



digitalsloths@gmail.com

Specifica Tecnica

Informazioni sul documento

Nome documento	Specifica Tecnica
Versione	v3.0.0
Data redazione	2016-03-08
Redattori	<ul style="list-style-type: none">• Ugo Padoan• Filinesi Skrypnyk Oleksandr• Andria Umberto
Verificatori	<ul style="list-style-type: none">• Parise Ivan• Saccon Daniele
Approvazione	<ul style="list-style-type: none">• Pinton Federico
Lista distribuzione	<ul style="list-style-type: none">• <i>Digital Sloths</i>• <i>Prof. Tullio Vardanega</i>• <i>Prof. Riccardo Cardin</i>• <i>Miriade</i>
Uso	<ul style="list-style-type: none">• Esterno

Sommario

Il documento contiene la specifica tecnica e architettura del prodotto CLIPS.

Diario delle Modifiche

Versione	Modifica	Autore & Ruolo	Data
v 3.0.0	<i>Approvazione documento</i>	Pinton Federico <i>Responsabile di Progetto</i>	2016-05-15
v 2.1.0	<i>Verifica documento</i>	Parise Ivan <i>Verificatore</i>	2016-05-15
v 2.0.2	<i>Aggiornamento Classi e Package con relative immagini dopo codifica</i>	Saccon Daniele <i>Progettista</i>	2016-06-14
v 2.0.1	<i>Aggiornamento figure capitolo Design Pattern</i>	Pinton Federico <i>Progettista</i>	2016-06-06
v 2.0.0	<i>Approvazione documento</i>	Pinton Federico <i>Responsabile di Progetto</i>	2016-05-13
v 1.1.0	<i>Verifica documento</i>	Saccon Daniele <i>Verificatore</i>	2016-05-12
v 1.0.6	<i>Aggiornamento figure</i>	Ugo Padoan <i>Progettista</i>	2016-05-11
v 1.0.5	<i>Aggiornamento e modifica sezione componenti e classi</i>	Padoan Ugo <i>Progettista</i>	2016-05-10
v 1.0.4	<i>Aggiornamento e modifica sezione architettura database</i>	Casotto Federico <i>Progettista</i>	2016-05-02
v 1.0.3	<i>Aggiornamento e modifica sezione design pattern</i>	Andria Umberto <i>Progettista</i>	2016-04-30
v 1.0.2	<i>Aggiornamento e modifica sezione tecnologie</i>	Parise Ivan <i>Progettista</i>	2016-04-28
v 1.0.1	<i>Aggiornamento e modifica sezione diagrammi attività</i>	Saccon Daniele <i>Progettista</i>	2016-04-27
v 1.0.0	<i>Approvazione documento</i>	Federico Pinton <i>Responsabile di Progetto</i>	2016-04-11
v 0.4.0	<i>Verifica documento</i>	Saccon Daniele <i>Verificatore</i>	2016-04-11
v 0.3.4	<i>Inserimento immagini sezione Componenti</i>	Filinesi Skrypnyk Oleksandr <i>Progettista</i>	2016-04-11
v 0.3.3	<i>Caricamento immagini sezione design pattern</i>	Pinton Federico <i>Progettista</i>	2016-04-10
v 0.3.2	<i>Aggiornamento sezione descrizione design pattern</i>	Pinton Federico <i>Progettista</i>	2016-04-09
v 0.3.1	<i>Aggiornamento sezione design pattern</i>	Andria Umberto <i>Progettista</i>	2016-04-08
v 0.3.0	<i>Verifica del documento</i>	Parise Ivan <i>Verificatore</i>	2016-04-06
v 0.2.4	<i>Inserimento Classi</i>	Ugo Padoan <i>Progettista</i>	2016-04-06
v 0.2.3	<i>Inserimento Package</i>	Filinesi Skrypnyk Oleksandr <i>Progettista</i>	2016-04-06
v 0.2.2	<i>Creazione capitolo Descrizione Design Pattern</i>	Andria Umberto <i>Progettista</i>	2016-04-06

v 0.2.1	<i>Aggiornamento sezione design pattern</i>	Andria Umberto <i>Progettista</i>	2016-04-06
v 0.2.0	<i>Verifica capitolo Architettura Database e Descrizione Architettura</i>	Parise Ivan <i>Verificatore</i>	2016-04-04
v 0.1.4	<i>Creazione capitolo Architettura Database</i>	Casotto Federico <i>Progettista</i>	2016-04-03
v 0.1.3	<i>Creazione capitolo Descrizione Architettura</i>	Casotto Federico <i>Progettista</i>	2016-04-02
v 0.1.2	<i>Creazione capitolo Descrizione Architettura</i>	Casotto Federico <i>Progettista</i>	2016-04-02
v 0.1.1	<i>Aggiornamento sezione design pattern</i>	Pinton Federico <i>Progettista</i>	2016-04-01
v 0.1.0	<i>Verifica sezione diagrammi di attività</i>	Padoan Ugo <i>Verificatore</i>	2016-04-01
v 0.0.9	<i>Aggiornamento sezione diagrammi delle attività</i>	Saccon Daniele <i>Progettista</i>	2016-03-31
v 0.0.8	<i>Aggiornamento sezione Tecnologie utilizzate</i>	Parise Ivan <i>Progettista</i>	2016-03-31
v 0.0.7	<i>Creazione delle sezioni giochi e Battleship nei diagrammi delle attività</i>	Saccon Daniele <i>Progettista</i>	2016-03-22
v 0.0.6	<i>Creazione delle sezioni profilo, bacheca e chat nei diagrammi delle attività</i>	Saccon Daniele <i>Progettista</i>	2016-03-21
v 0.0.5	<i>Creazione Capitolo Tecnologie utilizzate</i>	Parise Ivan <i>Progettista</i>	2016-03-20
v 0.0.4	<i>Creazione Capitolo Introduzione</i>	Parise Ivan <i>Progettista</i>	2016-03-19
v 0.0.3	<i>Creazione delle sezioni diagrammi delle attività e attività principali</i>	Saccon Daniele <i>Progettista</i>	2016-03-18
v 0.0.2	<i>Creazione Introduzione</i>	Parise Ivan <i>Progettista</i>	2016-03-09
v 0.0.1	<i>Creazione dello scheletro del documento</i>	Filinesi Skrypnyk Oleksandr <i>Progettista</i>	2016-03-08



Indice

1 Introduzione	1
1.1 Scopo del documento	1
1.2 Scopo del prodotto	1
1.3 Glossario	1
1.4 Riferimenti	1
1.4.1 Normativi	1
1.4.2 Informativi	1
2 Tecnologie Utilizzate	2
2.1 Linguaggio di Programmazione	2
2.1.1 Java	2
2.2 Stili Architetturali	2
2.2.1 REST	2
2.3 Framework	4
2.3.1 Akka	4
2.3.2 Play	5
2.4 Infrastruttura Server	7
2.4.1 Amazon Web Services	7
2.5 Database	8
2.5.1 MySQL	8
2.5.2 JDBC	9
2.6 Client	9
2.6.1 Android SDK	9
2.6.2 Picasso	10
2.6.3 AltBeacon	10
3 Descrizione Architettura	11
3.1 Metodo e formalismo di specifica	11
3.2 Architettura Generale	11
3.2.1 Architettura Frontend _G	12
3.2.2 Architettura Backend _G	12
4 Architettura Database	14
4.1 Struttura Database	14
4.1.1 Sezione Base	14
4.1.2 Sezione Profilo	15
4.1.3 Sezione Gioco	16
4.1.4 Sezione Classifica	17
5 Componenti e classi	18
5.1 Client	18
5.1.1 Informazioni generali	18
5.2 Client::BaseFunctions	19
5.2.1 Informazioni generali	19
5.3 Client::BaseFunctions::Model	20
5.3.1 Informazioni generali	20
5.3.2 Classi	20
5.3.2.1 Client::BaseFunctions::Model::EnviromentModel	20
5.3.2.2 Client::BaseFunctions::Model::EnviromentModelImpl	21



Indice

5.3.2.3	Client::BaseFunctions::Model::EnviromentServiceModel	21
5.3.2.4	Client::BaseFunctions::Model::EnviromentServiceModelImpl	21
5.4	Client::BaseFunctions::Model::Communication	21
5.4.1	Informazioni generali	21
5.4.2	Classi	21
5.4.2.1	Client::BaseFunctions::Model::Communication ::Communication	21
5.4.2.2	Client::BaseFunctions::Model::Communication ::CommunicationImpl	22
5.4.2.3	Client::BaseFunctions::Model::Communication ::EnviromentCommunication	22
5.4.2.4	Client::BaseFunctions::Model::Communication ::EnviromentCommunicationImpl	22
5.5	Client::BaseFunctions::Presenter	22
5.5.1	Informazioni generali	22
5.5.2	Classi	23
5.5.2.1	Client::BaseFunctions::Presenter::BaseFunctionsManager	23
5.5.2.2	Client::BaseFunctions::Presenter::BaseFunctionsManagerImpl	23
5.5.2.3	Client::BaseFunctions::Presenter::EnviromentPresenter	24
5.5.2.4	Client::BaseFunctions::Presenter::EnviromentPresenterImpl	24
5.5.2.5	Client::BaseFunctions::Presenter::ErrorPresenter . .	24
5.5.2.6	Client::BaseFunctions::Presenter::ErrorPresenterImpl	25
5.5.2.7	Client::BaseFunctions::Presenter::MainMenuPresenter	25
5.5.2.8	Client::BaseFunctions::Presenter::MainMenuPresenterImpl	25
5.6	Client::BaseFunctions::Types	25
5.6.1	Informazioni generali	25
5.6.2	Classi	26
5.6.2.1	Client::BaseFunctions::Types::Beacon	26
5.6.2.2	Client::BaseFunctions::Types::BeaconImpl	26
5.6.2.3	Client::BaseFunctions::Types::ModuleIndex	26
5.6.2.4	Client::BaseFunctions::Types::ModuleIndexImpl . .	26
5.6.2.5	Client::BaseFunctions::Types::Profile	27
5.6.2.6	Client::BaseFunctions::Types::ProfileImpl	27
5.7	Client::BaseFunctions::Types::Errors	27
5.7.1	Informazioni generali	27
5.7.2	Classi	27
5.7.2.1	Client::BaseFunctions::Types::Errors::BluetoothError	27
5.7.2.2	Client::BaseFunctions::Types::Errors::BluetoothErrorImpl	28
5.7.2.3	Client::BaseFunctions::Types::Errors::EmptyTeamNameError	28
5.7.2.4	Client::BaseFunctions::Types::Errors::EmptyTeamNameErrorImpl	28
5.7.2.5	Client::BaseFunctions::Types::Errors::Error	28
5.7.2.6	Client::BaseFunctions::Types::Errors::ErrorImpl . .	29
5.7.2.7	Client::BaseFunctions::Types::Errors::InternetError .	29
5.7.2.8	Client::BaseFunctions::Types::Errors::InternetErrorImpl	29
5.7.2.9	Client::BaseFunctions::Types::Errors::ServerError . .	30
5.7.2.10	Client::BaseFunctions::Types::Errors::ServerErrorImpl	30
5.8	Client::BaseFunctions::View	30
5.8.1	Informazioni generali	30
5.8.2	Classi	30
5.8.2.1	Client::BaseFunctions::View::EnviromentErrorView .	30



Indice

5.8.2.2	Client::BaseFunctions::View::EnviromentErrorViewImpl	31
5.8.2.3	Client::BaseFunctions::View::MainMenuView	31
5.8.2.4	Client::BaseFunctions::View::MainMenuViewImpl	31
5.9	Client::Games	32
5.9.1	Informazioni generali	32
5.10	Client::Games::Battleship	33
5.10.1	Informazioni generali	33
5.11	Client::Games::Battleship::Model	34
5.11.1	Informazioni generali	34
5.11.2	Classi	34
5.11.2.1	Client::Games::Battleship::Model::AttackPhaseModel	34
5.11.2.2	Client::Games::Battleship::Model::AttackPhaseModelImpl	34
5.11.2.3	Client::Games::Battleship::Model::BattleshipModel	35
5.11.2.4	Client::Games::Battleship::Model::BattleshipModelImpl	36
5.11.2.5	Client::Games::Battleship::Model::CreateTeamModel	36
5.11.2.6	Client::Games::Battleship::Model::CreateTeamModelImpl	36
5.11.2.7	Client::Games::Battleship::Model::EnemyAttackPhaseModel	37
5.11.2.8	Client::Games::Battleship::Model::EnemyAttackPhaseModelImpl	37
5.11.2.9	Client::Games::Battleship::Model::JoinTeamModel	37
5.11.2.10	Client::Games::Battleship::Model::JoinTeamModelImpl	37
5.11.2.11	Client::Games::Battleship::Model::LeaderboardModel	37
5.11.2.12	Client::Games::Battleship::Model::LeaderboardModelImpl	38
5.11.2.13	Client::Games::Battleship::Model::ProfileModel	38
5.11.2.14	Client::Games::Battleship::Model::ProfileModelImpl	38
5.11.2.15	Client::Games::Battleship::Model::SelectShipPositionModel	38
5.11.2.16	Client::Games::Battleship::Model::SelectShipPositionModelImpl	38
5.11.2.17	Client::Games::Battleship::Model::ShowFinalScoreModel	39
5.11.2.18	Client::Games::Battleship::Model::ShowFinalScoreModelImpl	39
5.11.2.19	Client::Games::Battleship::Model::TeamManagementModel	39
5.11.2.20	Client::Games::Battleship::Model::TeamManagementModelImpl	39
5.11.2.21	Client::Games::Battleship::Model::WaitingOpponentModel	39
5.11.2.22	Client::Games::Battleship::Model::WaitingOpponentModelImpl	40
5.12	Client::Games::Battleship::Model::Communication	40
5.12.1	Informazioni generali	40
5.12.2	Classi	40
5.12.2.1	Client::Games::Battleship::Model::Communication ::AttackPhaseCommunication	40
5.12.2.2	Client::Games::Battleship::Model::Communication ::AttackPhaseCommunicationImpl	40
5.12.2.3	Client::Games::Battleship::Model::Communication ::BattleshipCommunication	40
5.12.2.4	Client::Games::Battleship::Model::Communication ::BattleshipCommunicationImpl	41
5.12.2.5	Client::Games::Battleship::Model::Communication ::CreateTeamCommunication	41
5.12.2.6	Client::Games::Battleship::Model::Communication ::CreateTeamCommunicationImpl	41
5.12.2.7	Client::Games::Battleship::Model::Communication ::EnemyAttackPhaseCommunication	42



Indice

5.12.2.8	Client::Games::Battleship::Model::Communication ::EnemyAttackPhaseCommunicationImpl	42
5.12.2.9	Client::Games::Battleship::Model::Communication ::JoinTeamCommunication	42
5.12.2.10	Client::Games::Battleship::Model::Communication ::JoinTeamCommunicationImpl	42
5.12.2.11	Client::Games::Battleship::Model::Communication ::LeaderboardCommunication	43
5.12.2.12	Client::Games::Battleship::Model::Communication ::LeaderboardCommunicationImpl	43
5.12.2.13	Client::Games::Battleship::Model::Communication ::ProfileCommunication	43
5.12.2.14	Client::Games::Battleship::Model::Communication ::ProfileCommunicationImpl	43
5.12.2.15	Client::Games::Battleship::Model::Communication ::SelectShipPositionCommunication	43
5.12.2.16	Client::Games::Battleship::Model::Communication ::SelectShipPositionCommunicationImpl	44
5.12.2.17	Client::Games::Battleship::Model::Communication ::ShowFinalScoreCommunication	44
5.12.2.18	Client::Games::Battleship::Model::Communication ::ShowFinalScoreCommunicationImpl	44
5.12.2.19	Client::Games::Battleship::Model::Communication ::TeamManagementCommunication	44
5.12.2.20	Client::Games::Battleship::Model::Communication ::TeamManagementCommunicationImpl	44
5.12.2.21	Client::Games::Battleship::Model::Communication ::WaitingOpponentCommunication	45
5.12.2.22	Client::Games::Battleship::Model::Communication ::WaitingOpponentCommunicationImpl	45
5.13	Client::Games::Battleship::Presenter	45
5.13.1	Informazioni generali	45
5.13.2	Classi	45
5.13.2.1	Client::Games::Battleship::Presenter::AttackPhasePresenter	45
5.13.2.2	Client::Games::Battleship::Presenter::AttackPhasePresenterImpl	46
5.13.2.3	Client::Games::Battleship::Presenter::BattleshipManager	46
5.13.2.4	Client::Games::Battleship::Presenter::BattleshipManagerImpl	47
5.13.2.5	Client::Games::Battleship::Presenter::CreateTeamPresenter	47
5.13.2.6	Client::Games::Battleship::Presenter::CreateTeamPresenterImpl	47
5.13.2.7	Client::Games::Battleship::Presenter::EnemyAttackPhasePresenter	47
5.13.2.8	Client::Games::Battleship::Presenter::EnemyAttackPhasePresenterImpl	48
5.13.2.9	Client::Games::Battleship::Presenter::FinalScorePresenter	48
5.13.2.10	Client::Games::Battleship::Presenter::FinalScorePresenterImpl	49
5.13.2.11	Client::Games::Battleship::Presenter::JoinTeamPresenter	49
5.13.2.12	Client::Games::Battleship::Presenter::JoinTeamPresenterImpl	49
5.13.2.13	Client::Games::Battleship::Presenter::LeaderboardPresenter	50
5.13.2.14	Client::Games::Battleship::Presenter::LeaderboardPresenterImpl	50
5.13.2.15	Client::Games::Battleship::Presenter::SelectShipPositionPresenter	50
5.13.2.16	Client::Games::Battleship::Presenter::SelectShipPositionPresenterImpl	51
5.13.2.17	Client::Games::Battleship::Presenter::SenderShotPresenter	51



