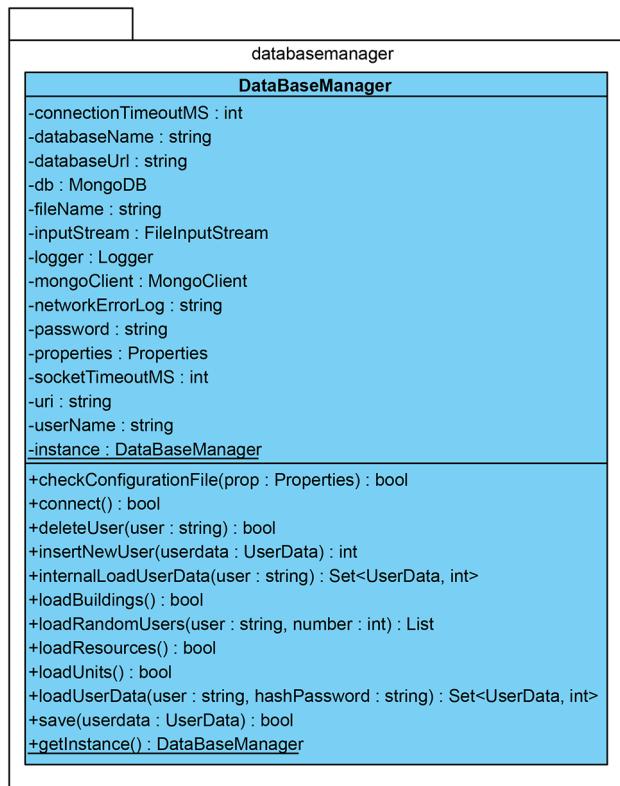


Figura 49: Diagramma della componente sgad::servtier::dataaccess::databaseaccess::databasemanager

- **Descrizione**: componente per la gestione dell’interazione tra il server e il database, esegue azioni come il salvataggio e il caricamento dei dati. Esso è l’unico punto di accesso con il database.
- **Padre**: databaseaccess

- **Interazioni con altri componenti**

- **casbah**.
- **logic**: componente che gestisce il controllo delle operazioni richieste dal client.
- **data**: componente che gestisce l'insieme dei dati necessari per mantenere memorizzata, sul server, la sessione dell'utente.
- **shareddata**: componente per la gestione dei dati che non dipendono dall'utente ma dall'implementazione del mondo di gioco.
- **shareddatadao**: componente per l'istanziazione delle classi del pacchetto shareddata a partire dai dati letti dal database.
- **userdatadao**: componente per l'istanziazione delle classi del pacchetto userdata a partire dai dati letti dal database.

4.38.2 Classi

4.38.2.1 sgad::servtier::dataaccess::databaseaccess::databasemanager::DataBa-seManager

- **Descrizione**: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
- **Utilizzo**: viene utilizzata per la connessione al database e l'esecuzione di operazioni *CRUD_{|g|}* su di esso. Astrae l'implementazione della base di dati al resto dell'applicazione.

- **Relazioni con altre classi**

- ← **Application**: classe per la gestione del bootstrap dell'applicativo lato server.
- ← **ChangeAccountData**: classe che rappresenta l'operazione di modifica dei dati dell'account utente.
- ← **DeleteAccount**: classe che rappresenta l'operazione di eliminazione di un account utente.
- ← **InternalLogin**: classe che rappresenta l'operazione di login.
- ← **LoadUserList**: classe per la gestione dell'operazione di scelta degli utenti con i quali un utente può interagire.
- ← **Login**: classe che rappresenta l'operazione di login.
- ← **Registration**: classe per la gestione dell'operazione di registrazione.
- ← **SaveUser**: classe che rappresenta l'operazione di salvataggio sul database i dati aggiornati dell'utente in input.
- → **casbah::DB**.
- → **casbah::DBCollection**.
- → **casbah::DBObject**.

- → **casbah::MongoClient**.
- → **BuildingWithLevel**: classe per la gestione di un edificio ad un particolare livello.
- → **DataFactory**: classe che ritorna un'istanza delle componenti di gioco.
- → **Resource**: classe che rappresenta una risorsa.
- → **Unit**: classe per la gestione delle informazioni di una unità.
- → **UserData**: classe per la gestione dei dati di un utente.
- → **BuildingWithLevelDAO**: classe per l'interazione con il database di una classe di tipo BuildingWithLevel.
- → **ResourceDAO**: classe per l'interazione con il database di una classe di tipo Resource.
- → **UnitDAO**: classe per l'interazione con il database di una classe di tipo Unit.
- → **UserDataDAO**: classe per l'interazione con il database di una classe di tipo UserData.

4.39 Componente sgad::servtier::dataaccess::databaseaccess::shareddata Dao

4.39.1 Informazioni sul package

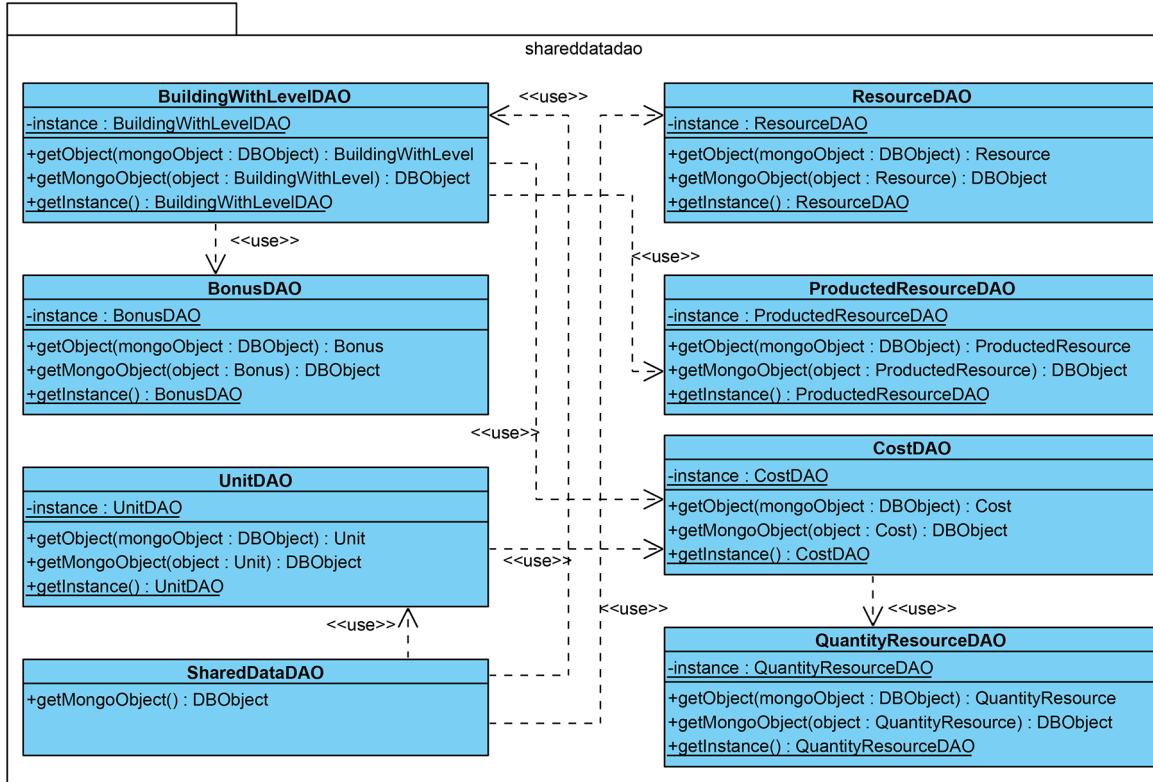


Figura 50: Diagramma della componente sgad::servtier::dataaccess::databaseaccess::shareddata Dao

- **Descrizione:** componente per l'istanziazione delle classi del pacchetto shareddata a partire dai dati letti dal database.
- **Padre:** databaseaccess
- **Interazioni con altri componenti**
 - casbah.
 - shareddata: componente per la gestione dei dati che non dipendono dall'utente ma dall'implementazione del mondo di gioco.
 - databasemanager: componente per la gestione dell'interazione tra il server e il database, esegue azioni come il salvataggio e il caricamento dei dati. Esso è l'unico punto di accesso con il database.

4.39.2 Classi

4.39.2.1 sgad::servtier::dataaccess::databaseaccess::shareddatadao::BonusDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo Bonus.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe Bonus a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `BuildingWithLevelDAO`: classe per l'interazione con il database di una classe di tipo `BuildingWithLevel`.
 - → `casbah::DBObject`.
 - → `Bonus`: classe per la gestione dei bonus eventualmente disponibili per certi edifici.

4.39.2.2 sgad::servtier::dataaccess::databaseaccess::shareddatadao::BuildingWithLevelDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo `BuildingWithLevel`.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe `BuildingWithLevel` a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `DataBaseManager`: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
 - ← `SharedDataDAO`: classe per la trasformazione dei dati di gioco generali contenuti nel `DataFactory` in formato JSON.
 - → `casbah::DBObject`.
 - → `BuildingWithLevel`: classe per la gestione di un edificio ad un particolare livello.
 - → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
 - → `BonusDAO`: classe per l'interazione con il database di una classe di tipo `Bonus`.
 - → `CostDAO`: classe per l'interazione con il database di una classe di tipo `Cost`.
 - → `ProductedResourceDAO`: classe per l'interazione con il database di una classe di tipo `ProductedResource`.

4.39.2.3 sgad::servertier::dataaccess::databaseaccess::shareddatadao::CostDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo Cost.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe Cost a partire da oggetti MongoDBObject propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← BuildingWithLevelDAO: classe per l'interazione con il database di una classe di tipo BuildingWithLevel.
 - ← UnitDAO: classe per l'interazione con il database di una classe di tipo Unit.
 - → casbah::DBObject.
 - → Cost: classe per la gestione di un costo in termini di risorse e tempo.
 - → QuantityResourceDAO: classe per l'interazione con il database di una classe di tipo QuantityResource.

4.39.2.4 sgad::servertier::dataaccess::databaseaccess::shareddatadao::ProductedResourceDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo ProductedResource.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe ProductedResource a partire da oggetti MongoDBObject propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← BuildingWithLevelDAO: classe per l'interazione con il database di una classe di tipo BuildingWithLevel.
 - → casbah::DBObject.
 - → DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - → ProductedResource: classe per la gestione delle informazioni relative alla risorsa prodotta da un edificio.
 - → Resource: classe che rappresenta una risorsa.

4.39.2.5 sgad::servertier::dataaccess::databaseaccess::shareddatadao::QuantityResourceDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo QuantityResource.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe QuantityResource a partire da oggetti MongoDBObject propri della tecnologia di persistenza adottata.

- **Relazioni con altre classi**

- ← `CostDAO`: classe per l'interazione con il database di una classe di tipo `Cost`.
- → `casbah::DBObject`.
- → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
- → `QuantityResource`: classe per la gestione della quantità di una particolare risorsa.

4.39.2.6 sgad::servtier::dataaccess::databaseaccess::shareddatadao::ResourceDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo `Resource`.

- **Utilizzo:** viene utilizzata per istanziare oggetti della classe `Resource` a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.

- **Relazioni con altre classi**

- ← `DataBaseManager`: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
- ← `SharedDataDAO`: classe per la trasformazione dei dati di gioco generali contenuti nel `DataFactory` in formato JSON.
- → `casbah::DBObject`.
- → `Resource`: classe che rappresenta una risorsa.

4.39.2.7 sgad::servtier::dataaccess::databaseaccess::shareddatadao::SharedDataDAO

- **Descrizione:** classe per la trasformazione dei dati di gioco generali contenuti nel `DataFactory` in formato JSON.

- **Utilizzo:** viene usata per ottenere il JSON dei dati di gioco generali.

- **Relazioni con altre classi**

- ← `LoadGlobalData`: classe per la gestione del caricamento da parte del client dei dati globali di gioco.
- → `casbah::DBObject`.
- → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
- → `BuildingWithLevelDAO`: classe per l'interazione con il database di una classe di tipo `BuildingWithLevel`.
- → `ResourceDAO`: classe per l'interazione con il database di una classe di tipo `Resource`.
- → `UnitDAO`: classe per l'interazione con il database di una classe di tipo `Unit`.

4.39.2.8 sgad::servertier::dataaccess::databaseaccess::shareddatadao::UnitDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo Unit.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe Unit a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `DataBaseManager`: classe per la gestione del database. Essa astrae l'utilizzo di un particolare database al resto dell'applicativo.
 - ← `SharedDataDAO`: classe per la trasformazione dei dati di gioco generali contenuti nel DataFactory in formato JSON.
 - → `casbah::DBObject`.
 - → `Unit`: classe per la gestione delle informazioni di una unità.
 - → `CostDAO`: classe per l'interazione con il database di una classe di tipo Cost.

4.40 Componente sgad::servertier::dataaccess::databaseaccess::userdatadao

4.40.1 Informazioni sul package

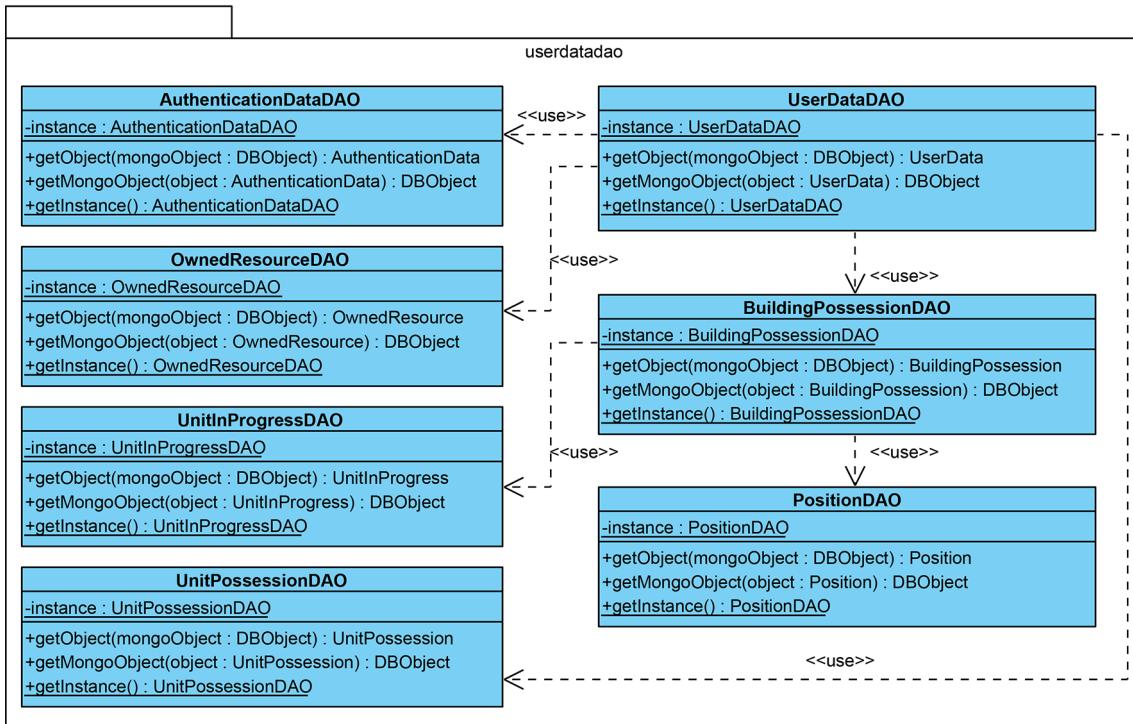


Figura 51: Diagramma della componente sgad::servertier::dataaccess::databaseaccess::userdatadao

- **Descrizione:** componente per l'istanziazione delle classi del pacchetto userdata a partire dai dati letti dal database.
- **Padre:** databaseaccess
- **Interazioni con altri componenti**
 - casbah.
 - shareddata: componente per la gestione dei dati che non dipendono dall'utente ma dall'implementazione del mondo di gioco.
 - userdata: componente per la gestione dei dati relativi alla sessione dell'utente mantenuta dal server.
 - databasemanager: componente per la gestione dell'interazione tra il server e il database, esegue azioni come il salvataggio e il caricamento dei dati. Esso è l'unico punto di accesso con il database.

4.40.2 Classi

4.40.2.1 sgad::servertier::dataaccess::databaseaccess::userdatadao::AuthenticationDataDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo AuthenticationData.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe AuthenticationData a partire da oggetti MongoDBObject propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← UserDataDAO: classe per l'interazione con il database di una classe di tipo UserData.
 - → casbah::DBObject.
 - → AuthenticationData: classe per la gestione delle informazioni personali e di accesso di un utente.

4.40.2.2 sgad::servertier::dataaccess::databaseaccess::userdatadao::BuildingPossessionDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo BuildingPossession.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe BuildingPossession a partire da oggetti MongoDBObject propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← ReceiveAttack: classe che gestisce la ricezione dell'attacco ed il computo del risultato della battaglie.
 - ← UserDataDAO: classe per l'interazione con il database di una classe di tipo UserData.
 - → casbah::DBObject.
 - → DataFactory: classe che ritorna un'istanza delle componenti di gioco.
 - → BuildingPossession: classe per la gestione di un edificio posseduto da un utente.
 - → PositionDAO: classe per l'interazione con il database di una classe di tipo Position.
 - → UnitInProgressDAO: classe per l'interazione con il database di una classe di tipo UnitInProgress.

4.40.2.3 sgad::servertier::dataaccess::databaseaccess::userdatadao::OwnedResourceDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo OwnedResource.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe OwnedResource a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `UserDataDAO`: classe per l'interazione con il database di una classe di tipo `UserData`.
 - → `casbah::DBObject`.
 - → `DataFactory`: classe che ritorna un'istanza delle componenti di gioco.
 - → `OwnedResource`: classe che rappresenta una risorsa posseduta da un utente.

4.40.2.4 sgad::servertier::dataaccess::databaseaccess::userdatadao::PositionDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo `Position`.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe `Position` a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `BuildingPossessionDAO`: classe per l'interazione con il database di una classe di tipo `BuildingPossession`.
 - → `casbah::DBObject`.
 - → `Position`: classe per la gestione di una posizione sulla griglia del villaggio.

4.40.2.5 sgad::servertier::dataaccess::databaseaccess::userdatadao::UnitInProgressDAO

- **Descrizione:** classe per l'interazione con il database di una classe di tipo `UnitInProgress`.
- **Utilizzo:** viene utilizzata per istanziare oggetti della classe `UnitInProgress` a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `BuildingPossessionDAO`: classe per l'interazione con il database di una classe di tipo `BuildingPossession`.
 - → `casbah::DBObject`.

- → **DataFactory**: classe che ritorna un’istanza delle componenti di gioco.
- → **UnitInProgress**: classe per la gestione della coda di costruzione di unità di un particolare edificio.

4.40.2.6 sgad::servertier::dataaccess::databaseaccess::userdatadao::UnitPossessionDAO

- **Descrizione**: classe per l’interazione con il database di una classe di tipo UnitPossession.
- **Utilizzo**: viene utilizzata per istanziare oggetti della classe UnitPossession a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `UserDataDAO`: classe per l’interazione con il database di una classe di tipo `UserData`.
 - → `casbah::DBObject`.
 - → **DataFactory**: classe che ritorna un’istanza delle componenti di gioco.
 - → **UnitPossession**: classe per la rappresentazione di un’unità con associata una quantità.

4.40.2.7 sgad::servertier::dataaccess::databaseaccess::userdatadao::UserDataDAO

- **Descrizione**: classe per l’interazione con il database di una classe di tipo `UserData`.
- **Utilizzo**: viene utilizzata per istanziare oggetti della classe `UserData` a partire da oggetti `MongoDBObject` propri della tecnologia di persistenza adottata.
- **Relazioni con altre classi**
 - ← `LoadVillage`: classe per la gestione dell’operazione di caricamento del villaggio richiesta del client.
 - ← `DataBaseManager`: classe per la gestione del database. Essa astrae l’utilizzo di un particolare database al resto dell’applicativo.
 - → `casbah::DBObject`.
 - → **BuildingPossession**: classe per la gestione di un edificio posseduto da un utente.
 - → **OwnedResource**: classe che rappresenta una risorsa posseduta da un utente.
 - → **UnitPossession**: classe per la rappresentazione di un’unità con associata una quantità.
 - → **UserData**: classe per la gestione dei dati di un utente.

- → `AuthenticationDataDAO`: classe per l'interazione con il database di una classe di tipo `AuthenticationData`.
- → `BuildingPossessionDAO`: classe per l'interazione con il database di una classe di tipo `BuildingPossession`.
- → `OwnedResourceDAO`: classe per l'interazione con il database di una classe di tipo `OwnedResource`.
- → `UnitPossessionDAO`: classe per l'interazione con il database di una classe di tipo `UnitPossession`.

4.41 Componente sgad::servertier::presentation

4.41.1 Informazioni sul package

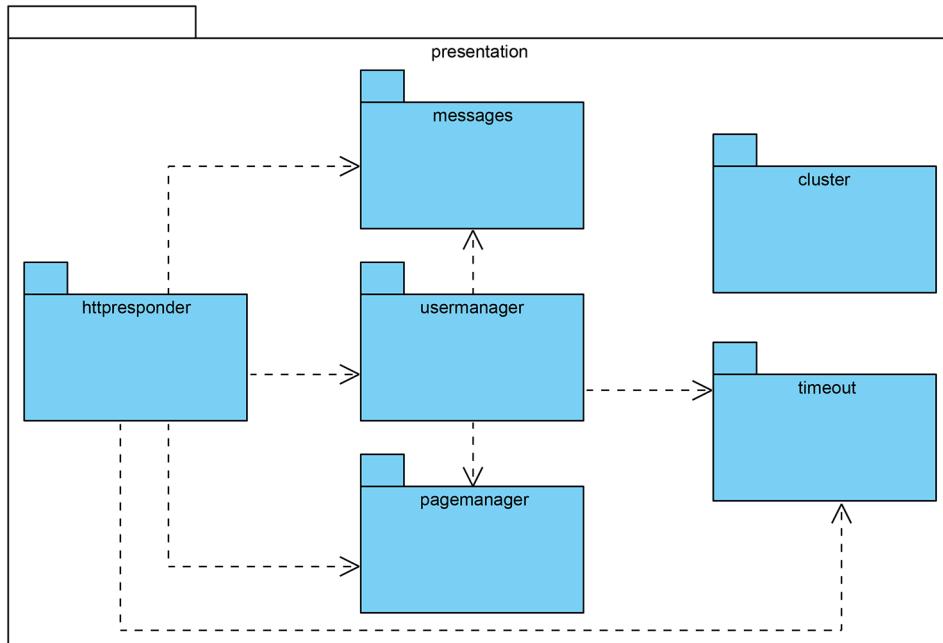


Figura 52: Diagramma della componente `sgad::servertier::presentation`

- **Descrizione:** componente per la gestione della comunicazione con i client. Rappresenta il livello superiore e visibile dall'esterno dell'architettura Three-Tier. Si occupa di smistare le richieste in arrivo alle corrette componenti della logica.
- **Padre:** `servertier`
- **Interazioni con altri componenti**
 - `akka::actors`.
 - `akka::contrib::pattern`.
 - `application`: componente che si occupa del bootstrap dell'applicazione.
 - `businesslogic`: componente corrispondente al livello business logic dell'architettura Three-Tier. Essa si occupa di effettuare le operazioni richieste dagli utenti. Nel caso le operazioni siano relative a interazioni tra più utenti, comunicherà con la componente `presentation` per instradare correttamente la richiesta verso gli altri client coinvolti.
 - `spray`.
- **Package contenuti**
 - `cluster`: componente per la gestione e la partecipazione del singolo server a cluster di server.

- **httpresponder**: componente per la gestione delle richieste HTTP ricevute dai vari client.
- **messages**: componente per i messaggi scambiati tra gli attori del sistema.
- **pagemanager**: componente per la gestione delle pagine HTML, CSS e sorgenti JavaScript.
- **timeout**: componente per la gestione dei timeout.
- **usermanager**: componente per la gestione degli attori relativi alla gestione delle richieste generate dai client. Si occupa di ricercare in tutto il cluster di server l'attore associato alla richiesta.

4.42 Componente sgad::servertier::presentation::cluster

4.42.1 Informazioni sul package

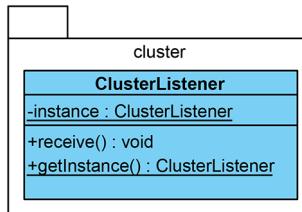


Figura 53: Diagramma della componente `sgad::servertier::presentation::cluster`

- **Descrizione**: componente per la gestione e la partecipazione del singolo server a cluster di server.
- **Padre**: `presentation`
- **Interazioni con altri componenti**
 - `application`: componente che si occupa del bootstrap dell'applicazione.

4.42.2 Classi

4.42.2.1 `sgad::servertier::presentation::cluster::ClusterListener`

- **Descrizione**: classe che gestisce l'interazione del server con gli altri server del cluster.
- **Utilizzo**: viene utilizzato per gestire i messaggi in arrivo dagli eventi del cluster.
- **Relazioni con altre classi**
 - ← `Application`: classe per la gestione del bootstrap dell'applicativo lato server.

4.43 Componente sgad::servertier::presentation::httpresponder

4.43.1 Informazioni sul package

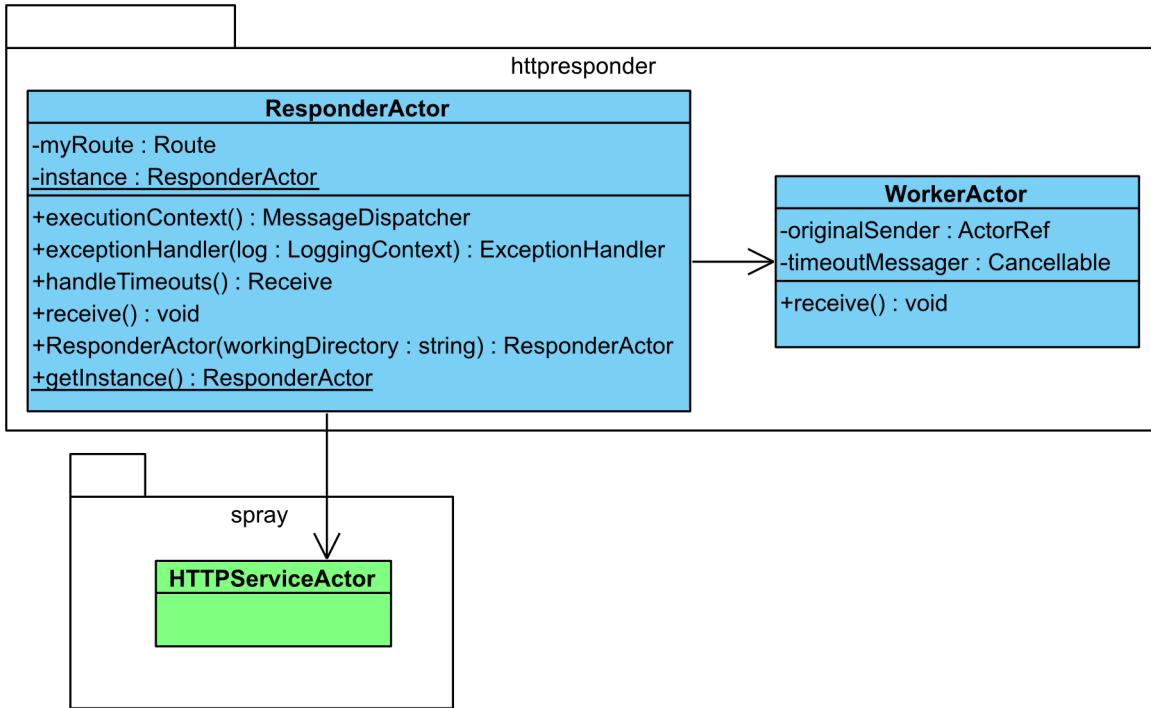


Figura 54: Diagramma della componente sgad::servertier::presentation::httpresponder

- **Descrizione:** componente per la gestione delle richieste HTTP ricevute dai vari client.
- **Padre:** presentation
- **Interazioni con altri componenti**
 - requester: componente per la gestione dell'invio di richieste al server.
 - messages: componente per i messaggi scambiati tra gli attori del sistema.
 - pagemanager: componente per la gestione delle pagine HTML, CSS e sorgenti JavaScript.
 - timeout: componente per la gestione dei timeout.
 - usermanager: componente per la gestione degli attori relativi alla gestione delle richieste generate dai client. Si occupa di ricercare in tutto il cluster di server l'attore associato alla richiesta.
 - spray.

4.43.2 Classi

4.43.2.1 sgad::servertier::presentation::httpresponder::ResponderActor

- **Descrizione:** classe per la gestione degli end point per le richieste HTTP inviate dal client.
- **Utilizzo:** viene utilizzato per poter esporre un pin di connessione ai client.
- **Classi ereditate**
 - spray::HttpServiceActor
- **Relazioni con altre classi**
 - ← Application: classe per la gestione del bootstrap dell'applicativo lato server.
 - → WorkerActor: classe per la gestione asincrona delle richieste in arrivo dai client.
 - → ToWorkerLoginRequest: classe per il messaggio di elaborazione di una richiesta di login al WorkerActor.
 - → ToWorkerRegistrationRequest: classe per il messaggio di elaborazione di una richiesta di registrazione al WorkerActor.
 - → ToWorkerUserRequest: classe per il messaggio di elaborazione di una richiesta di gioco al WorkerActor.
 - → PageFactory: classe che gestisce le pagine HTML da inviare ai client.
 - → STimeout: classe che rappresenta i timeout delle richieste all'interno dell'applicazione.
 - → spray::Route.