Indice

5.13.2.18 Client::Games::Battleship::Presenter::SenderShotPresenterImpl	51
5.13.2.19 Client::Games::Battleship::Presenter::StartGamePresenter	51
5.13.2.20 Client::Games::Battleship::Presenter::StartGamePresenterImpl	52
5.13.2.21 Client::Games::Battleship::Presenter::TeamManagementPresenter	52
5.13.2.22 Client::Games::Battleship::Presenter::TeamManagementPresenterImpl	53
5.13.2.23 Client::Games::Battleship::Presenter::WaitingOpponentPresenter	53
5.13.2.24 Client::Games::Battleship::Presenter::WaitingOpponentPresenterImpl	53
5.14 Client::Games::Battleship::Types	53
5.14.1 Informazioni generali	53
5.14.2 Classi	53
5.14.2.1 Client::Games::Battleship::Types::Cell	53
5.14.2.2 Client::Games::Battleship::Types::CellImpl	54
5.14.2.3 Client::Games::Battleship::Types::Field	54
5.14.2.4 Client::Games::Battleship::Types::FieldImpl	55
5.14.2.5 Client::Games::Battleship::Types::Grid	55
5.14.2.6 Client::Games::Battleship::Types::GridImpl	55
5.14.2.7 Client::Games::Battleship::Types::Position	56
5.14.2.8 Client::Games::Battleship::Types::PositionImpl	56
5.14.2.9 Client::Games::Battleship::Types::Ship	56
5.14.2.10 Client::Games::Battleship::Types::ShipImpl	57
5.14.2.11 Client::Games::Battleship::Types::ShipType	57
5.14.2.12 Client::Games::Battleship::Types::ShipTypeImpl	57
5.14.2.13 Client::Games::Battleship::Types::Shoot	57
5.14.2.14 Client::Games::Battleship::Types::ShootImpl	58
5.15 Client::Games::Battleship::View	58
5.15.1 Informazioni generali	58
5.15.2 Classi	58
5.15.2.1 Client::Games::Battleship::View::AttackPhaseView	58
5.15.2.2 Client::Games::Battleship::View::AttackPhaseViewImpl	59
5.15.2.3 Client::Games::Battleship::View::ChooseRoleView	59
5.15.2.4 Client::Games::Battleship::View::ChooseRoleViewImpl	59
5.15.2.5 Client::Games::Battleship::View::CreateTeamView	60
5.15.2.6 Client::Games::Battleship::View::CreateTeamViewImpl	60
5.15.2.7 Client::Games::Battleship::View::EnemyAttackPhaseView	60
5.15.2.8 Client::Games::Battleship::View::EnemyAttackPhaseViewImpl	61
5.15.2.9 Client::Games::Battleship::View::FinalScoreView	61
5.15.2.10 Client::Games::Battleship::View::FinalScoreViewImpl	61
5.15.2.11 Client::Games::Battleship::View::JoinTeamView	61
5.15.2.12 Client::Games::Battleship::View::JoinTeamViewImpl	62
5.15.2.13 Client::Games::Battleship::View::MembersAdapter	62
5.15.2.14 Client::Games::Battleship::View::SelectShipPositionView	62
5.15.2.15 Client::Games::Battleship::View::SelectShipPositionViewImpl	63
5.15.2.16 Client::Games::Battleship::View::TeamManagementView	63
5.15.2.17 Client::Games::Battleship::View::TeamManagementViewImpl	63
5.15.2.18 Client::Games::Battleship::View::WaitingOpponentView	64
5.15.2.19 Client::Games::Battleship::View::WaitingOpponentViewImpl	64
5.16 Client::Games::Utility	64
5.16.1 Informazioni generali	64
5.17 Client::Games::Utility::Model	65
5.17.1 Informazioni generali	65



Indice

5.17.2 Classi	65
5.17.2.1 Client::Games::Utility::Model::GamesListModel	65
5.17.2.2 Client::Games::Utility::Model::GamesListModelImpl	65
5.18 Client::Games::Utility::Presenter	65
5.18.1 Informazioni generali	65
5.18.2 Classi	66
5.18.2.1 Client::Games::Utility::Presenter::GamesListPresenter	66
5.18.2.2 Client::Games::Utility::Presenter::GamesListPresenterImpl	66
5.18.2.3 Client::Games::Utility::Presenter::UtilityManager	66
5.18.2.4 Client::Games::Utility::Presenter::UtilityManagerImpl	67
5.19 Client::Games::Utility::Types	67
5.19.1 Informazioni generali	67
5.19.2 Classi	67
5.19.2.1 Client::Games::Utility::Types::Game	67
5.19.2.2 Client::Games::Utility::Types::GameImpl	67
5.19.2.3 Client::Games::Utility::Types::Score	68
5.19.2.4 Client::Games::Utility::Types::ScoreImpl	68
5.19.2.5 Client::Games::Utility::Types::Team	68
5.19.2.6 Client::Games::Utility::Types::TeamImpl	69
5.20 Client::Games::Utility::View	69
5.20.1 Informazioni generali	69
5.20.2 Classi	69
5.20.2.1 Client::Games::Utility::View::GamesAdapter	69
5.20.2.2 Client::Games::Utility::View::GamesListView	69
5.20.2.3 Client::Games::Utility::View::GamesListViewImpl	70
5.20.2.4 Client::Games::Utility::View::LeaderboardView	70
5.20.2.5 Client::Games::Utility::View::LeaderboardViewImpl	70
5.20.2.6 Client::Games::Utility::View::ScoreAdapter	70
5.21 Client::Profile	71
5.21.1 Informazioni generali	71
5.22 Client::Profile::Model	71
5.22.1 Informazioni generali	71
5.22.2 Classi	72
5.22.2.1 Client::Profile::Model::ProfileModel	72
5.22.2.2 Client::Profile::Model::ProfileModelImpl	72
5.23 Client::Profile::Model::Communication	72
5.23.1 Informazioni generali	72
5.23.2 Classi	72
5.23.2.1 Client::Profile::Model::Communication::ProfileCommunication	72
5.23.2.2 Client::Profile::Model::Communication::ProfileCommunicationImpl	73
5.24 Client::Profile::Presenter	73
5.24.1 Informazioni generali	73
5.24.2 Classi	73
5.24.2.1 Client::Profile::Presenter::ProfileManager	73
5.24.2.2 Client::Profile::Presenter::ProfileManagerImpl	74
5.24.2.3 Client::Profile::Presenter::ProfilePresenter	74
5.24.2.4 Client::Profile::Presenter::ProfilePresenterImpl	74
5.25 Client::Profile::View	74
5.25.1 Informazioni generali	74
5.25.2 Classi	75



Indice

5.25.2.1 Client::Profile::View::ProfileView	75
5.25.2.2 Client::Profile::View::ProfileViewImpl	75
5.26 Client::Types	75
5.26.1 Informazioni generali	75
5.26.2 Classi	76
5.26.2.1 Client::Types::Profile	76
5.26.2.2 Client::Types::ProfileImpl	76
5.26.2.3 Client::Types::Session	76
5.26.2.4 Client::Types::SessionImpl	76
5.27 Server	77
5.27.1 Informazioni generali	77
5.28 Server::Microservices	77
5.28.1 Informazioni generali	77
5.29 Server::Microservices::Battleship	78
5.29.1 Informazioni generali	78
5.30 Server::Microservices::Battleship::Business	79
5.30.1 Informazioni generali	79
5.30.2 Classi	79
5.30.2.1 Server::Microservices::Battleship::Business::BattleshipBusiness	79
5.30.2.2 Server::Microservices::Battleship::Business::BattleshipBusinessImpl	79
5.31 Server::Microservices::Battleship::Communication	80
5.31.1 Informazioni generali	80
5.31.2 Classi	80
5.31.2.1 Server::Microservices::Battleship::Communication::BattleshipCommunication	80
5.31.2.2 Server::Microservices::Battleship::Communication::BattleshipCommunicationImpl	80
5.32 Server::Microservices::Battleship::Persistence	81
5.32.1 Informazioni generali	81
5.32.2 Classi	81
5.32.2.1 Server::Microservices::Battleship::Persistence::BattleshipPersistence	81
5.32.2.2 Server::Microservices::Battleship::Persistence::BattleshipPersistenceImpl	81
5.33 Server::Microservices::Battleship::Persistence::Dao	81
5.33.1 Informazioni generali	81
5.33.2 Classi	82
5.33.2.1 Server::Microservices::Battleship::Persistence::Dao::BattleshipDao	82
5.33.2.2 Server::Microservices::Battleship::Persistence::Dao::DaoFactory	82
5.34 Server::Microservices::Battleship::Persistence::Dao::Sql	82
5.34.1 Informazioni generali	82
5.34.2 Classi	82
5.34.2.1 Server::Microservices::Battleship::Persistence::Dao::Sql::SqlBattleshipDaoImpl	82
5.34.2.2 Server::Microservices::Battleship::Persistence::Dao::Sql::SqlDaoFactoryImpl	82
5.35 Server::Microservices::Battleship::Services	83
5.35.1 Informazioni generali	83
5.35.2 Classi	83
5.35.2.1 Server::Microservices::Battleship::Services::BattleshipServices	83
5.35.2.2 Server::Microservices::Battleship::Services::BattleshipServicesImpl	83
5.36 Server::Microservices::Battleship::Types	83
5.36.1 Informazioni generali	83
5.36.2 Classi	84
5.36.2.1 Server::Microservices::Battleship::Types::AdminField	84



Indice

5.36.2.2	Server::Microservices::Battleship::Types::AdminFieldImpl	84
5.36.2.3	Server::Microservices::Battleship::Types::Beacon . . .	84
5.36.2.4	Server::Microservices::Battleship::Types::BeaconImpl	84
5.36.2.5	Server::Microservices::Battleship::Types::Cell	84
5.36.2.6	Server::Microservices::Battleship::Types::CellImpl . .	85
5.36.2.7	Server::Microservices::Battleship::Types::Field	85
5.36.2.8	Server::Microservices::Battleship::Types::FieldImpl . .	85
5.36.2.9	Server::Microservices::Battleship::Types::Game	85
5.36.2.10	Server::Microservices::Battleship::Types::GameImpl	86
5.36.2.11	Server::Microservices::Battleship::Types::Grid	86
5.36.2.12	Server::Microservices::Battleship::Types::GridImpl . .	86
5.36.2.13	Server::Microservices::Battleship::Types::Local	86
5.36.2.14	Server::Microservices::Battleship::Types::LocalImpl . .	87
5.36.2.15	Server::Microservices::Battleship::Types::Position . . .	87
5.36.2.16	Server::Microservices::Battleship::Types::PositionImpl	87
5.36.2.17	Server::Microservices::Battleship::Types::Profile	87
5.36.2.18	Server::Microservices::Battleship::Types::ProfileImpl	87
5.36.2.19	Server::Microservices::Battleship::Types::Score	88
5.36.2.20	Server::Microservices::Battleship::Types::ScoreImpl . .	88
5.36.2.21	Server::Microservices::Battleship::Types::Ship	88
5.36.2.22	Server::Microservices::Battleship::Types::ShipImpl . .	88
5.36.2.23	Server::Microservices::Battleship::Types::ShipType . .	89
5.36.2.24	Server::Microservices::Battleship::Types::ShipTypeImpl	89
5.36.2.25	Server::Microservices::Battleship::Types::Shot	89
5.36.2.26	Server::Microservices::Battleship::Types::ShotImpl . .	89
5.36.2.27	Server::Microservices::Battleship::Types::Table	89
5.36.2.28	Server::Microservices::Battleship::Types::TableImpl . .	90
5.36.2.29	Server::Microservices::Battleship::Types::Team	90
5.36.2.30	Server::Microservices::Battleship::Types::TeamField	90
5.36.2.31	Server::Microservices::Battleship::Types::TeamFieldImpl	90
5.36.2.32	Server::Microservices::Battleship::Types::TeamImpl . .	91
5.37	Server::Microservices::Profile	91
5.37.1	Informazioni generali .	91
5.38	Server::Microservices::Profile::Business	92
5.38.1	Informazioni generali .	92
5.38.2	Classi .	92
5.38.2.1	Server::Microservices::Profile::Business::ProfileBusiness	92
5.38.2.2	Server::Microservices::Profile::Business::ProfileBusinessImpl	92
5.39	Server::Microservices::Profile::Communication	93
5.39.1	Informazioni generali .	93
5.39.2	Classi .	93
5.39.2.1	Server::Microservices::Profile::Communication::ProfileCommunication	93
5.39.2.2	Server::Microservices::Profile::Communication::ProfileCommunicationImpl	93
5.40	Server::Microservices::Profile::Persistence	94
5.40.1	Informazioni generali .	94
5.40.2	Classi .	94
5.40.2.1	Server::Microservices::Profile::Persistence::ProfilePersistence	94
5.40.2.2	Server::Microservices::Profile::Persistence::ProfilePersistenceImpl	94
5.41	Server::Microservices::Profile::Persistence::Dao	95
5.41.1	Informazioni generali .	95



Indice

5.41.2 Classi	95
5.41.2.1 Server::Microservices::Profile::Persistence::Dao::BeaconDao	95
5.41.2.2 Server::Microservices::Profile::Persistence::Dao::DaoFactory	95
5.41.2.3 Server::Microservices::Profile::Persistence::Dao::ProfileDao	96
5.42 Server::Microservices::Profile::Persistence::Dao::Sql	96
5.42.1 Informazioni generali	96
5.42.2 Classi	96
5.42.2.1 Server::Microservices::Profile::Persistence::Dao::Sql::SqlBeaconDaoImpl	96
5.42.2.2 Server::Microservices::Profile::Persistence::Dao::Sql::SqlDaoFactoryImpl	96
5.42.2.3 Server::Microservices::Profile::Persistence::Dao::Sql::SqlProfileDaoImpl	97
5.43 Server::Microservices::Profile::Services	97
5.43.1 Informazioni generali	97
5.43.2 Classi	97
5.43.2.1 Server::Microservices::Profile::Services::ProfileServices	97
5.43.2.2 Server::Microservices::Profile::Services::ProfileServicesImpl	98
5.44 Server::Microservices::Profile::Types	98
5.44.1 Informazioni generali	98
5.44.2 Classi	98
5.44.2.1 Server::Microservices::Profile::Types::Beacon	98
5.44.2.2 Server::Microservices::Profile::Types::BeaconImpl	99
5.44.2.3 Server::Microservices::Profile::Types::Profile	99
5.44.2.4 Server::Microservices::Profile::Types::ProfileImpl	99
5.45 Server::Microservices::Session	100
5.45.1 Informazioni generali	100
5.46 Server::Microservices::Session::Business	101
5.46.1 Informazioni generali	101
5.46.2 Classi	101
5.46.2.1 Server::Microservices::Session::Business::SessionBusiness	101
5.46.2.2 Server::Microservices::Session::Business::SessionBusinessImpl	101
5.47 Server::Microservices::Session::Communication	101
5.47.1 Informazioni generali	101
5.47.2 Classi	102
5.47.2.1 Server::Microservices::Session::Communication::SessionCommunication	102
5.47.2.2 Server::Microservices::Session::Communication::SessionCommunicationImpl	102
5.48 Server::Microservices::Session::Persistence	102
5.48.1 Informazioni generali	102
5.48.2 Classi	103
5.48.2.1 Server::Microservices::Session::Persistence::SessionPersistence	103
5.48.2.2 Server::Microservices::Session::Persistence::SessionPersistenceImpl	103
5.49 Server::Microservices::Session::Persistence::Dao	103
5.49.1 Informazioni generali	103
5.49.2 Classi	103
5.49.2.1 Server::Microservices::Session::Persistence::Dao::DaoFactory	103
5.49.2.2 Server::Microservices::Session::Persistence::Dao::SessionDao	104
5.50 Server::Microservices::Session::Persistence::Dao::Sql	104
5.50.1 Informazioni generali	104
5.50.2 Classi	104
5.50.2.1 Server::Microservices::Session::Persistence::Dao::Sql::SqlDaoFactoryImpl	104
5.50.2.2 Server::Microservices::Session::Persistence::Dao::Sql::SqlSessionDaoImpl	105
5.51 Server::Microservices::Session::Services	105



Indice

5.51.1	Informazioni generali	105
5.51.2	Classi	105
5.51.2.1	Server::Microservices::Session::Services::SessionServices	105
5.51.2.2	Server::Microservices::Session::Services::SessionServicesImpl	106
5.52	Server::Microservices::Session::Types	106
5.52.1	Informazioni generali	106
5.52.2	Classi	106
5.52.2.1	Server::Microservices::Session::Types::Error	106
5.52.2.2	Server::Microservices::Session::Types::ErrorConstants	106
5.52.2.3	Server::Microservices::Session::Types::ErrorImpl . . .	106
5.52.2.4	Server::Microservices::Session::Types::Session	107
5.52.2.5	Server::Microservices::Session::Types::SessionImpl . .	107
6	Design Pattern	108
6.1	Design Pattern Architetturali	109
6.1.1	Microservices Architecture	109
6.1.2	Model View Presenter	110
6.2	Design Pattern Creazionali	111
6.2.1	Singleton	111
6.2.2	Abstract Factory	111
6.3	Design Pattern Strutturali	112
6.3.1	Adapter Pattern	112
6.3.2	Data Access Object DAO	112
6.3.3	Facade	113
6.4	Pattern comportamentali	114
6.4.1	Observer Pattern	114
7	Stime di Fattibilità e Bisogno di Risorse	115
8	Tracciamento	116
8.1	Tracciamento Classi-Requisiti	116
8.2	Tracciamento Requisiti-Classi	127
A	Descrizione Design Pattern	142
A.1	Design Pattern Architetturali	142
A.1.1	Microservice Architecture	142
A.1.2	Model View Presenter	142
A.2	Design Pattern Creazionali	143
A.2.1	Abstract Factory	143
A.3	Design Pattern Strutturali	144
A.3.1	Adapter Pattern	144
A.3.2	Data Access Object DAO	145
A.3.3	Facade	145
A.4	Design Pattern Comportamentali	146
A.4.1	Observer Pattern	146



Elenco delle tabelle

Elenco delle tabelle

1	Tracciamento Classi-Requisiti	126
2	Tracciamento Requisiti-Classi	141



Elenco delle figure

Elenco delle figure

1	Tecnologie utilizzate - Play Framework MVC pattern	6
2	Descrizione architettura - Architettura generale	11
3	Descrizione architettura - Model View Presenter	12
4	Descrizione architettura - Architettura Backend	13
5	Sezione schema E-R, relazioni base.	14
6	Sezione schema E-R, relazioni sul profilo utente.	15
7	Sezione schema E-R, relazioni sul gioco Battaglia Navale.	16
8	Sezione schema E-R, relazioni sulle classifiche.	17
9	Client	18
10	Client::BaseFunctions	19
11	Client::Games	32
12	Client::Games::Battleship	33
13	Client::Games::Utility	64
14	Client::Profile	71
15	Client::Types	75
16	Server	77
17	Server::Microservices	77
18	Server::Microservices::Battleship	78
19	Server::Microservices::Profile	91
20	Server::Microservices::Session	100
21	Diagramma della Microservices Architecture applicata al progetto CLIPS109	
22	Pattern MVP applicato al progetto CLIPS	110
23	Pattern Abstract Factory applicato al progetto CLIPS	111
24	Pattern Abstract Factory applicato al progetto CLIPS	111
25	DAO applicato al progetto CLIPS	112
26	Facade applicato al progetto CLIPS	113
27	Pattern Abstract Factory applicato al progetto CLIPS	114
28	Diagramma della Microservices Architecture	142
29	Diagramma del pattern Model View Presenter	143
30	Diagramma del pattern Abstract Factory	144
31	Diagramma del pattern Adapter	144
32	Diagramma del pattern DAO	145
33	Diagramma del pattern Facade	146
34	Diagramma del pattern Observer	146



1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione ad alto livello del progetto *Clips*.