4.43.2.2 sgad::servertier::presentation::httpresponder::WorkerActor

- **Descrizione:** classe per la gestione asincrona delle richieste in arrivo dai client.
- **Utilizzo:** viene utilizzata dal ResponderActor per rendere asincrone le risposte ai client.
- **Relazioni con altre classi**
 - ← ResponderActor: classe per la gestione degli end point per le richieste HTTP inviate dal client.
 - → ToLoginActorRequest: classe per il messaggio di elaborazione di una richiesta di login al LoginActor.
 - → ToPublisherAndUserRequest: classe per il messaggio di elaborazione di una richiesta di gioco al PublisherActor.

- → `ToRegistrationActorRequest`: classe per il messaggio di elaborazione di una richiesta di registrazione al RegistrationActor.
- → `ToWorkerLoginRequest`: classe per il messaggio di elaborazione di una richiesta di login al WorkerActor.
- → `ToWorkerRegistrationRequest`: classe per il messaggio di elaborazione di una richiesta di registrazione al WorkerActor.
- → `ToWorkerUserRequest`: classe per il messaggio di elaborazione di una richiesta di gioco al WorkerActor.
- → `WorkerActorTimeoutMessage`: classe che rappresenta il messaggio di timeout verso il WorkerActor a cui l'attore deve reagire uccidendosi.
- → `STimeout`: classe che rappresenta i timeout delle richieste all'interno dell'applicazione.
- → `LoginActor`: classe per la gestione delle richieste di login.
- → `RegistrationActor`: classe per la gestione delle richieste di registrazione.

4.44 Componente sgad::servertier::presentation::messages

4.44.1 Informazioni sul package

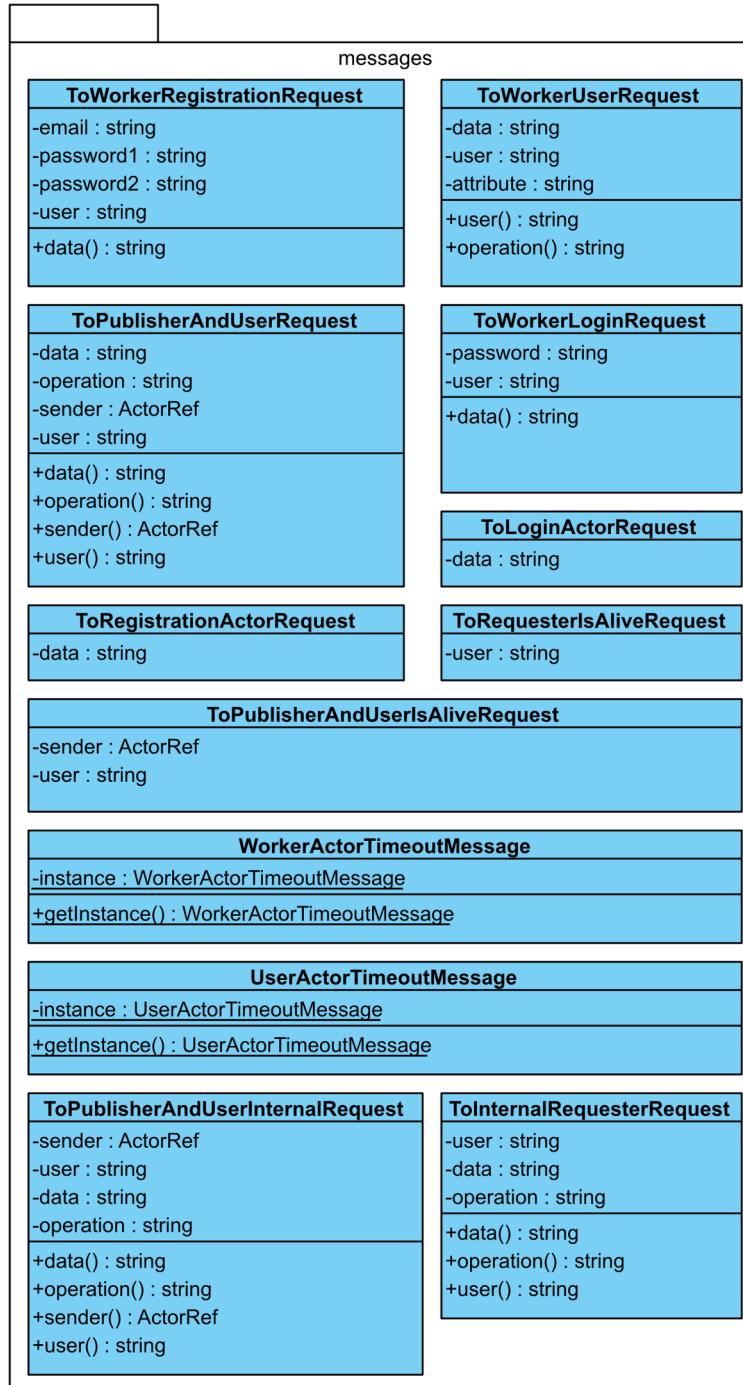


Figura 55: Diagramma della componente sgad::servertier::presentation::messages

- **Descrizione:** componente per i messaggi scambiati tra gli attori del sistema.
- **Padre:** presentation
- **Interazioni con altri componenti**
 - **httpresponder:** componente per la gestione delle richieste HTTP ricevute dai vari client.
 - **userManager:** componente per la gestione degli attori relativi alla gestione delle richieste generate dai client. Si occupa di ricercare in tutto il cluster di server l'attore associato alla richiesta.

4.44.2 Classi

4.44.2.1 sgad::servertier::presentation::messages::ToInternalRequesterRequest

- **Descrizione:** classe per una richiesta interna verso il InternalRequester.
- **Utilizzo:** viene utilizzata per inviare all'internalRequester una richiesta di gestione di un messaggio interno.
- **Relazioni con altre classi**
 - ← InternalRequester: classe di gestione della richiesta di user ad autorizzazione interna incapsulandone l'algoritmo.

4.44.2.2 sgad::servertier::presentation::messages::ToLoginActorRequest

- **Descrizione:** classe per il messaggio di elaborazione di una richiesta di login al LoginActor.
- **Utilizzo:** viene utilizzata dal WorkerActor per ottenere una risposta alla richiesta di login ricevuta.
- **Relazioni con altre classi**
 - ← WorkerActor: classe per la gestione asincrona delle richieste in arrivo dai client.
 - ← LoginActor: classe per la gestione delle richieste di login.

4.44.2.3 sgad::servertier::presentation::messages::ToPublisherAndUserInternalRequest

- **Descrizione:** classe per la gestione delle richieste interne tra user.
- **Utilizzo:** viene utilizzata per mandare un messaggio di richiesta interna tra UserActor.

- Relazioni con altre classi

- ← **InternalRequester**: classe di gestione della richiesta di user ad autorizzazione interna incapsulandone l'algoritmo.

4.44.2.4 sgad::servertier::presentation::messages::ToPublisherAndUserIsAliveRequest

- **Descrizione**: classe per il messaggio isAlive?.

- **Utilizzo**: viene utilizzata per scoprire se un attore è attivo.

- Relazioni con altre classi

- ← **IsUserActorAliveRequester**: attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.
 - ← **IsUserActorAliveResponder**: classe per la risposta della richiesta di esistenza dell'UserActor in modo asincrono.
 - ← **PublisherActor**: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto **usermanager**.

4.44.2.5 sgad::servertier::presentation::messages::ToPublisherAndUserRequest

- **Descrizione**: classe per il messaggio di elaborazione di una richiesta di gioco al PublisherActor.

- **Utilizzo**: viene utilizzata dal WorkerActor per ottenere una risposta alla richiesta di gioco ricevuta.

- Relazioni con altre classi

- ← **WorkerActor**: classe per la gestione asincrona delle richieste in arrivo dai client.
 - ← **PublisherActor**: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto **usermanager**.
 - ← **UserActor**: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.44.2.6 sgad::servertier::presentation::messages::ToRegistrationActorRequest

- **Descrizione**: classe per il messaggio di elaborazione di una richiesta di registrazione al RegistrationActor.

- **Utilizzo:** viene utilizzata dal WorkerActor per ottenere una risposta alla richiesta di registrazione ricevuta.
- **Relazioni con altre classi**
 - ← `WorkerActor`: classe per la gestione asincrona delle richieste in arrivo dai client.
 - ← `RegistrationActor`: classe per la gestione delle richieste di registrazione.

4.44.2.7 sgad::servertier::presentation::messages::ToRequesterIsAliveRequest

- **Descrizione:** classe per la richiesta di esistenza dell'UserActor associato al user in input.
- **Utilizzo:** la classe viene utilizzata per richiedere l'esistenza dell'UserActor associato ad un utente.
- **Relazioni con altre classi**
 - ← `IsUserActorAliveRequester`: attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.

4.44.2.8 sgad::servertier::presentation::messages::ToWorkerLoginRequest

- **Descrizione:** classe per il messaggio di elaborazione di una richiesta di login al WorkerActor.
- **Utilizzo:** viene utilizzata ogni qualvolta il server ottiene dal client una richiesta di login.
- **Relazioni con altre classi**
 - ← `ResponderActor`: classe per la gestione degli end point per le richieste HTTP inviate dal client.
 - ← `WorkerActor`: classe per la gestione asincrona delle richieste in arrivo dai client.

4.44.2.9 sgad::servertier::presentation::messages::ToWorkerRegistrationRequest

- **Descrizione:** classe per il messaggio di elaborazione di una richiesta di registrazione al WorkerActor.
- **Utilizzo:** viene utilizzata ogni qualvolta il server ottiene dal client una richiesta di registrazione.
- **Relazioni con altre classi**

- ← `ResponderActor`: classe per la gestione degli end point per le richieste HTTP inviate dal client.
- ← `WorkerActor`: classe per la gestione asincrona delle richieste in arrivo dai client.

4.44.2.10 sgad::servtier::presentation::messages::ToWorkerUserRequest

- **Descrizione:** classe per il messaggio di elaborazione di una richiesta di gioco al WorkerActor.
- **Utilizzo:** viene utilizzata ogni qualvolta il server ottiene dal client una richiesta di gioco.
- **Relazioni con altre classi**
 - ← `ResponderActor`: classe per la gestione degli end point per le richieste HTTP inviate dal client.
 - ← `WorkerActor`: classe per la gestione asincrona delle richieste in arrivo dai client.

4.44.2.11 sgad::servtier::presentation::messages::UserActorTimeoutMessage

- **Descrizione:** classe che rappresenta il messaggio di timeout verso l'UserActor a cui l'attore deve reagire uccidendosi.
- **Utilizzo:** viene utilizzata per rappresentare il messaggio di timeout diretto all'UserActor a cui l'attore deve reagire.
- **Relazioni con altre classi**
 - ← `UserActor`: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.44.2.12 sgad::servtier::presentation::messages::WorkerActorTimeoutMessage

- **Descrizione:** classe che rappresenta il messaggio di timeout verso il WorkerActor a cui l'attore deve reagire uccidendosi.
- **Utilizzo:** viene utilizzata per rappresentare il messaggio di timeout diretto al WorkerActor a cui l'attore deve reagire.
- **Relazioni con altre classi**
 - ← `WorkerActor`: classe per la gestione asincrona delle richieste in arrivo dai client.

4.45 Componente sgad::servertier::presentation::pagemanager

4.45.1 Informazioni sul package

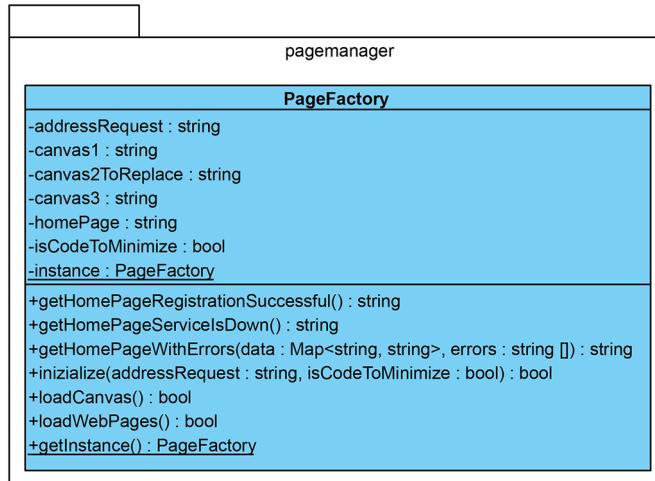


Figura 56: Diagramma della componente `sgad::servertier::presentation::-pagemanager`

- **Descrizione:** componente per la gestione delle pagine HTML, CSS e sorgenti JavaScript.
- **Padre:** `presentation`
- **Interazioni con altri componenti**
 - `httpresponder`: componente per la gestione delle richieste HTTP ricevute dai vari client.
 - `userManager`: componente per la gestione degli attori relativi alla gestione delle richieste generate dai client. Si occupa di ricercare in tutto il cluster di server l'attore associato alla richiesta.

4.45.2 Classi

4.45.2.1 `sgad::servertier::presentation::pagemanager::PageFactory`

- **Descrizione:** classe che gestisce le pagine HTML da inviare ai client.
- **Utilizzo:** viene usata per ottenere le pagine HTML personalizzate e non da inviare ai client.
- **Relazioni con altre classi**
 - ← `Application`: classe per la gestione del bootstrap dell'applicativo lato server.

- ← **Login**: classe che rappresenta l'operazione di login.
- ← **Registration**: classe per la gestione dell'operazione di registrazione.
- ← **ResponderActor**: classe per la gestione degli end point per le richieste HTTP inviate dal client.

4.46 Componente sgad::servertier::presentation::timeout

4.46.1 Informazioni sul package

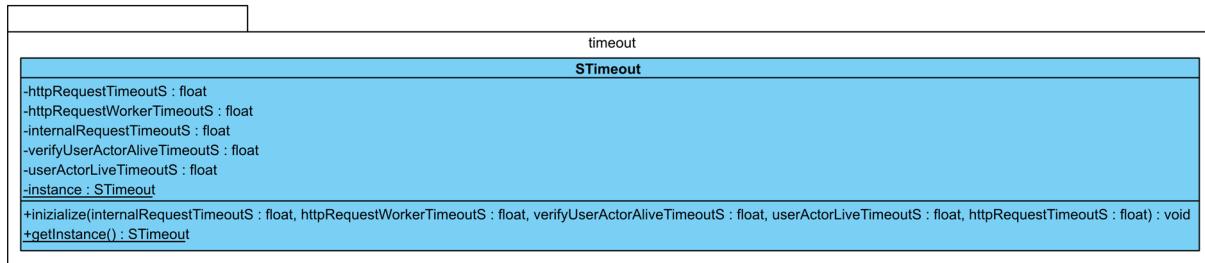


Figura 57: Diagramma della componente sgad::servertier::presentation::timeout

- **Descrizione:** componente per la gestione dei timeout.
- **Padre:** presentation
- **Interazioni con altri componenti**
 - **httpresponder:** componente per la gestione delle richieste HTTP ricevute dai vari client.
 - **usermanager:** componente per la gestione degli attori relativi alla gestione delle richieste generate dai client. Si occupa di ricercare in tutto il cluster di server l'attore associato alla richiesta.

4.46.2 Classi

4.46.2.1 sgad::servertier::presentation::timeout::STimeout

- **Descrizione:** classe che rappresenta i timeout delle richieste all'interno dell'applicazione.
- **Utilizzo:** viene utilizzata per rappresentare i timeout delle richieste.
- **Relazioni con altre classi**
 - ← **Application:** classe per la gestione del bootstrap dell'applicativo lato server.
 - ← **ResponderActor:** classe per la gestione degli end point per le richieste HTTP inviate dal client.
 - ← **WorkerActor:** classe per la gestione asincrona delle richieste in arrivo dai client.
 - ← **InternalRequester:** classe di gestione della richiesta di user ad autorizzazione interna incapsulandone l'algoritmo.

- ← **IsUserActorAliveRequester**: attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.

- ← **UserActor**: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.47 Componente sgad::servertier::presentation::usermanager

4.47.1 Informazioni sul package

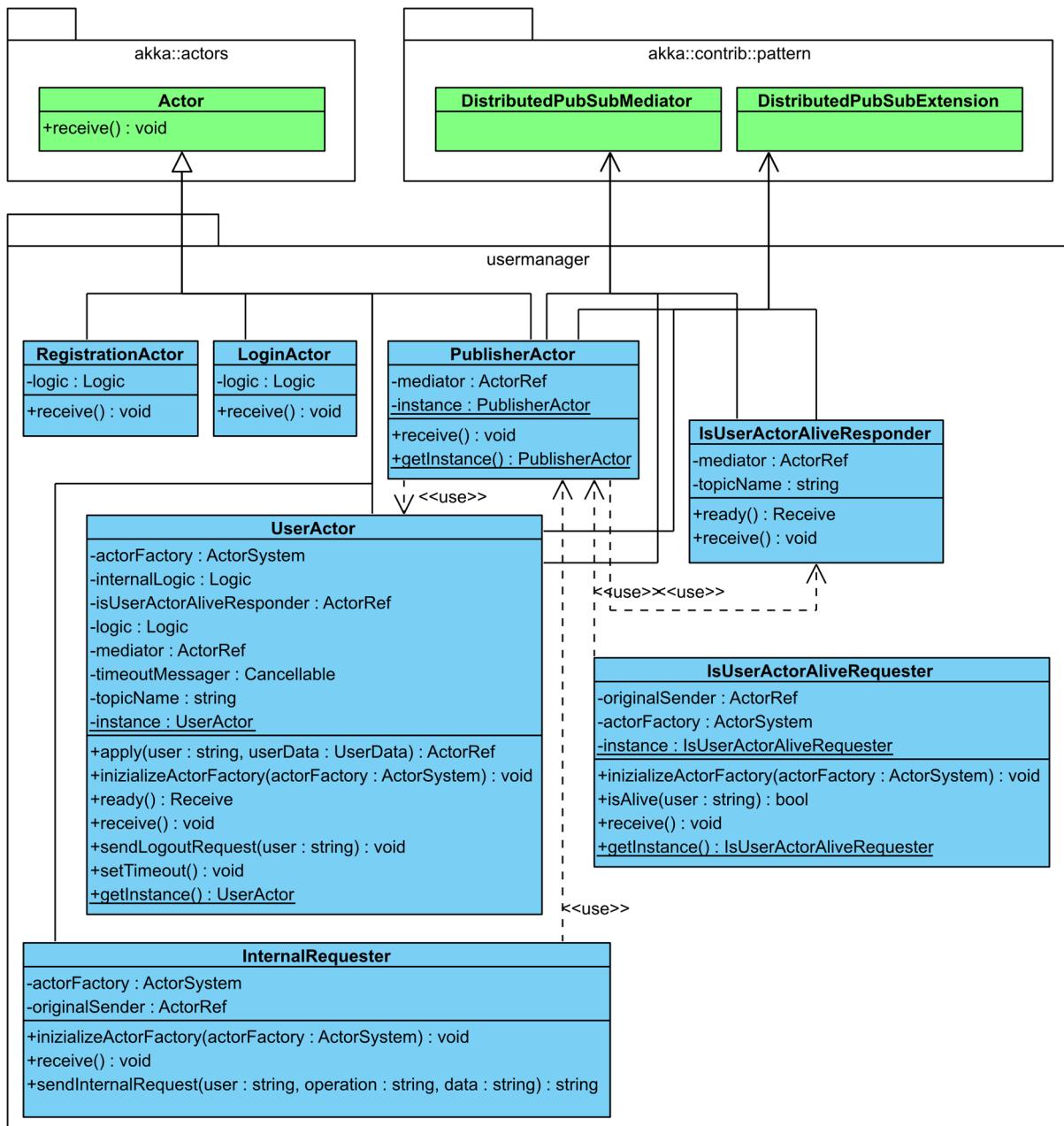


Figura 58: Diagramma della componente `sgad::servertier::presentation::usermanager`

- **Descrizione:** componente per la gestione degli attori relativi alla gestione delle richieste generate dai client. Si occupa di ricercare in tutto il cluster di server l'attore associato alla richiesta.
- **Padre:** presentation
- **Interazioni con altri componenti**
 - akka::actors.
 - akka::contrib::pattern.
 - logic: componente che gestisce il controllo delle operazioni richieste dal client.
 - httpresponder: componente per la gestione delle richieste HTTP ricevute dai vari client.
 - messages: componente per i messaggi scambiati tra gli attori del sistema.
 - pagemanager: componente per la gestione delle pagine HTML, CSS e sorgenti JavaScript.
 - timeout: componente per la gestione dei timeout.

4.47.2 Classi

4.47.2.1 sgad::servertier::presentation::usermanager::InternalRequester

- **Descrizione:** classe di gestione della richiesta di user ad autorizzazione interna incapsulandone l'algoritmo.
- **Utilizzo:** viene utilizzata per mandare richieste interne fra attori.
- **Classi ereditate**
 - akka::actors::Actor
- **Relazioni con altre classi**
 - ← Attack: classe che rappresenta l'operazione di lancio attacco ad un altro utente.
 - ← StealResource: classe che rappresenta l'operazione di saccheggio di risorse da un edificio di un altro utente.
 - → ToInternalRequesterRequest: classe per una richiesta interna verso il InternalRequester.
 - → ToPublisherAndUserInternalRequest: classe per la gestione delle richieste interne tra user.
 - → STimeout: classe che rappresenta i timeout delle richieste all'interno dell'applicazione.
 - → PublisherActor: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto usermanager.

4.47.2.2 sgad::servertier::presentation::usermanager::IsUserActorAliveRequester

- **Descrizione:** attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.
- **Utilizzo:** la classe viene utilizzata per verificare l'esistenza di altri UserActor.
- **Relazioni con altre classi**
 - ← **Application**: classe per la gestione del bootstrap dell'applicativo lato server.
 - ← **InternalLogin**: classe che rappresenta l'operazione di login.
 - ← **Login**: classe che rappresenta l'operazione di login.
 - → **ToPublisherAndUserIsAliveRequest**: classe per il messaggio isAlive?.
 - → **ToRequesterIsAliveRequest**: classe per la richiesta di esistenza dell'UserActor associato al user in input.
 - → **STimeout**: classe che rappresenta i timeout delle richieste all'interno dell'applicazione.
 - → **PublisherActor**: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto **usermanager**.

4.47.2.3 sgad::servertier::presentation::usermanager::IsUserActorAliveResponder

- **Descrizione:** classe per la risposta della richiesta di esistenza dell'UserActor in modo asincrono.
- **Utilizzo:** la classe viene utilizzata per verificare in modo asincrono l'esistenza dell'UserActor.
- **Relazioni con altre classi**
 - ← **PublisherActor**: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto **usermanager**.
 - → **akka::contrib::pattern::DistributedPubSubExtension**.
 - → **akka::contrib::pattern::DistributedPubSubMediator**.
 - → **ToPublisherAndUserIsAliveRequest**: classe per il messaggio isAlive?.

4.47.2.4 sgad::servertier::presentation::usermanager::LoginActor

- **Descrizione:** classe per la gestione delle richieste di login.

- **Utilizzo:** viene utilizzata per controllare i dati di una richiesta di login. Nel caso di login positivo essa istanzia un attore per l'utente e lo registra presso PublisherActor del server corretto.
- **Classi ereditate**
 - akka::actors::Actor
- **Relazioni con altre classi**
 - ← WorkerActor: classe per la gestione asincrona delle richieste in arrivo dai client.
 - → Logic: classe per la gestione dell'operazione richiesta dall'utente. Essa si occupa di instradare la richiesta tramite l'invocazione dell'operazione giusta.
 - → ToLoginActorRequest: classe per il messaggio di elaborazione di una richiesta di login al LoginActor.

4.47.2.5 sgad::servertier::presentation::usermanager::PublisherActor

- **Descrizione:** classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto usermanager.
- **Utilizzo:** viene utilizzato da per l'inoltro della richiesta del client all'attore corretto in grado di gestirla e rispondere. È in grado di capire dove trovare un attore indipendentemente da quale server lo ospita.
- **Classi ereditate**
 - akka::actors::Actor
- **Relazioni con altre classi**
 - ← Application: classe per la gestione del bootstrap dell'applicativo lato server.
 - ← GetAllServerData: classe per la fornitura delle informazioni sullo stato di tutto il cluster.
 - ← InternalRequester: classe di gestione della richiesta di user ad autorizzazione interna incapsulandone l'algoritmo.
 - ← IsUserActorAliveRequester: attore che gestisce la risoluzione della richiesta di esistenza di altri UserActor.
 - → akka::contrib::pattern::DistributedPubSubExtension.
 - → akka::contrib::pattern::DistributedPubSubMediator.
 - → ToPublisherAndUserIsAliveRequest: classe per il messaggio isAlive?.
 - → ToPublisherAndUserRequest: classe per il messaggio di elaborazione di una richiesta di gioco al PublisherActor.

- → `IsUserActorAliveResponder`: classe per la risposta della richiesta di esistenza dell'UserActor in modo asincrono.
- → `UserActor`: classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.

4.47.2.6 `sgad::servertier::presentation::usermanager::RegistrationActor`

- **Descrizione:** classe per la gestione delle richieste di registrazione.
- **Utilizzo:** viene utilizzata per la gestione delle richieste di registrazione. Essa controllerà la validità dei dati e nel caso procederà con la scrittura nel database.
- **Classi ereditate**
 - `akka::actors::Actor`
- **Relazioni con altre classi**
 - ← `WorkerActor`: classe per la gestione asincrona delle richieste in arrivo dai client.
 - → `Logic`: classe per la gestione dell'operazione richiesta dall'utente. Essa si occupa di instradare la richiesta tramite l'invocazione dell'operazione giusta.
 - → `ToRegistrationActorRequest`: classe per il messaggio di elaborazione di una richiesta di registrazione al RegistrationActor.

4.47.2.7 `sgad::servertier::presentation::usermanager::UserActor`

- **Descrizione:** classe per la gestione della sessione di un particolare utente. Essa è l'unico accesso per le richieste di manipolazione ai dati di quell'utente.
- **Utilizzo:** viene usata per effettuare le operazioni richieste sui dati di ogni utente. Essa si occupa di caricare i dati dell'utente, di tenerli aggiornati e infine salvarli sul database. Unificando la via di accesso ai dati dell'utente si eliminano potenziali inconsistenze a causa di modifiche concorrenti. La sua logica interna si occuperà anche di controllare la fonte e la validità delle richieste in arrivo prima di eseguirle.

- **Classi ereditate**
 - `akka::actors::Actor`
- **Relazioni con altre classi**
 - ← `Application`: classe per la gestione del bootstrap dell'applicativo lato server.
 - ← `InternalLogin`: classe che rappresenta l'operazione di login.
 - ← `Login`: classe che rappresenta l'operazione di login.

- ← **Logout**: classe che rappresenta l'operazione di logout.
- ← **PublisherActor**: classe che gestisce l'inoltro dei messaggi alle istanze giuste delle classi (attori Akka) del pacchetto **usermanager**.
- → **akka::contrib::pattern::DistributedPubSubExtension**.
- → **akka::contrib::pattern::DistributedPubSubMediator**.
- → **Logic**: classe per la gestione dell'operazione richiesta dall'utente. Essa si occupa di instradare la richiesta tramite l'invocazione dell'operazione giusta.
- → **ToPublisherAndUserRequest**: classe per il messaggio di elaborazione di una richiesta di gioco al PublisherActor.
- → **UserActorTimeoutMessage**: classe che rappresenta il messaggio di timeout verso l'UserActor a cui l'attore deve reagire uccidendosi.
- → **STimeout**: classe che rappresenta i timeout delle richieste all'interno dell'applicazione.

5 Design Pattern

Verranno in seguito presentati i vari design pattern utilizzati per la progettazione architettonale.

I design pattern sono soluzioni a problemi ricorrenti. Adottarli porta diversi benefici:

- favorisce il riutilizzo del codice;
- semplifica l'attività di progettazione;
- rende l'architettura più manutenibile.

I design pattern possono essere suddivisi in:

- **architetturali**: definiscono l'architettura dell'applicazione ad un livello elevato;
- **creazionali**: permettono di nascondere i costruttori delle classi, consentendo la creazione di oggetti senza conoscerne la loro implementazione;
- **strutturali**: consentono di riutilizzare classi preesistenti, fornendo un'interfaccia più adatta;
- **comportamentali**: definiscono soluzioni per le interazioni tra oggetti.

Per una descrizione più approfondita dei design pattern utilizzati si faccia riferimento all'appendice A. I vari diagrammi che riprendono l'architettura non espongono tutte le sottoclassi e i metodi delle stesse. Lo scopo dei diagrammi è di mostrare le caratteristiche del design pattern adottato e come le varie classi interagiscono tra di loro.

Nella realizzazione del progetto SGAD si è deciso di implementare i seguenti design pattern.

5.1 Design pattern architetturali

5.1.1 DAO

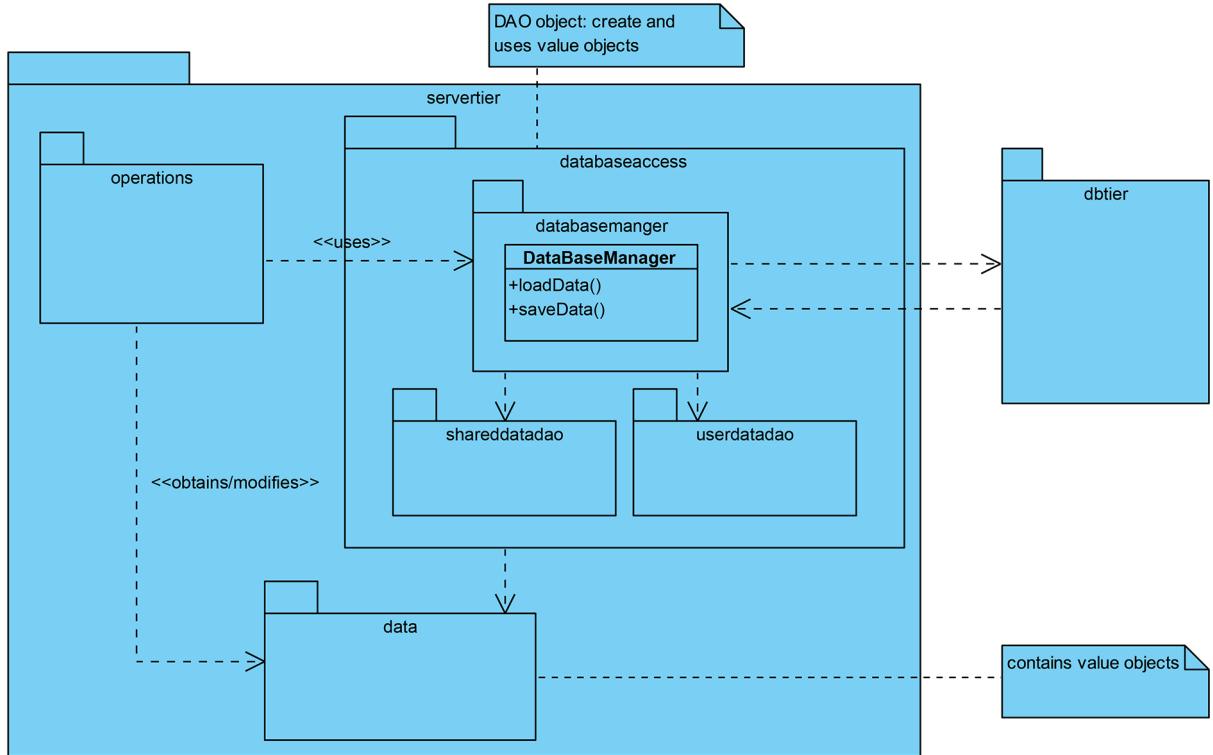


Figura 59: Diagramma I del design pattern DAO in SGAD

- **Scopo dell'utilizzo:** l'intento del pattern DAO (Data Access Object) è di disaccoppiare la logica di business dalla logica di accesso al database. Questo si ottiene spostando la logica di accesso ai dati dai componenti di business ad una classe $DAO_{[g]}$ rendendo le componenti della prima indipendenti dalla natura del dispositivo di persistenza. Questo approccio garantisce che un eventuale cambiamento del dispositivo di persistenza non comporti modifiche sui componenti di business. Inoltre legando la logica di accesso a un database ad una particolare istanza di un oggetto DAO, si favorisce la manutenibilità del sistema.
- **Contesto di utilizzo:** è stato usato per l'accesso ad database MongoDB usato per la persistenza dei dati. La creazione di un oggetto avviene tramite la lettura del DataBaseManager sul database e l'invocazione del metodo dell'oggetto DAO associato che si occupa della creazione dell'oggetto. Esiste un oggetto DAO per ogni classe del pacchetto data. Le classi DAO espongono metodi dell'interfaccia del database poiché essi sono rappresentazioni di stringhe JSON veloci e ben implementate. Questa scelta permette il vantaggio di utilizzare queste classi anche in contesti nei quali serve la conversione in JSON degli oggetti del pacchetto data (esempio: la comunicazione con

il client).

Vi partecipano i componenti del pacchetto **databaseaccess**.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::servtier::dataaccess::databaseaccess::databasemanager::DataBaseManager*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::BonusDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::BuildingWithLevelDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::CostDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::ProducedResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::QuantityResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::ResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::UnitDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::AuthenticationDataDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::BuildingPossessionDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::OwnedResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::PositionDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::UnitInProgressDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::UnitPossessionDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::UserDataDAO*

5.1.2 MVC

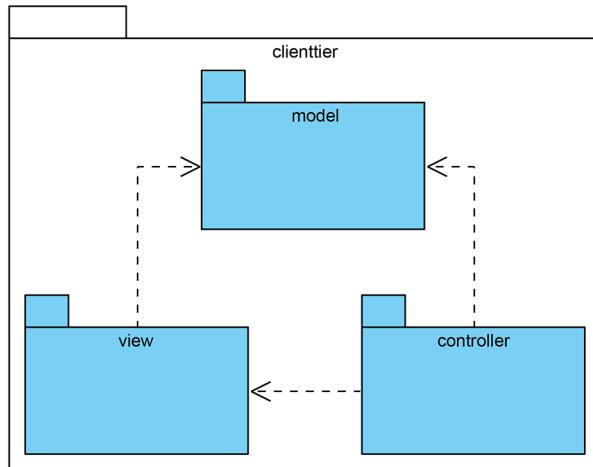


Figura 60: Diagramma I del design pattern MVC in SGAD

- **Scopo dell'utilizzo:** è stato scelto il pattern MVC per la separazione del modello di dati, la parte di logica e le viste.
- **Contesto di utilizzo:** è utilizzato dall'architettura generale dell'applicativo client. Ogni modifica sul model fa scattare il meccanismo di aggiornamento sulle view, le quali andranno a recuperarsi i dati per il corretto aggiornamento dal model. L'implementazione del meccanismo di aggiornamento segue il pattern Observer.

Per implementare tale design pattern non è stato adottato un particolare framework. Tale decisione si basa sulle caratteristiche dei vari framework che offrono l'implementazione di tale design pattern. Molti framework si associano al DOM della pagina. Le nostre esigenze invece ricadono nell'utilizzo di un unico elemento quale un canvas. I framework non risultano quindi adatti a modellare tale design pattern secondo le nostre necessità. Altri framework proponevano la gestione dell'elemento canvas ma con un approccio poco maturo oppure con uno scopo che non è idoneo ai nostri obiettivi.

Nel contesto JavaScript il caricamento delle viste avverrà mediante chiamate a metodi delle classi componenti la parte di View, e consisterà nella costruzione o nell'aggiornamento di componenti grafiche.

5.1.3 Three-Tier

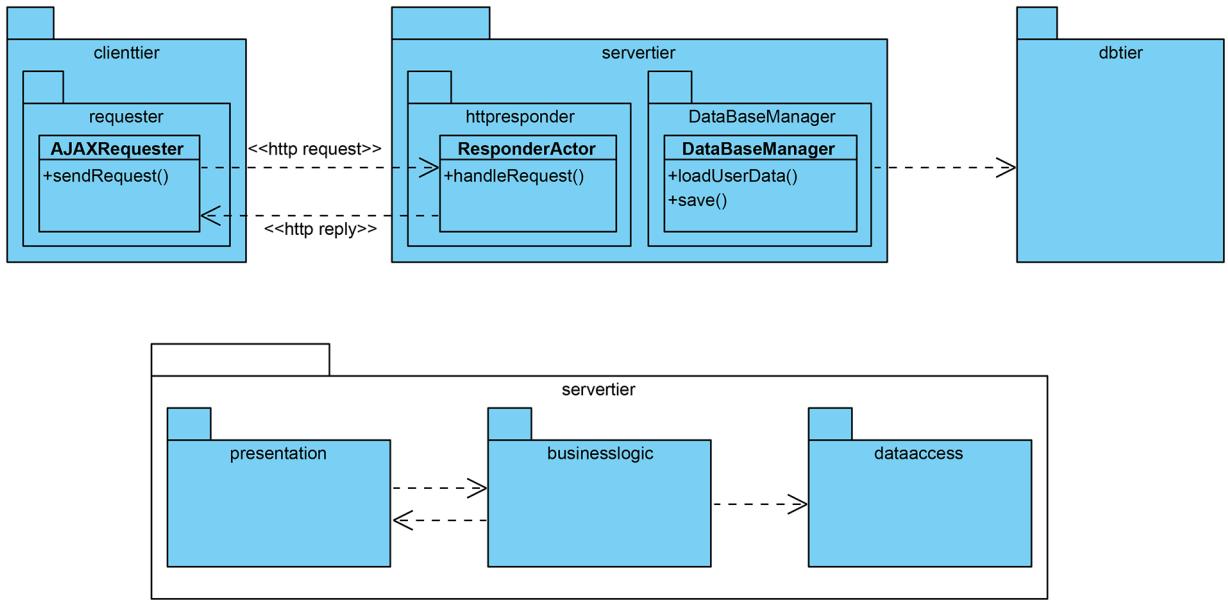


Figura 61: Diagramma I del design pattern Three-Tier in SGAD

- **Scopo dell'utilizzo:** il pattern Three-Tier si pone di dividere l'architettura su tre livelli: uno per il data model, uno per la business logic e l'ultimo per la presentazione. Il pattern impone comunicazioni solo fra livelli adiacenti.
- **Contesto di utilizzo:** è stato utilizzato dall'architettura generale identificando tre livelli quali: applicativo lato client, applicativo lato server, database.

È stato utilizzato anche dall'architettura generale dell'applicativo client. La presentazione rappresenta il punto di connessione con l'applicativo lato client. Il data model rappresenta la comunicazione con il database.

5.2 Design pattern creazionali

5.2.1 Factory Method

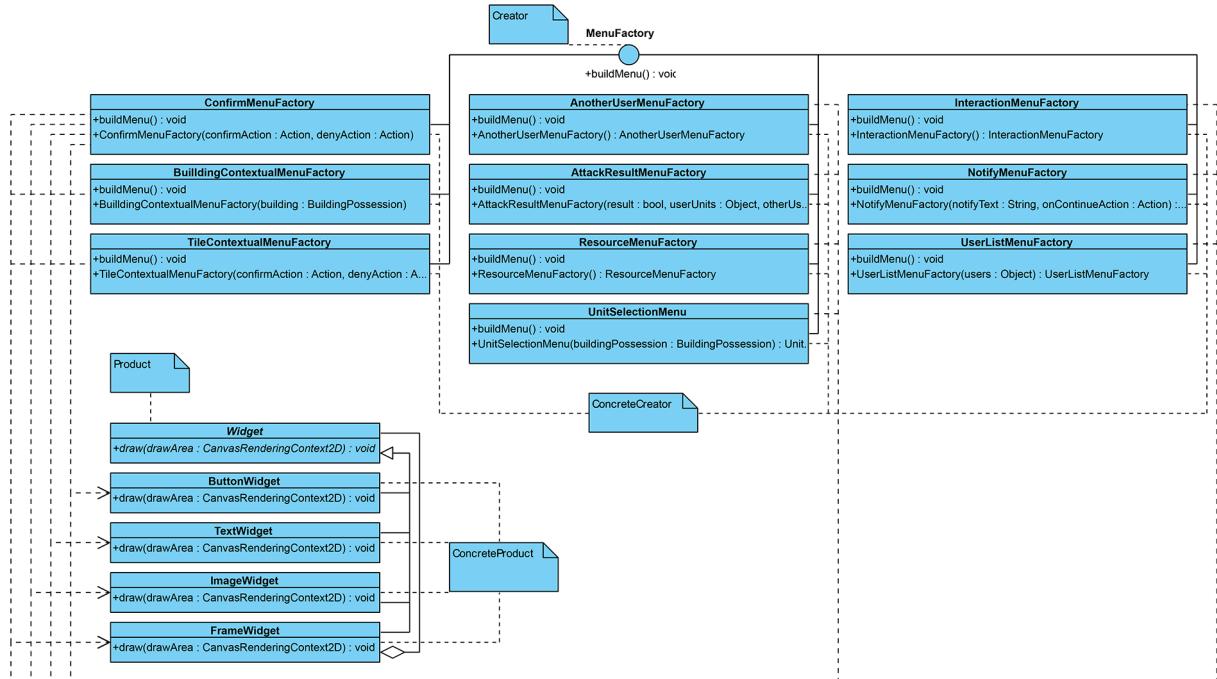


Figura 62: Diagramma I del design pattern Factory Method in SGAD

- **Scopo dell'utilizzo:** il pattern Factory Method definisce un'interfaccia per la creazione di un oggetto, lasciando alle sottoclassi la decisione sulla classe che deve essere istanziata. Il pattern consente di definire l'istanziazione di una classe alle sottoclassi.
- **Contesto di utilizzo:** viene utilizzato per delegare alle sottoclassi la realizzazione delle varie interfacce grafiche.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::controller::menufactory::AccountManagerMenuFactory*
- *sgad::clienttier::controller::menufactory::AnotherUserMenuFactory*
- *sgad::clienttier::controller::menufactory::AttackResultMenuFactory*
- *sgad::clienttier::controller::menufactory::BuildingContextualMenuFactory*
- *sgad::clienttier::controller::menufactory::ConfirmMenuFactory*
- *sgad::clienttier::controller::menufactory::InteractionMenuFactory*
- *sgad::clienttier::controller::menufactory::MenuFactory*
- *sgad::clienttier::controller::menufactory::NotifyMenuFactory*
- *sgad::clienttier::controller::menufactory::ResourceMenuFactory*

- *sgad::clienttier::controller::menufactory::TileContextualMenuFactory*
- *sgad::clienttier::controller::menufactory::UnitSelectionMenu*
- *sgad::clienttier::controller::menufactory::UserListMenuFactory*
- *sgad::clienttier::view::graphicobjects::widget::ButtonWidget*
- *sgad::clienttier::view::graphicobjects::widget::FrameWidget*
- *sgad::clienttier::view::graphicobjects::widget::ImageWidget*
- *sgad::clienttier::view::graphicobjects::widget::TextWidget*
- *sgad::clienttier::view::graphicobjects::widget::Widget*

5.2.2 Singleton

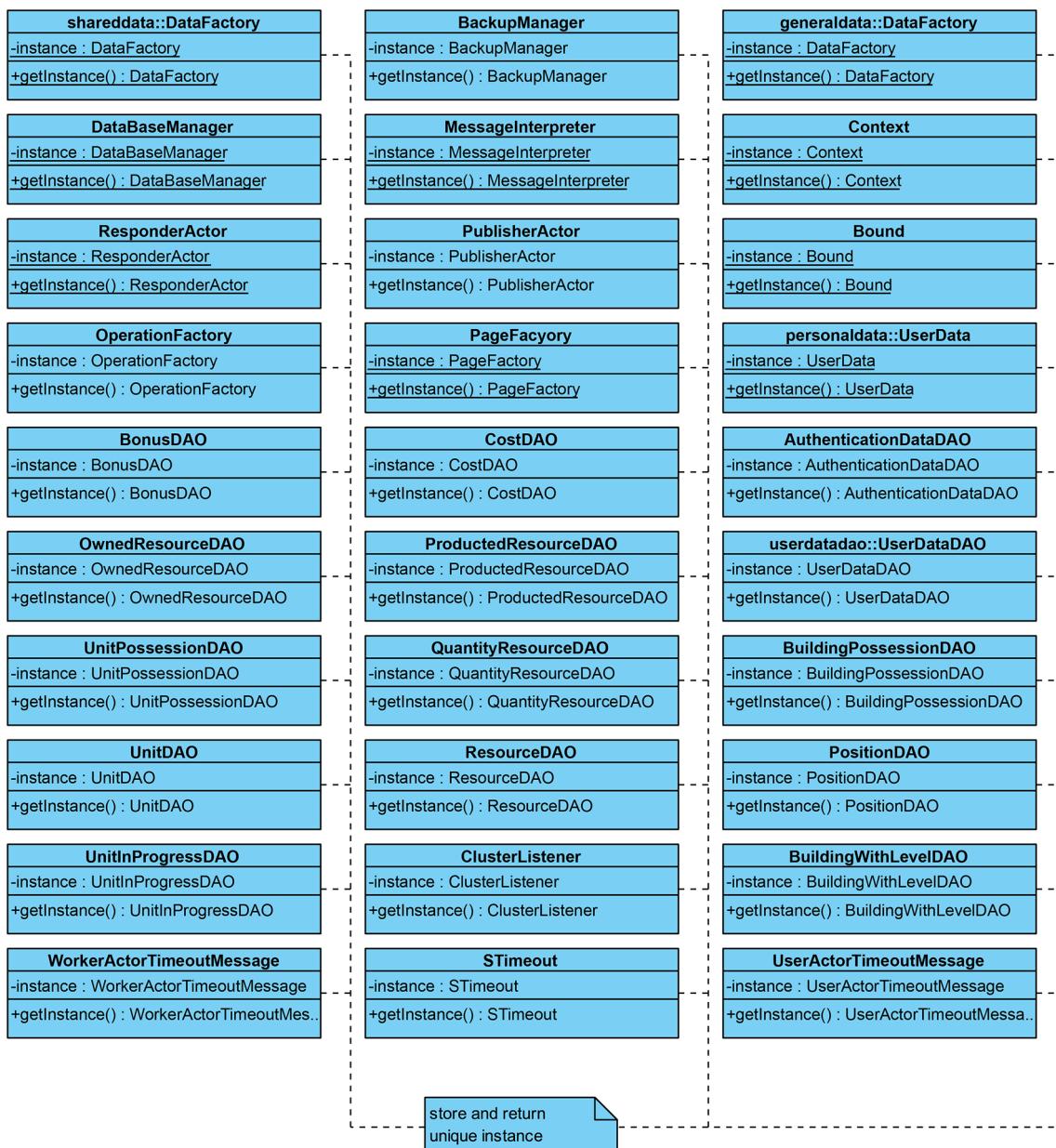


Figura 63: Diagramma I del design pattern Singleton in SGAD

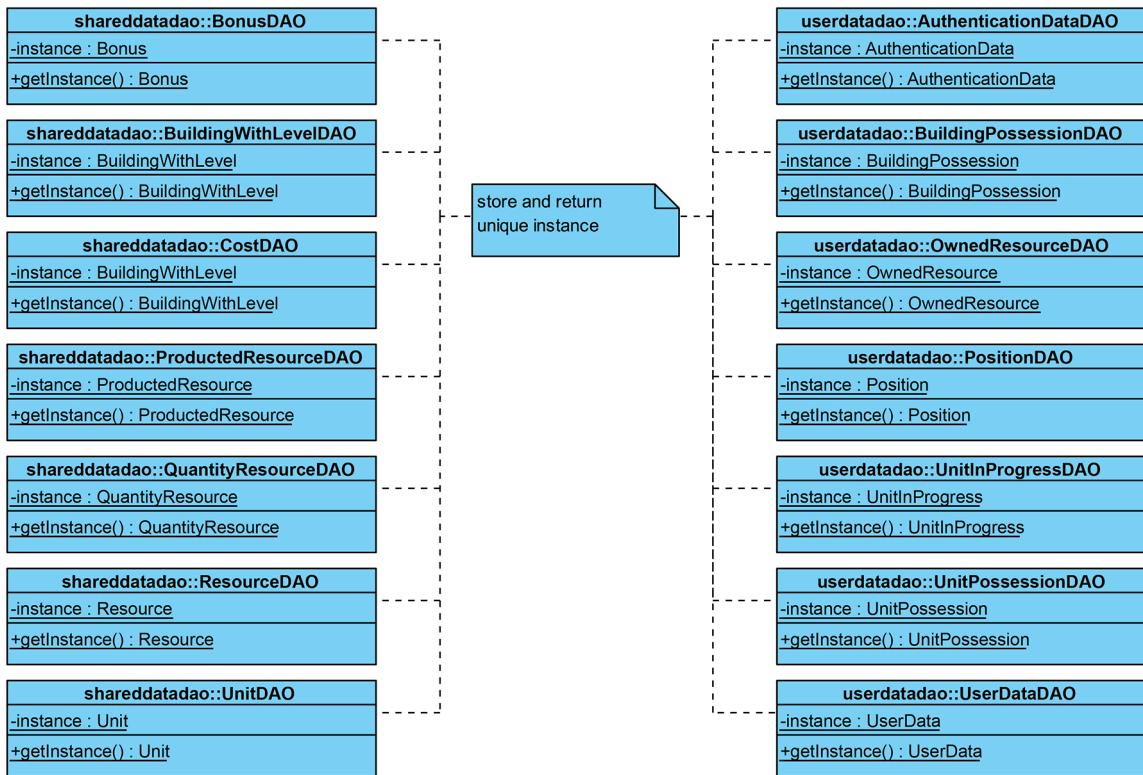


Figura 64: Diagramma II del design pattern Singleton in SGAD

- **Scopo dell'utilizzo:** il pattern Singleton assicura che una classe abbia una sola istanza e fornisce un punto d'acceso globale a tale istanza.
- **Contesto di utilizzo:** è stato usato per assicurare la presenza di una sola istanza delle classi, indicate in seguito, che implementano questo pattern.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::controller::backupmanager::BackupManager*
- *sgad::clienttier::controller::messageinterpreter::MessageInterpreter*
- *sgad::clienttier::model::generaldata::DataFactory*
- *sgad::clienttier::model::personaldatas::UserData*
- *sgad::clienttier::view::context::Context*
- *sgad::clienttier::view::graphicobjects::bound::Bound*
- *sgad::servertier::businesslogic::operations::OperationFactory*
- *sgad::servertier::dataaccess::data::shareddata::DataFactory*
- *sgad::servertier::dataaccess::databaseaccess::databasemanager::DataBaseManager*
- *sgad::servertier::dataaccess::databaseaccess::shareddata dao::BonusDAO*
- *sgad::servertier::dataaccess::databaseaccess::shareddata dao::BuildingWithLevelDAO*

- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::CostDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::ProducedResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::QuantityResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::ResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::shareddatadao::UnitDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::AuthenticationDataDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::BuildingPossessionDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::OwnedResourceDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::PositionDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::UnitInProgressDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::UnitPossessionDAO*
- *sgad::servtier::dataaccess::databaseaccess::userdatadao::UserDataDAO*
- *sgad::servtier::presentation::cluster::ClusterListener*
- *sgad::servtier::presentation::httpresponder::ResponderActor*
- *sgad::servtier::presentation::messages::UserActorTimeoutMessage*
- *sgad::servtier::presentation::messages::WorkerActorTimeoutMessage*
- *sgad::servtier::presentation::pagemanager::PageFactory*
- *sgad::servtier::presentation::timeout::STimeout*
- *sgad::servtier::presentation::usermanager::IsUserActorAliveRequester*
- *sgad::servtier::presentation::usermanager::PublisherActor*
- *sgad::servtier::presentation::usermanager::UserActor*

5.3 Design pattern strutturali

5.3.1 Composite

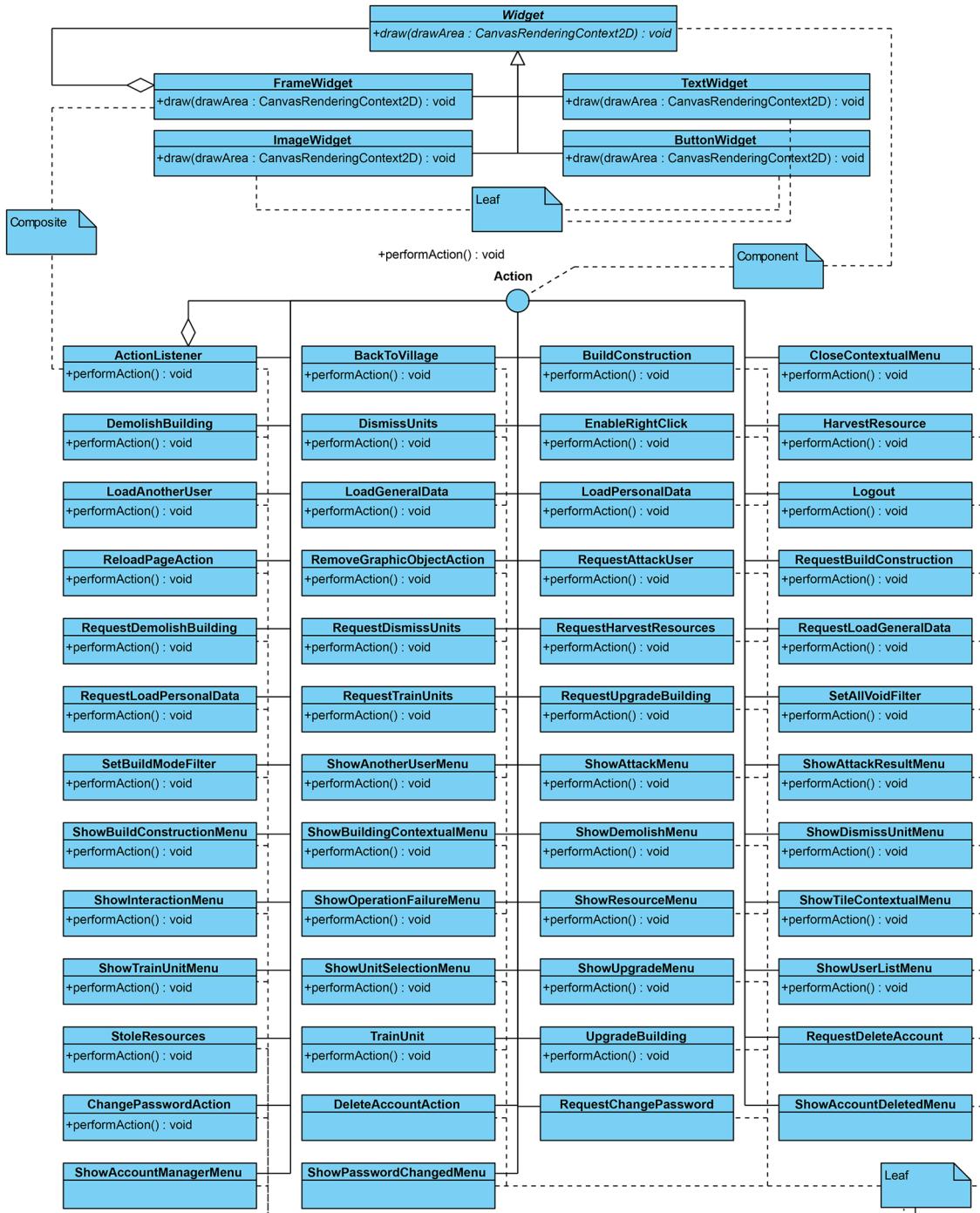


Figura 65: Diagramma I del design pattern Composite in SGAD

- **Scopo dell'utilizzo:** il pattern Composite permette di comporre oggetti in strutture ad albero per rappresentare gerarchie parte-tutto e consentire ai client di trattare oggetti singoli e composizioni in modo uniformi.
- **Contesto di utilizzo:** viene utilizzato per la gestione delle azioni e dei widget. Ciò permette di associare generiche azioni o widget ad oggetti senza che questi conoscano se stanno interagendo con insiemi di azioni o widget oppure con singole entità.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::controller::actions::Action*
- *sgad::clienttier::controller::actions::ActionListener*
- *sgad::clienttier::controller::actions::BackToVillage*
- *sgad::clienttier::controller::actions::BuildConstruction*
- *sgad::clienttier::controller::actions::ChangePasswordAction*
- *sgad::clienttier::controller::actions::CloseContextualMenu*
- *sgad::clienttier::controller::actions::DeleteAccountAction*
- *sgad::clienttier::controller::actions::DemolishBuilding*
- *sgad::clienttier::controller::actions::DismissUnits*
- *sgad::clienttier::controller::actions::EnableRightClick*
- *sgad::clienttier::controller::actions::HarvestResources*
- *sgad::clienttier::controller::actions::LoadAnotherUser*
- *sgad::clienttier::controller::actions::LoadGeneralData*
- *sgad::clienttier::controller::actions::LoadPersonalData*
- *sgad::clienttier::controller::actions::Logout*
- *sgad::clienttier::controller::actions::ReloadPageAction*
- *sgad::clienttier::controller::actions::RemoveGraphicObjectAction*
- *sgad::clienttier::controller::actions::RequestAttackUser*
- *sgad::clienttier::controller::actions::RequestBuildConstruction*
- *sgad::clienttier::controller::actions::RequestChangePassword*
- *sgad::clienttier::controller::actions::RequestDeleteAccount*
- *sgad::clienttier::controller::actions::RequestDemolishBuilding*
- *sgad::clienttier::controller::actions::RequestDismissUnits*
- *sgad::clienttier::controller::actions::RequestHarvestResources*
- *sgad::clienttier::controller::actions::RequestLoadGeneralData*
- *sgad::clienttier::controller::actions::RequestLoadPersonalData*
- *sgad::clienttier::controller::actions::RequestTrainUnits*
- *sgad::clienttier::controller::actions::RequestUpgradeBuilding*
- *sgad::clienttier::controller::actions::SetAllVoidFilter*
- *sgad::clienttier::controller::actions::SetBuildModeFilter*

- *sgad::clienttier::controller::actions::ShowAccountDeletedMenu*
- *sgad::clienttier::controller::actions::ShowAccountManagerMenu*
- *sgad::clienttier::controller::actions::ShowAnotherUserMenu*
- *sgad::clienttier::controller::actions::ShowAttackMenu*
- *sgad::clienttier::controller::actions::ShowAttackResultMenu*
- *sgad::clienttier::controller::actions::ShowBuildConstructionMenu*
- *sgad::clienttier::controller::actions::ShowBuildingContextualMenu*
- *sgad::clienttier::controller::actions::ShowDemolishMenu*
- *sgad::clienttier::controller::actions::ShowDismissUnitMenu*
- *sgad::clienttier::controller::actions::ShowInteractionMenu*
- *sgad::clienttier::controller::actions::ShowOperationFailureMenu*
- *sgad::clienttier::controller::actions::ShowResourceMenu*
- *sgad::clienttier::controller::actions::ShowTileContextualMenu*
- *sgad::clienttier::controller::actions::ShowTrainUnitMenu*
- *sgad::clienttier::controller::actions::ShowUnitSelectionMenu*
- *sgad::clienttier::controller::actions::ShowUpgradeMenu*
- *sgad::clienttier::controller::actions::ShowUserListMenu*
- *sgad::clienttier::controller::actions::StealResources*
- *sgad::clienttier::controller::actions::TrainUnit*
- *sgad::clienttier::controller::actions::UpgradeBuilding*
- *sgad::clienttier::view::graphicobjects::widget::ButtonWidget*
- *sgad::clienttier::view::graphicobjects::widget::FrameWidget*
- *sgad::clienttier::view::graphicobjects::widget::ImageWidget*
- *sgad::clienttier::view::graphicobjects::widget::TextWidget*
- *sgad::clienttier::view::graphicobjects::widget::Widget*