Verrà presentata l'architettura generale secondo la quale saranno organizzate le varie componenti software e saranno descritti i Design Pattern_G utilizzati.

1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di un'applicazione in grado di sfruttare i Beacon_G all'interno di un locale/bar per fare interagire i vari clienti dei tavoli mediante chat tavolo/-tavolo, bacheca del locale sulla quale pubblicare i contenuti e dei semplici giochi multiplayer. L'utilizzo dei giochi sarà la parte core dell'applicazione, in quanto l'utente sarà invogliato a sfidare gli altri tavoli, potrà partecipare al premio del giorno scalando la classifica in base ai punteggi ottenuti dalle varie sfide affrontate.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento Glossario v3.0.0. Ogni occorrenza di vocaboli presenti nel Glossario è marcata da una "G" maiuscola in pedice.

1.4 Riferimenti

1.4.1 Normativi

- Analisi dei requisiti: **Analisi dei Requisiti v.4.0.0**;
- Norme di Progetto: **Norme di Progetto v.4.0.0**.

1.4.2 Informativi

- Java coding Conventions & Style
<http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>
- Android SDK
<http://developer.android.com/index.html>
- Amazon Web Services
https://aws.amazon.com/it/about-aws/global-infrastructure/?nc2=h_12_cc
- MySQL
<https://www.mysql.it/>
- Descrizione dei Design Pattern_G
http://sourcemaking.com/design_patterns;
- Play
<http://www.playframework.com/documentation/2.5.x/Philosophy>
- Design Patterns: Elementi per il riuso di software a oggetti - E. Gamma, R. Helm, R. Johnson, J. Vlissides - Prima Edizione (2002)
- Rest API Documentation <http://v2.wp-api.org/>
- AltBeacon
<http://altbeacon.org/>



2 Tecnologie Utilizzate

2 Tecnologie Utilizzate

In questa sezione vengono descritte le tecnologie su cui si basa lo sviluppo del progetto e la motivazione del loro utilizzo.

2.1 Linguaggio di Programmazione

2.1.1 Java

Si è deciso di utilizzare Java per questi motivi:

- Compatibilità multipiattaforma: grazie alla Java Virtual Machine_G che trasla il codice in bytecode rendendolo compatibile con il codice macchina in accordo al sistema operativo su cui è installata.
- Conoscenza: il gruppo ha già avuto modo di utilizzare il linguaggio Java, sia per progetti di altri corsi universitari sia per esperienze personali, permettendo di raggiungere un buon grado di conoscenza;
- Android: Java è il linguaggio di programmazione utilizzato per sviluppare Apps su Android_G.

Svantaggi: Per quanto riguarda gli aspetti negativi, il linguaggio presenta un maggiore utilizzo di memoria e offre un'efficienza minore in termini computazionali, rispetto a linguaggi concorrenti come, per esempio, il C++.

2.2 Stili Architetturali

2.2.1 REST

REpresentational State Transfer (REST) è un tipo di architettura software per i sistemi di ipertesto distribuiti come il World Wide Web.

REST si riferisce ad un insieme di principi di architetture di rete, i quali delineano come le risorse sono definite e indirizzate. Il termine è spesso usato nel senso di descrivere ogni semplice interfaccia che trasmette dati su HTTP_G senza un livello opzionale come SOAP_G o la gestione della sessione tramite i cookie. Questi due concetti possono andare in conflitto così come in sovrapposizione. È possibile progettare ogni sistema software complesso in accordo con l'architettura REST senza usare HTTP_G e senza interagire con il World Wide Web. È anche possibile progettare una semplice interfaccia XML+HTTP che non sia conforme ai principi REST, e invece segua un modello di Remote Procedure Call.

I sistemi che seguono i principi REST sono spesso definiti "RESTful". REST prevede che la scalabilità del Web e la crescita siano diretti risultati di pochi principi chiave di progettazione:

- lo stato dell'applicazione e le funzionalità sono divisi in risorse web;
- ogni risorsa è unica e indirizzabile usando sintassi universale per uso nei link ipertestuali;
- tutte le risorse sono condivise come interfaccia uniforme per il trasferimento di stato tra client e risorse, questo consiste in:
 - un insieme vincolato di operazioni ben definite;
 - un insieme vincolato di contenuti, optionalmente supportato da codice a richiesta; un protocollo che è:
 - * client-server;
 - * privo di stato (stateless);
 - * memorizzabile in cache (cacheable);
 - * a livelli.



2 Tecnologie Utilizzate

L'approccio architetturale REST è definito dai seguenti sei vincoli applicati ad una architettura, mentre si lascia libera l'implementazione dei singoli componenti.

- **Client-server:** un insieme di interfacce uniformi separa i client dai server. Questa separazione di ruoli e preoccupazioni significa che, per esempio, il client non si deve preoccupare del salvataggio delle informazioni, che rimane all'interno di ogni singolo server, in questo modo la portabilità del codice del client ne trae vantaggio. I server non si devono preoccupare dell'interfaccia grafica o dello stato dell'utente, in questo modo i server sono più semplici e maggiormente scalabili. Server e client possono essere sostituiti e sviluppati indipendentemente fintanto che l'interfaccia non viene modificata;
- **Stateless:** la comunicazione client-server è ulteriormente vincolata in modo che nessun contesto client venga memorizzato sul server tra le richieste. Ogni richiesta da ogni client contiene tutte le informazioni necessarie per richiedere il servizio, e lo stato della sessione è contenuto sul client. Nota importante lo stato della sessione può essere trasferito al server attraverso un altro servizio posto a persistere ad esempio lo stato su database;
- **Cacheable:** come nel World Wide Web, i client possono fare caching delle risposte. Le risposte devono in ogni modo definirsi implicitamente o esplicitamente cacheable o no, in modo da prevenire che i client possano riusare stati vecchi o dati errati. Una gestione ben fatta della cache può ridurre o parzialmente eliminare le comunicazioni client server, migliorando scalabilità e performance;
- **Layered system:** un client non può dire se è connesso direttamente ad un server di livello più basso od intermedio, i server intermedi possono migliorare la scalabilità del sistema con load-balancing o con cache distribuite. Layer intermedi possono offrire inoltre politiche di sicurezza;
- **Code on demand:** (opzionale) I server possono temporaneamente estendere o personalizzare le funzionalità del client trasferendo del codice eseguibile. Ad esempio questo può includere componenti compilati come Applet Java o linguaggi di scripting client side come JavaScript. "Code on demand" è l'unico vincolo opzionale per la definizione di un'architettura REST.
- **Uniform interface:** un'interfaccia di comunicazione omogenea tra client e server permette di semplificare e disaccoppiare l'architettura, la quale si può evolvere separatamente.

Risorse

Un concetto importante in REST è l'esistenza di risorse (fonti di informazioni), a cui si può accedere tramite un identificatore globale (un URI). Per utilizzare le risorse, le componenti di una rete (componenti client e server) comunicano attraverso una interfaccia standard (ad es. **HTTP_G**) e si scambiano rappresentazioni di queste risorse (il documento che trasmette le informazioni). Ad esempio, una risorsa cerchio potrebbe accettare e restituire una rappresentazione che specifica un punto per il centro e il raggio, formattati nel formato SVG, ma potrebbe anche accettare e restituire una rappresentazione che specifica tre punti distinti qualsiasi lungo la circonferenza nel formato CSV.

Un numero qualsiasi di connettori (client, server, cache, tunnel, ecc...) può mediare la richiesta, ma ogni connettore interviene senza conoscere la "storia passata" delle altre richieste. Di conseguenza una applicazione può interagire con una risorsa conoscendo due cose: l'identificatore della risorsa e l'azione richiesta - non ha bisogno di sapere se ci sono proxy, gateway, firewalls, tunnel, ecc... tra essa e il server su cui è presente l'informazione cercata. L'applicazione comunque deve conoscere il formato dell'informazione (rappresentazione) restituita, tipicamente un documento JSON nel nostro caso, ma potrebbe essere anche un'immagine o qualsiasi altro contenuto.

Comunicazione

La comunicazione REST avviene solitamente tramite lo scambio di oggetti XML o JSON, il



2 Tecnologie Utilizzate

sistema nè utilizzerà di quest'ultimo tipo.

JSON, acronimo di JavaScript Object Notation, è un formato adatto all'interscambio di dati fra applicazioni client-server.

Viene usato in AJAX come alternativa a XML/XSLT. La semplicità di JSON ne ha decretato un rapido utilizzo specialmente nella programmazione in AJAX. Il suo uso tramite JavaScript è particolarmente semplice, infatti l'interprete è in grado di eseguirne il parsing tramite una semplice chiamata alla funzione eval(). Questo fatto lo ha reso velocemente molto popolare a causa della diffusione della programmazione in JavaScript nel mondo del Web.

Svantaggi Rest

- **Limitato al caso sincrono:** lo stile REST è legato a doppio nodo al protocollo HTTP. Pertanto l'unica comunicazione supportata è quella sincrona.
- **Governance ridotta:** le interfacce REST sono uniformi e implicite. Ciò è un grande vantaggio da diversi punti di vista, ma limita fortemente la possibilità di instaurare opportuni livelli di governance.
- **Supporto - Medio:** lo stile REST sta ottenendo sempre maggiore attenzione da parte della comunità IT e ciò si ripercuote sia sui framework disponibili sia sulle infrastrutture che ne prevedono l'adozione. Tuttavia ancora non si è giunti al livello di supporto offerto dai Web Service.
- **Tempo/Budget:** i servizi RESTful richiedono un investimento iniziale limitato e sono ottimali in scenari semplici. La difficoltà di dar luogo ad una vera propria governance e la limitazione alle sole comunicazioni sincrone, fa sì che in contesti enterprise l'utilizzo esclusivo di soluzioni REST richieda maggiori investimenti e renda il sistema più complesso e quindi costoso da gestire.

2.3 Framework

2.3.1 Akka

Akka è un framework ideato per realizzare facilmente applicazione distribuite e con un elevato grado di concorrenza sulla JVM. Essendo stato realizzato dalla Typesafe, l'azienda fondata dall'ideatore di Scala, è stato progettato alla luce di tutte le feature più avanzate del linguaggio Scala, ed è il framework consigliato dall'azienda stessa per chi vuole scrivere software in Scala utilizzando il modello ad Attori. Come già detto però Akka allo stesso tempo permette di scrivere codice anche in linguaggio Java, senza perdere nessuna delle funzionalità più importanti offerte dal framework. In questo modo chi conosce il linguaggio Scala e deve scrivere del codice ex novo può utilizzare Scala, mentre chi conosce solo il linguaggio Java oppure chi ha la necessità di fare un porting di codice Java già esistente, può utilizzare tranquillamente il linguaggio Java senza rinunciare alle potenzialità di Akka.

Sviluppando software ad Attori in Akka, oltre all'evidente vantaggio di poter produrre facilmente del software concorrente senza dover usare tecniche di locking, si hanno altri grossi vantaggi dati dalle funzionalità di remoting e di fault tolerance: una volta realizzata un'applicazione in Akka, si può decidere anche a runtime di spostare parte degli Attori che la compongono in un altro computer, in modo completamente trasparente al resto dell'applicazione. Inoltre, se l'applicazione è realizzata correttamente, dovrebbe poter tollerare un malfunzionamento di ogni sua parte, grazie alle funzioni di gracefully failure.



2 Tecnologie Utilizzate

2.3.2 Play

Play framework è una libreria di codice basato su Akka che fornisce tutte le funzionalità necessarie per sviluppare un'applicazione Web e metterla in esecuzione lato server.

Play è stato scritto completamente in Scala, utilizzando Akka ed il modello ad Attori, tuttavia da allo sviluppatore Web la possibilità di utilizzare sia il linguaggio Scala sia il linguaggio Java, in modo analogo ad Akka. Un'applicazione Web realizzata con Play è a tutti gli effetti un software utilizzante il modello ad Attori implementato su Akka, tuttavia Play è stato progettato in modo da rendere semplice ed intuitivo lo sviluppo di applicazioni web anche agli sviluppatori che non conoscono le fondamenta del modello ad attori. Lo sviluppo di applicazioni su Play segue infatti il semplice e consueto pattern Model-View-Controller, al quale molti sviluppatori sono già abituati dato che esso è utilizzato in molti altri framework per lo sviluppo Web, per esempio Ruby on Rails o Symphony. Il Play framework integra già dentro di se il server HTTP Netty, opportunamente adattato per essere perfettamente integrato con il resto della piattaforma basata su Akka. Esso supporta una gestione completamente asincrona ad eventi del trasferimento di dati.

Scalabilità

Play è stato progettato per lo sviluppo di applicazioni ad alta scalabilità, cioè che possano essere distribuite a runtime su più server in caso di necessità, in modo da poter servire un numero di utenti teoricamente illimitato semplicemente aggiungendo altri server.

Per far sì che ciò sia possibile, occorre che le richieste HTTP provenienti dai vari client, siano spartite tra i vari server, e che ognuna di esse interessi solo il server a cui è stata diretta. In molti framework Web tradizionali come JSF¹ o Spring ed in linguaggi come il PHP, questa regola non viene rispettata, poiché essi devono offrire allo sviluppatore la possibilità di memorizzare nelle variabili di sessione dei dati arbitrariamente grandi, che devono quindi essere memorizzati lato server.

Per memorizzare le variabili di sessione lato server generalmente viene utilizzata questa tecnica: la prima volta che un utente visita l'applicazione Web, il server genera un ID casuale che identifica l'utente, memorizza questo ID nel browser dell'utente tramite un Cookie HTTP, e da quel momento in poi memorizza tutte le variabili di sessione di quell'utente su un database sul server, associandole all'ID di sessione dell'utente. Questa tecnica ha il vantaggio di poter memorizzare grandi quantità di dati nelle variabili di sessione, e funziona piuttosto bene nel caso l'applicazione Web sia eseguita su un solo server. Ma cosa succede se invece dobbiamo eseguire l'applicazione su un numero elevato di server? In questo caso, uno stesso utente potrebbe effettuare più richieste che vengono indirizzate ognuna su server diversi. Siccome l'applicazione lato server deve sempre poter leggere e scrivere le variabili di sessione per ogni richiesta HTTP, se uno stesso utente esegue più richieste su server diversi, per ogni richiesta le variabili di sessione di quell'utente devono essere necessariamente trasferite dal server nel quale sono memorizzate a quello che sta servendo la richiesta. Questo fa sì che una buona parte delle richieste HTTP per essere gestite richiedano l'intervento di 2 server invece che uno solo, in quanto per ogni richiesta va contattato anche il server nel quale sono memorizzate le variabili di sessione.

Come è facile intuire questo rappresenta un problema per la scalabilità dell'applicazione Web, in quanto può crearsi un collo di bottiglia nel sistema di memorizzazione variabili di sessione: anche aumentando i server sul quale viene eseguita l'applicazione Web, se non viene parallelamente aumentato anche il throughput del database delle variabili di sessione, il sistema complessivo non riesce a gestire un numero più alto di utenti.

Il Play framework risolve questo problema in modo piuttosto radicale: invece di memorizzare le variabili di sessione lato server come fanno la quasi totalità degli altri framework Web, Play memorizza le variabili di sessione esclusivamente lato client, cioè nei browser degli utenti, utilizzando i Cookie HTTP. Questo approccio ha il grosso vantaggio di rendere le richieste HTTP effettuate dagli utenti completamente stateless, come dovrebbero in effetti essere secondo le specifiche del protocollo HTTP: infatti, memorizzando le variabili di

¹https://it.wikipedia.org/wiki/Java_Server_Faces

2 Tecnologie Utilizzate

sessione esclusivamente nei Cookie, ogni sessione può essere gestita interamente da un server Web qualsiasi senza che esso debba anche contattare un altro server per ottenere le variabili di sessione.

Con questo approccio quindi si guadagna in scalabilità, ma ha anche degli svantaggi: siccome i Cookie HTTP hanno una dimensione massima di 4 kilobyte ed essi devono essere inviati ad ogni singola richiesta HTTP, le variabili di sessione in essi contenute devono essere di dimensioni molto piccole. Questo tuttavia non è un problema, in quanto nella maggior parte dei casi questi dati sono molto piccoli, e nei rari casi in cui sia necessario memorizzare dei dati più grandi basta inserire nella variabile di sessione un ID che in un database centrale lato server corrisponde al dato vero e proprio.

Inoltre è necessario risolvere un problema di sicurezza: l'utente non deve essere in grado di alterare i valori nelle variabili di sessione, in quanto per esempio in esse potrebbe essere memorizzato se l'utente ha il permesso o no di accedere ad un'area riservata. Questo problema viene completamente risolto da Play inserendo nei Cookie HTTP anche una firma digitale del contenuto delle variabili di sessione generata da una chiave di cui solo i server sono in possesso. In questo modo se il client altera i dati, i server se ne accorgono e rifiutano la richiesta.

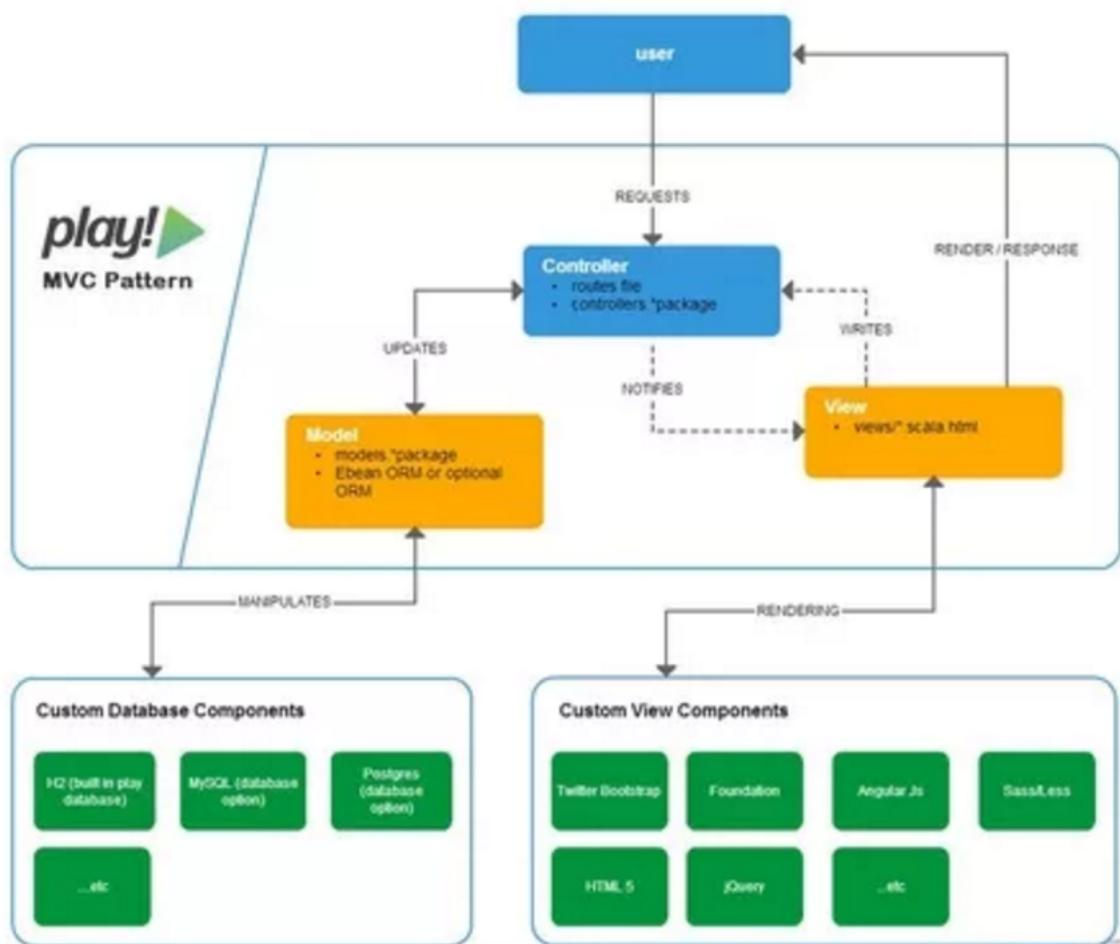


Figura 1: Tecnologie utilizzate - Play Framework MVC pattern

Utilizzo esplicito librerie Akka



2 Tecnologie Utilizzate

Dato che Play è basato su Akka, esso è particolarmente adatto per realizzare applicazioni che devono scambiarsi in modo bidirezionale dei messaggi asincroni. Il protocollo migliore utilizzato in ambito Web per scambiarsi messaggi di questo tipo è quello dei Websocket. Grazie ad esso infatti possiamo dal server web inviare in qualunque momento messaggi al client. In Play è possibile utilizzare i Websocket in modo molto semplice ma al tempo stesso estremamente potente, utilizzando gli Attori del sistema Akka sottostante: all'interno dell'applicazione Play, è possibile creare Attori che all'arrivo di certi messaggi da un qualsiasi altro Attore, inviano dei dati attraverso la connessione Websocket.

Se si vuole per esempio realizzare una applicazione Web che svolga lato server una computazione piuttosto complessa, si può progettare il tutto in questo modo: quando il browser effettua la richiesta al server, all'interno del sistema ad Attori Akka sul quale viene eseguita l'applicazione Play, verrà creato un Attore che eseguirà la computazione, eventualmente delegandola ad altri Attori, grazie allo scambio di messaggi. Quando gli Attori coinvolti avranno terminato il lavoro, il risultato della computazione verrà in qualche modo trasmesso con dei messaggi all'Attore supervisore che aveva dato inizio al lavoro. Esso infine, alla ricezione di questo risultato, potrà inviarlo tramite Websocket al browser dell'utente.

Svantaggi

- **Immaturo:** il framework è relativamente "giovane", anche se ormai presenta un'ampia diffusione, è ancora facile riscontrare qualche bug o problema inaspettato nel suo utilizzo. Inoltre, offre una vasta rosa di plugin di cui usufruire ma anche quest'ultimi a volte risultano problematici;
- **Linguaggio Scala:** come precedentemente illustrato il framework è realizzato tramite linguaggio Scala e rilasciato poi su JVM. La mancanza di conoscenza del linguaggio da parte del team può portare ad una non piena comprensione dei meccanismi operativi del framework;
- **Retro-compatibilità:** attualmente il linguaggio è alla sua seconda versione, una nota negativa risulta essere la non compatibilità tra progetti realizzati con la scorsa versione e il nuovo framework. Ciò deve far tener conto di un possibile problema fra la realizzazione del progetto con questa versione e un possibile sviluppo del framework.

2.4 Infrastruttura Server

2.4.1 Amazon Web Services

Amazon Web Services definito anche AWS è una infrastruttura in cloud computing che permette ai fruitori del servizio di creare piattaforme web on demand offerte da Amazon.com. È un servizio che opera da 12 regioni geografiche nel mondo, perciò con politiche di affidabilità e ridondanza del dato molto alte. Uno dei maggiori (e anche più conosciuti) servizi è Amazon Elastic Compute Cloud, conosciuto anche come EC2, a cui segue poi Amazon Simple Storage Service, meglio conosciuto come S3. Il servizio AWS di Amazon fornisce una vastissima capacità computazionale in modo immediato e più economico rispetto a quello che comporterebbe da parte del fruitore del servizio acquistare la stessa architettura server e mantenerla home premise. Nella realizzazione del progetto *CLIPS*, Amazon Web Services ha permesso al team di ottenere un server virtuale remoto con sistema operativo Ubuntu 12.04 LTS, che al suo interno ospiterà Play e MySQL. Questo server è gestito da una console e si possono modificare le risorse come CPU e RAM, pagandole come un servizio on demand. Questo permette al team di poter intervenire in modo molto semplice in caso di esigenze prestazionali, senza dover prendersi l'onere di manutenere un server in casa assorbendo costi di manutenzione/energia elettrica/politiche di firewalling ecc..

Svantaggi: L'unico svantaggio evidente di tale servizio è rappresentato dal costo, ma questo è stato ammortizzato dall'azienda proponente, la quale se ne è sobbarcata l'onere.



2 Tecnologie Utilizzate

2.5 Database

2.5.1 MySQL

MySQL o Oracle MySQL è un Relational database management system (RDBMS) composto da un client a riga di comando e un server. Entrambi i software sono disponibili sia per sistemi Unix e Unix-like che per Windows; le piattaforme principali di riferimento sono Linux e Oracle Solaris.

MySQL è un software libero rilasciato a doppia licenza, compresa la GNU General Public License ed è sviluppato per essere il più possibile conforme agli standard ANSI SQL e ODBC SQL. I sistemi e i linguaggi di programmazione che supportano MySQL sono molto numerosi: ODBC, Java, Mono, .NET, PHP, Python e molti altri. Le piattaforme LAMP e WAMP incorporano MySQL per l'implementazione di server per gestire siti web dinamici, inoltre molti dei Content Management System di successo come WordPress, Joomla e Drupal nascono proprio con il supporto predefinito a MySQL.

Dal 1996 ad oggi, MySQL si è affermato molto velocemente prestando le sue capacità a moltissimi software e siti Internet. I motivi di tale successo risiedono nella sua capacità di mantenere fede agli impegni presi sin dall'inizio:

- alta efficienza nonostante le moli di dati affidate;
- integrazione di tutte le funzionalità che offrono i migliori DBMS: indici, trigger e stored procedure ne sono un esempio, e saranno approfonditi nel corso della guida;
- altissima capacità di integrazione con i principali linguaggi di programmazione, ambienti di sviluppo e suite di programmi da ufficio.

In MySQL una tabella può essere di diversi tipi (o Storage Engine). Ogni tipo di tabella presenta proprietà e caratteristiche differenti (transazionale o meno, migliori prestazioni, diverse strategie di locking, funzioni particolari, ecc.). Esiste poi un'API G che si può utilizzare per creare in modo relativamente facile un nuovo tipo di tabella, che poi si può installare senza dover ricompilare o riavviare il server.

Storage Engine ufficiali

I tipi di tabella predefiniti sono:

- MyISAM;
- InnoDB (transazionale, sviluppata da InnoBase Oy, società ora comprata da Oracle);
- Memory (una volta si chiamava Heap);
- Merge;
- NDB, o ClusterDB (introdotta nella 5.0);
- CSV (introdotta nella 5.1);
- Federated (introdotta nella 5.0);
- Archive (introdotta nella 5.0);
- BLACKHOLE (introdotta nella 5.0);
- Falcon (non è mai stato terminato e il progetto è abbandonato).

Svantaggi

Questo RDBMS soffre di alcune problematiche minori:

- **Stabilità:** sotto determinate condizioni (utilizzo di certe funzioni complesse), può presentare problemi di stabilità. Sono situazioni poco probabili per il progetto in questione;
- **Scalabilità:** il DBMS riesce a gestire piuttosto bene ampie quantità di dati, quando si tratta però di gestire molte richieste di operazioni simultanee il sistema risulta poco efficiente. Nel caso di sviluppi futuri con grandi numeri da gestire è probabile dover considerare un cambio di tecnologia;



2 Tecnologie Utilizzate

- **Poco estendibile:** per quanto facile da installare, risulta problematica l'aggiunta di Add-On che possono servire nel progetto.

2.5.2 JDBC

JDBC (Java DataBase Connectivity), è un connettore (driver) per database che consente l'accesso e la gestione della persistenza dei dati sulle basi di dati da qualsiasi programma scritto con il linguaggio di programmazione Java, indipendentemente dal tipo di DBMS utilizzato. È costituito da un'APIG object oriented orientata ai database relazionali, raggruppata nel package java.sql, che serve ai client per connettersi a un database fornendo i metodi per interrogare e modificare i dati.

La piattaforma Java 2 Standard Edition contiene le API JDBC, insieme all'implementazione di un bridge JDBC-ODBC, che permette di connettersi a database relazionali che supportino ODBC, che è in codice nativo e non in Java. Tipicamente ciascun DB ha il suo specifico driver JDBC per interfacciarsi con l'applicazione. Spesso i framework di persistenza in ambito Java (es. Hibernate) nella loro implementazione a più alto livello si interfacciano a più basso livello proprio con uno strato software JDBC.

L'architettura di JDBC, così come quella di ODBC, prevede l'utilizzo di un "driver manager", che espone alle applicazioni un insieme di interfacce standard e si occupa di caricare a "run-time" i driver opportuni per "pilotare" gli specifici DBMS. Le applicazioni Java utilizzano le "JDBC API" per parlare con il JDBC driver manager, mentre il driver manager usa le JDBC driver APIG per parlare con i singoli driver che pilotano i DBMS specifici. Esiste un driver particolare, il "JDBC-ODBC Bridge", che consente di interfacciarsi con qualsiasi driver ODBC in ambiente Windows.

Svantaggi: tale tecnologia risulta molto verbosa, richiede molto codice per effettuare operazioni relativamente semplici.

2.6 Client

2.6.1 Android SDK

È un pacchetto, realizzato in linguaggio Java, che offre un ricco assortimento di strumenti per la costruzione di applicazioni per dispositivi mobile facenti girare l'omonimo sistema operativo.

L'approccio di questo SDK si discosta dal classico stile di programmazione Java, non necessita di un punto di accesso iniziale (rappresentato dal metodo *main* in Java), possiede una propria Virtual Machine e più in generale è maggiormente guidato dagli eventi rispetto al linguaggio di programmazione originale.

È stato preferito l'ecosistema AndroidG per la maggior diffusione che attualmente può vantare rispetto alla controparte Apple/iOS.

Svantaggi:

- **Vasta gamma di versioni e dispositivi:** scegliendo di programmare su questa piattaforma bisogna tenere in considerazione l'ampio e estremamente diversificato insieme di dispositivo che ne fa uso. Per questa ragione è possibile riscontrare dei problemi nell'applicazione su dispositivi di marche differenti facenti girare la medesima versione di AndroidG. Oppure, si potrebbero rilevare problemi di compatibilità fra le diverse versioni di AndroidG;
- **Sistema unico:** anche se, come detto, il sistema scelto copre gran parte dei dispositivi, comunque la scelta di non sviluppare per gli altri sistemi operativi mobile (iOS e Windows Phone) preclude la possibilità di arrivare ad una fetta significativa di pubblico.



2 Tecnologie Utilizzate

2.6.2 Picasso

Picasso è una libreria per la gestione di immagini per Android_G. È creata e mantenuta da Square e provvede al caricamento e al processo delle immagini. Semplifica il processo di visualizzazione delle immagini caricate da posizioni remote. In molti casi sono richieste solo poche righe di codice per implementare questa libreria. Picasso eccelle nella visualizzazione di immagini caricate da remoto. La libreria gestisce ogni stadio del processo: dalla richiesta HTTP iniziale al caching dell'immagine. Scrivere codice da sé per portare a termine queste operazioni, può risultare abbastanza lungo. Dal momento che Picasso è relativamente facile da utilizzare, e dal momento che stiamo creando un'app che carica immagini di frequente, Picasso può davvero semplificare il questo processo.

2.6.3 AltBeacon

AltBeacon è una specifica di protocollo per il broadcast di Beacon_G open ideata da Radius Networks3 nel 2014, per sopperire alla mancanza di uno standard aperto e alla disponibilità di beacon per la piattaforma Android_G. Inizialmente era stata sviluppata una libreria per interagire con gli iBeacon, ma Apple non ne permetteva l'utilizzo all'interno di sistemi Android_G ed è quindi stata ritirata. Successivamente è quindi stata ideata una piattaforma compatibile con quella di Apple, ma aperta e disponibile per qualsiasi sistema operativo, così da poter garantire un supporto anche se in maniera non ufficiale. Radius Networks, oltre ad aver ideato un protocollo, vende diversi tipi dei Beacon_G di cui viene garantita la compatibilità con i tre diversi tipi di protocollo. In questi è possibile utilizzare contemporaneamente iBeacon e AltBeacon, alternandoli.

Svantaggi: le classi che implementano tale protocollo, e disponibili liberamente, mancano di documentazione di supporto adeguato. Tal situazione può rappresentare un problema per chi dovesse approcciarsi per la prima volta a tali classi, gli verrà richiesto di capire il funzionamento dei vari componenti esclusivamente dal codice prodotto.



3 Descrizione Architettura

3 Descrizione Architettura

3.1 Metodo e formalismo di specifica

Ai fini di illustrare l'architettura dell'applicazione si procederà con un approccio top down_G, descrivendo inizialmente l'architettura a livello generale, successivamente scendendo nel particolare. Si procederà quindi nella descrizione dei package_G e dei componenti, per poi descrivere in dettaglio le singole, classi specificando per ognuna, l'obiettivo, la funzione e le relazione in ingresso e in uscita.