5.3.2 Flyweight

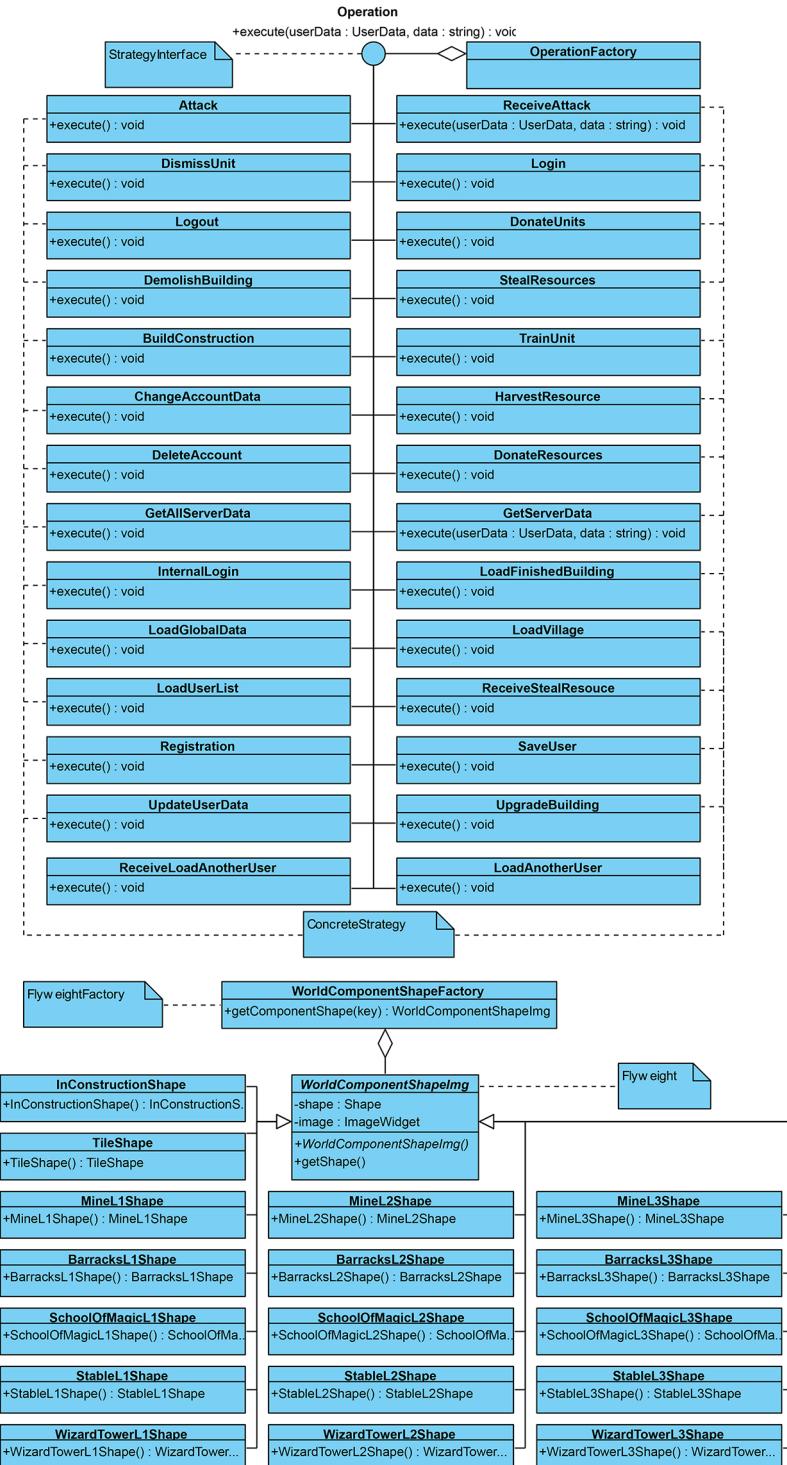


Figura 66: Diagramma I del design pattern Flyweight in SGAD

- **Scopo dell'utilizzo:** il design Flyweight permette la condivisione allo scopo di supportare in modo efficiente un gran numero di oggetti a granularità fine.
- **Contesto di utilizzo:** nel server è usato insieme al pattern Strategy per condividere le istanze delle operazioni concrete da eseguire. Permette inoltre di istanziare le classi solo quando effettivamente richieste. Nel client viene impiegato per gestire la condivisione delle forme che costituiscono gli edifici o le unità. Essendo una forma identica per ogni edificio dello stesso tipo, questo permette di ridurre la quantità di memoria utilizzata per mantenere il poligono e l'immagine.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::Barracks-L1Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::Barracks-L2Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::Barracks-L3Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::InConstructionShape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::MineL1-Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::MineL2-Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::MineL3-Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::School-OfMagicL1Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::School-OfMagicL2Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::School-OfMagicL3Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::StableL1-Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::StableL2-Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::StableL3-Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::TileShape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::Wizard-TowerL1Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::Wizard-TowerL2Shape*

- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WizardTowerL3Shape*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WorldComponentShapeFactory*
- *sgad::clienttier::view::graphicobjects::components::worldcomponentshape::WorldComponentShapeImg*
- *sgad::servertier::businesslogic::operations::Attack*
- *sgad::servertier::businesslogic::operations::BuildConstruction*
- *sgad::servertier::businesslogic::operations::ChangeAccountData*
- *sgad::servertier::businesslogic::operations::DeleteAccount*
- *sgad::servertier::businesslogic::operations::DemolishBuilding*
- *sgad::servertier::businesslogic::operations::DismissUnit*
- *sgad::servertier::businesslogic::operations::DonateResources*
- *sgad::servertier::businesslogic::operations::DonateUnits*
- *sgad::servertier::businesslogic::operations::GetAllServerData*
- *sgad::servertier::businesslogic::operations::GetServerData*
- *sgad::servertier::businesslogic::operations::HarvestResource*
- *sgad::servertier::businesslogic::operations::InternalLogin*
- *sgad::servertier::businesslogic::operations::LoadGlobalData*
- *sgad::servertier::businesslogic::operations::LoadVillage*
- *sgad::servertier::businesslogic::operations::LoadUserList*
- *sgad::servertier::businesslogic::operations::Login*
- *sgad::servertier::businesslogic::operations::Logout*
- *sgad::servertier::businesslogic::operations::Operation*
- *sgad::servertier::businesslogic::operations::OperationFactory*
- *sgad::servertier::businesslogic::operations::ReceiveAttack*
- *sgad::servertier::businesslogic::operations::ReceiveStealResource*
- *sgad::servertier::businesslogic::operations::Registration*
- *sgad::servertier::businesslogic::operations::SaveUser*
- *sgad::servertier::businesslogic::operations::StealResource*
- *sgad::servertier::businesslogic::operations::TrainUnit*
- *sgad::servertier::businesslogic::operations::UpdateUserData*
- *sgad::servertier::businesslogic::operations::UpgradeBuilding*

5.4 Design pattern comportamentali

5.4.1 Command

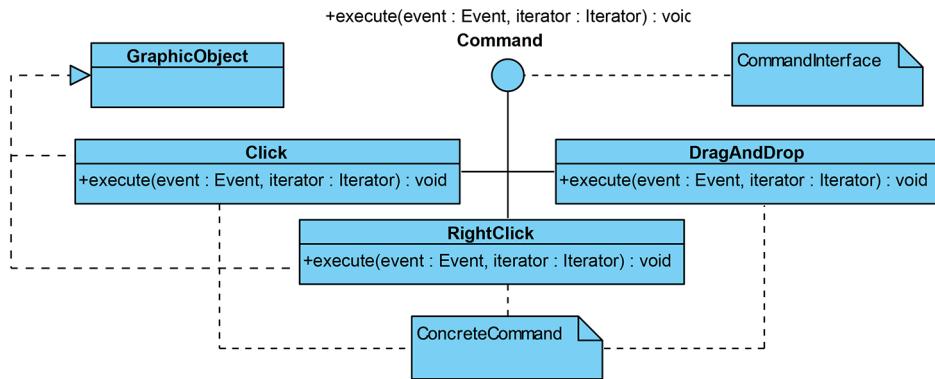


Figura 67: Diagramma I del design pattern Command in SGAD

- **Scopo dell'utilizzo:** il pattern Command disaccoppia l'oggetto che invoca un'operazione da quello che conosce come portarla a termine. Di conseguenza incapsula una richiesta in un oggetto permettendo agli oggetti di inoltrare richieste a oggetti sconosciuti dell'applicazione.
- **Contesto di utilizzo:** viene utilizzato per incapsulare la gestione di un evento che si verifica durante l'interazione dell'utente con il mondo di gioco. Uno specifico evento viene gestito da un determinato comando. Ogni comando determinerà se trasferire la richiesta all'oggetto su cui si è verificato l'evento. Oppure la richiesta verrà gestita globalmente.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::view::commands::Click*
- *sgad::clienttier::view::commands::Command*
- *sgad::clienttier::view::commands::DragAndDrop*
- *sgad::clienttier::view::commands::RightClick*
- *sgad::clienttier::view::graphicobjects::graphicobject::GraphicObject*

5.4.2 Iterator

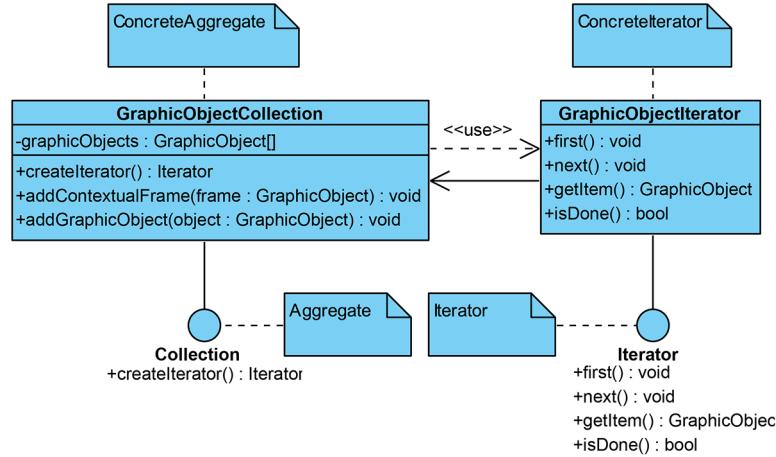


Figura 68: Diagramma I del design pattern Iterator in SGAD

- **Scopo dell'utilizzo:** il pattern Iterator fornisce un modo di accesso sequenziale agli elementi che formano un oggetto composito, senza esporre all'esterno la struttura interna dell'oggetto composito.
- **Contesto di utilizzo:** il pattern viene utilizzato per fornire un accesso sequenziale all'aggregazione di oggetti di tipo **GraphicObject** mantenuta in **GraphicObjectCollection**. Tale classe manterrà una particolare struttura dati per permettere di invocare i metodi di disegno degli oggetti grafici in ordine corretto. Ciò permette di evitare la sovrapposizione inesatta di oggetti che hanno una maggiore profondità rispetto a quelli che ne hanno poca.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::view::graphicobjects::collection::Collection*
- *sgad::clienttier::view::graphicobjects::collection::GraphicObjectCollection*
- *sgad::clienttier::view::graphicobjects::collection::GraphicObjectIterator*
- *sgad::clienttier::view::graphicobjects::collection::Iterator*

5.4.3 Observer

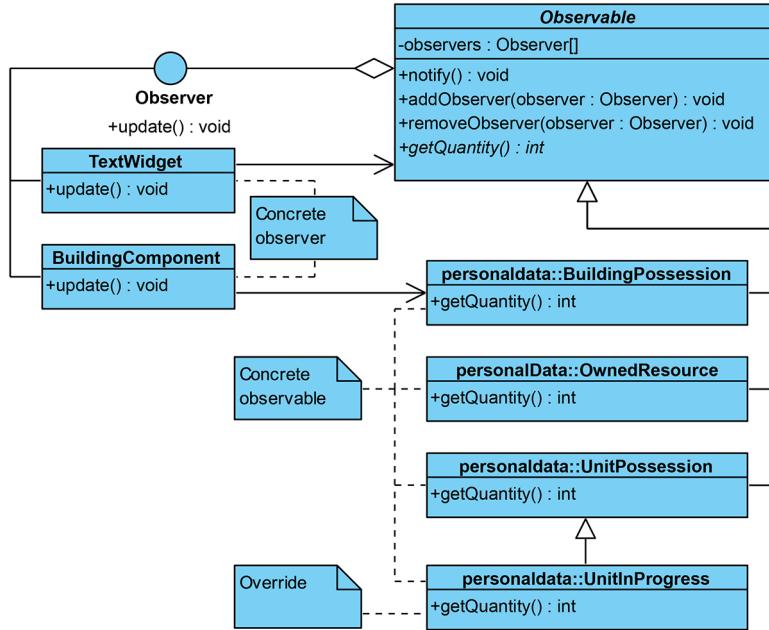


Figura 69: Diagramma I del design pattern Observer in SGAD

- **Scopo dell'utilizzo:** il pattern Observer definisce una dipendenza uno a molti fra oggetti, in modo tale che se un oggetto cambia il proprio stato, tutti gli oggetti dipendenti da questo siano notificati e aggiornati automaticamente.
- **Contesto di utilizzo:** è stato usato per legare l'aggiornamento della view lato client all'aggiornamento del data model lato client. La view alla richiesta di aggiornamento si salverà i nuovi dati aggiornati per procedere velocemente alle continue richieste del ciclo di disegno.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::model::observer::Observable*
- *sgad::clienttier::model::observer::Observer*
- *sgad::clienttier::model::personaldata::BuildingPossession*
- *sgad::clienttier::model::personaldata::OwnedResource*
- *sgad::clienttier::model::personaldata::UnitInProgress*
- *sgad::clienttier::model::personaldata::UnitPossession*
- *sgad::clienttier::view::graphicobjects::components::worldcomponent::BuildingComponent*
- *sgad::clienttier::view::graphicobjects::widget::TextWidget*

5.4.4 State

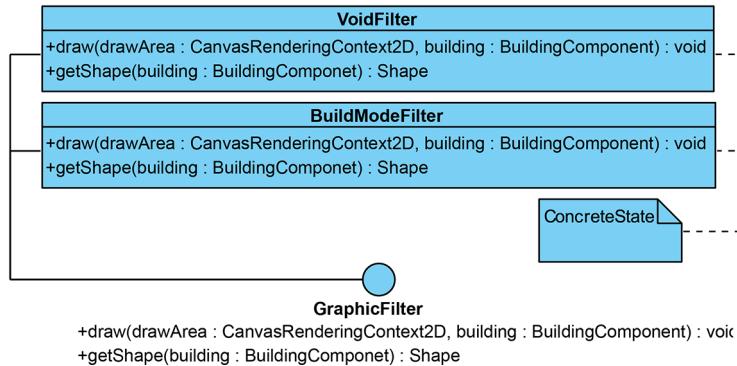


Figura 70: Diagramma I del design pattern State in SGAD

- **Scopo dell'utilizzo:** il pattern State permette a un oggetto di cambiare il proprio comportamento al cambiare del suo stato interno. L'oggetto si comporterà come se avesse cambiato la sua classe.
- **Contesto di utilizzo:** viene utilizzato per la gestione dei filtri per la visualizzazione dei vari edifici. Il pattern State si adatta a tale situazione poiché in base al tipo di filtro, l'oggetto grafico deve rappresentarsi diversamente. Questo può essere visto come un cambiamento di stato.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::clienttier::view::graphicobjects::components::filter::BuildModeFilter*
- *sgad::clienttier::view::graphicobjects::components::filter::GraphicFilter*
- *sgad::clienttier::view::graphicobjects::components::filter::VoidFilter*

5.4.5 Strategy

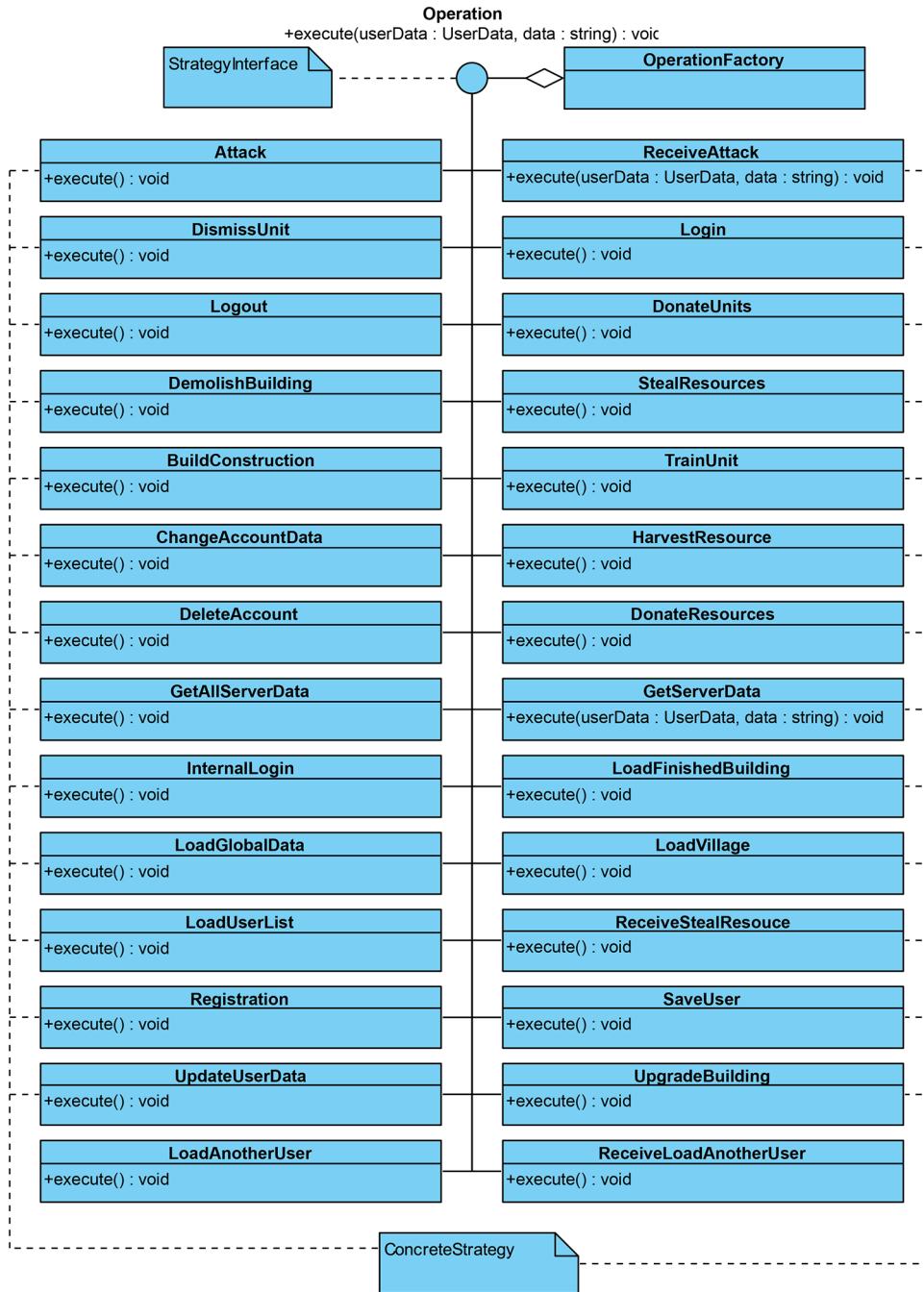


Figura 71: Diagramma I del design pattern Strategy in SGAD

- **Scopo dell'utilizzo:** il pattern Strategy definisce una famiglia di algoritmi, incapsulati e resi intercambiabili. Strategy permette agli algoritmi di variare indipendentemente dai client che ne fanno uso.

- **Contesto di utilizzo:** è usato dalla logica di controllo del server per eseguire operazioni diverse in base al tipo di richiesta del client. Essendo l'operazione un algoritmo diverso per analizzare i dati, il pattern risulta adatto allo scopo. Ogni operazione non mantiene uno stato interno che varia per ogni utente quindi non risulta necessario istanziare un oggetto per effettuare l'operazione per ogni utente. Il pattern viene utilizzato assieme al pattern Flyweight per mantenere un unico oggetto per ogni operazione. Ciò comporta significativi risparmi di spazio.

Partecipano alla realizzazione di questo pattern le seguenti classi:

- *sgad::servertier::businesslogic::operations::Attack*
- *sgad::servertier::businesslogic::operations::BuildConstruction*
- *sgad::servertier::businesslogic::operations::ChangeAccountData*
- *sgad::servertier::businesslogic::operations::DeleteAccount*
- *sgad::servertier::businesslogic::operations::DemolishBuilding*
- *sgad::servertier::businesslogic::operations::DismissUnit*
- *sgad::servertier::businesslogic::operations::DonateResources*
- *sgad::servertier::businesslogic::operations::DonateUnits*
- *sgad::servertier::businesslogic::operations::GetAllServerData*
- *sgad::servertier::businesslogic::operations::GetServerData*
- *sgad::servertier::businesslogic::operations::HarvestResource*
- *sgad::servertier::businesslogic::operations::InternalLogin*
- *sgad::servertier::businesslogic::operations::LoadGlobalData*
- *sgad::servertier::businesslogic::operations::LoadVillage*
- *sgad::servertier::businesslogic::operations::LoadUserList*
- *sgad::servertier::businesslogic::operations::Login*
- *sgad::servertier::businesslogic::operations::Logout*
- *sgad::servertier::businesslogic::operations::Operation*
- *sgad::servertier::businesslogic::operations::OperationFactory*
- *sgad::servertier::businesslogic::operations::ReceiveAttack*
- *sgad::servertier::businesslogic::operations::ReceiveStealResource*
- *sgad::servertier::businesslogic::operations::Registration*
- *sgad::servertier::businesslogic::operations::SaveUser*
- *sgad::servertier::businesslogic::operations::StealResource*
- *sgad::servertier::businesslogic::operations::TrainUnit*
- *sgad::servertier::businesslogic::operations::UpdateUserData*
- *sgad::servertier::businesslogic::operations::UpgradeBuilding*

6 Diagrammi delle attività

In questa sezione vengono rappresentati i diagrammi di attività che descrivono le interazioni dell'utente con il prodotto. È stato illustrato un diagramma generale per descrivere le principali azioni che l'utente può eseguire nel mondo di gioco. Ogni attività viene specificata in un sotto diagramma specifico. Tali attività verranno individuate dal formato grassetto del testo.

6.1 Attività principali dell'utente giocatore

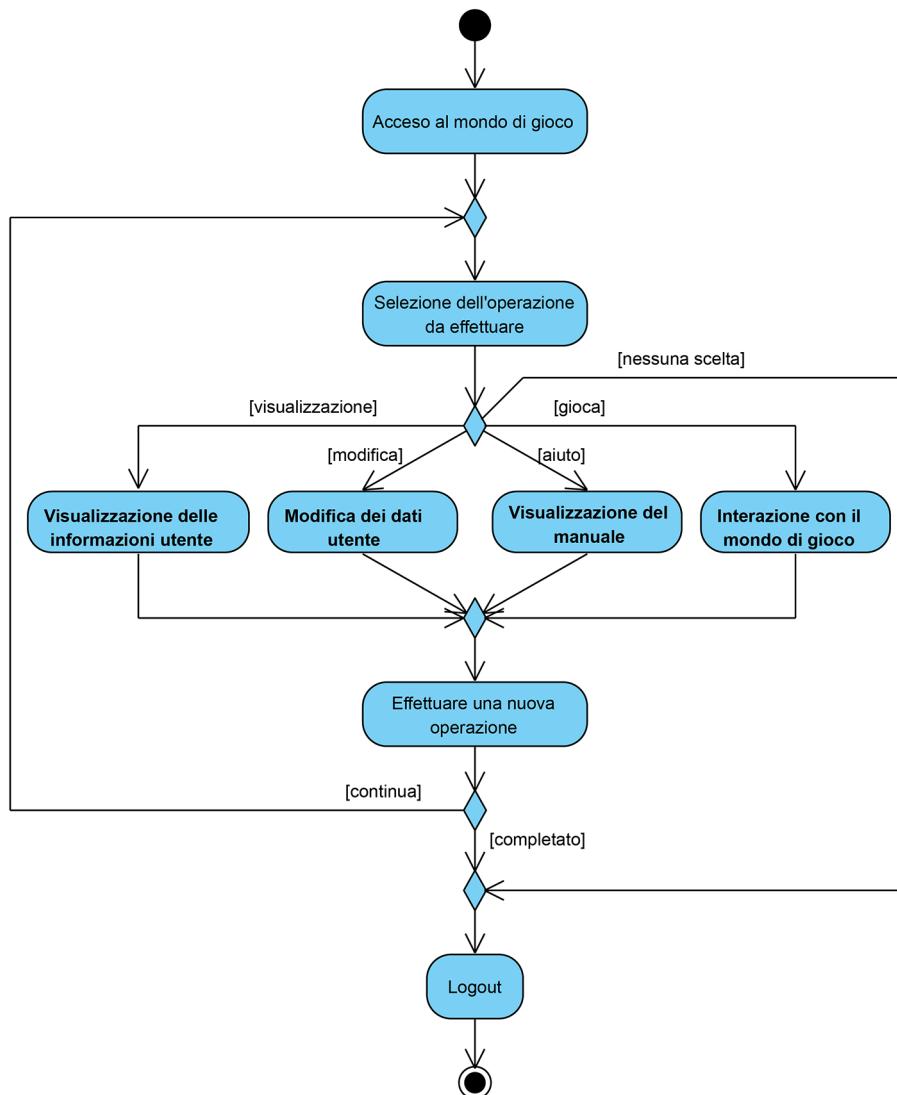


Figura 72: Diagramma delle attività principali dell'utente giocatore

Una volta che l'utente giocatore si è autenticato all'interno del sistema, questi avrà la scelta di che operazione effettuare. La scelta può ricadere tra:

- visualizzazione delle informazioni dell’utente;
- modifica dei dati dell’utente;
- visualizzazione del manuale utente;
- interazione con il mondo di gioco.

Dopo aver individuato l’operazione e averla eseguita, può decidere se proseguire con un’altra azione oppure effettuare il logout dal sistema. L’utente potrebbe non eseguire alcuna operazione. Di conseguenza effettuerà solamente il logout dal sistema.

6.2 Visualizzazione delle informazioni utente

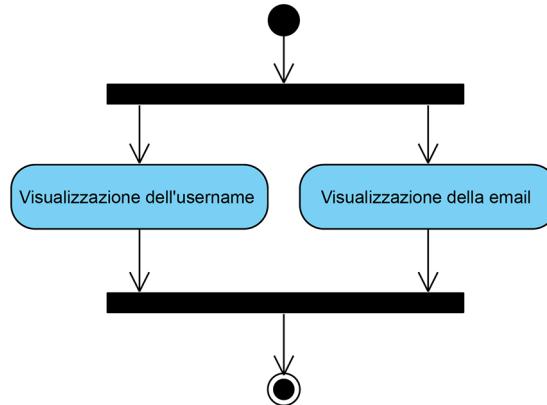


Figura 73: Diagramma per la visualizzazione delle informazioni utente

Vengono visualizzate in parallelo l’username e l’email associate all’account dell’utente.

6.3 Modifica dei dati dell'utente

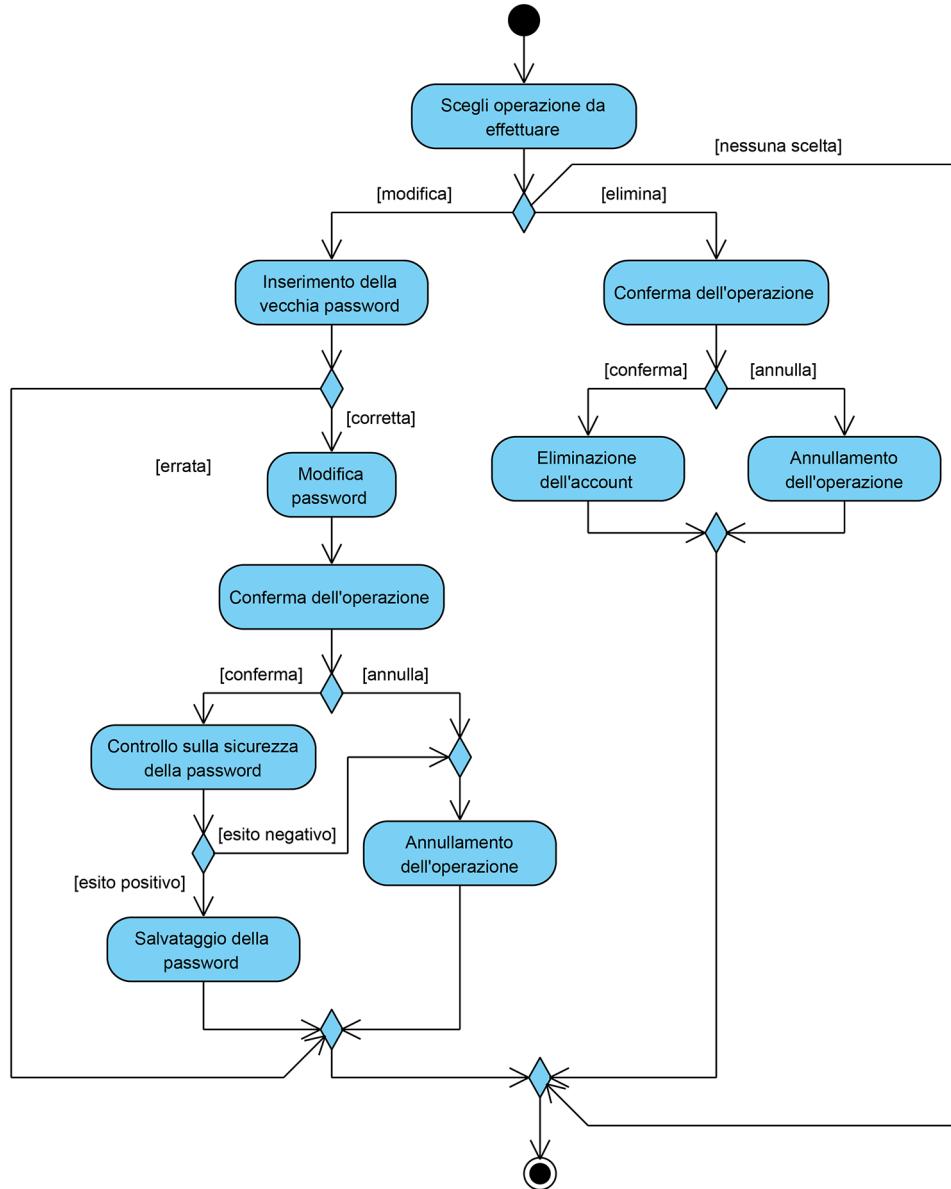


Figura 74: Diagramma per le modifiche dei dati dell'utente

Quando viene scelto di modificare i dati dell'utente viene richiesto se modificare la password o eliminare l'account. Se l'utente richiede di modificare i dati dell'account, viene richiesto l'inserimento della vecchia password. Se l'inserimento è errato verrà terminata l'operazione di modifica. In caso contrario, proseguirà inserendo la nuova password. Se conferma l'operazione il sistema procederà con il controllo che la password soddisfi i requisiti di sicurezza richiesti. Se vengono soddisfatti, si procederà con il salvataggio della password, altrimenti l'azione verrà annullata. Se l'utente non desidera confermare l'operazione, questa

verrà annullata. Se l'utente richiede invece di eliminare l'account dovrà procedere con la conferma dell'operazione. Se confermerà l'azione l'account verrà eliminato, altrimenti la richiesta verrà annullata.