Per descrivere l'architettura mediante diagrammi di package, di classe e di attività si farà uso dei formalismi grafici messi a disposizione dal linguaggio di modellazione UML_G2.x.

Come successivamente verrà illustrato, l'applicazione si può considerare come suddivisibile in servizi, indipendenti fra di loro, per questa ragione nella sezione di descrizione generale dell'architettura si procederà illustrando la singola struttura di un servizio; è da intendersi che la stessa verrà ripetuta per i restanti in maniera pressoché identica.

3.2 Architettura Generale

Il software è stato progettato seguendo uno schema nella quale sia presente un Server centrale, con compito di gestione della logica dell'applicazione, e una serie di Client, i quali avranno principalmente il compito di gestire la presentazione dei servizi all'utente finale. La descrizione sarà quindi suddivisa in due parti distinte: una componente **Frontend_G** e una **Backend_G**.

Come illustrato precedentemente l'applicazione dovrà essere in grado di gestire più servizi distinti, che possono essere considerati indipendenti fra di loro. In particolare si individua una divisione netta tra il servizio di autenticazione e gestione dell'utente, quello di controllo delle chat e quello necessario al coordinamento del gioco proposto.

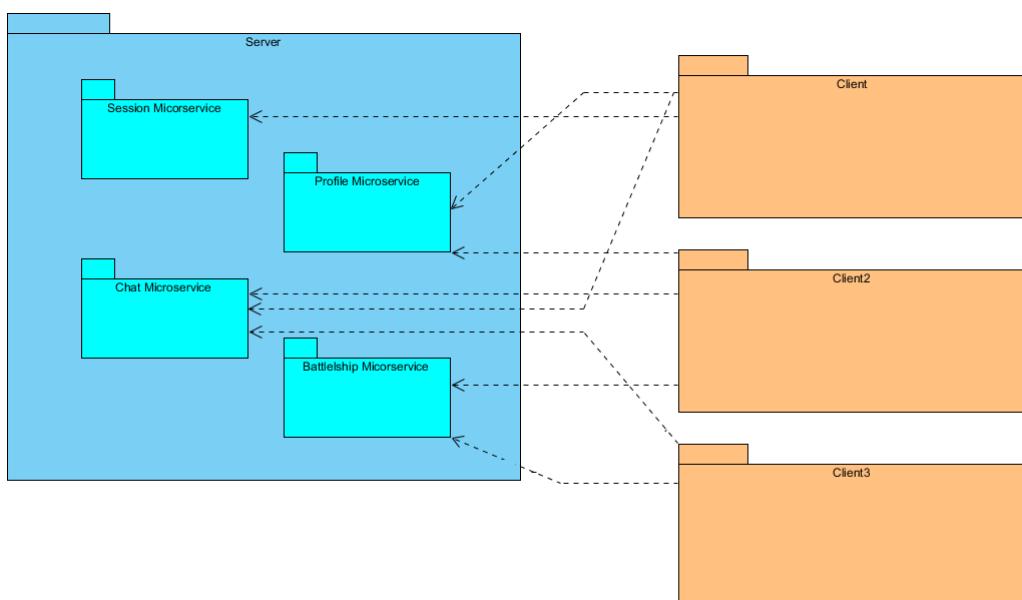


Figura 2: Descrizione architettura - Architettura generale



3 Descrizione Architettura

3.2.1 Architettura Frontend_G

Tenendo presente che è stato scelto di sviluppare l'applicazione per dispositivi Android_G, lo stile architettonico selezionato per definire questa parte è il Model View Presenter, ciò è dovuto principalmente al fatto che la stessa Google consiglia l'utilizzo di tale stile per produrre questo tipo di applicazioni.

La struttura così definita, esporrà un serie di componenti dedite alla presentazione delle informazioni all'utente (la parte di View), le quali avranno anche il compito di catturare gli input che saranno successivamente gestiti dal Presenter. Quest'ultimo dovrà mettersi in comunicazione con il Server, grazie ai servizi messi a disposizione nel Model, per permettere di processare nel modo corretto gli eventi ed infine avrà il compito di aggiornare la parte di View secondo le risposte ottenute.

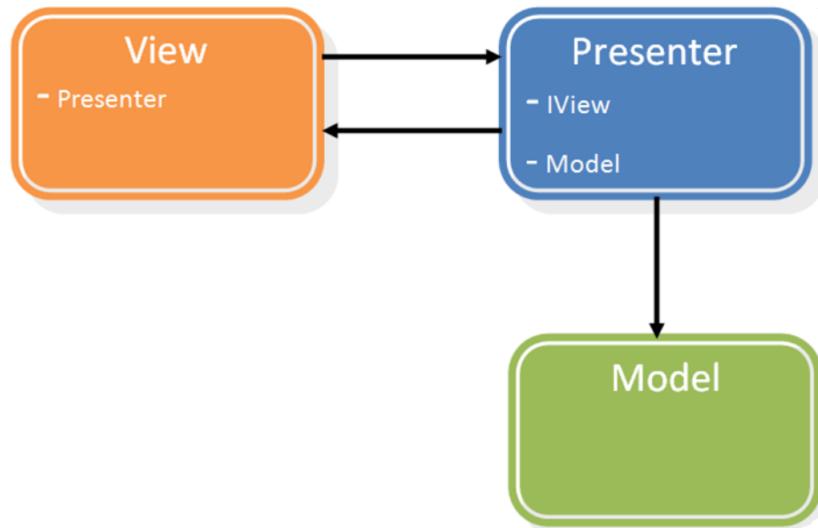


Figura 3: Descrizione architettura - Modem View Presenter

3.2.2 Architettura Backend_G

Dopo una iniziale valutazione di quale stile architettonico utilizzare per progettare l'applicazione, è stato deciso di adoperare uno stile a Microservices_G che permetta una maggiore flessibilità nello sviluppo dell'applicazione stessa.

Come sopra accennato sono stati individuate delle componenti in grado di esistere indipendentemente dall'esistenza di altre parti dell'applicazione, per questa ragione è stato deciso l'utilizzo di questo stile. L'architettura quindi conterrà i servizi di: Chat, Autenticazione, Gioco di Battaglia Navale, Gestione Squadra e Visualizzazione della Classifica legata al gioco. Ogni servizio sarà strutturato secondo il più classico stile architettonico Multi-Tier_G, in quanto sono facilmente individuabili delle suddivisioni logiche all'interno dello stesso. Ognuno di essi sarà suddiviso in un sezione di Persistence, Services, Business e Communication. Ogni strato avrà bisogno del precedente e dovrà fornire un servizio utile al successivo, in questo modo si otterrà un catena virtuosa in grado di adempiere allo scopo per cui è stata progettata; è stato ritenuto necessario inoltre, aggiungere un sezione contenente i tipi comuni per ogni singolo servizio, in quanto saranno necessari per più di uno degli strati del microservizio.

Le sezioni o strati sono state progettate tenendo in mente la seguente separazione logica:

- **Persistence:** questo strato ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato, dovrà fornire al livello successivo una serie di componenti "interpretate" per un linguaggio ad oggetti;



3 Descrizione Architettura

- **Services:** il compito di questa sezione è invece quello di raggruppare e suddividere in maniera logica le componenti grezze fornite dallo strato sottostante e renderle quindi utili la parte di Business;
- **Business:** il vero cuore pulsante del servizio, questo strato dovrà gestire la logica di funzionamento del servizio e rispondere agli input correttamente alle richieste che arrivano dall'esterno;
- **Communication:** l'ultima sezione ha invece il compito di comunicare e rimanere in ascolto delle richieste che arrivano dal esterno del sistema Server, dovrà lavorare da intermediario tra la parte di Business e l'applicazione Client.

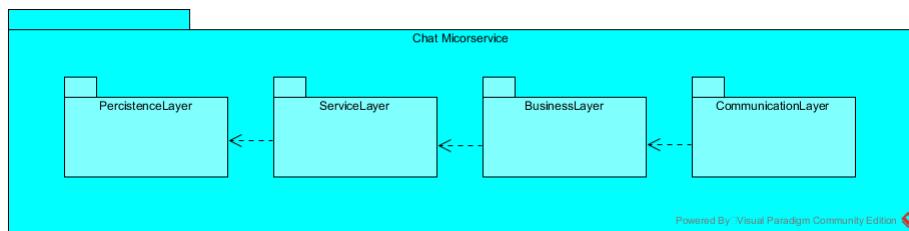


Figura 4: Descrizione architettura - Architettura Backend



4 Architettura Database

I microservizi, presenti nella parte server del sistema, si appoggiano su un database di tipo relazionale.

Lo stile architettonico a microservizi permette di gestire un proprio database specifico per singolo servizio oppure uno comune fra tutti. Nel nostro caso tutti i microservizi condivideranno un singolo database, ciò è dovuto principalmente alla necessità di alcuni di loro ad accedere a dati comuni.

Il RDBMS_G scelto è MySQL_G, la scelta è dovuta principalmente alla buona conoscenza pregressa del team rispetto ad altre soluzioni simili, inoltre il sistema presenta un buon livello di performance rispetto a soluzioni concorrenti.

4.1 Struttura Database

4.1.1 Sezione Base

L'applicazione basa tutto il sua struttura su un concetto fondamentale: ad ogni tavolo di un dato locale è presente un Beacon_G. Proprio in virtù di questo principio il database dovrà contenere tre relazioni basilari ma fondamentali: Locals, Tables e Beacons_G.

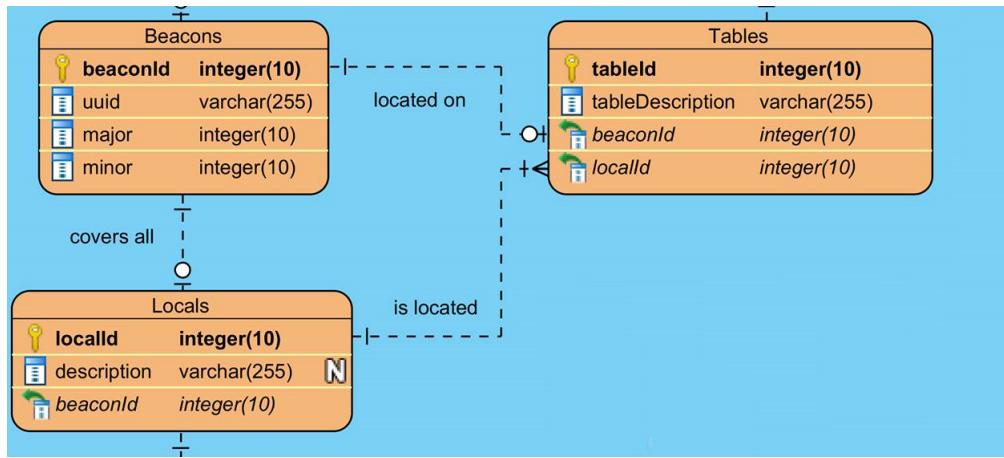


Figura 5: Sezione schema E-R, relazioni base.

- **Locals:** Questa relazione contiene le informazioni essenziali per rappresentare un locale, in questo modo è possibile distinguere Beacons_G appartenente a locali differenti;
- **Tables:** Tramite questa relazione si crea un entità in grado di rappresentare i tavoli a cui gli utenti, che stanno interagendo con l'applicazione, sono seduti. Permette inoltre, di dare consistenza alla presenza di squadre collegate ad essi;
- **Beacons:** La relazione fondamentale, essa infatti contiene le informazioni base dei Beacons_G, ovvero i tre codici. Combinando quest'ultima, con le due precedenti relazioni, si crea la struttura in grado di stabilire la posizione dell'utente all'interno del locale e permettergli di usufruire delle funzionalità in base ad essa.



4 Architettura Database

4.1.2 Sezione Profilo

Una volta determinata la struttura fondamentale in grado di sostenere l'applicazione, è importante stabilire una relazione in grado di salvare il profilo di un nuovo utente che sta utilizzando l'applicazione all'interno del locale. Questa relazione è chiamata Profiles.

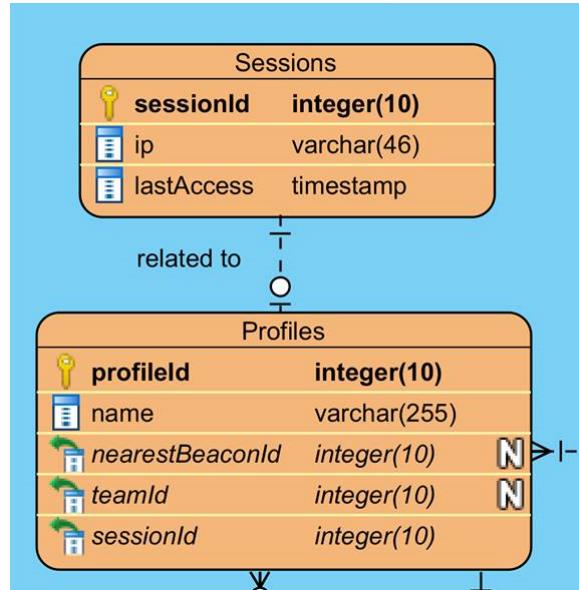


Figura 6: Sezione schema E-R, relazioni sul profilo utente.

- **Profiles:** La relazione mantiene lo username univoco, assegnato al momento dell'autenticazione, ad un utente. Inoltre salva informazioni su dove è seduto e quale BeaconG ad esso è collegato;
- **Session:** Il profilo di un utente è stato progettato come temporaneo, legato alla permanenza della persona all'interno del locale. Per questa ragione e per eventuali disconnessioni accidentali, è stato progettata una relazione in grado di mantenere vivo il profilo utente creando una sottospecie di sessione oraria che ne preservi lo stato. Dopo quest'intervallo il profilo sarà perso e si dovrà necessariamente crearne uno nuovo.



4 Architettura Database

4.1.3 Sezione Gioco

Questa sezione del database salva e permette la gestione del gioco offerto agli utenti dell'applicazione ovvero, la Battaglia Navale. Permetterà di mantenere lo stato delle squadre, dei leader, la disposizione delle navi, dei campi di battaglia e dello svolgimento delle partite.

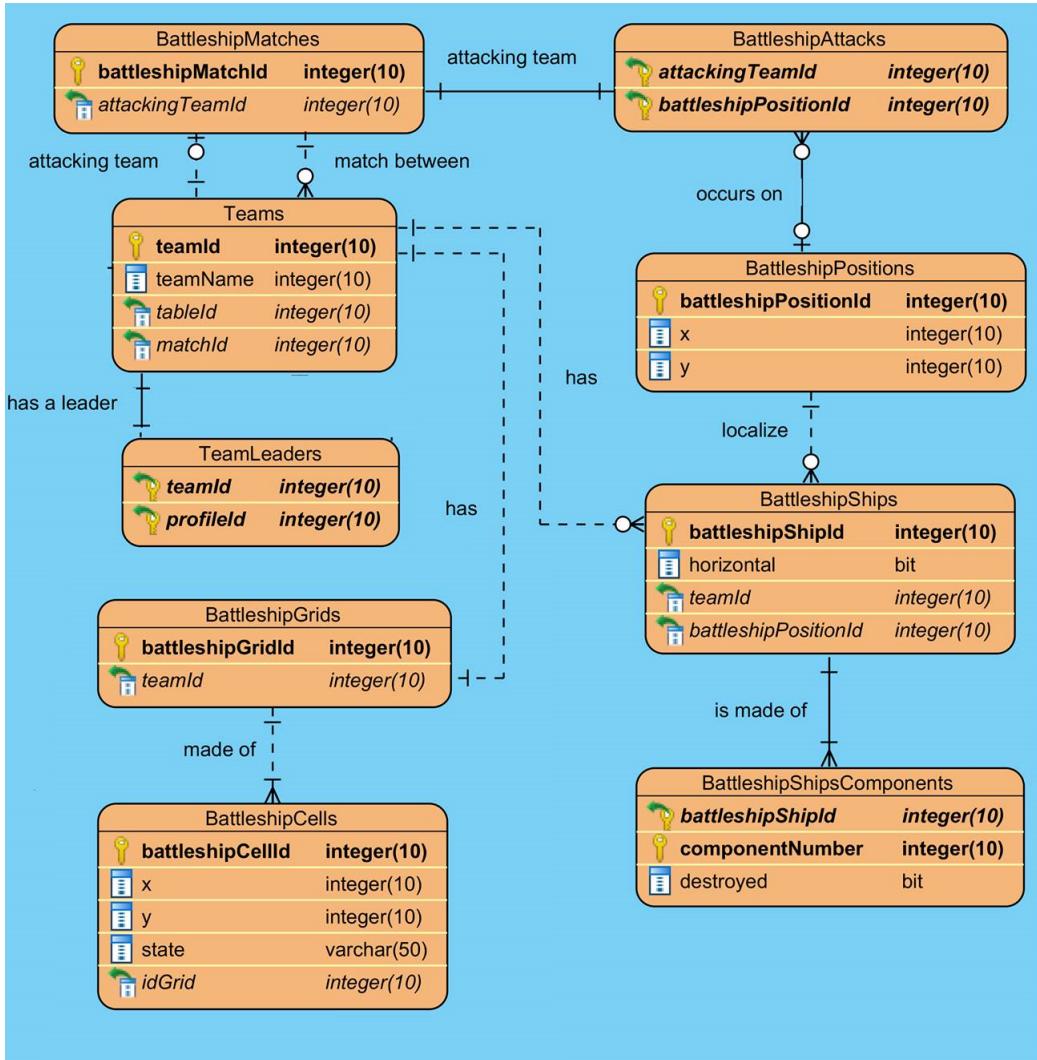


Figura 7: Sezione schema E-R, relazioni sul gioco Battaglia Navale.

- **Teams:** Questa relazione rappresenta le squadre pronte a sfidarsi, ne viene salvato il nome e i membri, ognuna è legata al proprio tavolo di appartenenza;
- **TeamLeaders:** La seguente relazione identifica semplicemente il capitano della squadra, ovviamente esso sarà un utente del locale e membro di quella stessa squadra;
- **BattleshipGrids:** Qui vengono mantenuti i dati riguardanti i campi di battaglia delle varie partite che vengono effettuate, si tiene anche un riferimento alla squadra presente al suo interno;
- **BattleshipCells:** Molto facilmente rappresentano le varie celle facenti parte dei vari campi di battaglia;
- **BattleshipMatches:** La relazione mantiene riferimento del match in corso di svolgimento, tenendo conto della squadra attaccante;
- **BattleshipAttacks:** Questa relazione serve a rappresentare i vari colpi che vengono sparati, ad ogni turno, dai componenti delle squadre in competizione fra loro;



4 Architettura Database

- **BattleshipPositions:** La seguente relazione mantiene informazione sul collocamento delle varie navi posizionate nella fase iniziale di ogni sfida;
- **BattleshipShips:** un delle relazioni fondamentali di questa sezione, rappresenta le varie navi a disposizione delle squadre sfidanti, viene tenuto conto anche dell'orientamento con la quale è stata collocata sul terreno di gioco;
- **BattleshipComponents:** Una relazione di supporto per stabilire se la nave collegata sia stata affondata dai colpi sparati o meno.

4.1.4 Sezione Classifica

L'ultima sezione riguarda ciò che viene mantenuto maggiormente, insieme a quella base, ovvero la classifica dei migliori punteggi ottenuti. Essa è consultabile in ogni momento e conserva la coppia punteggio e nome della squadra che ha conseguito il risultato.

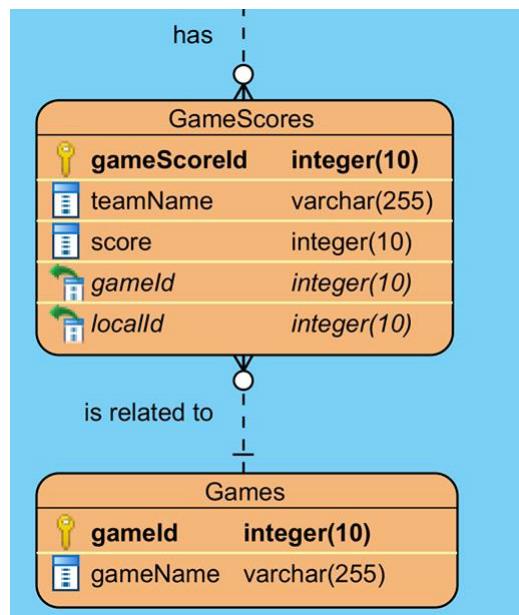


Figura 8: Sezione schema E-R, relazioni sulle classifiche.

- **GamesScore:** La relazione indicata, rappresenta i vari punteggi ottenuti con il loro valore e il nome della squadra che lo ha ottenuto, inoltre viene ovviamente mantenuto il riferimento a quale gioco il punteggio è stato ottenuto;
- **Games:** Molto semplicemente, la relazione mantiene in memoria i vari giochi disponibili sul momento permettendo di distinguere i punteggi ottenuti in base ad essi.



5 Componenti e classi

5 Componenti e classi

5.1 Client

5.1.1 Informazioni generali

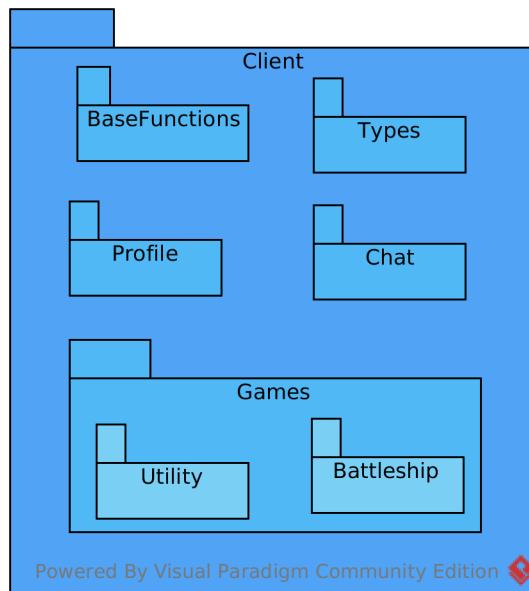


Figura 9: Client

- **Descrizione**

Package che contiene il front end del progetto.

- **Package contenuti:**

- **BaseFunctions**

Package che contiene le componenti che hanno il compito di gestire le iterazioni con l'ambiente di esecuzione dell'applicazione ed esporre un menù di navigazione tra le componenti dell'applicazione.

- **Games**

Package che contiene le componenti relative ai giochi.

- **Profile**

Package che contiene le componenti che realizzano la gestione del profilo.

- **Types**

Package che contiene i tipi comuni usati dalle componenti dell'applicazione.



5 Componenti e classi

5.2 Client::BaseFunctions

5.2.1 Informazioni generali

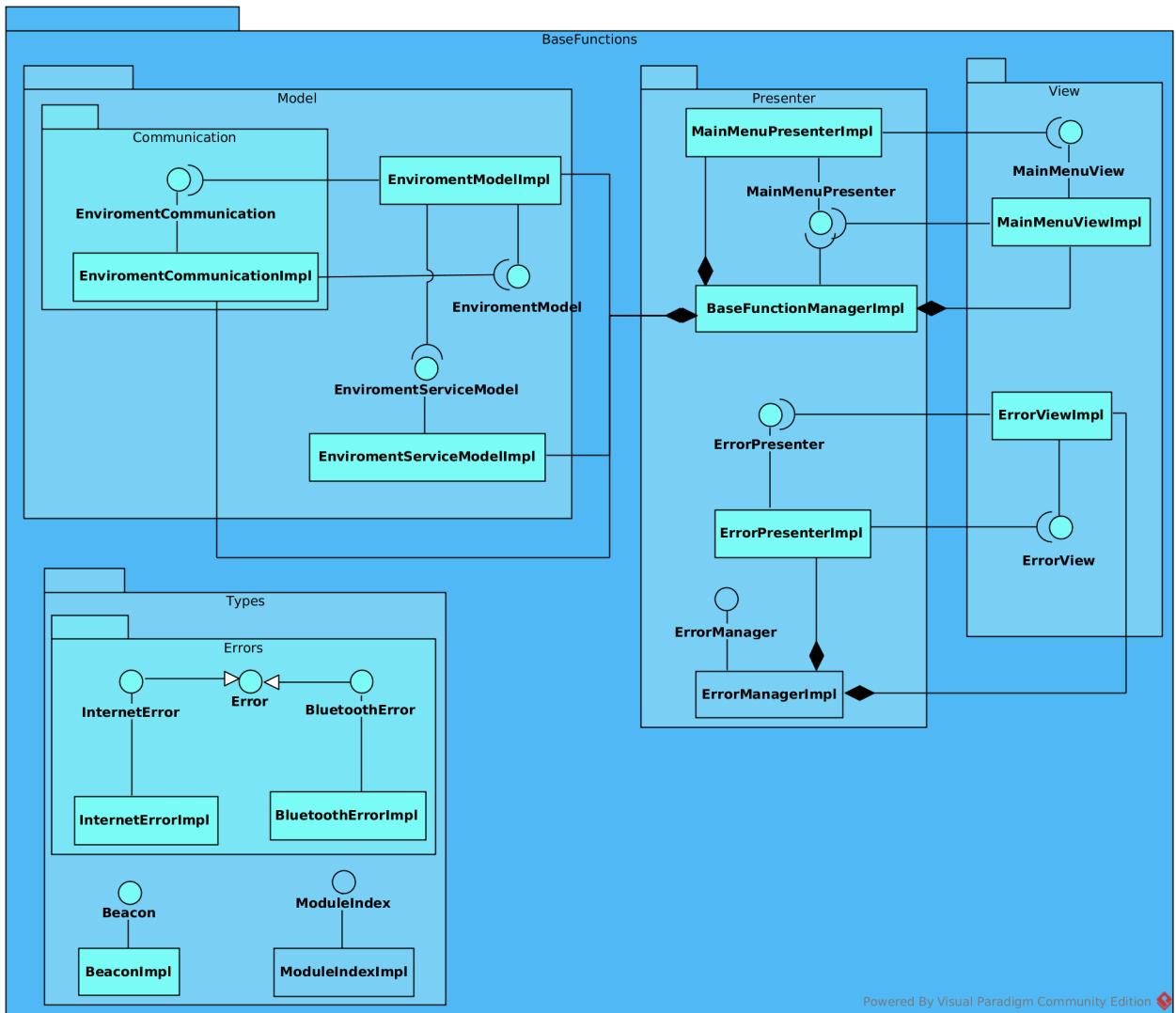


Figura 10: Client::BaseFunctions

- **Descrizione**

Package che contiene le componenti che hanno il compito di gestire le iterazioni con l'ambiente di esecuzione dell'applicazione ed esporre un menù di navigazione tra le componenti dell'applicazione.

- **Padre:** Client

- **Package contenuti:**

- **Model**

Package che contiene le componenti che hanno il compito di fornire il modello dei dati delle funzioni base.

- **Presenter**

Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.

- **Types**

Package contenente i tipi.



5 Componenti e classi

- **View**

Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative alle funzioni base.

5.3 Client::BaseFunctions::Model

5.3.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di fornire il modello dei dati delle funzioni base.

- **Padre:** BaseFunctions

- **Interazioni con altri componenti:**

- **Presenter**

Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.

- **Package contenuti:**

- **Communication**

Package che contiene le componenti che hanno il compito di mettere in comunicazione client e server per le funzionalità riguardanti le funzionalità di base.

5.3.2 Classi

5.3.2.1 Client::BaseFunctions::Model::EnviromentModel

- **Descrizione**

Gestisce il modello dei dati dell'ambiente di esecuzione; classe statica;

- **Utilizzo**

Viene utilizzata quando bisogna controllare lo stato del dispositivo;

- **Sottoclassi:**

- **EnviromentModelImpl**

- **Relazioni con altre classi:**

- **IN EnviromentCommunication**

Gestisce la comunicazione tra client e server per le funzionalità di base;

- **IN EnviromentPresenter**

Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi con l'ambiente di esecuzione;

- **OUT EnviromentCommunication**

Gestisce la comunicazione tra client e server per le funzionalità di base;

- **OUT EnviromentServiceModel**

Gestisce le interrogazioni legate all'ambiente di esecuzione; classe statica;

- **OUT Beacon**

Gestisce le informazioni associate ad un beacon;

- **OUT Error**

Gestisce le informazioni riguardanti un errore generico;

- **OUT Profile**

Gestisce le informazioni che rappresentano un profilo;



5 Componenti e classi

5.3.2.2 Client::BaseFunctions::Model::EnviromentModelImpl

- **Descrizione**

La classe rappresenta le operazioni di controllo sullo stato del dispositivo;

- **Utilizzo**

Viene utilizzata quando bisogna controllare lo stato del dispositivo;

- **Classi ereditate:**

- EnviromentModel

5.3.2.3 Client::BaseFunctions::Model::EnviromentServiceModel

- **Descrizione**

Gestisce le interrogazioni legate all'ambiente di esecuzione; classe statica;

- **Utilizzo**

Viene utilizzata quando bisogna controllare lo stato del dispositivo;

- **Relazioni con altre classi:**

- *IN* EnviromentModel

Gestisce il modello dei dati dell'ambiente di esecuzione; classe statica;

- *OUT* Beacon

Gestisce le informazioni associate ad un beacon;

- *OUT* BluetoothError

Gestisce le informazioni riguardanti un errore della connessione Bluetooth;

- *OUT* InternetError

Gestisce le informazioni riguardanti la connessione Internet;

5.3.2.4 Client::BaseFunctions::Model::EnviromentServiceModelImpl

- **Descrizione**

Gestisce le interrogazioni legate all'ambiente di esecuzione; classe statica;

- **Utilizzo**

Viene utilizzata quando bisogna controllare lo stato del dispositivo;

5.4 Client::BaseFunctions::Model::Communication

5.4.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di mettere in comunicazione client e server per le funzionalità riguardanti le funzionalità di base.

- **Padre: Model**

5.4.2 Classi

5.4.2.1 Client::BaseFunctions::Model::Communication ::Communication

- **Descrizione**

Interfaccia che gestisce le funzionalità di comunicazione con il server;

- **Utilizzo**

Utilizzata dai package communication per avere le funzionalità di comunicazione di base;



5 Componenti e classi

5.4.2.2 Client::BaseFunctions::Model::Communication ::CommunicationImpl

- **Descrizione**

Classe che implementa l’interfaccia che gestisce le funzionalità di comunicazione con il server;

- **Utilizzo**

Utilizzata dai package communication per avere le funzionalità di comunicazione di base;

5.4.2.3 Client::BaseFunctions::Model::Communication ::EnviromentCommunication

- **Descrizione**

Gestisce la comunicazione tra client e server per le funzionalità di base;

- **Utilizzo**

Viene utilizzata per comunicare con il server;

- **Sottoclassi:**

- EnviromentCommunicationImpl

- **Relazioni con altre classi:**

- *IN* EnviromentModel

Gestisce il modello dei dati dell’ambiente di esecuzione; classe statica;

- *OUT* EnviromentModel

Gestisce il modello dei dati dell’ambiente di esecuzione; classe statica;

- *OUT* Beacon

Gestisce le informazioni associate ad un beacon;

- *OUT* Profile

Gestisce le informazioni che rappresentano un profilo;

- *OUT* SessionCommunication

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate alle sessioni, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

5.4.2.4 Client::BaseFunctions::Model::Communication ::EnviromentCommunicationImpl

- **Descrizione**

Gestisce la comunicazione tra client e server per le funzionalità di base;

- **Utilizzo**

Viene utilizzata per comunicare con il server;

- **Classi ereditate:**

- EnviromentCommunication

5.5 Client::BaseFunctions::Presenter

5.5.1 Informazioni generali

- **Descrizione**

Contiene tutte le componenti che gestiscono l’application logic_G e permettono la comunicazione tra model e view.

- **Padre: BaseFunctions**



5 Componenti e classi

- **Interazioni con altri componenti:**

- **Model**

Package che contiene le componenti che hanno il compito di fornire il modello dei dati delle funzioni base.

- **View**

Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative alle funzioni base.

5.5.2 Classi

5.5.2.1 Client::BaseFunctions::Presenter::BaseFunctionsManager

- **Descrizione**

Gestisce quale presenter deve essere in esecuzione; classe statica;

- **Utilizzo**

Viene utilizzata per gestire il flusso dell'applicazione;

- **Sottoclassi:**

- **BaseFunctionsManagerImpl**

- **Relazioni con altre classi:**

- **OUT EnviromentPresenter**

Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi com l'ambiente di esecuzione;

- **OUT MainMenuPresenter**

Gestisce View::MainMenu implementando la application logic_G e la presentation logic_G per la selezione e l'avvio di una delle funzionalità offerte (Gestione profilo, Chat, Bacheca, Battaglia Navale);

- **OUT Profile**

Gestisce le informazioni che rappresentano un profilo;

- **OUT BattleshipManager**

Interfaccia che gestisce quale presenter di Battleship deve entrare in esecuzione;

- **OUT UtilityManager**

Interfaccia che gestisce quale presenter del gestore dei giochi deve essere in esecuzione;

- **OUT ProfileManager**

Gestisce quale presenter del gestore dei giochi deve essere in esecuzione;

5.5.2.2 Client::BaseFunctions::Presenter::BaseFunctionsManagerImpl

- **Descrizione**

Gestisce quale presenter deve essere in esecuzione; classe statica;

- **Utilizzo**

Viene utilizzata per gestire il flusso dell'applicazione;

- **Classi ereditate:**

- **BaseFunctionsManager**



5 Componenti e classi

5.5.2.3 Client::BaseFunctions::Presenter::EnviromentPresenter

- **Descrizione**

Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi con l'ambiente di esecuzione;

- **Utilizzo**

Viene utilizzata per notificare all'utente eventuali problemi con l'ambiente di esecuzione;

- **Sottoclassi:**

- EnviromentPresenterImpl

- **Relazioni con altre classi:**

- *IN BaseFunctionsManager*

Gestisce quale presenter deve essere in esecuzione; classe statica;

- *IN EnviromentErrorView*

Gestisce la visualizzazione di errori legati al ambiente di esecuzione;

- *OUT EnviromentModel*

Gestisce il modello dei dati dell'ambiente di esecuzione; classe statica;

- *OUT BluetoothError*

Gestisce le informazioni riguardanti un errore della connessione Bluetooth;

- *OUT Error*

Gestisce le informazioni riguardanti un errore generico;

- *OUT InternetError*

Gestisce le informazioni riguardanti la connessione Internet;

- *OUT Profile*

Gestisce le informazioni che rappresentano un profilo;

- *OUT EnviromentErrorView*

Gestisce la visualizzazione di errori legati al ambiente di esecuzione;

5.5.2.4 Client::BaseFunctions::Presenter::EnviromentPresenterImpl

- **Descrizione**

Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi con l'ambiente di esecuzione;

- **Utilizzo**

Viene utilizzata per notificare all'utente eventuali problemi con l'ambiente di esecuzione;

- **Classi ereditate:**

- EnviromentPresenter

5.5.2.5 Client::BaseFunctions::Presenter::ErrorPresenter

- **Descrizione**

La classe permette di visualizzare gli errori;

- **Utilizzo**

Viene utilizzata per notificare all'utente gli errori riscontrati;

- **Sottoclassi:**

- ErrorPresenterImpl



5 Componenti e classi

5.5.2.6 Client::BaseFunctions::Presenter::ErrorPresenterImpl

- **Descrizione**

La classe permette di visualizzare gli errori;

- **Utilizzo**

Viene utilizzata per notificare all'utente gli errori riscontrati;

- **Classi ereditate:**

- ErrorPresenter

5.5.2.7 Client::BaseFunctions::Presenter::MainMenuPresenter

- **Descrizione**

Gestisce View::MainMenu implementando la application logic_G e la presentation logic_G per la selezione e l'avvio di una delle funzionalità offerte (Gestione profilo, Chat, Bacheca, Battaglia Navale);

- **Utilizzo**

Viene utilizzata per visualizzare il menù iniziale dell'applicazione;

- **Sottoclassi:**

- MainMenuPresenterImpl

- **Relazioni con altre classi:**

- *IN* BaseFunctionsManager

Gestisce quale presenter deve essere in esecuzione; classe statica;

- *IN* MainMenuView

Gestisce la GUI riguardante la visualizzazione e la scelta delle funzionalità offerte dal sistema;

- *OUT* MainMenuView

Gestisce la GUI riguardante la visualizzazione e la scelta delle funzionalità offerte dal sistema;

5.5.2.8 Client::BaseFunctions::Presenter::MainMenuPresenterImpl

- **Descrizione**

La classe gestisce la view iniziale dell'applicazione (start menu);

- **Utilizzo**

Viene utilizzata per visualizzare il menù iniziale dell'applicazione;

- **Classi ereditate:**

- MainMenuPresenter

5.6 Client::BaseFunctions::Types

5.6.1 Informazioni generali

- **Descrizione**

Package contenente i tipi.