6.4 Visualizzazione del manuale utente giocatore

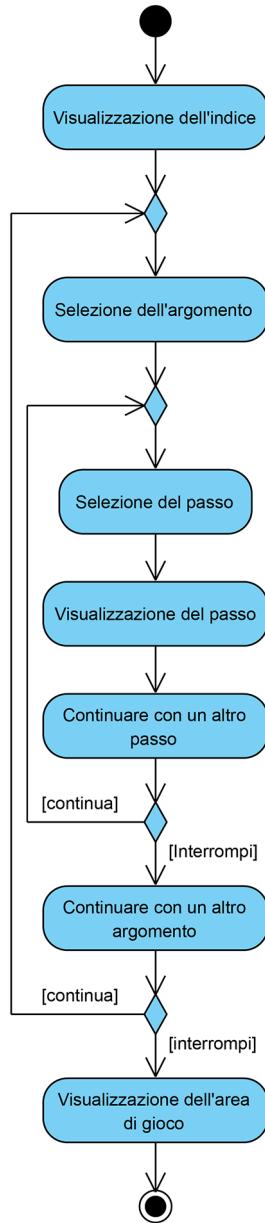


Figura 75: Diagramma per la visualizzazione del manuale utente giocatore

Alla visualizzazione del manuale utente, viene rappresentato l'indice degli argomenti contenuti. L'utente procederà quindi con la scelta dell'argomento e poi del passo specifico.

Dopo aver visualizzato il relativo passo può scegliere se visualizzare un altro passo o ritornare all'argomento. Se sceglie di proseguire con un altro passo o argomento ripeterà le operazioni precedenti. Altrimenti verrà ripresentata l'area di gioco.

6.5 Interazione con il mondo di gioco

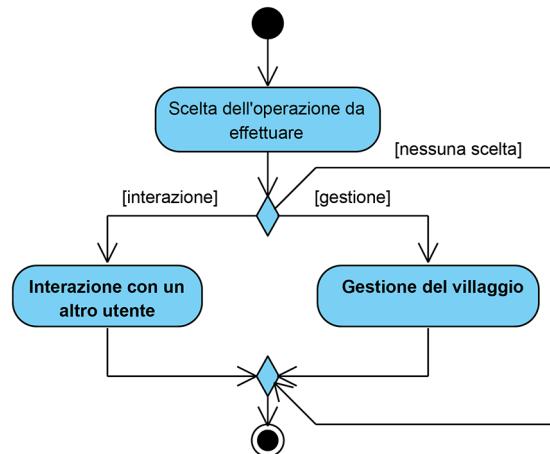


Figura 76: Diagramma per le interazioni con il mondo di gioco

Per interagire con il mondo di gioco l'utente può scegliere se gestire il proprio villaggio o interagire con un altro utente. L'utente potrebbe decidere di non eseguire alcuna operazione.

6.6 Interazione con un altro utente

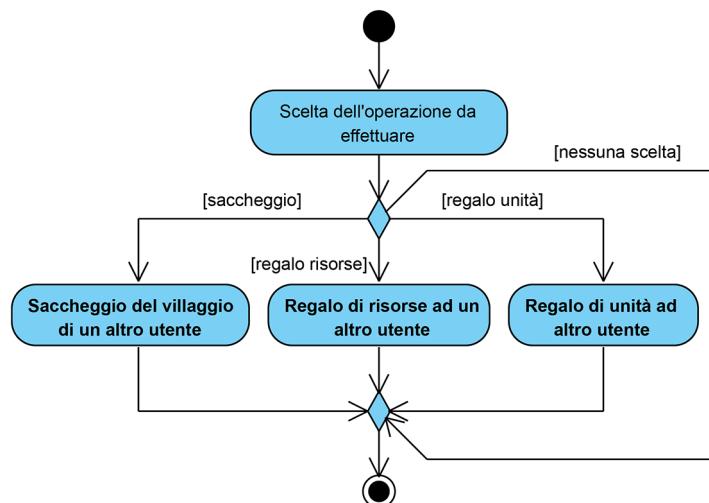


Figura 77: Diagramma per le interazioni con un altro utente

L'interazione con un altro utente richiede la scelta dell'operazione da effettuare. L'utente può decidere di saccheggiare un altro villaggio, regalare risorse oppure unità. L'utente potrebbe decidere di non eseguire alcuna azione.

6.7 Saccheggio del villaggio di un altro utente

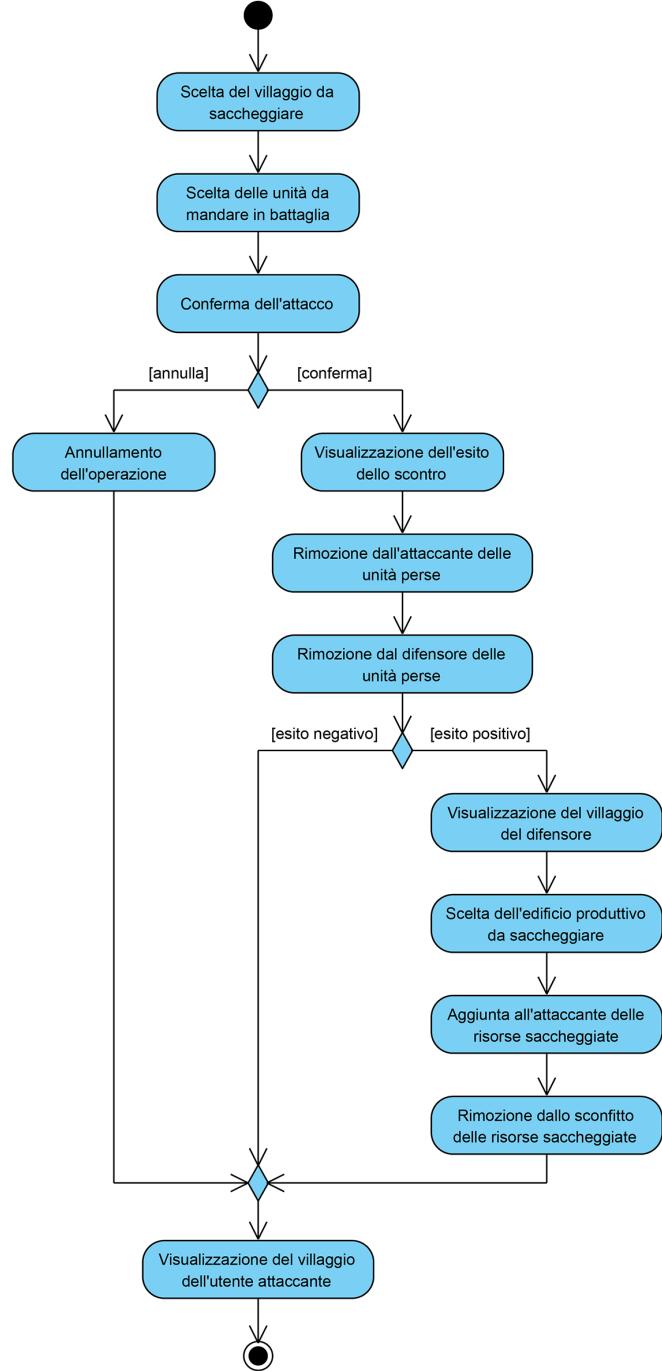


Figura 78: Diagramma per il saccheggio del villaggio di un altro utente

Una volta che l'utente ha scelto di saccheggiare un villaggio, questi dovrà scegliere lo specifico villaggio da attaccare. Successivamente sceglierà le unità da mandare in

battaglia per poi confermare l'attacco. Se l'utente decide di non confermare l'attacco, l'operazione verrà annullata. In caso di conferma vengono calcolate e rimosse le unità perse dall'attaccante e quelle perse dal difensore. Viene poi visualizzato l'esito della battaglia. Se l'esito è negativo verrà ripresentato il villaggio dell'attaccante. In caso di esito positivo verrà presentato il villaggio del difensore. Successivamente l'utente seleziona l'edificio da saccheggiare, azzerando le risorse prodotte dallo stesso. Vengono quindi sommate alle risorse possedute dall'utente attaccante. Viene infine ripresentata la schermata del villaggio dell'attaccante.

6.8 Regalo di risorse ad un altro utente

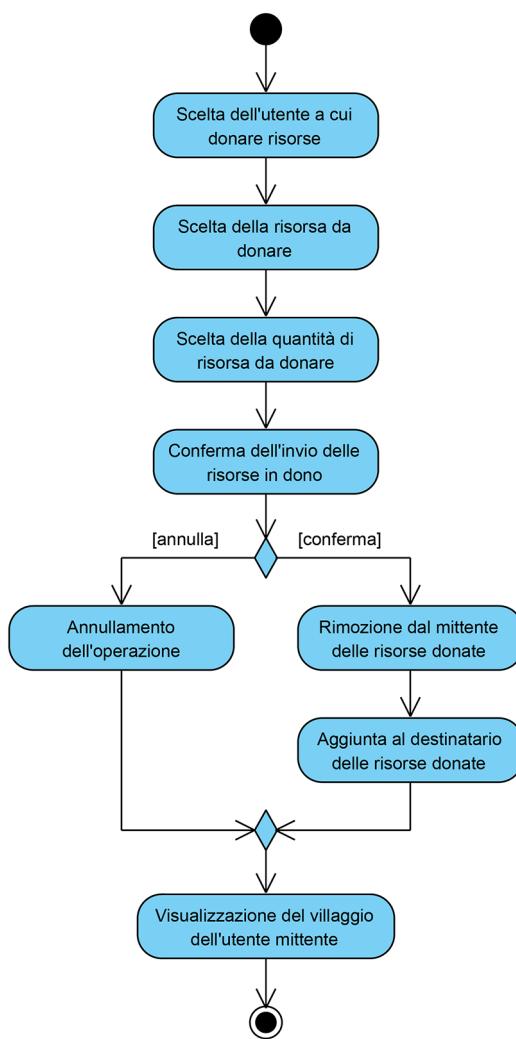


Figura 79: Diagramma per il regalo di risorse ad un altro utente

Per il regalo di risorse, l'utente dovrà selezionare l'utente destinatario. Seleziona quindi la risorsa da donare e la relativa quantità. Potrà procedere quindi con la conferma

6 Diagrammi delle attività

dell'operazione. Se l'utente non vuole confermare l'azione, questa verrà annullata. In caso di conferma, le risorse verranno sottratte al mittente e aggiunte al destinatario. Infine, viene ripresentato il villaggio del mittente.

6.9 Regalo di unità ad un altro utente

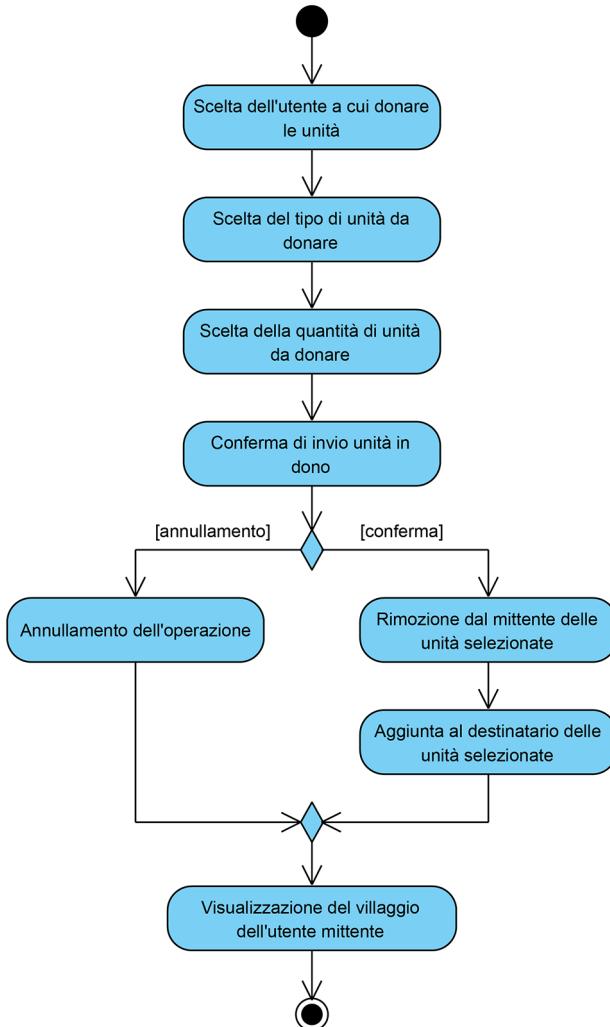


Figura 80: Diagramma per il regalo di unità ad un altro utente

Per il regalo di unità, l'utente dovrà selezionare il destinatario. Seleziona quindi l'unità da donare e la relativa quantità. Potrà procedere con la conferma dell'operazione. Se l'utente non vuole confermare l'azione, questa verrà annullata. In caso di conferma le unità verranno sottratte al mittente e aggiunte al destinatario. Infine, viene ripresentato il villaggio del mittente.

6.10 Gestione del villaggio

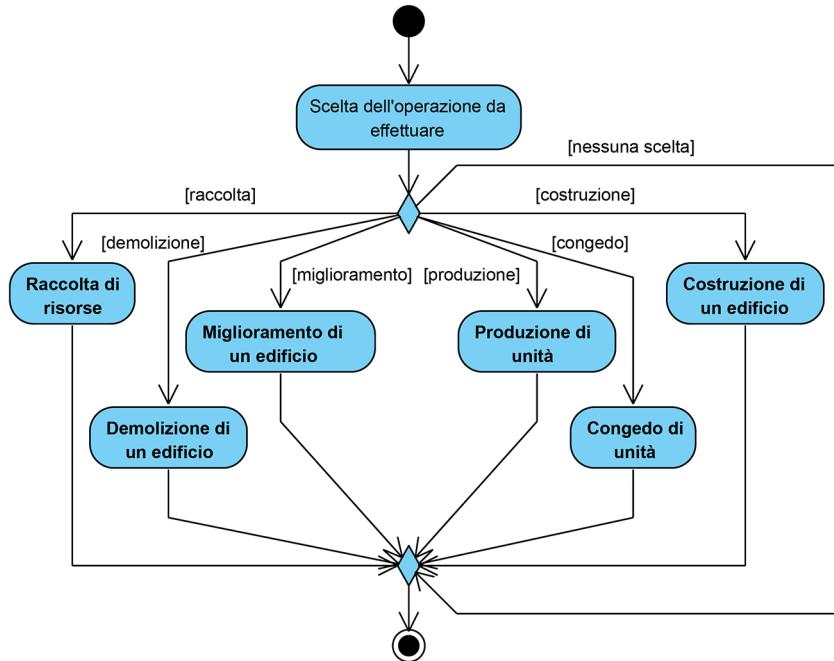


Figura 81: Diagramma per la gestione del villaggio

Per la gestione del villaggio l'utente può:

- raccogliere risorse;
- demolire un edificio;
- migliorare un edificio;
- produrre unità;
- congedare unità;
- costruire un edificio.

L'utente potrebbe decidere di non eseguire alcuna operazione.

6.11 Raccolta di risorse

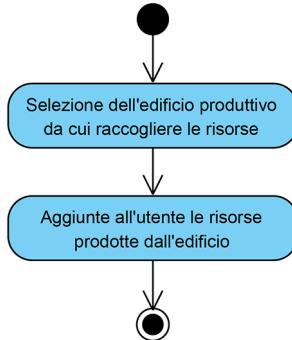


Figura 82: Diagramma per la raccolta di risorse

L'utente, per raccogliere le risorse prodotte, deve selezionare l'edificio produttivo. Vengono quindi rimosse le risorse prodotte dall'edificio e sommate a quelle totali dell'utente.

6.12 Demolizione di un edificio

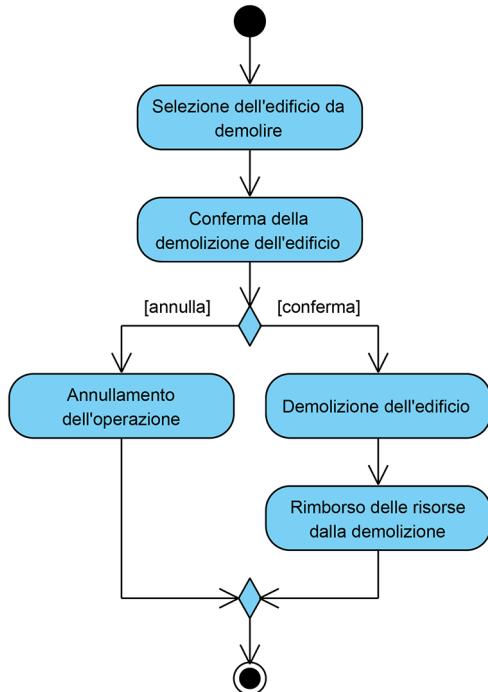


Figura 83: Diagramma per la demolizione di un edificio

L'utente, per la demolizione di un edificio, deve selezionare l'edificio da demolire. Deve quindi procedere con la conferma dell'operazione. Se decide di non confermare, verrà

annullata l'azione. In caso di conferma, viene demolito l'edificio e rimborsate le risorse ottenute dalla demolizione.

6.13 Miglioramento di un edificio

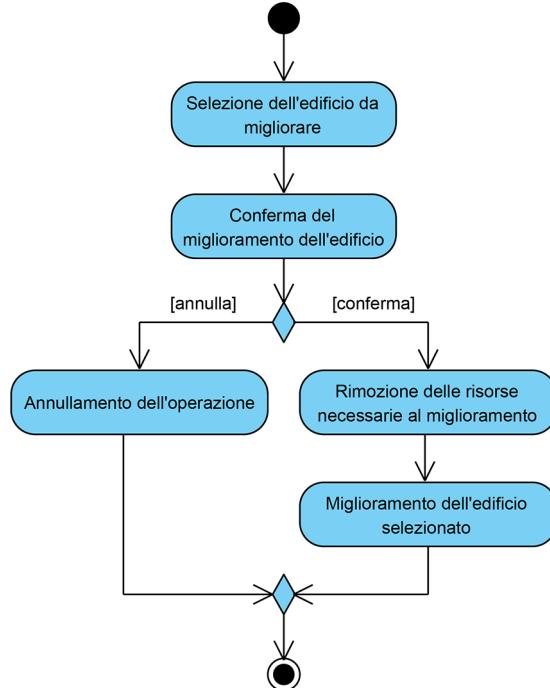


Figura 84: Diagramma per il miglioramento di un edificio

L'utente, per il miglioramento di un edificio, deve selezionare l'edificio da migliorare. Deve quindi procedere con la conferma dell'operazione. Se decide di non confermare, verrà annullata l'azione. In caso di conferma, vengono rimosse all'utente le risorse necessarie all'avanzamento. Viene quindi visualizzato l'edificio in miglioramento. Una volta scaduto il tempo richiesto per l'avanzamento al livello successivo, viene visualizzato l'edificio completato.

6.14 Produzione di unità

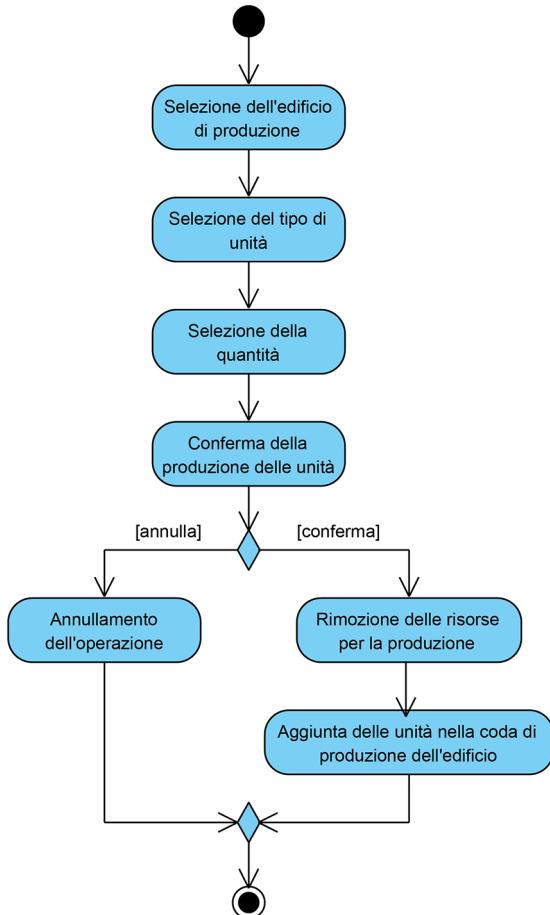


Figura 85: Diagramma per la produzione di unità

L'utente, per la produzione di un'unità, deve selezionare l'edificio produttivo. Deve quindi indicare il tipo di unità e la quantità da produrre. Procede poi con la conferma dell'operazione. Se decide di non confermare, verrà annullata l'azione. In caso di conferma, vengono rimosse all'utente le risorse necessarie alla produzione. Vengono quindi aggiunte le unità alla coda di produzione dell'edificio.

6.15 Congedo di unità

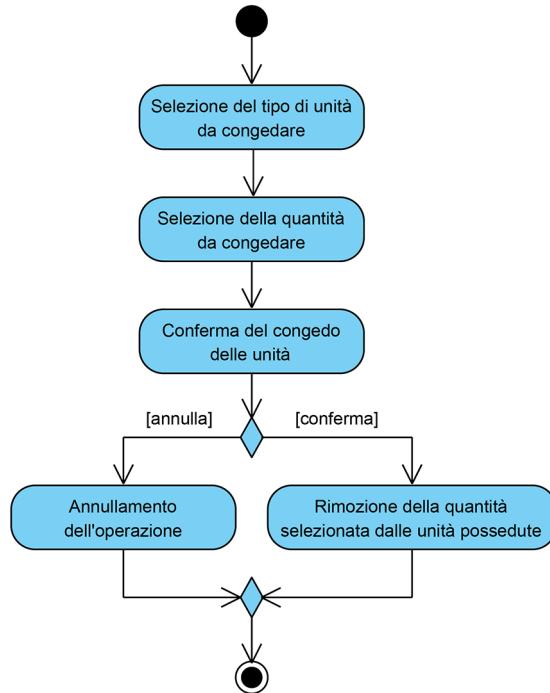


Figura 86: Diagramma per il congedo di unità

L'utente, per il congedo di unità, deve selezionare il tipo di unità e la quantità da congedare. Deve quindi procedere con la conferma dell'operazione. Se decide di non confermare, verrà annullata l'azione. In caso di conferma, viene rimossa la quantità di unità selezionate dalle unità possedute.

6.16 Costruzione di un edificio

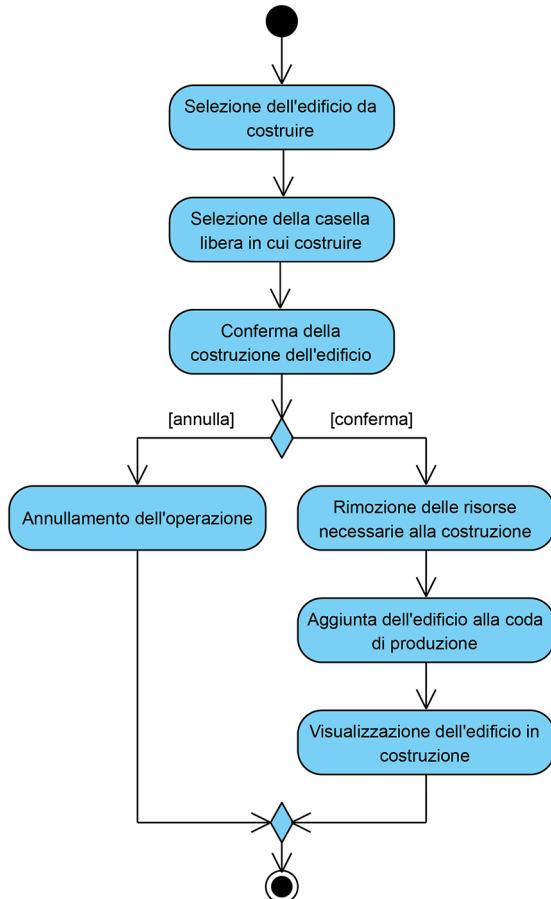


Figura 87: Diagramma per la costruzione di un edificio

L'utente, per la costruzione di un edificio, deve selezionare l'edificio da costruire e una casella libera dove collocarlo. Deve quindi procedere con la conferma dell'operazione. Se decide di non confermare, verrà annullata l'azione. In caso di conferma, vengono rimosse all'utente le risorse necessarie alla costruzione. Viene quindi aggiunto l'edificio alla coda di costruzione e visualizzato. Una volta scaduto il tempo richiesto per la costruzione, viene visualizzato l'edificio terminato.

6.17 Attività principali dell'amministratore

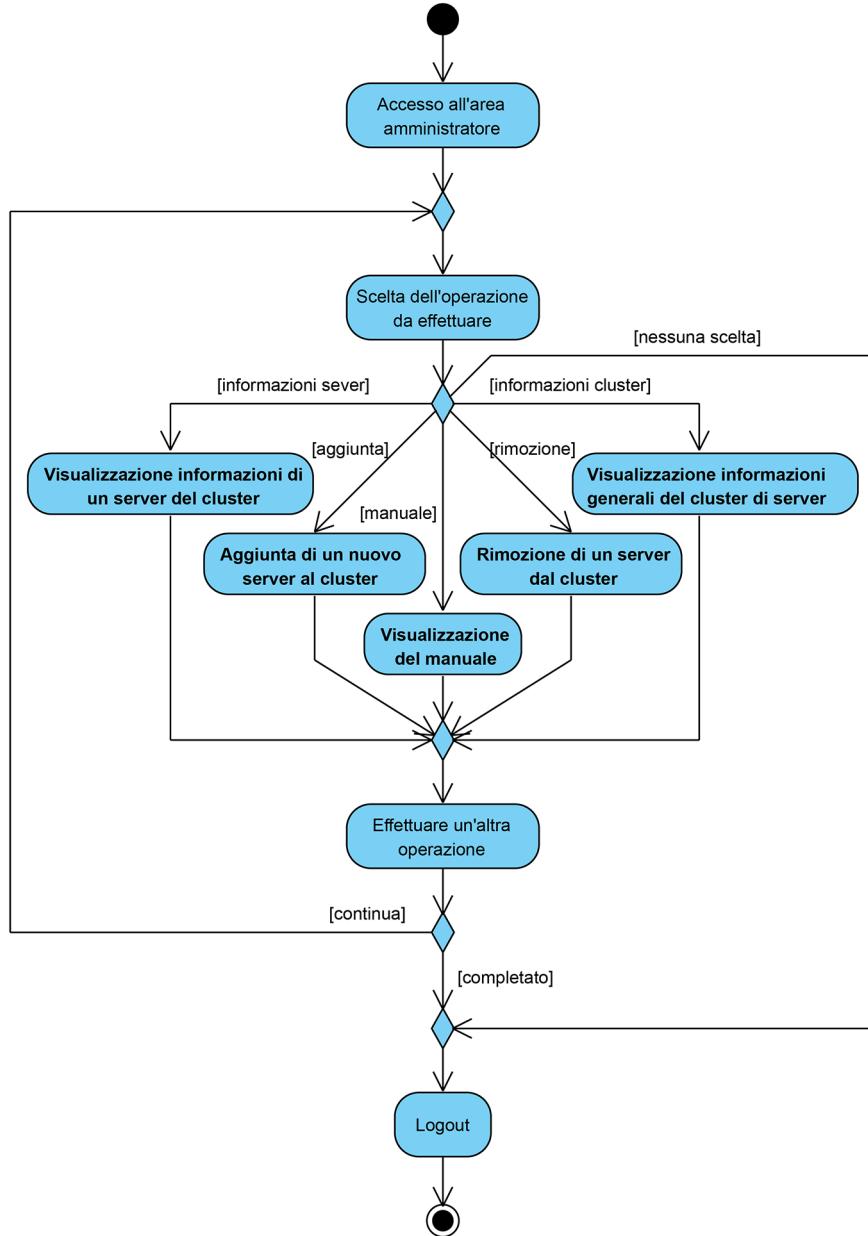


Figura 88: Diagramma delle attività principali dell'utente amministratore

Una volta che l'utente amministratore si è autenticato all'interno del sistema, queste potranno scegliere quale operazione effettuare. La scelta può ricadere tra:

- visualizzazione informazioni di un server del cluster;
- aggiunta di un nuovo server al cluster;

- visualizzazione del manuale;
- rimozione di un server dal cluster;
- visualizzazione informazioni generali del cluster di server.