- **Padre: BaseFunctions**

- **Package contenuti:**

- Errors

Package che contiene le componenti che hanno il compito di memorizzare informazioni su errori.



5 Componenti e classi

5.6.2 Classi

5.6.2.1 Client::BaseFunctions::Types::Beacon

- **Descrizione**

Gestisce le informazioni associate ad un beacon;

- **Utilizzo**

Viene utilizzata per gestire le informazioni relative ad un beacon;

- **Sottoclassi:**

- BeaconImpl

- **Relazioni con altre classi:**

- *IN EnviromentCommunication*

Gestisce la comunicazione tra client e server per le funzionalità di base;

- *IN EnviromentModel*

Gestisce il modello dei dati dell'ambiente di esecuzione; classe statica;

- *IN EnviromentServiceModel*

Gestisce le interrogazioni legate all'ambiente di esecuzione; classe statica;

5.6.2.2 Client::BaseFunctions::Types::BeaconImpl

- **Descrizione**

Gestisce le informazioni associate ad un beacon;

- **Utilizzo**

Viene utilizzata per gestire le informazioni relative ad un beacon;

- **Classi ereditate:**

- Beacon

5.6.2.3 Client::BaseFunctions::Types::ModuleIndex

- **Descrizione**

La classe specifica la funzionalità da aprire;

- **Utilizzo**

Viene utilizzata per specificare la funzionalità;

- **Sottoclassi:**

- ModuleIndexImpl

5.6.2.4 Client::BaseFunctions::Types::ModuleIndexImpl

- **Descrizione**

La classe specifica la funzionalità da aprire;

- **Utilizzo**

Viene utilizzata per specificare la funzionalità;

- **Classi ereditate:**

- ModuleIndex



5 Componenti e classi

5.6.2.5 Client::BaseFunctions::Types::Profile

- **Descrizione**

Gestisce le informazioni che rappresentano un profilo;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti un profilo;

- **Sottoclassi:**

- ProfileImpl

- **Relazioni con altre classi:**

- *IN EnviromentCommunication*

Gestisce la comunicazione tra client e server per le funzionalità di base;

- *IN EnviromentModel*

Gestisce il modello dei dati dell'ambiente di esecuzione; classe statica;

- *IN BaseFunctionsManager*

Gestisce quale presenter deve essere in esecuzione; classe statica;

- *IN EnviromentPresenter*

Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi com l'ambiente di esecuzione;

5.6.2.6 Client::BaseFunctions::Types::ProfileImpl

- **Descrizione**

Gestisce le informazioni che rappresentano un profilo;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti un profilo;

- **Classi ereditate:**

- Profile

5.7 Client::BaseFunctions::Types::Errors

5.7.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di memorizzare informazioni su errori.

- **Padre: Types**

5.7.2 Classi

5.7.2.1 Client::BaseFunctions::Types::Errors::BluetoothError

- **Descrizione**

Gestisce le informazioni riguardanti un errore della connessione Bluetooth;

- **Utilizzo**

Viene utilizzata per gestire le informazioni di un errore della connessione Bluetooth;

- **Classi ereditate:**

- Error

- **Sottoclassi:**



5 Componenti e classi

- `BluetoothErrorImpl`
- **Relazioni con altre classi:**
 - *IN EnviromentServiceModel*
Gestisce le interrogazioni legate all’ambiente di esecuzione; classe statica;
 - *IN EnviromentPresenter*
Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all’utente eventuali problemi com l’ambiente di esecuzione;

5.7.2.2 Client::BaseFunctions::Types::Errors::BluetoothErrorImpl

- **Descrizione**
Gestisce le informazioni riguardanti un errore della connessione Bluetooth;
- **Utilizzo**
Viene utilizzata per gestire le informazioni di un errore della connessione Bluetooth;
- **Classi ereditate:**
 - `BluetoothError`

5.7.2.3 Client::BaseFunctions::Types::Errors::EmptyTeamNameError

- **Descrizione**
Gestisce gli errori causati dall’inserimento del nome del team;
- **Utilizzo**
Quando viene inserito un nome di un team vuoto viene visualizzato l’errore;
- **Classi ereditate:**
 - `Error`
- **Sottoclassi:**
 - `EmptyTeamNameErrorImpl`

5.7.2.4 Client::BaseFunctions::Types::Errors::EmptyTeamNameErrorImpl

- **Descrizione**
Gestisce gli errori causati dall’inserimento del nome del team;
- **Utilizzo**
Quando viene inserito un nome di un team vuoto viene visualizzato l’errore;
- **Classi ereditate:**
 - `EmptyTeamNameError`

5.7.2.5 Client::BaseFunctions::Types::Errors::Error

- **Descrizione**
Gestisce le informazioni riguardanti un errore generico;
- **Utilizzo**
Viene utilizzata per gestire le informazioni riguardanti un errore generico;
- **Sottoclassi:**
 - `BluetoothError`
 - `EmptyTeamNameError`
 - `ErrorImpl`



5 Componenti e classi

- InternetError
- ServerError

- **Relazioni con altre classi:**

- *IN EnviromentModel*
Gestisce il modello dei dati dell'ambiente di esecuzione; classe statica;
- *IN EnviromentPresenter*
Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi com l'ambiente di esecuzione;

5.7.2.6 Client::BaseFunctions::Types::Errors::ErrorImpl

- **Descrizione**
Gestisce le informazioni riguardanti un errore generico;
- **Utilizzo**
Viene utilizzata per gestire le informazioni riguardanti un errore generico;
- **Classi ereditate:**

- Error

5.7.2.7 Client::BaseFunctions::Types::Errors::InternetError

- **Descrizione**
Gestisce le informazioni riguardanti la connessione Internet;
- **Utilizzo**
Viene utilizzata per gestire le informazioni riguardanti la connessione Internet;
- **Classi ereditate:**
- Error
- **Sottoclassi:**
- InternetErrorImpl
- **Relazioni con altre classi:**
- *IN EnviromentServiceModel*
Gestisce le interrogazioni legate all'ambiente di esecuzione; classe statica;
- *IN EnviromentPresenter*
Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi com l'ambiente di esecuzione;

5.7.2.8 Client::BaseFunctions::Types::Errors::InternetErrorImpl

- **Descrizione**
Gestisce le informazioni riguardanti la connessione Internet;
- **Utilizzo**
Viene utilizzata per gestire le informazioni riguardanti la connessione Internet;
- **Classi ereditate:**
- InternetError



5 Componenti e classi

5.7.2.9 Client::BaseFunctions::Types::Errors::ServerError

- **Descrizione**

Gestisce le informazioni riguardanti un errore della connessione con il server;

- **Utilizzo**

Viene utilizzata per gestire le informazioni di un errore della connessione con il server

- **Classi ereditate:**

- Error

- **Sottoclassi:**

- ServerErrorImpl

5.7.2.10 Client::BaseFunctions::Types::Errors::ServerErrorImpl

- **Descrizione**

Gestisce le informazioni riguardanti un errore della connessione con il server;

- **Utilizzo**

Viene utilizzata per gestire le informazioni di un errore della connessione con il server;

- **Classi ereditate:**

- ServerError

5.8 Client::BaseFunctions::View

5.8.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative alle funzioni base.

- **Padre: BaseFunctions**

- **Interazioni con altri componenti:**

- **Presenter**

Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.

5.8.2 Classi

5.8.2.1 Client::BaseFunctions::View::EnviromentErrorView

- **Descrizione**

Gestisce la visualizzazione di errori legati al ambiente di esecuzione;

- **Utilizzo**

Viene utilizzata per visualizzare a schermo gli errori legati all'ambiente di esecuzione;

- **Sottoclassi:**

- EnviromentErrorViewImpl

- **Relazioni con altre classi:**

- **IN EnviromentPresenter**

Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all'utente eventuali problemi con l'ambiente di esecuzione;



5 Componenti e classi

– *OUT* EnviromentPresenter

Gestisce Model::Enviroment e View::EnviromentError implementando la application logic_G e la presentation logic_G per notificare all’utente eventuali problemi con l’ambiente di esecuzione;

5.8.2.2 Client::BaseFunctions::View::EnviromentErrorViewImpl

- **Descrizione**

Gestisce la visualizzazione di errori legati al ambiente di esecuzione;

- **Utilizzo**

Viene utilizzata per visualizzare a schermo gli errori legati all’ambiente di esecuzione;

- **Classi ereditate:**

 - EnviromentErrorView

5.8.2.3 Client::BaseFunctions::View::MainMenuView

- **Descrizione**

Gestisce la GUI riguardante la visualizzazione e la scelta delle funzionalità offerte dal sistema;

- **Utilizzo**

Viene utilizzata per dare la possibilità all’utente di selezionare la funzionalità desiderata;

- **Sottoclassi:**

 - MainMenuViewImpl

- **Relazioni con altre classi:**

 - *IN* MainMenuPresenter

Gestisce View::MainMenu implementando la application logic_G e la presentation logic_G per la selezione e l’avvio di una delle funzionalità offerte (Gestione profilo, Chat, Bacheca, Battaglia Navale);

 - *OUT* MainMenuPresenter

Gestisce View::MainMenu implementando la application logic_G e la presentation logic_G per la selezione e l’avvio di una delle funzionalità offerte (Gestione profilo, Chat, Bacheca, Battaglia Navale);

5.8.2.4 Client::BaseFunctions::View::MainMenuViewImpl

- **Descrizione**

Gestisce la GUI riguardante la visualizzazione e la scelta delle funzionalità offerte dal sistema;

- **Utilizzo**

Viene utilizzata per dare la possibilità all’utente di selezionare la funzionalità desiderata;

- **Classi ereditate:**

 - MainMenuView



5 Componenti e classi

5.9 Client::Games

5.9.1 Informazioni generali

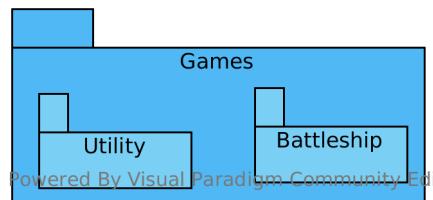


Figura 11: Client::Games

- **Descrizione**

Package che contiene le componenti relative ai giochi.

- **Padre:** Client

- **Package contenuti:**

- **Battleship**

Package che contiene le componenti relative al gioco Battleship.

- **Utility**

Package che contiene le componenti relative alle funzionalità di selezione del gioco e di visualizzazione delle classifiche relative ai giochi.



5 Componenti e classi

5.10 Client::Games::Battleship

5.10.1 Informazioni generali

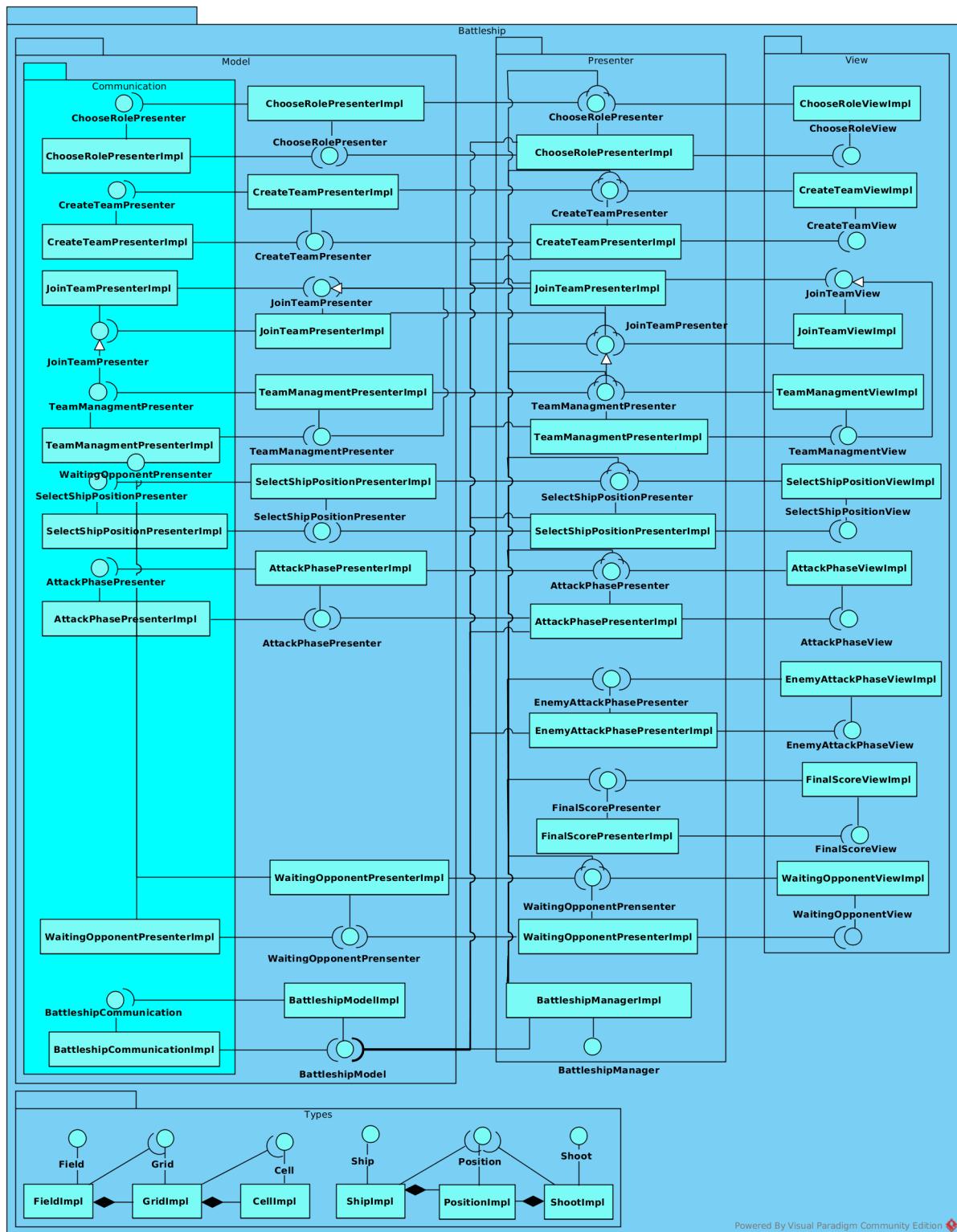


Figura 12: Client::Games::Battleship



5 Componenti e classi

- **Descrizione**

Package che contiene le componenti relative al gioco Battleship.

- **Padre:** Games

- **Package contenuti:**

- **Model**

Package che contiene le componenti che hanno il compito di fornire il modello dei dati.

- **Presenter**

Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.

- **Types**

Package contenente i tipi.

- **View**

Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative al gioco Battleship.

5.11 Client::Games::Battleship::Model

5.11.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di fornire il modello dei dati.

- **Padre:** Battleship

- **Package contenuti:**

- **Communication**

Package che contiene le componenti che hanno il compito di mettere in comunicazione client e server per le funzionalità riguardanti il gioco Battleship.