Dopo aver individuato l'operazione e averla eseguita, può decidere se proseguire con un'altra azione oppure effettuare il logout dal sistema. L'utente potrebbe non eseguire alcuna operazione. Di conseguenza effettuerà solamente il logout dal sistema.

6.18 Visualizzazione delle informazioni di un server del cluster

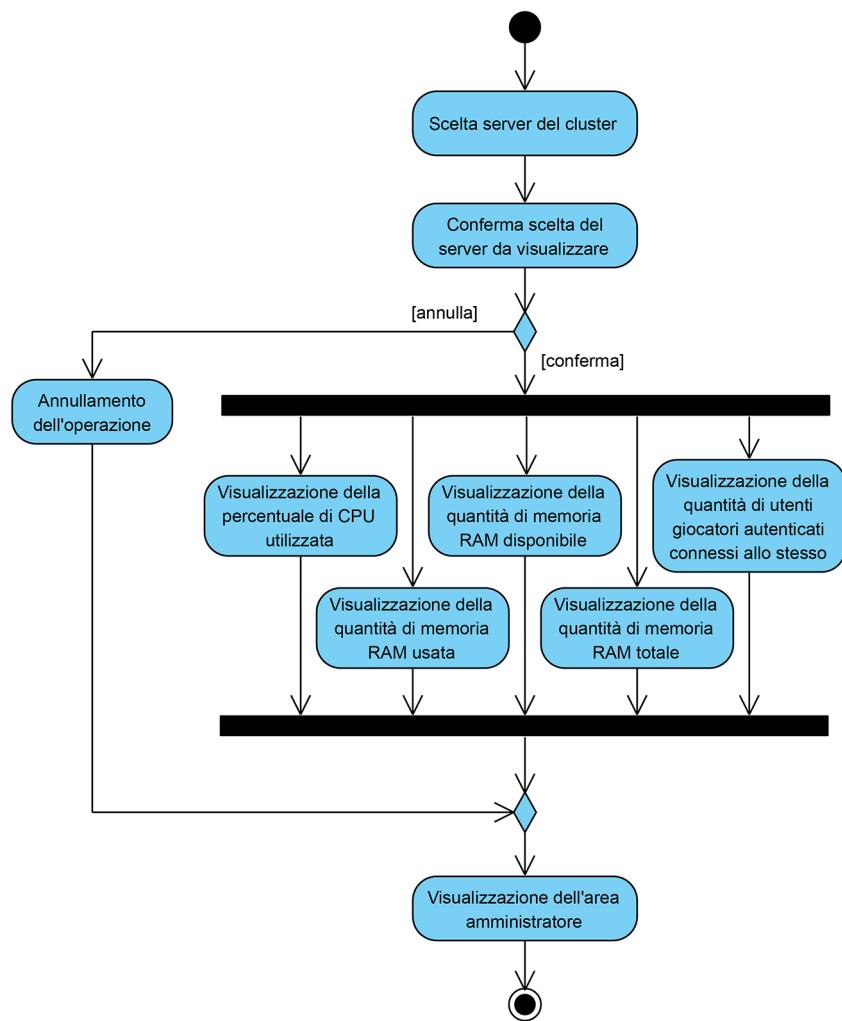


Figura 89: Diagramma per la visualizzazione delle informazioni di un server del cluster

L'amministratore, per la visualizzazione delle informazioni di un server del cluster, deve selezionare il server. Deve quindi procedere con la conferma dell'operazione. Se decide

di non confermare, verrà annullata l'azione. In caso di conferma, vengono mostrati in parallelo seguenti dati:

- percentuale di CPU utilizzata;
- quantità di memoria RAM usata;
- quantità di memoria RAM disponibile;
- quantità di memoria RAM totale;
- quantità di utenti giocatori autenticati connessi allo stesso.

Viene quindi ripresentata la schermata dell'area amministratore.

6.19 Aggiunta di un nuovo server al cluster

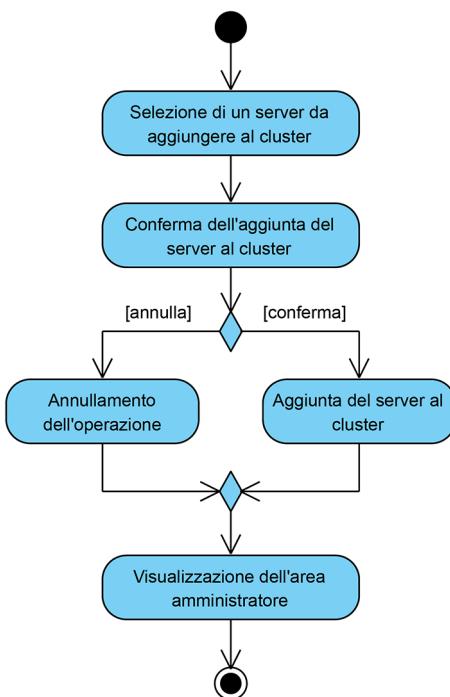


Figura 90: Diagramma per l'aggiunta di un nuovo server al cluster

L'amministratore, per l'aggiunta di un nuovo server al cluster, deve selezionare il server da aggiungere. Deve quindi procedere con la conferma dell'operazione. Se decide di non confermare, verrà annullata l'azione. In caso di conferma, viene aggiunto il server al cluster. Infine, viene ripresentata l'area amministratore.

6.20 Visualizzazione del manuale utente amministratore

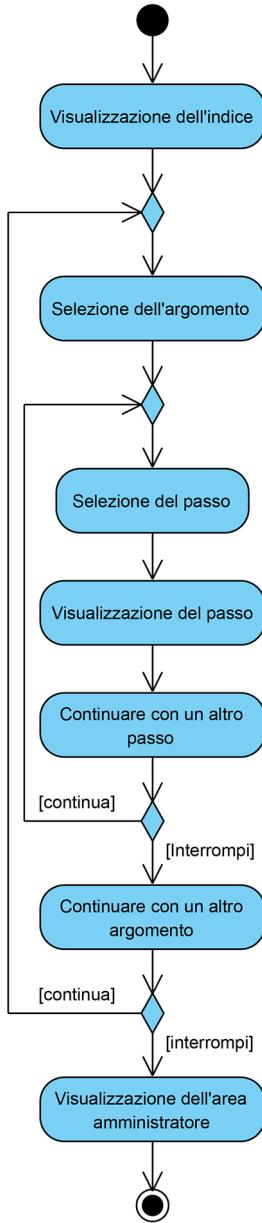


Figura 91: Diagramma per la visualizzazione del manuale amministratore

Alla visualizzazione del manuale utente amministratore, viene rappresentato l'indice degli argomenti in esso contenuti. L'utente procederà quindi con la scelta dell'argomento e poi del passo specifico. Dopo aver visualizzato il relativo passo può scegliere se visualizzare un altro passo o ritornare all'argomento. Se sceglie di proseguire con un altro passo o argomento ripeterà le operazioni precedenti. Altrimenti verrà ripresentata l'area amministratore.

6.21 Rimozione di un server dal cluster

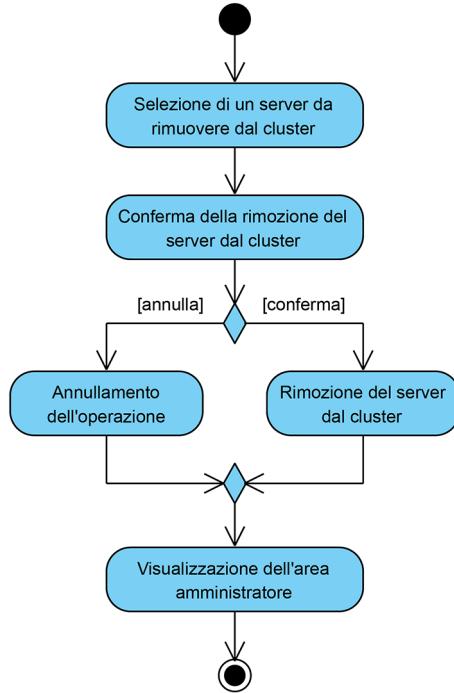


Figura 92: Diagramma per la rimozione di un server dal cluster

L'amministratore, per la rimozione di un server dal cluster, deve selezionare il server da rimuovere. Deve quindi procedere con la conferma dell'operazione. Se decide di non confermare, verrà annullata l'azione. In caso di conferma, viene rimosso il server dal cluster. Infine, viene ripresentata l'area amministratore.

6.22 Visualizzazione delle informazioni generali del cluster di server

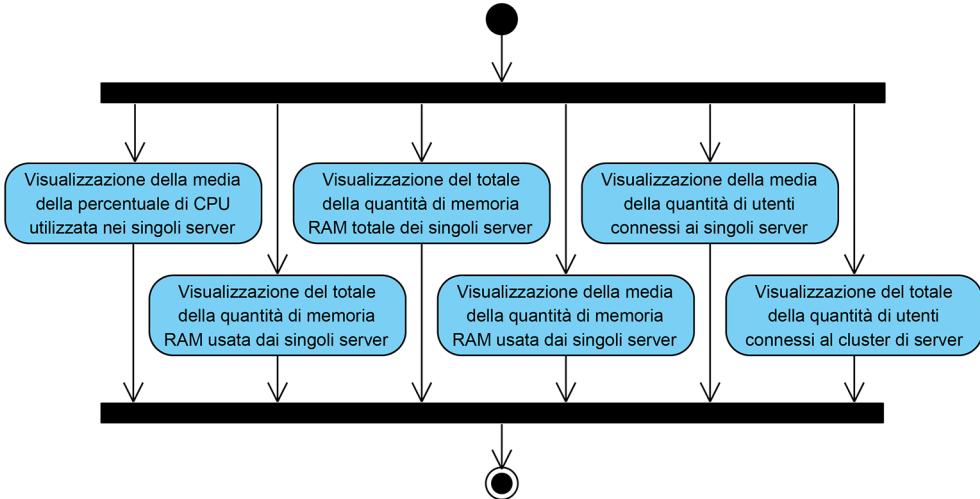


Figura 93: Diagramma per la visualizzazione delle informazioni generali del cluster di server

All'amministratore, se desidera visualizzare le informazioni del cluster, vengono mostrati in parallelo i seguenti dati:

- media della percentuale di CPU utilizzata nei singoli server;
- totale della quantità di memoria RAM usata dai singoli server;
- media della quantità di memoria RAM usata dai singoli server;
- totale della quantità di memoria RAM totale dei singoli server;
- media della quantità di utenti connessi ai singoli server;
- totale della quantità di utenti connessi al cluster di server.

Viene quindi ripresentata la schermata dell'area amministratore.

7 Stime di fattibilità e di bisogno di risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente a fornire una stima sulla fattibilità e di bisogno di risorse.

L'analisi dell'architettura ha messo in evidenza che le tecnologie scelte risultano sufficientemente adeguate per la realizzazione del prodotto. Riescono quindi a soddisfare le esigenze progettuali. Anche se JavaScript non supporta direttamente interfacce o classi astratte, verranno utilizzati dei metodi alternativi per ottenere uno stesso risultato.

La maggior parte delle tecnologie e degli strumenti sono nuove e poco conosciute per tutti i componenti del gruppo. Il $team_{|g|}$ si impegnerà quindi ad approfondire personalmente e tramite il materiale fornito dall'Amministratore le relative capacità di utilizzo.

In particolare il linguaggio Scala offre un costrutto per l'implementazione del pattern Singleton senza dover ripetere, per ogni classe, lo stesso codice.

8 Tracciamento

Questa sezione contiene le tabelle di tracciamento tra componenti e requisiti.

Per agevolare la lettura, i requisiti associati ai figli di una componente non sono riportati anche nel padre.

Le componenti riguardanti il Controller risultano senza requisiti tracciati per la natura di semplice gestore del flusso di informazioni tra View e Model. Per quanto riguarda le tabelle a cui fanno riferimento i requisiti, queste possono essere trovate nel documento “*Analisi dei Requisiti v6.00*” in Appendice.

8.1 Tracciamento Componenti-Requisiti

Componenti	Requisiti
akka::actors	R0F5 R0F5.1
akka::contrib::pattern	R0F5 R0F5.1
casbah	R0F5 R0F5.1
sgad::clienttier::controller::actions	R0F4 R0F4.1 R0F4.1.1 R0F4.1.2 R0F4.1.3 R0F4.2 R0F4.2.1 R0F4.2.2 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.2 R0F5.1.1.3 R0F5.1.1.6 R0F5.1.2 R0F5.1.2.1 R0F5.1.2.2 R0F5.1.2.3 R0F5.1.2.3.1 R0F5.1.2.4 R0F5.1.2.4.1 R0F5.1.2.5 R0F5.1.2.7.1 R0F5.1.3 R0F5.1.3.1

	R0F5.1.3.2 R0F5.1.3.2.1 R0F5.1.4 R0F5.1.4.1 R0F5.1.4.3 R0F5.1.4.3.2 R1F5.1.5 R1F5.1.5.1 R1F5.1.5.2 R1F5.1.5.2.1 R1F5.1.5.2.2 R1F5.1.5.2.3 R1F5.1.5.2.3.1 R1F5.1.6.1 R1F5.1.6.1.1 R1F5.1.6.2 R1F5.1.6.3 R1F5.1.6.4 R1F5.1.6.5 R1F5.2 R2F5.2.1 R2F5.2.1.1 R2F5.2.1.1.1.1 R2F5.2.1.2 R2F5.2.1.3
sgad::clienttier::controller::- backupmanager	R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::controller::- menufactory	R0F3 R0F3.1 R0F3.2 R0F3.3 R0F4 R0F4.1 R0F4.1.1 R0F4.1.2 R0F4.1.3

	R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.2.1 R0F5.1.2.7 R0F5.1.3 R0F5.1.3.1 R0F5.1.3.1.1 R0F5.1.3.1.3 R0F5.1.4 R0F5.1.4.1 R0F5.1.4.1.1 R1F5.1.5 R1F5.1.5.1 R1F5.1.5.1.1 R1F5.1.6.1 R1F5.1.6.1.1 R1F5.2 R2F5.2.1 R2F5.2.1.1 R2F5.2.1.3
sgad::clienttier::controller::: messageinterpreter	R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.2 R0F5.1.2 R0F5.1.2.4 R0F5.1.3 R0F5.1.3.2 R0F5.1.4 R0F5.1.4.3 R1F5.1.5 R1F5.1.5.2 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::controller::requester	R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.2 R0F5.1.2 R0F5.1.2.4 R0F5.1.3 R0F5.1.3.2

	R0F5.1.4 R0F5.1.4.3 R1F5.1.5 R1F5.1.5.2 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::model::generaldata	R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.2 R0F5.1.1.3 R0F5.1.1.4 R0F5.1.1.5 R0F5.1.2 R0F5.1.2.1.1 R0F5.1.2.2.1 R0F5.1.2.4.1 R0F5.1.2.4.3 R0F5.1.2.5 R0F5.1.2.6 R0F5.1.2.7 R0F5.1.2.7.2 R0F5.1.3 R0F5.1.3.1.1 R0F5.1.3.1.3 R0F5.1.3.2 R0F5.1.3.2.1 R0F5.1.3.2.2 R0F5.1.4 R0F5.1.4.1 R0F5.1.4.1.1 R0F5.1.4.1.2 R0F5.1.4.3.1 R0F5.1.4.3.2 R0F5.1.4.3.3 R1F5.1.5 R1F5.1.5.1.1 R1F5.1.5.2.3 R1F5.1.5.2.3.1 R1F5.1.6.1 R1F5.2 R2F5.2.1 R2F5.2.1.1.1 R2F5.2.1.1.2
sgad::clienttier::model::observer	R0F5

	R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::model::- personaldata	R0F3 R0F3.1 R0F3.2 R0F3.3 R0F4 R0F4.1 R0F4.1.1 R0F4.1.2 R0F4.1.3 R0F4.2 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.2 R0F5.1.1.3 R0F5.1.1.4 R0F5.1.2 R0F5.1.2.3.1 R0F5.1.2.4 R0F5.1.2.4.1 R0F5.1.2.4.2 R0F5.1.2.4.3 R0F5.1.2.5 R0F5.1.2.7 R0F5.1.2.7.1 R0F5.1.3 R0F5.1.3.1.1 R0F5.1.3.1.3 R0F5.1.3.2 R0F5.1.3.2.1 R0F5.1.3.2.2 R0F5.1.4 R0F5.1.4.1.1 R0F5.1.4.3 R0F5.1.4.3.1

	R0F5.1.4.3.2 R0F5.1.4.3.3 R1F5.1.5 R1F5.1.5.2 R1F5.1.5.2.3 R1F5.1.5.2.3.1 R1F5.1.6.1 R1F5.2 R2F5.2.1 R2F5.2.1.1.1.1 R2F5.2.1.4.1
sgad::clienttier::model::- userdatamanager	R0F3 R0F3.1 R0F3.2 R0F3.3 R0F4 R0F4.1 R0F4.1.1 R0F4.1.2 R0F4.1.3 R0F4.2 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.2 R0F5.1.2 R0F5.1.2.3.1 R0F5.1.2.4 R0F5.1.2.4.1 R0F5.1.2.5 R0F5.1.2.7 R0F5.1.3 R0F5.1.3.1.1 R0F5.1.3.1.3 R0F5.1.3.2.1 R0F5.1.4 R0F5.1.4.1.1 R0F5.1.4.3.2 R1F5.1.5 R1F5.1.5.2.3 R1F5.1.5.2.3.1 R1F5.1.6.1 R1F5.2 R2F5.2.1

	R2F5.2.1.1.1.1 R2F5.2.1.4.1
sgad::clienttier::view::commands	R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.1 R0F5.1.2 R0F5.1.2.1 R0F5.1.2.2 R0F5.1.3 R0F5.1.3.1 R0F5.1.4 R0F5.1.4.1 R0F5.1.4.2 R0F5.1.4.2.1 R1F5.1.5 R1F5.1.5.1 R1F5.1.6.1 R1F5.1.6.1.1 R1F5.2 R2F5.2.1 R2F5.2.1.1 R2F5.2.1.4 R2F5.2.1.4.2
sgad::clienttier::view::context	R0F3 R0F3.1 R0F3.2 R0F3.3 R0F4 R0F4.1 R0F4.1.1 R0F4.1.2 R0F4.1.3 R0F4.2 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.1 R0F5.1.2 R0F5.1.2.1 R0F5.1.2.2 R0F5.1.3 R0F5.1.3.1 R0F5.1.4 R0F5.1.4.1

	R0F5.1.4.2 R0F5.1.4.2.1 R1F5.1.5 R1F5.1.5.1 R1F5.1.5.2.1 R1F5.1.6.1 R1F5.1.6.1.1 R1F5.2 R2F5.2.1 R2F5.2.1.1 R2F5.2.1.4 R2F5.2.1.4.2
sgad::clienttier::view::- graphicobjects::bound	R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::view::- graphicobjects::collection	R0F3 R0F3.1 R0F3.2 R0F3.3 R0F4 R0F4.1 R0F4.1.1 R0F4.1.2 R0F4.1.3 R0F4.2 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.1 R0F5.1.2 R0F5.1.2.1 R0F5.1.2.2 R0F5.1.3 R0F5.1.3.1 R0F5.1.4 R0F5.1.4.1

	R1F5.1.5 R1F5.1.5.1 R1F5.1.6.1 R1F5.1.6.1.1 R1F5.2 R2F5.2.1 R2F5.2.1.1 R2F5.2.1.4 R2F5.2.1.4.2
sgad::clienttier::view::- graphicobjects::components::filter	R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::view::- graphicobjects::components::- worldcomponent	R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.1 R0F5.1.2 R0F5.1.2.1 R0F5.1.3 R0F5.1.3.1 R0F5.1.4 R0F5.1.4.1 R0F5.1.4.2 R0F5.1.4.2.1 R1F5.1.5 R1F5.1.5.1 R1F5.1.5.2 R1F5.1.5.2.1 R1F5.1.6.1 R1F5.1.6.1.1 R1F5.2 R2F5.2.1 R2F5.2.1.4.2
sgad::clienttier::view::- graphicobjects::components::- worldcomponentshape	R0F5

	R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::view::- graphicobjects::graphicobject	R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::view::- graphicobjects::point	R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::view::- graphicobjects::shape	R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R1F5.1.6.1 R1F5.2 R2F5.2.1
sgad::clienttier::view::- graphicobjects::widget	R0F3 R0F3.1 R0F3.2

	R0F3.3 R0F4 R0F4.1 R0F4.1.1 R0F4.1.2 R0F4.1.3 R0F4.2 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.2.1 R0F5.1.2.2 R0F5.1.3 R0F5.1.3.1 R0F5.1.3.1.1 R0F5.1.4 R0F5.1.4.1 R0F5.1.4.1.1 R1F5.1.5 R1F5.1.5.1 R1F5.1.6.1 R1F5.1.6.1.1 R1F5.2 R2F5.2.1 R2F5.2.1.3
sgad::servtier::application	R0F5 R0F5.1 R1F5.2
sgad::servtier::businesslogic::logic	R0F2 R0F4.1.1 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.3 R0F5.1.4 R1F5.1.5 R2F5.2.1
sgad::servtier::businesslogic::operations	R0F1 R0F1.1.1 R0F1.2.1 R0F1.2.2 R0F1.3.1

R0F1.4
R0F1.4.1
R0F1.4.2
R0F1.4.3
R0F1.4.4
R0F1.4.5
R0F1.5
R0F1.5.1
R0F1.5.2
R0F1.5.3
R0F2
R0F2.3.1
R0F2.3.2
R0F4
R0F4.1
R0F4.1.1
R0F4.1.1.1
R0F4.1.2
R0F4.2
R0F4.2.1
R0F4.2.2
R0F4.3
R0F5
R0F5.1
R0F5.1.1
R0F5.1.1.1
R0F5.1.1.3
R0F5.1.1.6
R0F5.1.2
R0F5.1.2.1
R0F5.1.2.2
R0F5.1.2.3
R0F5.1.2.4
R0F5.1.2.4.1
R0F5.1.2.5
R0F5.1.2.7
R0F5.1.2.7.1
R0F5.1.3
R0F5.1.3.1
R0F5.1.3.1.2
R0F5.1.4
R0F5.1.4.1
R0F5.1.4.1.1
R0F5.1.4.2
R0F5.1.4.2.1
R0F5.1.4.3

	R0F5.1.4.3.1 R0F5.1.4.3.2 R0F5.1.4.3.4 R1F5.1.5 R1F5.1.5.1 R1F5.1.5.2.2 R1F5.1.5.2.3 R1F5.1.6.1 R1F5.1.6.2 R1F5.1.6.2.1 R1F5.1.6.3 R1F5.1.6.4 R1F5.1.6.5 R2F5.2.1 R2F5.2.1.1 R2F5.2.1.1.1 R2F5.2.1.2 R2F5.2.1.3 R2F5.2.1.4
sgad::servtier::dataaccess::data::-shareddata	R0F1 R0F1.4 R0F1.4.1 R0F1.4.2 R0F1.4.3 R0F1.4.4 R0F1.4.5 R0F2 R0F5 R0F5.1 R0F5.1.1.2 R0F5.1.1.3 R0F5.1.1.4 R0F5.1.1.5 R0F5.1.2.1 R0F5.1.2.1.1 R0F5.1.2.2.1 R0F5.1.2.4 R0F5.1.2.4.1 R0F5.1.2.5 R0F5.1.2.6 R0F5.1.2.7.2 R0F5.1.3.1.1 R0F5.1.3.1.2 R0F5.1.3.1.3

	R0F5.1.3.2 R0F5.1.3.2.1 R0F5.1.3.2.2 R0F5.1.4.1.1 R0F5.1.4.1.2 R0F5.1.4.3.1 R0F5.1.4.3.2 R0F5.1.4.3.3 R0F5.1.4.3.4 R1F5.1.5.1.1 R1F5.1.5.2.3 R1F5.1.5.2.3.1 R1F5.1.6.3 R2F5.2.1 R2F5.2.1.1.1.1 R2F5.2.1.1.1.2 R2F5.2.1.2 R2F5.2.1.4.1
sgad::servtier::dataaccess::data::-userdata	R0F1 R0F1.1.1 R0F1.2.2 R0F1.4 R0F1.4.1 R0F1.4.2 R0F1.4.3 R0F1.4.4 R0F1.4.5 R0F2 R0F3.1 R0F4 R0F4.1 R0F4.1.1 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.1.2 R0F5.1.1.3 R0F5.1.1.4 R0F5.1.2 R0F5.1.2.1 R0F5.1.2.3.1 R0F5.1.2.4 R0F5.1.2.4.1 R0F5.1.2.4.2

	R0F5.1.2.5 R0F5.1.2.7 R0F5.1.2.7.1 R0F5.1.3 R0F5.1.3.1.1 R0F5.1.3.1.2 R0F5.1.3.1.3 R0F5.1.3.2 R0F5.1.3.2.1 R0F5.1.3.2.2 R0F5.1.4 R0F5.1.4.1.1 R0F5.1.4.2 R0F5.1.4.3 R0F5.1.4.3.1 R0F5.1.4.3.2 R0F5.1.4.3.3 R0F5.1.4.3.4 R1F5.1.5 R1F5.1.5.2 R1F5.1.5.2.1 R1F5.1.5.2.3 R1F5.1.5.2.3.1 R1F5.1.6.3 R2F5.2.1 R2F5.2.1.1.1.1 R2F5.2.1.2 R2F5.2.1.4.1
sgad::servtier::dataaccess::- databaseaccess:::databasemanager	R0F1 R0F1.1.1 R0F1.2.2 R0F1.4 R0F2 R0F5 R0F5.1
sgad::servtier::dataaccess::- databaseaccess:::shareddatadao	R0F5 R0F5.1
sgad::servtier::dataaccess::- databaseaccess:::userdatadao	R0F1 R0F2 R0F5 R0F5.1
sgad::servtier::presentation::- httpresponder	R0F1

	R0F1.1 R0F1.2 R0F1.3 R0F1.5 R0F2 R0F2.1 R0F2.2 R0F4 R0F4.1 R0F4.1.1 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R0P1 R1F5.1.5 R2F5.2.1
sgad::servtier::presentation::- messages	R0F1 R0F2 R0F4 R0F4.1 R0F4.1.1 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R2F5.2.1
sgad::servtier::presentation::- pagemanager	R0F1 R0F1.5 R0F2 R0F2.3 R0F5 R0F5.1 R0P1
sgad::servtier::presentation::- timeout	R0F1 R0F2

	R0F4 R0F4.1 R0F4.1.1 R0F4.3 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R2F5.2.1
sgad::servtier::presentation::- usermanager	R0F1 R0F2 R0F4 R0F4.1 R0F4.1.1 R0F4.3 R0F5 R0F5.1 R0F5.1.1 R0F5.1.2 R0F5.1.3 R0F5.1.4 R1F5.1.5 R2F5.2.1 R2F5.2.1.4

Tabella 2: Tracciamento Componenti-Requisiti

8.2 Tracciamento Requisiti-Componenti

Requisito	Descrizione	Componenti
F1	Un utente può creare un account per usufruire delle funzionalità offerte.	sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata sgad::servtier::- dataaccess::databaseaccess::- databasemanager sgad::servtier::dataaccess::- databaseaccess::userdatadao

		sgad::servtier::- presentation::httpresponder sgad::servtier::- presentation::messages sgad::servtier::- presentation::pagemanager sgad::servtier::- presentation::timeout sgad::servtier::- presentation::usermanager
F1.1	La registrazione richiede lo username dell'utente.	sgad::servtier::- presentation::httpresponder
F1.1.1	Lo username deve essere univoco in tutto il sistema. La lunghezza deve essere compresa tra i 4 e i 14 caratteri.	sgad::servtier::- businesslogic::operations sgad::servtier::: dataaccess::: data::userdata sgad::servtier::- dataaccess::: databaseaccess::: databasemanager
F1.2	La registrazione richiede l'email dell'utente.	sgad::servtier::- presentation::httpresponder
F1.2.1	La email deve avere un formato valido. Deve contenere il carattere '@' preceduto da altri caratteri e seguito da un dominio. Non deve superare i 100 caratteri.	sgad::servtier::- businesslogic::operations
F1.2.2	La email non deve già essere in uso da un altro utente.	sgad::servtier::- businesslogic::operations sgad::servtier::: dataaccess::: data::userdata sgad::servtier::- dataaccess::: databaseaccess::: databasemanager
F1.3	La registrazione richiede una password.	sgad::servtier::- presentation::httpresponder
F1.3.1	La password inserita deve avere una lunghezza di almeno 8 e al massimo di 16 caratteri. La password deve contenere almeno un carattere e almeno un numero.	sgad::servtier::- businesslogic::operations

F1.4	Alla creazione dell'account viene creato anche il villaggio dell'utente.	sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata sgad::servtier::- dataaccess::databaseaccess::- databasemanager sgad::servtier::- businesslogic::operations
F1.4.1	Alla creazione del villaggio dell'utente è presente la torre dello stregone di livello 1.	 sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata
F1.4.2	Alla creazione del villaggio dell'utente è presente una miniera di livello 1.	 sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata
F1.4.3	Alla creazione del villaggio dell'utente è presente una scuola di magia di livello 1.	 sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata
F1.4.4	Alla creazione del villaggio dell'utente sono presenti due lavoratori e nessun'altra unità.	 sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata
F1.4.5	L'utente possiede tutte le risorse indicate nella tabella "Lista delle risorse" con una quantità pari a zero.	 sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata

F1.5	Il sistema comunica all'utente gli eventuali errori sui dati inseriti durante la procedura di registrazione.	sgad::servertier::-businesslogic::operations sgad::servertier::-presentation::httpresponder sgad::servertier::-presentation::pagemanager
F1.5.1	Il sistema interrompe la procedura di registrazione se lo username inserito è già in possesso di un altro utente.	sgad::servertier::-businesslogic::operations
F1.5.2	Il sistema interrompe la procedura di registrazione se l'email è già in uso da un altro utente oppure non è valida.	sgad::servertier::-businesslogic::operations
F1.5.3	Il sistema interrompe la procedura di registrazione se la password inserita non soddisfa i requisiti di qualità richiesti.	sgad::servertier::-businesslogic::operations
F2	Il sistema permette all'utente di autenticarsi.	sgad::servertier::-businesslogic::logic sgad::servertier::-businesslogic::operations sgad::servertier:::dataaccess::-data::shareddata sgad::servertier:::dataaccess::-data::userdata sgad::servertier::-dataaccess:::databaseaccess::-databasemanager sgad::servertier:::dataaccess:::databaseaccess:::userdatadao sgad::servertier::-presentation::httpresponder sgad::servertier::-presentation::messages sgad::servertier::-presentation::pagemanager sgad::servertier::-presentation::timeout sgad::servertier::-presentation::usermanager
F2.1	Il sistema richiede l'username dell'utente.	sgad::servertier::-presentation::httpresponder

F2.2	Il sistema richiede la password dell'utente.	sgad::servtier::-presentation::httpresponder
F2.3	Il sistema deve comunicare all'utente l'eventuale errore durante la procedura di autenticazione.	sgad::servtier::-presentation::pagemanager
F2.3.1	Il sistema interrompe la procedura di autenticazione se lo username inserito è inesistente.	sgad::servtier::-businesslogic::operations
F2.3.2	Il sistema interrompe la procedura di autenticazione se la password inserita non corrisponde a quella associata allo username.	sgad::servtier::-businesslogic::operations
F3	Il sistema permette di visualizzare i dati dell'utente giocatore autenticato.	sgad::clienttier::controller::-menufactory sgad::clienttier::model::-personaldatas sgad::clienttier::model::-userdatamanager sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::widget
F3.1	Il sistema permette di visualizzare lo username dell'utente giocatore autenticato.	sgad::clienttier::controller::-menufactory sgad::clienttier::model::-personaldatas sgad::clienttier::model::-userdatamanager sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::widget sgad::servtier::dataaccess::data::userdata
F3.2	Il sistema permette di visualizzare l'email associata all'account dell'utente giocatore autenticato.	sgad::clienttier::controller::-menufactory sgad::clienttier::model::-personaldatas

		<pre>sgad::clienttier::model::- userdatamanager sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::widget</pre>
F3.3	Il sistema non permette all'utente giocatore autenticato di visualizzare la propria password.	<pre>sgad::clienttier::controller::- menufactory</pre> <pre>sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::widget</pre>
F4	Il sistema permette all'utente giocatore autenticato di modificare il proprio account.	<pre>sgad::clienttier::controller::- actions</pre> <pre>sgad::clienttier::controller::- menufactory sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::userdata sgad::servertier::- presentation::httpresponder sgad::servertier::- presentation::messages sgad::servertier::- presentation::timeout sgad::servertier::- presentation::usermanager</pre>

F4.1	<p>Il sistema permette all'utente giocatore autenticato di modificare i dati del proprio account.</p>	<p>sgad::clienttier::controller::-actions sgad::clienttier::controller::-menufactory sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::widget sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::-data::userdata sgad::servertier::-presentation::httpresponder sgad::servertier::-presentation::messages sgad::servertier::-presentation::timeout sgad::servertier::-presentation::usermanager</p>
F4.1.1	<p>Il sistema permette all'utente giocatore autenticato di modificare la propria password.</p>	<p>sgad::clienttier::controller::-actions sgad::clienttier::controller::-menufactory sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::widget sgad::servertier::-businesslogic::logic sgad::servertier::-businesslogic::operations</p>

		sgad::servtier::dataaccess:: data::userdata sgad::servtier::- presentation::httpresponder sgad::servtier::- presentation::messages sgad::servtier::- presentation::timeout sgad::servtier::- presentation::usermanager
F4.1.1.1	La password deve soddisfare gli standard previsti durante la procedura di registrazione.	sgad::servtier::- businesslogic::operations
F4.1.2	Il sistema richiede la vecchia password dell'utente giocatore autenticato.	sgad::clienttier::controller::- actions sgad::clienttier::controller::- menufactory sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::widget sgad::servtier::- businesslogic::operations
F4.1.3	Il sistema permette all'utente giocatore autenticato di confermare le modifiche apportate ai suoi dati.	sgad::clienttier::controller::- actions sgad::clienttier::controller::- menufactory sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::widget

F4.2	Il sistema avvisa l'utente giocatore autenticato che, durante la procedura di modifica dei dati, è avvenuto un errore.	sgad::clienttier::controller::-actions sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::widget sgad::servertier::-businesslogic::operations
F4.2.1	Il sistema interrompe la procedura di modifica dei dati se la nuova password inserita non soddisfa i requisiti.	sgad::clienttier::controller::-actions sgad::servertier::-businesslogic::operations
F4.2.2	Il sistema interrompe la procedura di modifica dei dati se la vecchia password inserita non corrisponde a quella salvata.	sgad::clienttier::controller::-actions sgad::servertier::-businesslogic::operations
F4.3	Il sistema permette all'utente giocatore autenticato di rimuovere il proprio account dal sistema.	sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::widget sgad::servertier::-businesslogic::logic sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::data::userdata

		<pre> sgad::servtier::- presentation::httpresponder sgad::servtier::- presentation::messages sgad::servtier::- presentation::timeout sgad::servtier::- presentation::usermanager akka::actors </pre>
F5	Il sistema permette all'utente giocatore autenticato di interagire con il mondo di gioco.	<pre> akka::contrib::pattern casbah sgad::clienttier::controller::- actions sgad::clienttier::controller::- backupmanager sgad::clienttier::controller::- menufactory sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- generaldata sgad::clienttier::model::- observer sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::bound sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- filter sgad::clienttier::view::- graphicobjects::components::- worldcomponent </pre>

	<p>F5.1</p> <p>L'utente giocatore autenticato può gestire il proprio villaggio.</p>	<pre> sgad::clienttier::view::- graphicobjects::components::- worldcomponentshape sgad::clienttier::view::- graphicobjects::graphicobject sgad::clienttier::view::- graphicobjects::point sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::application sgad::servertier::- businesslogic::logic sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::shareddata sgad::servertier::dataaccess::- data::userdata sgad::servertier::- dataaccess::databaseaccess::- databasemanager sgad::servertier::- dataaccess::databaseaccess::- shareddatadao sgad::servertier::dataaccess::- databaseaccess::userdatadao sgad::servertier::- presentation::httpresponder sgad::servertier::- presentation::messages sgad::servertier::- presentation::pagemanager sgad::servertier::- presentation::usermanager akka::actors akka::contrib::pattern casbah sgad::clienttier::controller::- actions sgad::clienttier::controller::- backupmanager sgad::clienttier::controller::- menufactory </pre>
--	---	---

```
sgad::clienttier::controller::-
messageinterpreter
sgad::clienttier::controller::-
requester
sgad::clienttier::model::-
generaldata
sgad::clienttier::model::-
observer
sgad::clienttier::model::-
personaldatal
sgad::clienttier::model::-
userdatamanager
sgad::clienttier::view::-
commands
sgad::clienttier::view::context
sgad::clienttier::view::-
graphicobjects::bound
sgad::clienttier::view::-
graphicobjects::collection
sgad::clienttier::view::-
graphicobjects::components::-
filter
sgad::clienttier::view::-
graphicobjects::components::-
worldcomponent
sgad::clienttier::view::-
graphicobjects::components::-
worldcomponentshape
sgad::clienttier::view::-
graphicobjects::graphicobject
sgad::clienttier::view::-
graphicobjects::point
sgad::clienttier::view::-
graphicobjects::shape
sgad::clienttier::view::-
graphicobjects::widget
sgad::servertier::application
sgad::servertier::-
businesslogic::logic
sgad::servertier::-
businesslogic::operations
sgad::servertier::dataaccess::-
data::shareddata
sgad::servertier::dataaccess::-
data::userdata
```

		sgad::servtier::- dataaccess::databaseaccess::- databasemanager sgad::servtier::- dataaccess::databaseaccess::- shareddatadao sgad::servtier::dataaccess::- databaseaccess::userdatadao sgad::servtier::- presentation::httpresponder sgad::servtier::- presentation::messages sgad::servtier::- presentation::pagemanager sgad::servtier::- presentation::usermanager
F5.1.1	L'utente giocatore autenticato può raccogliere le risorse prodotte dagli edifici produttivi.	sgad::clienttier::controller::- actions sgad::clienttier::controller::- backupmanager sgad::clienttier::controller::- menufactory sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- generaldata sgad::clienttier::model::- observer sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::bound sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- filter

		<pre> sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::components::- worldcomponentshape sgad::clienttier::view::- graphicobjects::graphicobject sgad::clienttier::view::- graphicobjects::point sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::logic sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::userdata sgad::servertier::- presentation::httpresponder sgad::servertier::- presentation::messages sgad::servertier::- presentation::timeout sgad::servertier::- presentation::usermanager </pre>
F5.1.1.1	La raccolta di risorse richiede la selezione di un edificio produttivo.	<pre> sgad::clienttier::view::- commands </pre> <pre> sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::servertier::- businesslogic::operations </pre>
F5.1.1.2	Le risorse raccolte dall'edificio selezionato vengono sommate alle risorse possedute.	<pre> sgad::clienttier::controller::- actions </pre> <pre> sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester </pre>

		sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldatal sgad::clienttier::model::-userdatamanager sgad::servtier::dataaccess::-data::shareddata sgad::servtier::dataaccess::-data::userdata sgad::clienttier::controller::-actions
F5.1.1.3	L'edificio comincia subito a produrre nuove risorse dopo la raccolta.	sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldatal sgad::servtier::businesslogic::operations sgad::servtier::dataaccess::-data::shareddata sgad::servtier::dataaccess::-data::userdata
F5.1.1.4	Ogni edificio ha un numero massimo di risorse accumulabili. Raggiunto il limite la produzione di tale edificio si ferma. I limiti sono indicati nella tabella “Lista degli edifici”.	sgad::clienttier::model::-generaldata
F5.1.1.5	Ogni tipo di edificio ha un particolare ritmo di produzione come indicato nella tabella “Lista degli edifici”.	sgad::clienttier::model::-personaldatal sgad::servtier::dataaccess::-data::shareddata sgad::servtier::dataaccess::-data::userdata
F5.1.1.6	La raccolta non richiede attesa di tempo.	sgad::clienttier::controller::-actions sgad::servtier::businesslogic::operations

F5.1.2	<p>L'utente giocatore autenticato può produrre unità.</p>	<pre> sgad::clienttier::controller::- actions sgad::clienttier::controller::- backupmanager sgad::clienttier::controller::- menufactory sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- generaldata sgad::clienttier::model::- observer sgad::clienttier::model::- personaldatal sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::bound sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- filter sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::components::- worldcomponentsshape sgad::clienttier::view::- graphicobjects::graphicobject sgad::clienttier::view::- graphicobjects::point sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::userdata </pre>
--------	---	---

		<pre> sgad::servtier::- presentation::httpresponder sgad::servtier::- presentation::messages sgad::servtier::- presentation::timeout sgad::servtier::- presentation::usermanager </pre>
F5.1.2.1	<p>La creazione di unità richiede la selezione di un edificio adatto allo scopo. Gli edifici adatti sono illustrati nella tabella “Lista degli edifici”.</p>	<pre> sgad::clienttier::controller::- actions </pre> <pre> sgad::clienttier::controller::- menufactory sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::widget sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata sgad::clienttier::model::- generaldata </pre> <pre> sgad::servtier::dataaccess::- data::shareddata </pre>
F5.1.2.1.1	<p>Ogni edificio può creare solo certi tipi di unità come indicato nella tabella “Lista degli edifici”.</p>	<pre> sgad::servtier::dataaccess::- data::shareddata </pre>
F5.1.2.2	<p>La creazione di unità richiede la selezione del tipo da creare.</p>	<pre> sgad::clienttier::controller::- actions sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::widget </pre>

		sgad::servtier::- businesslogic::operations sgad::clienttier::model::- generaldata
F5.1.2.2.1	I tipi di unità creabili sono indicati nella tabella “Lista delle unità”.	sgad::servtier::dataaccess::- data::shareddata
F5.1.2.3	La produzione di unità richiede la selezione della quantità da produrre.	sgad::clienttier::controller::- actions sgad::servtier::- businesslogic::operations
F5.1.2.3.1	La quantità selezionabile è vincolata al numero di risorse possedute.	sgad::clienttier::controller::- actions sgad::clienttier::model::- personaldta sgad::clienttier::model::- userdatamanager sgad::servtier::dataaccess::- data::userdata
F5.1.2.4	Le unità richieste sono aggiunte in coda di produzione dell'edificio selezionato.	sgad::clienttier::controller::- actions sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- personaldta sgad::clienttier::model::- userdatamanager sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata sgad::clienttier::controller::- actions
F5.1.2.4.1	Le risorse richieste per la produzione delle unità sono detratte dal totale di risorse dell'utente. Le risorse richieste sono specificate nella tabella “Lista degli edifici”.	sgad::clienttier::model::- generaldata

		sgad::clienttier::model::-personalldata sgad::clienttier::model::-userdatamanager sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::-data::shareddata sgad::servertier::dataaccess::-data::userdata
F5.1.2.4.2	Un edificio può produrre una sola unità alla volta.	sgad::clienttier::model::-personalldata sgad::servertier::dataaccess::-data::userdata
F5.1.2.4.3	La produzione di un'unità richiede l'attesa di tempo. La quantità varia per ogni tipo di unità. Le quantità sono specificate nella tabella “Lista delle unità”.	sgad::clienttier::model::-generaldata sgad::clienttier::model::-personalldata
F5.1.2.5	Le unità aggiunte in coda di produzione occupano già il posto.	sgad::clienttier::controller::-actions sgad::clienttier::model::-generaldata sgad::clienttier::model::-personalldata sgad::clienttier::model::-userdatamanager sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::-data::shareddata sgad::servertier::dataaccess::-data::userdata
F5.1.2.6	Ogni unità di un certo tipo richiede un certo numero di risorse come illustrato nella tabella “Lista delle unità”.	sgad::clienttier::model::-generaldata sgad::servertier::dataaccess::-data::shareddata

F5.1.2.7	La creazione di nuove unità è permessa se la quantità di posti disponibili per le unità è sufficiente.	sgad::clienttier::controller::-menufactory sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::-data::userdata
F5.1.2.7.1	Ogni unità occupa un posto.	sgad::clienttier::controller::-actions sgad::clienttier::model::-personaldata sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::-data::userdata
F5.1.2.7.2	Alcuni edifici offrono un certo numero di posti. Nella tabella “Lista degli edifici” è indicato il numero di posti offerti da ogni edificio.	sgad::clienttier::model::-generaldata sgad::servertier::dataaccess::-data::shareddata
F5.1.3	L’utente giocatore autenticato può migliorare un edificio già esistente.	sgad::clienttier::controller::-actions sgad::clienttier::controller::-backupmanager sgad::clienttier::controller::-menufactory sgad::clienttier::controller::-messageinterpreter sgad::clienttier::controller::-requester sgad::clienttier::model::-generaldata sgad::clienttier::model::-observer sgad::clienttier::model::-personaldata

		<pre> sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::bound sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- filter sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::components::- worldcomponentshape sgad::clienttier::view::- graphicobjects::graphicobject sgad::clienttier::view::- graphicobjects::point sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::logic sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::userdata sgad::servertier::- presentation::httpresponder sgad::servertier::- presentation::messages sgad::servertier::- presentation::timeout sgad::servertier::- presentation::usermanager sgad::clienttier::controller::- actions sgad::clienttier::controller::- menufactory </pre>
F5.1.3.1	Il miglioramento richiede la selezione dell'edificio da migliorare.	

		<pre> sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::operations sgad::clienttier::controller::- menufactory </pre>
F5.1.3.1.1	<p>Un edificio è migliorabile se non ha raggiunto il massimo livello. La tabella “Lista degli edifici” indica i livelli disponibili per ciascun edificio.</p>	<pre> sgad::clienttier::model::- generaldata sgad::clienttier::model::- personaldatal sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- graphicobjects::widget sgad::servertier::dataaccess::- data::shareddata sgad::servertier::dataaccess::- data::userdata </pre>
F5.1.3.1.2	<p>Un edificio è migliorabile se nel villaggio sono presenti gli edifici richiesti. Gli edifici richiesti per ogni livello sono indicati nella tabella “Lista degli edifici”.</p>	<pre> sgad::servertier::- businesslogic::operations </pre>
F5.1.3.1.3	<p>Un edificio è migliorabile se sono presenti le risorse richieste per il miglioramento. La tabella “Lista degli edifici” indica le risorse richieste per il miglioramento.</p>	<pre> sgad::servertier::dataaccess::- data::shareddata sgad::servertier::dataaccess::- data::userdata </pre>
		<pre> sgad::clienttier::controller::- menufactory </pre>

		<pre>sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::servertier::dataaccess::data::shareddata sgad::servertier::dataaccess::data::userdata sgad::clienttier::controller::actions</pre>
F5.1.3.2	Il miglioramento di un edificio ne migliora le caratteristiche come riportato nella tabella “Lista degli edifici”.	<pre>sgad::clienttier::controller::messageinterpreter sgad::clienttier::controller::requester sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldata sgad::servertier::dataaccess::data::shareddata sgad::servertier::dataaccess::data::userdata</pre>
F5.1.3.2.1	Le risorse richieste per il miglioramento di un edificio sono detratte dal totale di risorse dell’utente. Le risorse richieste sono specificate nella tabella “Lista degli edifici”.	<pre>sgad::clienttier::controller::actions</pre> <pre>sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::servertier::dataaccess::data::shareddata sgad::servertier::dataaccess::data::userdata</pre>

F5.1.3.2.2	Il miglioramento richiede l'attesa di tempo. La quantità varia per tipo di edificio e livello. Le quantità sono specificate nella tabella "Lista degli edifici".	sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldata sgad::servertier::dataaccess::-data::shareddata sgad::servertier::dataaccess::-data::userdata
F5.1.4	L'utente giocatore autenticato può costruire nuovi edifici.	sgad::clienttier::controller::-actions sgad::clienttier::controller::-backupmanager sgad::clienttier::controller::-menufactory sgad::clienttier::controller::-messageinterpreter sgad::clienttier::controller::-requester sgad::clienttier::model::-generaldata sgad::clienttier::model::-observer sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::clienttier::view::-commands sgad::clienttier::view::context sgad::clienttier::view::graphicobjects::bound sgad::clienttier::view::graphicobjects::collection sgad::clienttier::view::graphicobjects::components::filter sgad::clienttier::view::graphicobjects::components::worldcomponent sgad::clienttier::view::graphicobjects::components::worldcomponentshape

		<pre> sgad::clienttier::view::- graphicobjects::graphicobject sgad::clienttier::view::- graphicobjects::point sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::logic sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::userdata sgad::servertier::- presentation::httpresponder sgad::servertier::- presentation::messages sgad::servertier::- presentation::timeout sgad::servertier::- presentation::usermanager </pre>
F5.1.4.1	La costruzione richiede la selezione di un tipo di edificio.	<pre> sgad::clienttier::controller::- actions sgad::clienttier::controller::- menufactory sgad::clienttier::model::- generaldata sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::operations </pre>
F5.1.4.1.1	È selezionabile solo un edificio che richiede un quantitativo di risorse posseduto dall'utente giocatore autenticato.	<pre> sgad::clienttier::controller::- menufactory </pre> <pre> sgad::clienttier::model::- generaldata </pre>

		sgad::clienttier::model::- personalldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::shareddata sgad::servertier::dataaccess::- data::userdata
F5.1.4.1.2	I tipi di edifici costruibili sono indicati nella tabella “Lista degli edifici”.	sgad::clienttier::model::- generaldata sgad::servertier::dataaccess::- data::shareddata
F5.1.4.2	La costruzione richiede la selezione di una casella libera dove posizionare il nuovo edificio.	sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::userdata
F5.1.4.2.1	La costruzione è possibile solo se nel villaggio sono rimaste caselle libere.	sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::servertier::- businesslogic::operations
F5.1.4.3	La costruzione è aggiunta alla coda di costruzione del villaggio.	sgad::clienttier::controller::- actions sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- personalldata

		sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::userdata
F5.1.4.3.1	Appena l'edificio è stato costruito, se è di tipo produttivo comincia subito a produrre risorse.	sgad::clienttier::model::- generaldata sgad::clienttier::model::- personaldta sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata
F5.1.4.3.2	Le risorse richieste per la costruzione dell'edificio sono derivate dal totale dell'utente giocatore autenticato. Le risorse richieste sono specificate nella tabella "Lista degli edifici".	sgad::clienttier::controller::- actions sgad::clienttier::model::- generaldata sgad::clienttier::model::- personaldta sgad::clienttier::model::- userdatamanager sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata
F5.1.4.3.3	La costruzione richiede l'attesa di tempo. La quantità varia per tipo di edificio e livello. Le quantità sono specificate nella tabella "Lista degli edifici".	sgad::clienttier::model::- generaldata sgad::clienttier::model::- personaldta sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata

F5.1.4.3.4	Ogni lavoratore può costruire un edificio alla volta.	sgad::servtier::-businesslogic::operations sgad::servtier::dataaccess::-data::shareddata sgad::servtier::dataaccess::-data::userdata
F5.1.5	L'utente giocatore autentico può demolire un edificio già esistente.	sgad::clienttier::controller::-actions sgad::clienttier::controller::-backupmanager sgad::clienttier::controller::-menufactory sgad::clienttier::controller::-messageinterpreter sgad::clienttier::controller::-requester sgad::clienttier::model::-generaldata sgad::clienttier::model::-observer sgad::clienttier::model::-personaldatal sgad::clienttier::model::-userdatamanager sgad::clienttier::view::-commands sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::bound sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::components::-filter sgad::clienttier::view::-graphicobjects::components::-worldcomponent sgad::clienttier::view::-graphicobjects::components::-worldcomponentshape sgad::clienttier::view::-graphicobjects::graphicobject sgad::clienttier::view::-graphicobjects::point

		<pre> sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::logic sgad::servertier::- businesslogic::operations sgad::servertier::dataaccess::- data::userdata sgad::servertier::- presentation::httpresponder sgad::servertier::- presentation::messages sgad::servertier::- presentation::timeout sgad::servertier::- presentation::usermanager sgad::clienttier::controller::- actions </pre>
F5.1.5.1	La demolizione richiede la selezione di un edificio già costruito.	<pre> sgad::clienttier::controller::- menufactory sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::operations </pre>
F5.1.5.1.1	È selezionabile solo un edificio diverso dalla torre dello stregone.	<pre> sgad::clienttier::controller::- menufactory </pre>
F5.1.5.2	L'edificio è distrutto.	<pre> sgad::clienttier::model::- generaldata sgad::servertier::dataaccess::- data::shareddata </pre>
		<pre> sgad::clienttier::controller::- actions sgad::clienttier::controller::- messageinterpreter </pre>

		sgad::clienttier::controller::-requester sgad::clienttier::model::-personaldatal sgad::clienttier::view::-graphicobjects::components::-worldcomponent sgad::servertier::dataaccess::-data::userdata
F5.1.5.2.1	La casella dove risiedeva l'edificio diventa nuovamente libera.	sgad::clienttier::controller::-actions sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::components::-worldcomponent sgad::servertier::dataaccess::-data::userdata
F5.1.5.2.2	La demolizione non richiede attesa di tempo.	sgad::clienttier::controller::-actions sgad::servertier::-businesslogic::operations
F5.1.5.2.3	La metà delle risorse richieste per la costruzione del livello attuale dell'edificio demolito viene rimborsata.	sgad::clienttier::controller::-actions sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldatal sgad::clienttier::model::-userdatamanager sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::-data::shareddata sgad::servertier::dataaccess::-data::userdata
F5.1.5.2.3.1	Le risorse rimborsate vengono sommate alle risorse possedute.	sgad::clienttier::controller::-actions sgad::clienttier::model::-generaldata sgad::clienttier::model::-personaldatal

		sgad::clienttier::model::- userdatamanager sgad::servertier::dataaccess::- data::shareddata sgad::servertier::dataaccess::- data::userdata
F5.1.6	L'utente giocatore autenticato può congedare delle unità precedentemente arruolate.	
F5.1.6.1	Il congedo di unità richiede la selezione del tipo.	sgad::clienttier::controller::- actions sgad::clienttier::controller::- backupmanager sgad::clienttier::controller::- menufactory sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- generaldata sgad::clienttier::model::- observer sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::bound sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- filter sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::components::- worldcomponentshape sgad::clienttier::view::- graphicobjects::graphicobject

		sgad::clienttier::view::-graphicobjects::point sgad::clienttier::view::-graphicobjects::shape sgad::clienttier::view::-graphicobjects::widget sgad::servtier::-businesslogic::operations
F5.1.6.1.1	Sono selezionabili solo i tipi per i quali l'utente giocatore autenticato abbia almeno un'unità.	sgad::clienttier::controller::-actions sgad::clienttier::controller::-menufactory sgad::clienttier::view::-commands sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::clienttier::view::-graphicobjects::components::-worldcomponent sgad::clienttier::view::-graphicobjects::widget
F5.1.6.2	Il congedo di unità richiede la selezione della quantità.	sgad::clienttier::controller::-actions sgad::servtier::-businesslogic::operations
F5.1.6.2.1	La quantità selezionabile è minore o uguale alla quantità di unità possedute del tipo selezionato.	sgad::servtier::-businesslogic::operations
F5.1.6.3	Le unità richieste sono state congedate.	sgad::clienttier::controller::-actions sgad::servtier::-businesslogic::operations sgad::servtier::dataaccess::-data::shareddata sgad::servtier::dataaccess::-data::userdata
F5.1.6.4	Le unità congedate lasciano liberi i posti per nuove unità.	sgad::clienttier::controller::-actions sgad::servtier::-businesslogic::operations
F5.1.6.5	Il congedo di unità non richiede l'attesa di tempo.	sgad::clienttier::controller::-actions

		sgad::servertier::- businesslogic::operations sgad::clienttier::controller::- actions sgad::clienttier::controller::- backupmanager sgad::clienttier::controller::- menufactory sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- generaldata sgad::clienttier::model::- observer sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::bound sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- filter sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::components::- worldcomponentshape sgad::clienttier::view::- graphicobjects::graphicobject sgad::clienttier::view::- graphicobjects::point sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::application
F5.2	L'utente giocatore autenticato può interagire con un altro utente.	

F5.2.1	<p>L'utente giocatore autenticato può saccheggiare il villaggio di un altro utente.</p>	<pre> sgad::clienttier::controller::- actions sgad::clienttier::controller::- backupmanager sgad::clienttier::controller::- menufactory sgad::clienttier::controller::- messageinterpreter sgad::clienttier::controller::- requester sgad::clienttier::model::- generaldata sgad::clienttier::model::- observer sgad::clienttier::model::- personaldatal sgad::clienttier::model::- userdatamanager sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::bound sgad::clienttier::view::- graphicobjects::collection sgad::clienttier::view::- graphicobjects::components::- filter sgad::clienttier::view::- graphicobjects::components::- worldcomponent sgad::clienttier::view::- graphicobjects::components::- worldcomponentshape sgad::clienttier::view::- graphicobjects::graphicobject sgad::clienttier::view::- graphicobjects::point sgad::clienttier::view::- graphicobjects::shape sgad::clienttier::view::- graphicobjects::widget sgad::servertier::- businesslogic::logic </pre>
--------	---	--

		sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata sgad::servtier::- presentation::httpresponder sgad::servtier::- presentation::messages sgad::servtier::- presentation::timeout sgad::servtier::- presentation::usermanager
F5.2.1.1	Il saccheggio richiede la selezione di un utente destinatario.	sgad::clienttier::controller::- actions sgad::clienttier::controller::- menufactory sgad::clienttier::view::- commands sgad::clienttier::view::context sgad::clienttier::view::- graphicobjects::collection sgad::servtier::- businesslogic::operations
F5.2.1.1.1	Si considera vinto un attacco se le unità perse dallo schieramento difensivo sono maggiori di quelle dello schieramento attaccante.	sgad::servtier::- businesslogic::operations
F5.2.1.1.1.1	Le unità perse sono sottratte dal totale di unità dei due utenti giocatore coinvolti.	sgad::clienttier::controller::- actions sgad::clienttier::model::- generaldata sgad::clienttier::model::- personaldata sgad::clienttier::model::- userdatamanager sgad::servtier::- businesslogic::operations sgad::servtier::dataaccess::- data::shareddata sgad::servtier::dataaccess::- data::userdata

F5.2.1.1.2	Ogni unità ha un attributo di attacco e uno di difesa, come indicato in tabella “Lista delle unità”.	sgad::clienttier::model::-generaldata sgad::servertier::dataaccess::-data::shareddata
F5.2.1.2	Il saccheggio richiede la scelta delle unità da mandare all’attacco.	sgad::clienttier::controller::-actions sgad::servertier::-businesslogic::operations sgad::servertier::dataaccess::-data::shareddata sgad::servertier::dataaccess::-data::userdata
F5.2.1.3	Il sistema deve mostrare l’esito dello scontro.	sgad::clienttier::controller::-actions sgad::clienttier::controller::-menufactory sgad::clienttier::view::-graphicobjects::widget sgad::servertier::-businesslogic::operations
F5.2.1.4	Se l’attacco è vinto l’utente giocatore autenticato ha la possibilità di prelevare le risorse da un edificio avversario.	sgad::clienttier::view::-commands sgad::clienttier::view::context sgad::clienttier::view::-graphicobjects::collection sgad::servertier::-businesslogic::operations sgad::servertier::-presentation::usermanager
F5.2.1.4.1	Le risorse prelevate sono aggiunte al totale dell’utente attaccante.	sgad::clienttier::model::-personaldata sgad::clienttier::model::-userdatamanager sgad::servertier::dataaccess::-data::shareddata sgad::servertier::dataaccess::-data::userdata
F5.2.1.4.2	È richiesta la selezione dell’edificio da cui prelevare le risorse.	sgad::clienttier::view::-commands

		sgad::clienttier::view::context sgad::clienttier::view::graphicobjects::collection sgad::clienttier::view::graphicobjects::components::worldcomponent
F5.2.2	L'utente giocatore autenticato può regalare risorse a un altro utente.	
F5.2.2.1	Il regalo richiede la selezione di un utente destinatario.	
F5.2.2.2	Il dono richiede la scelta del tipo di risorsa da donare.	
F5.2.2.3	Il dono richiede la scelta della quantità di risorsa.	
F5.2.2.3.1	La quantità selezionabile è minore o uguale a quella posseduta.	
F5.2.2.4	Le risorse donate sono detratte dal totale dell'utente mittente.	
F5.2.2.5	Le risorse donate sono aggiunte al totale dell'utente destinatario.	
F5.2.3	L'utente autenticato può regalare unità ad un altro utente.	
F5.2.3.1	Il dono richiede la selezione di un utente destinatario.	
F5.2.3.2	Il dono richiede la scelta del tipo di unità da donare.	
F5.2.3.3	Il dono richiede la scelta della quantità di unità.	
F5.2.3.3.1	La quantità selezionabile è minore o uguale a quella posseduta.	
F5.2.3.4	Le unità donate sono detratte dal totale dell'utente mittente.	
F5.2.3.5	Le unità donate sono aggiunte al totale dell'utente destinatario.	
F6	L'utente giocatore autenticato deve potersi deautenticare dal sistema.	
F7	L'utente amministratore autenticato può gestire il cluster di server.	

F7.1	L'utente amministratore autenticato può visualizzare le informazioni di un server del cluster.	
F7.1.1	La visualizzazione richiede la selezione di un server appartenente al cluster.	
F7.1.2	Il sistema mostra come informazione del server la percentuale di CPU utilizzata.	
F7.1.3	Il sistema mostra come informazione del server la quantità di memoria RAM disponibile.	
F7.1.4	Il sistema mostra come informazione del server la quantità di memoria RAM utilizzata.	
F7.1.5	Il sistema mostra come informazione del server la quantità di memoria RAM totale.	
F7.1.6	Il sistema mostra come informazione del server il numero di utenti connessi allo stesso.	
F7.2	L'utente amministratore autenticato può aggiungere un nuovo server al cluster.	
F7.2.1	L'aggiunta richiede la selezione di un server non appartenente al cluster.	
F7.3	L'utente amministratore autenticato può rimuovere un server dal cluster.	
F7.3.1	La rimozione richiede la selezione di un server appartenente al cluster.	
F7.4	L'utente amministratore autenticato può visualizzare le informazioni generali del cluster di server.	
F7.4.1	Il sistema mostra come informazione del cluster la media della percentuale di CPU utilizzata nei singoli server.	
F7.4.2	Il sistema mostra come informazione del cluster il totale di memoria RAM utilizzata dai singoli server.	

F7.4.3	Il sistema mostra come informazione del cluster la media di memoria RAM utilizzata dai singoli server.	
F7.4.4	Il sistema mostra come informazione del cluster il totale di memoria RAM installata dai singoli server.	
F7.4.5	Il sistema mostra come informazione del cluster la media della quantità di utenti connessi ai singoli server.	
F7.4.6	Il sistema mostra come informazione del cluster il totale della quantità di utenti connessi al cluster di server.	
Q1	Viene fornito il manuale per l'utente giocatore tramite link all'interno del sito.	
Q1.1	Viene fornito il manuale per l'utente giocatore in lingua inglese.	
Q2	Tutto il codice rispetta le norme e le metriche sulla stesura di codice riportate nei documenti “Norme di Progetto v3.00” e “Piano di Qualifica v3.00”.	
Q4	Viene fornito il manuale per l'utente amministratore.	
Q4.1	Viene fornito il manuale per l'utente amministratore in lingua inglese.	

Tabella 3: Tracciamento Requisiti-Componenti

A Descrizione generale design pattern

A.1 Design pattern architetturali

A.1.1 MVC, Model-View-Controller

L'architettura utilizzata per la realizzazione dell'interfaccia utente nel progetto SGAD, riprende il design pattern MVC:

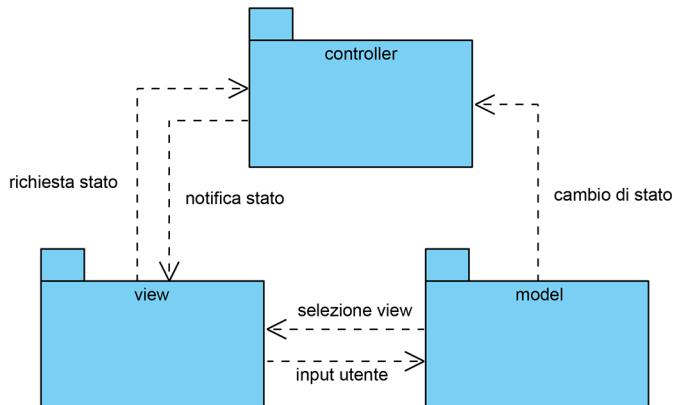


Figura 94: Diagramma del design pattern MVC

- **Descrizione:** il design pattern MVC permette una completa disgiunzione tra le funzionalità di vista e di modello dei dati. Si individuano tre componenti:
 1. Model rappresenta la logica di memorizzazione e recupero di dati utilizzati nel sistema;
 2. View visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti esterni;
 3. Controller riceve i comandi dell'utente (in genere attraverso la View) e li attua modificando lo stato degli altri due componenti.
- **Motivazione:** tale design pattern permette la separazione tra la modellazione del dominio, la presentazione, e le azioni basate sugli input degli utenti, all'interno di tre pacchetti separate. Questo schema implica anche la tradizionale separazione fra la logica applicativa, a carico del Controller e del Model, e l'interfaccia utente, a carico della View. Per il progetto SGAD si è deciso di implementare la strategia “pull-model”. Questa scelta è dovuta alla necessità di ridisegnare l'area di gioco ripetutamente.
- **Contesto applicativo:** il design pattern MVC verrà implementato nello sviluppo del front end del sistema. È stato scelto di utilizzarlo con tecnologie quali JavaScript/jQuery ed HTML5. Si individuano le seguenti macro componenti:
 1. View permette all'utente di interagire con il sistema attraverso un'interfaccia grafica. Questa viene utilizzata per visualizzare, modificare, aggiungere o eliminare i dati richiesti. Il tutto avviene attraverso l'invio di eventi, che vengono

gestiti da handler definiti, i quali si occupano di utilizzare tali informazioni per interagire con il Controller. La View non ha necessità di test approfonditi in quanto la logica interna è pressoché assente, limitandosi essa alla gestione di semplici eventi e visualizzazione di interfacce.

2. Controller mette in comunicazione la parte di View e quella di Model. Il Controller riceve le informazioni necessarie per il recupero o la modifica dei dati inviatigli dalla View conseguentemente all'interazione con l'utente, e le traduce in chiamate a metodi del Model. Successivamente alla ricezione della risposta asincrona proveniente del Model, il Controller indica alla View cosa mostrare all'utente. Ciò viene effettuato mediante chiamate mirate a metodi di quest'ultima.
3. Model ha il compito di fornire al Controller un insieme di metodi per il recupero, la modifica, l'inserimento e la cancellazione dei dati necessari al funzionamento dell'applicazione; l'implementazione di tali operazioni sarà resa trasparente al Controller. Compito del modello, inoltre, è fornire i dati richiesti dalla componente di controllo in un formato comprensibile e di facile utilizzo.

A.1.2 Three-Tier

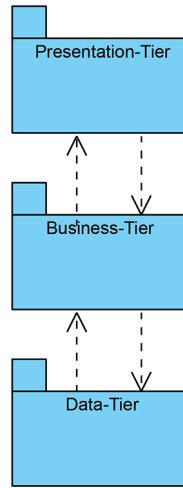


Figura 95: Diagramma del design pattern Three-Tier

- **Descrizione:** tale design pattern permette una disgiunzione tra i vari gruppi di entità che cooperano nell'erogazione del servizio. Esisterà un livello che si occuperà di interagire con il cliente offrendo l'interfaccia grafica, un altro livello che gestirà di eseguire la parte algoritmica dell'applicazione e un altro livello che si occuperà di persistere i dati e recuperarli. Ogni livello comunicherà solo con i livelli adiacenti.
- **Motivazione:** il beneficio principale apportato da tale paradigma risiede nel fatto che ogni livello può venir cambiato o aggiornato senza dover propagare le modifiche ai livelli adiacenti. Tale vantaggio deriva principalmente dal raggruppamento delle varie funzionalità in gruppi disgiunti ma in stretta collaborazione. Tuttavia, la struttura di tale design pattern risulta particolarmente calzante per dei servizi basati su architettura client-server, in quanto ogni livello non esiste semplicemente come raggruppamento logico a se stante, ma il suo ruolo viene adattato in relazione allo specifico ambiente di rete in cui viene eseguito: nel caso la morfologia della rete cambi, basterà aggiornare lo strato che lavorava in quel determinato ambiente.

Il fatto che ogni raggruppamento non risulti esclusivamente logico, ma anche un raggruppamento di funzionalità che eseguono su uno stesso ambiente fisico è un grande pregio per un applicativo web come SGAD.

- **Contesto applicativo:** tale design pattern verrà utilizzato come struttura portante nel back end del progetto, in cui:
 1. Presentation-Tier offrirà le varie interfacce grafiche agli utenti e si occuperà di reindirizzare le richieste dell'utente alle corrette operazioni;
 2. BusinessLogic-Tier si occuperà di eseguire i vari calcoli necessari per elaborare la risposta da fornire all'utente;
 3. DataAccess-Tier si occuperà di astrarre la base di dati e persistere o recuperare da essa i dati necessari per eseguire le computazioni.

A.1.3 Data Access Object

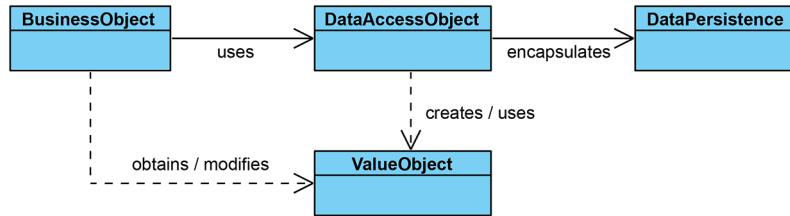


Figura 96: Diagramma del design pattern Data Access Object

- **Descrizione:** l'idea del pattern DAO si basa sulla possibilità di concentrare il codice per l'accesso al sistema di persistenza in una classe che si occupa di gestire la logica di accesso a questo.
- **Motivazione:** tale design pattern permette di disaccoppiare la logica di business dalla logica di accesso ai dati. Questo si ottiene spostando la logica di accesso ai dati dalle componenti di business ad una classe DAO, rendendo le componenti della prima indipendenti dalla natura del dispositivo di persistenza. Questo approccio garantisce che un eventuale cambiamento del dispositivo di persistenza non comporti modifiche sulle componenti di business.

Il fatto che ogni raggruppamento non risulti esclusivamente logico, ma anche un raggruppamento di funzionalità che eseguono su uno stesso ambiente fisico è un grande pregio per un applicativo web come SGAD.

- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. si desidera stratificare e isolare l'accesso ad una tabella dalla parte della logica di business;
 2. si desidera nascondere i dettagli implementativi della base di dati, permettendo di cambiare tipo di database senza dover modificare troppo codice.

A.2 Design pattern creazionali

A.2.1 Factory Method

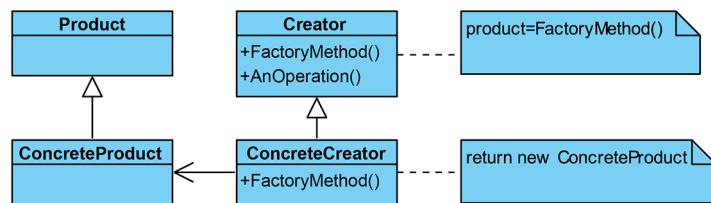


Figura 97: Diagramma del design pattern Factory Method

- **Descrizione:** il design pattern Factory Method indirizza il problema della creazione di oggetti senza specificarne l'esatta classe, fornendo un'interfaccia per creare un oggetto, ma lasciando che le sottoclassi decidano quale oggetto istanziare. Definisce un'interfaccia (Creator) per ottenere una nuova istanza di un oggetto (Product). Delega ad una classe derivata (ConcreteCreator) la scelta di quale classe istanziare (ConcreteProduct).
- **Motivazione:** la creazione di un oggetto può, spesso, richiedere processi complessi la cui collocazione all'interno della classe di composizione potrebbe non essere appropriata. Esso può, inoltre, comportare duplicazione di codice, richiedere informazioni non accessibili alla classe di composizione, o non fornire un sufficiente livello di astrazione. Il Factory Method indirizza questi problemi definendo un metodo separato per la creazione degli oggetti. Tale metodo può essere ridefinito dalle sottoclassi per definire il tipo derivato di prodotto che verrà effettivamente creato.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. si desidera che la creazione di un oggetto non precluda il suo riuso senza una significativa duplicazione di codice;
 2. si desidera che la creazione di un oggetto non richieda l'accesso ad informazioni o risorse che non dovrebbero essere contenute nella classe di composizione;
 3. si desidera che la gestione del ciclo di vita degli oggetti gestiti debba essere centralizzata in modo da assicurare un comportamento consistente all'interno dell'applicazione.

A.2.2 Singleton

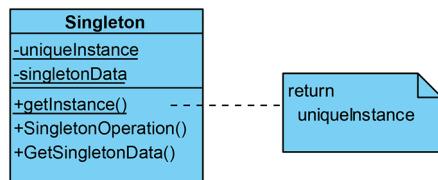


Figura 98: Diagramma del design pattern Singleton

- **Descrizione:** il design pattern Singleton assicura che una classe (Singleton) abbia una unica istanza e che questa sia globalmente accessibile in un punto ben noto.
- **Motivazione:** garantire che di una determinata classe venga creata una e una sola istanza, fornendo un punto di accesso globale a tale istanza. Si garantisce così un risparmio di risorse fisiche poiché la creazione di istanze dell'oggetto in questione è rimandata fino al momento dell'effettivo utilizzo dell'oggetto in questione.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:

1. si desidera la garanzia che nel sistema vi sia una sola istanza di oggetti di un determinato tipo, e che tali oggetti debbano essere accessibili da un punto di accesso ben preciso;
2. si desidera che non venga delegato ad altri il controllo di unicità di un oggetto;
3. si desidera che più oggetti condividano un unico pool di dati.

A.3 Design pattern strutturali

A.3.1 Composite

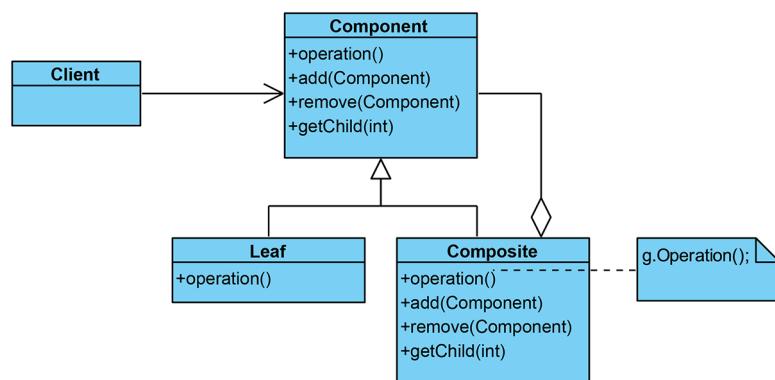


Figura 99: Diagramma del design pattern Composite

- **Descrizione:** il design pattern Composite permette di trattare un gruppo di oggetti come se fossero l’istanza di un oggetto singolo. Organizza gli oggetti in una struttura ad albero, nella quale i nodi sono delle Composite e le foglie sono oggetti semplici (Leaf).

In particolare, la sua struttura è:

1. Client manipola gli oggetti attraverso l’interfaccia Component.
 2. Component dichiara l’interfaccia per gli oggetti nella composizione, per l’accesso e la manipolazione di questi. Imposta un comportamento di default per l’interfaccia comune a tutte le classi, e può definire un’interfaccia per l’accesso al padre della componente e la implementa se è appropriata.
 3. Composite definisce il comportamento per le componenti aventi figli, salva i figli e implementa le operazioni ad essi connesse nell’interfaccia Component.
 4. Leaf definisce il comportamento delle foglie.
- **Motivazione:** dare la possibilità ai client di manipolare oggetti singoli e composizioni in modo uniforme, permettendo di costruire oggetti che possono essere a loro volta componenti di oggetti dello stesso tipo.
 - **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. si desidera rappresentare gerarchie tutto-parti;

2. si desidera rendere i client capaci di ignorare le differenze tra insiemi di oggetti ed oggetti individuali;
3. le funzionalità dell'oggetto composto sono le stesse delle sue componenti.

A.3.2 Flyweight

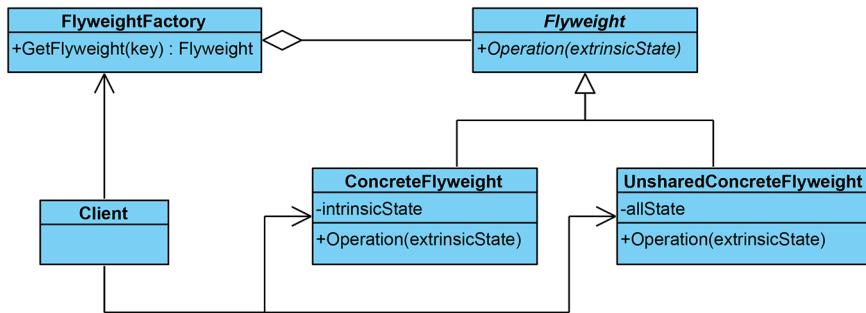


Figura 100: Diagramma del design pattern Flyweight

- **Descrizione:** il design pattern Flyweight permette di separare la parte variabile di una classe dalla parte che può essere riutilizzata, in modo tale da poter avere quest'ultima condivisa fra istanze differenti della parte variabile. L'oggetto flyweight deve essere un oggetto immutabile, per permettere la condivisione tra diversi client e $thread_{|g|}$.
- **Motivazione:** un flyweight è un oggetto condiviso che può essere usato in contesti multipli simultaneamente, ed è indistinguibile rispetto ad un'istanza non condivisa di un oggetto.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. l'applicazione utilizza un largo numero di oggetti;
 2. la maggior parte degli stati dell'oggetto possono essere resi estrinseci;
 3. si desidera rendere disponibili nuove operazioni che sono composizioni di altre operazioni;
 4. si desidera che l'applicazione non sia dipendente dall'identità dell'oggetto;
 5. si desidera delegare ad un oggetto l'invocazione di operazioni frequenti.

A.4 Design pattern comportamentali

A.4.1 Command

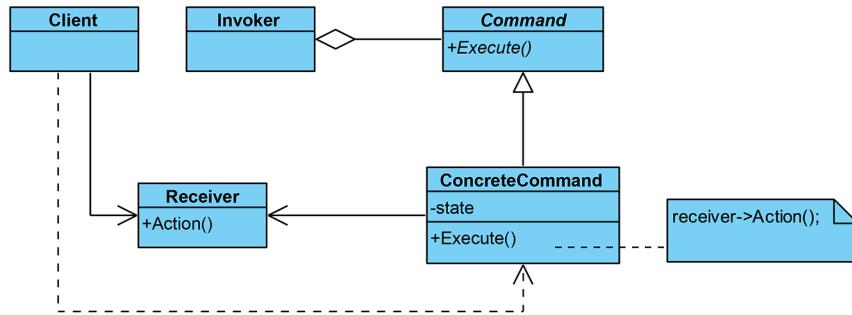


Figura 101: Diagramma del design pattern Command

- **Descrizione:** permette (lato Client) di encapsulare una richiesta, ossia un comando (Execute) da eseguire ed i suoi parametri (State), sotto forma di oggetto (ConcreteCommand). Isola la porzione di codice che effettua un'azione (eventualmente molto complessa) dal codice che ne richiede l'esecuzione. L'azione è encapsulata nell'oggetto Command.
- **Motivazione:** rendere variabile l'azione del client senza però conoscere i dettagli dell'operazione stessa. Un aspetto importante è che il destinatario della richiesta può non essere deciso staticamente all'atto dell'istanziazione del Command ma ricavato a tempo di esecuzione.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. si desidera parametrizzare oggetti rispetto ai comandi da eseguire;
 2. si desidera specificare, accodare ed eseguire richieste in tempi diversi;
 3. si desidera supportare l'*undo*_[g]: l'operazione di Execute può mantenere lo stato per annullare il proprio effetto;
 4. si desidera supportare il logging dei comandi in modo da consentire il *redo*_[g] in caso di crash del sistema;
 5. si desidera eseguire transazioni atomiche.

A.4.2 Iterator

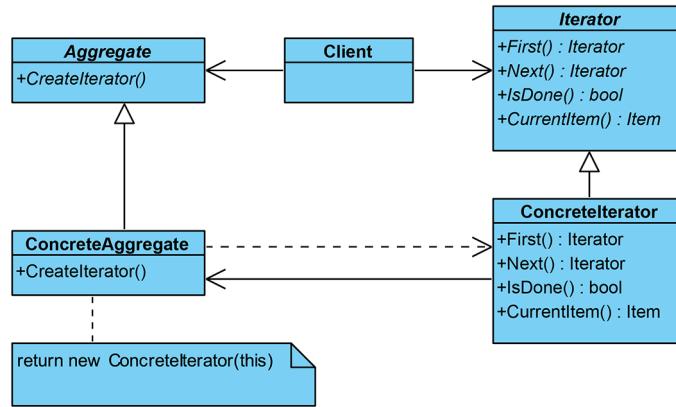


Figura 102: Diagramma del design pattern Iterator

- **Descrizione:** il design pattern Iterator viene utilizzato per fornire un meccanismo di accesso ad elementi di un oggetto in maniera sequenziale. Esso permette di scandire tutti gli elementi di una struttura senza conoscerne l'esatta implementazione.
- **Motivazione:** il pattern Iterator offre un'alternativa all'utilizzo di indici per l'accesso ad elementi di un contenitore. Tale alternativa è possibile mediante l'aggiunta di operazioni all'interfaccia di quest'ultimo. Questa soluzione può permettere di annullare la dipendenza verso i dettagli interni della classe contenitrice. Nondimeno, presenta alcuni inconvenienti, quali il sovraccarico di tale interfaccia e la mancanza di punti di accesso multipli.

In particolare, il design pattern Iterator definisce due gerarchie di classi: una per i contenitori e una per gli iteratori. Ne fanno parte:

- **Iterator:** definisce un'interfaccia per attraversare l'insieme degli elementi di un contenitore e accedere ai singoli elementi.
- **ConcreteIterator:** implementa l'interfaccia Iterator tenendo traccia della posizione corrente nel contenitore. Calcola inoltre qual è l'elemento successivo nella sequenza di attraversamento.
- **Aggregate:** definisce un'interfaccia per creare un oggetto Iterator.
- **ConcreteAggregate:** implementa l'interfaccia di creazione dell'Iterator. Ritorna un'istanza appropriata di ConcreteIterator.

- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:

1. si desidera semplificare l'interfaccia del contenitore, rimuovendo la parte che permette di navigare attraverso gli elementi contenuti;
2. si desidera isolare i dettagli di implementazione e la struttura del contenitore;
3. si desidera accedere agli elementi di un contenitore attraverso un'interfaccia uniforme e indipendente dal tipo di contenitore;
4. si desidera fornire un'interfaccia multipla per attraversare diverse strutture.

A.4.3 Observer

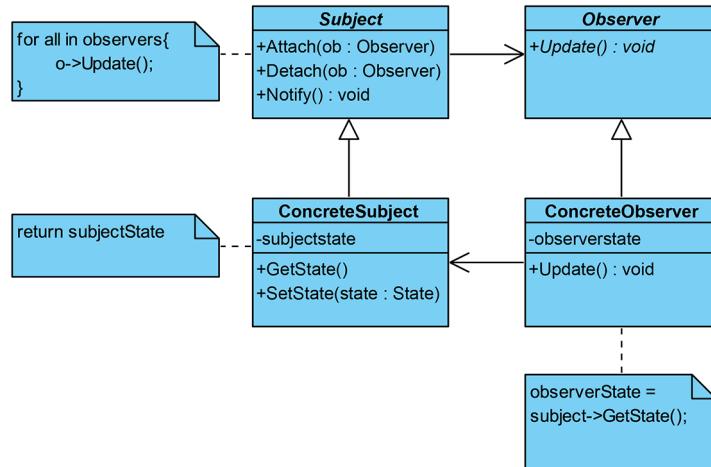


Figura 103: Diagramma del design pattern Observer

- **Descrizione:** il design pattern Observer viene utilizzato quando un oggetto (Subject) vuole notificare il suo cambio di stato a un gruppo di oggetti (ConcreteObserver). Si applica quando ci sono diversi oggetti interessati alle modifiche di un oggetto. Usare il pattern di osservazione permette di separare l'oggetto che è visualizzato dalla visualizzazione.
- **Motivazione:** definire una relazione uno-a-molti tra oggetti, in modo che quando un oggetto cambia stato tutti gli ascoltatori collegati sono notificati ed aggiornati.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. l'astrazione è composta da due aspetti, una dipendente dall'altra, e si desidera incapsulare le due astrazioni in oggetti separati;
 2. si desidera che il cambiamento di un oggetto richieda il cambiamento di altri, senza sapere quali;
 3. non si conosce a priori il numero degli oggetti dipendenti;
 4. si desidera che un oggetto notifichi ad altri un cambiamento senza conoscere la struttura degli oggetti dipendenti.

A.4.4 State

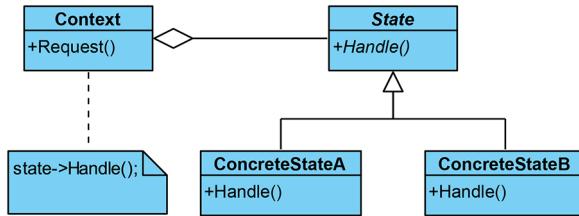


Figura 104: Diagramma del design pattern State

- **Descrizione:** il design pattern State consente ad un oggetto di cambiare il proprio comportamento (il risultato dell'operazione Request dell'oggetto Context) a *runtime_g* in funzione dello stato in cui si trova. Si applica quando c'è un oggetto che cambia stato. Invece di avere un metodo che effettua dei test per sapere quale codice usare, si usano gli oggetti.

La sua struttura è:

1. **Context:** definisce l'interfaccia del client e mantiene un riferimento ad un **ConcreteState**;
 2. **State:** definisce l'interfaccia, implementata dai **ConcreteState**, che incapsula la logica del comportamento associato ad un determinato stato;
 3. **ConcreteState:** implementa il comportamento associato ad un particolare stato.
- **Motivazione:** descrivere il comportamento di un sistema mediante la composizione di azioni intraprese in un numero finito di stati. In ciascuno stato vengono intraprese singole azioni. La somma delle azioni determina il comportamento complessivo del sistema. Consentire ad un oggetto di modificare il proprio comportamento quando il suo stato interno cambia.
 - **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. si desidera che l'astrazione descriva il sistema come un automa a stati finiti;
 2. si desidera che il cambiamento del comportamento del sistema in un determinato stato non comporti la modifica del comportamento di altri stati;
 3. non si conosce a priori il numero degli stati;
 4. si desidera che le azioni che vengono intraprese in uno stato non dipendano dalle azioni intraprese in altri stati;
 5. si desidera che il comportamento di un oggetto dipenda dal suo stato, e debba modificare il proprio comportamento a *runtime* in base alla variazione dello stato;
 6. le operazioni sono implementate mediante largo uso di *statement_g* condizionali dipendenti dai valori assunti dalle variabili di stato: lo State pone ciascun *branch_g* in una classe separata.

A.4.5 Strategy

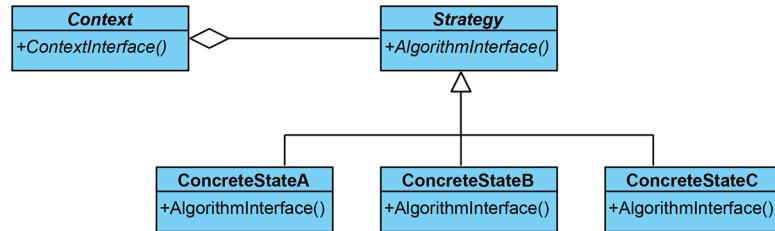


Figura 105: Diagramma del design pattern Strategy

- **Descrizione:** il design pattern Strategy incapsula una famiglia di algoritmi in una famiglia (Strategy) di classi (ConcreteStrategyA, ConcreteStrategyB, etc.) rendendoli intercambiabili in base alla situazione (Context). Prevede che gli algoritmi siano intercambiabili tra loro (in base ad una qualche condizione) in modo trasparente al client che ne fa uso. Si applica in situazioni dove esistono diverse strategie per fare una cosa ma l'utente (o l'oggetto) ne sceglie una solamente.
- **Motivazione:** definire una famiglia di algoritmi, incapsularli e renderli intercambiabili in maniera trasparente rispetto all'uso da parte del client.
- **Contesto applicativo:** tale design pattern verrà utilizzato nei seguenti casi:
 1. classi collegate differiscono solo per il loro comportamento;
 2. gli algoritmi utilizzano dati non a conoscenza del client;
 3. una classe definisce differenti comportamenti, espressi come statement condizionali multipli nelle operazioni (il problema si risolve in maniera simile al caso del pattern State).