5.11.2 Classi

5.11.2.1 Client::Games::Battleship::Model::AttackPhaseModel

- **Descrizione**

Interfaccia che gestisce la business logic associata alla fase di attacco;

- **Utilizzo**

Viene utilizzata dal AttackPhasePresenter per gestisce la business logic associata alla fase di attacco;

- **Sottoclassi:**

- **AttackPhaseModelImpl**

5.11.2.2 Client::Games::Battleship::Model::AttackPhaseModelImpl

- **Descrizione**

Classe che implementa l'interfaccia che gestisce la business logic associata alla fase di attacco;

- **Utilizzo**

Viene utilizzata dal AttackPhasePresenterImpl per gestisce la business logic associata alla fase di attacco;

- **Classi ereditate:**

- **AttackPhaseModel**



5 Componenti e classi

5.11.2.3 Client::Games::Battleship::Model::BattleshipModel

- **Descrizione**

Interfaccia che gestisce la business logic_G legata al gioco di Battleship;

- **Utilizzo**

Viene utilizzata dai presenter per gestire la business logic_G legata al gioco di Battleship;

- **Sottoclassi:**

- BattleshipModelImpl

- **Relazioni con altre classi:**

- *IN BattleshipCommunication*

Interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;

- *IN AttackPhasePresenter*

Interfaccia che gestisce l'application logic_G legata alla fase di attacco;

- *IN BattleshipManager*

Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;

- *IN CreateTeamPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di creare una squadra;

- *IN EnemyAttackPhasePresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il risultato degli attacchi nemici;

- *IN FinalScorePresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il punteggio finale;

- *IN JoinTeamPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

- *IN SelectShipPositionPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di posizionare le sue navi nel campo di battaglia;

- *IN StartGamePresenter*

Interfaccia che gestisce l'application logic_G legata alla fase iniziale dove si può visualizzare la classifica del locale o iniziare a giocare;

- *IN TeamManagementPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'capitano di gestire la propria squadra;

- *OUT BattleshipCommunication*

Interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;

- *OUT AttackPhasePresenter*

Interfaccia che gestisce l'application logic_G legata alla fase di attacco;

- *OUT BattleshipManager*

Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;

- *OUT CreateTeamPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di creare una squadra;



5 Componenti e classi

- **OUT EnemyAttackPhasePresenter**
Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il risultato degli attacchi nemici;
- **OUT FinalScorePresenter**
Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il punteggio finale;
- **OUT JoinTeamPresenter**
Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;
- **OUT SelectShipPositionPresenter**
Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di posizionare le sue navi nel campo di battaglia;
- **OUT StartGamePresenter**
Interfaccia che gestisce l'application logic_G legata alla fase iniziale dove si può visualizzare la classifica del locale o iniziare a giocare;
- **OUT TeamManagementPresenter**
Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'capitano di gestire la propria squadra;
- **OUT Team**
Interfaccia che gestisce le informazioni riguardanti un team;

5.11.2.4 Client::Games::Battleship::Model::BattleshipModelImpl

- **Descrizione**
Classe che implementa l'interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
- **Utilizzo**
Viene utilizzata dai presenter per gestire la bussines logic_G legata al gioco di Battleship;
- **Classi ereditate:**
 - BattleshipModel

5.11.2.5 Client::Games::Battleship::Model::CreateTeamModel

- **Descrizione**
Interfaccia che gestisce la business logic per la creazione della squadra;
- **Utilizzo**
Utilizzata da CreateTeamPresenter per la gestione della business logic per la creazione della squadra;
- **Sottoclassi:**
 - CreateTeamModelImpl

5.11.2.6 Client::Games::Battleship::Model::CreateTeamModelImpl

- **Descrizione**
Implementa l'interfaccia che gestisce la business logic per la creazione della squadra;
- **Utilizzo**
Utilizzata da CreateTeamPresenter per la gestione della business logic per la creazione della squadra;
- **Classi ereditate:**
 - CreateTeamModel



5 Componenti e classi

5.11.2.7 Client::Games::Battleship::Model::EnemyAttackPhaseModel

- **Descrizione**

Interfaccia che gestisce le funzionalità associate all'fase di attacco del team avversario;

- **Utilizzo**

Utilizzata da EnemyAttackPhasePresenterImpl per gestire la fase di attacco del team avversario;

5.11.2.8 Client::Games::Battleship::Model::EnemyAttackPhaseModelImpl

- **Descrizione**

Classe che implementa l'interfaccia che gestisce le funzionalità associate all'fase di attacco del team avversario;

- **Utilizzo**

Utilizzata da EnemyAttackPhasePresenterImpl per gestire la fase di attacco del team avversario;

5.11.2.9 Client::Games::Battleship::Model::JoinTeamModel

- **Descrizione**

Interfaccia che gestisce la business logic legata all'inserimento nella squadra da parte dell'utente;

- **Utilizzo**

Utilizzata da JointeamPresenter per la gestione della business logic legata all'inserimento nella squadra da parte dell'utente;

- **Sottoclassi:**

- JoinTeamModelImpl
- TeamManagementModel

5.11.2.10 Client::Games::Battleship::Model::JoinTeamModelImpl

- **Descrizione**

Implementa l'interfaccia che gestisce la business logic legata all'inserimento nella squadra da parte dell'utente;

- **Utilizzo**

Utilizzata da JointeamPresenter per la gestione della business logic legata all'inserimento nella squadra da parte dell'utente;

- **Classi ereditate:**

- JoinTeamModel

5.11.2.11 Client::Games::Battleship::Model::LeaderboardModel

- **Descrizione**

Interfaccia che gestisce la bussines logic_G legata alla gestione della classifica;

- **Utilizzo**

Utilizzata da LeaderboardPresenter per gestire la bussines logic_G legata alla gestione della classifica;

- **Relazioni con altre classi:**

- IN LeaderboardPresenter

Interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all'utente di aprire la classifica del gioco desiderato;



5 Componenti e classi

- **OUT Score**

Interfaccia che gestisce le informazioni riguardanti i punteggi di una squadra in classifica;

5.11.2.12 Client::Games::Battleship::Model::LeaderboardModelImpl

- **Descrizione**

Implementa l’interfaccia che gestisce la bussines logic_G legata alla gestione della classifica;

- **Utilizzo**

Utilizzata da LeaderboardPresenter per gestire la bussines logic_G legata alla gestione della classifica;

5.11.2.13 Client::Games::Battleship::Model::ProfileModel

- **Descrizione**

Interfaccia che gestisce la bussines logic_G legata al profilo;

- **Utilizzo**

Viene utilizzata dalle classi battleship per avere informazioni sull’utente;

5.11.2.14 Client::Games::Battleship::Model::ProfileModelImpl

- **Descrizione**

Classe che implementa l’interfaccia che gestisce la bussines logic_G legata al profilo;

- **Utilizzo**

Viene utilizzata dalle classi battleship per avere informazioni sull’utente;

5.11.2.15 Client::Games::Battleship::Model::SelectShipPositionModel

- **Descrizione**

Interfaccia che gestisce la business logic legata al posizionamento della nave;

- **Utilizzo**

Usata da SelectShipPositionPresenter per gestire la business logic legata al posizionamento della nave;

- **Sottoclassi:**

- SelectShipPositionModelImpl

5.11.2.16 Client::Games::Battleship::Model::SelectShipPositionModelImpl

- **Descrizione**

Implementa l’interfaccia che gestisce la business logic legata al posizionamento della nave;

- **Utilizzo**

Usata da SelectShipPositionPresenter per gestire la business logic legata al posizionamento della nave;

- **Classi ereditate:**

- SelectShipPositionModel



5 Componenti e classi

5.11.2.17 Client::Games::Battleship::Model::ShowFinalScoreModel

- **Descrizione**

Interfaccia che gestisce le funzionalità legate alla visualizzazione del punteggi finali della partita;

- **Utilizzo**

Utilizzata da ShowFinalScorePresenterImpl per visualizzazione del punteggi finali della partita;

5.11.2.18 Client::Games::Battleship::Model::ShowFinalScoreModelImpl

- **Descrizione**

Classe che implementa l’interfaccia che gestisce le funzionalità legate alla visualizzazione del punteggi finali della partita;

- **Utilizzo**

Utilizzata da ShowFinalScorePresenterImpl per visualizzazione del punteggi finali della partita;

5.11.2.19 Client::Games::Battleship::Model::TeamManagementModel

- **Descrizione**

Interfaccia che gestisce la business logic per la gestione del team;

- **Utilizzo**

Utilizzata da TeamManagementPresenter per la business logic per la gestione del team;

- **Classi ereditate:**

- JoinTeamModel

- **Sottoclassi:**

- TeamManagementModelImpl

5.11.2.20 Client::Games::Battleship::Model::TeamManagementModelImpl

- **Descrizione**

Implementa l’interfaccia che gestisce la business logic per la gestione del team;

- **Utilizzo**

Utilizzata da TeamManagementPresenter per la business logic per la gestione del team;

- **Classi ereditate:**

- TeamManagementModel

5.11.2.21 Client::Games::Battleship::Model::WaitingOpponentModel

- **Descrizione**

Interfaccia che gestisce la business logic delle funzionalità di attesa avversario;

- **Utilizzo**

Utilizzato da WaitingOpponentPresenter per gestire la business logic delle funzionalità di attesa avversario;

- **Sottoclassi:**

- WaitingOpponentModelImpl



5 Componenti e classi

5.11.2.22 Client::Games::Battleship::Model::WaitingOpponentModelImpl

- **Descrizione**

Implementa l’interfaccia che gestisce la business logic delle funzionalità di attesa avversario;

- **Utilizzo**

Utilizzato da WaitingOpponentPresenter per gestire la business logic delle funzionalità di attesa avversario;

- **Classi ereditate:**

- WaitingOpponentModel

5.12 Client::Games::Battleship::Model::Communication

5.12.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di mettere in comunicazione client e server per le funzionalità riguardanti il gioco Battleship.

- **Padre: Model**

5.12.2 Classi

5.12.2.1 Client::Games::Battleship::Model::Communication ::AttackPhaseCommunication

- **Descrizione**

Interfaccia per gestire la comunicazione nella fase di attacco;

- **Utilizzo**

Utilizzata da AttackPhaseModel per gestire la comunicazione nella fase di attacco;

- **Sottoclassi:**

- AttackPhaseCommunicationImpl

5.12.2.2 Client::Games::Battleship::Model::Communication ::AttackPhaseCommunicationImpl

- **Descrizione**

Classe che implementa l’interfaccia per gestire la comunicazione nella fase di attacco;

- **Utilizzo**

Utilizzata da AttackPhaseModelImpl per gestire la comunicazione nella fase di attacco;

- **Classi ereditate:**

- AttackPhaseCommunication

5.12.2.3 Client::Games::Battleship::Model::Communication ::BattleshipCommunication

- **Descrizione**

Interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;

- **Utilizzo**

Utilizzata da BattleshipModel per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;



5 Componenti e classi

- **Sottoclassi:**
 - `BattleshipCommunicationImpl`
- **Relazioni con altre classi:**
 - *IN BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
 - *OUT BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
 - *OUT Team*
Interfaccia che gestisce le informazioni riguardanti un team;
 - *OUT BattleshipCommunication*
Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al gioco Battleship, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

5.12.2.4 Client::Games::Battleship::Model::Communication ::BattleshipCommunicationImpl

- **Descrizione**
Implementa l'interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;
- **Utilizzo**
Utilizzata da BattleshipModelImpl per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;
- **Classi ereditate:**
 - `BattleshipCommunication`

5.12.2.5 Client::Games::Battleship::Model::Communication ::CreateTeamCommunication

- **Descrizione**
Interfaccia che espone metodi per la gestione della comunicazione con il server per le funzionalità di creazione della squadra;
- **Utilizzo**
Utilizzata da CreateTeamPresenter per gestire la comunicazione con il server per le funzionalità di creazione della squadra;
- **Sottoclassi:**
 - `CreateTeamCommunicationImpl`

5.12.2.6 Client::Games::Battleship::Model::Communication ::CreateTeamCommunicationImpl

- **Descrizione**
Implementa l'interfaccia per la gestione della comunicazione con il server per le funzionalità di attesa avversario;
- **Utilizzo**
Utilizzata da WaitingOpponentPresenter per la gestione della comunicazione con il server per le funzionalità di attesa avversario;
- **Classi ereditate:**
 - `CreateTeamCommunication`



5 Componenti e classi

5.12.2.7 Client::Games::Battleship::Model::Communication ::EnemyAttackPhaseCommunication

- **Descrizione**

Interfaccia che gestisce la comunicazione con il server per le funzionalità associate all'fase di attacco del team avversario;

- **Utilizzo**

Utilizzata da EnemyAttackPhaseModelImpl per gestire la fase di attacco del team avversario;

- **Sottoclassi:**

- EnemyAttackPhaseCommunicationImpl

5.12.2.8 Client::Games::Battleship::Model::Communication ::EnemyAttackPhaseCommunicationImpl

- **Descrizione**

Classe che implementa l'interfaccia che gestisce la comunicazione con il server per le funzionalità associate all'fase di attacco del team avversario;

- **Utilizzo**

Utilizzata da EnemyAttackPhaseModelImpl per gestire la fase di attacco del team avversario;

- **Classi ereditate:**

- EnemyAttackPhaseCommunication

5.12.2.9 Client::Games::Battleship::Model::Communication ::JoinTeamCommunication

- **Descrizione**

Interfaccia che gestisce la comunicazione con il per le funzionalità di inserimento nella squadra;

- **Utilizzo**

Utilizzata da JoinTeamPresenter per gestire la comunicazione con il per le funzionalità di inserimento nella squadra;

- **Sottoclassi:**

- JoinTeamCommunicationImpl
- TeamManagementCommunication

5.12.2.10 Client::Games::Battleship::Model::Communication ::JoinTeamCommunicationImpl

- **Descrizione**

Gestisce la comunicazione tra client e server per le funzionalità di base;

- **Utilizzo**

Viene utilizzata per comunicare con il server;

- **Classi ereditate:**

- JoinTeamCommunication



5 Componenti e classi

5.12.2.11 Client::Games::Battleship::Model::Communication ::LeaderboardCommunication

- **Descrizione**

Interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano la classifica di Battleship;

- **Utilizzo**

Utilizzata da LeaderboardModel per la gestione della comunicazione tra client e server per le funzionalità che riguardano la classifica di Battleship;

5.12.2.12 Client::Games::Battleship::Model::Communication ::LeaderboardCommunicationImpl

- **Descrizione**

Classe che implementa l'interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano la classifica di Battleship;

- **Utilizzo**

Utilizzata da LeaderboardModel per la gestione della comunicazione tra client e server per le funzionalità che riguardano la classifica di Battleship;

5.12.2.13 Client::Games::Battleship::Model::Communication ::ProfileCommunication

- **Descrizione**

Gestisce la comunicazione con in server per le funzionalità riguardanti la gestione del profilo assegnato al utente;

- **Utilizzo**

Viene utilizzata da ProfileModel per accedere ai dati del profilo memorizzati sul server;

5.12.2.14 Client::Games::Battleship::Model::Communication ::ProfileCommunicationImpl

- **Descrizione**

Classe che implementa l'interfacci che gestisce la comunicazione con in server per le funzionalità riguardanti la gestione del profilo assegnato al utente;

- **Utilizzo**

Viene utilizzata da ProfileModel per accedere ai dati del profilo memorizzati sul server;

5.12.2.15 Client::Games::Battleship::Model::Communication ::SelectShipPositionCommunication

- **Descrizione**

Interfaccia che gestisce la comunicazione con il server per la funzionalità del posizionamento navi;

- **Utilizzo**

Utilizzata da SelectShipPositionPresenter per gestire la comunicazione con il server per la funzionalità del posizionamento navi;

- **Sottoclassi:**

- SelectShipPositionCommunicationImpl



5 Componenti e classi

5.12.2.16 Client::Games::Battleship::Model::Communication ::SelectShipPositionCommunicationImpl

- **Descrizione**

Classe che implementa l’interfaccia che gestisce la comunicazione con il server per la funzionalità del posizionamento navi;

- **Utilizzo**

Utilizzata da SelectShipPositionPresenter per gestire la comunicazione con il server per la funzionalità del posizionamento navi;

- **Classi ereditate:**

- SelectShipPositionCommunication

5.12.2.17 Client::Games::Battleship::Model::Communication ::ShowFinalScoreCommunication

- **Descrizione**

Interfaccia che gestisce le funzionalità legate alla visualizzazione del punteggi finali della partita;

- **Utilizzo**

Utilizzata da ShowFinalScoreModelImpl per visualizzazione del punteggi finali della partita;

5.12.2.18 Client::Games::Battleship::Model::Communication ::ShowFinalScoreCommunicationImpl

- **Descrizione**

Classe che implementa l’interfaccia che gestisce le funzionalità legate alla visualizzazione del punteggi finali della partita;

- **Utilizzo**

Utilizzata da ShowFinalScoreModelImpl per visualizzazione del punteggi finali della partita;

5.12.2.19 Client::Games::Battleship::Model::Communication ::TeamManagementCommunication

- **Descrizione**

Gestisce la comunicazione con il server per le funzionalità legate alla gestione del team;

- **Utilizzo**

Viene utilizzata per comunicare con il server;

- **Classi ereditate:**

- JoinTeamCommunication

5.12.2.20 Client::Games::Battleship::Model::Communication ::TeamManagementCommunicationImpl

- **Descrizione**

Gestisce la comunicazione con il server per le funzionalità legate alla gestione del team;

- **Utilizzo**

Viene utilizzata per comunicare con il server;



5 Componenti e classi

5.12.2.21 Client::Games::Battleship::Model::Communication ::WaitingOpponentCommunication

- **Descrizione**

Interfaccia per la gestione della comunicazione con il server per le funzionalità di attesa avversario;

- **Utilizzo**

Utilizzata da WaitingOpponentPresenter per la gestione della comunicazione con il server per le funzionalità di attesa avversario;

5.12.2.22 Client::Games::Battleship::Model::Communication ::WaitingOpponentCommunicationImpl

- **Descrizione**

.Interfaccia per la gestione della comunicazione con il server per le funzionalità di attesa avversario;

- **Utilizzo**

Utilizzata da WaitingOpponentPresenter per la gestione della comunicazione con il server per le funzionalità di attesa avversario;

5.13 Client::Games::Battleship::Presenter

5.13.1 Informazioni generali

- **Descrizione**

Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.

- **Padre: Battleship**

5.13.2 Classi

5.13.2.1 Client::Games::Battleship::Presenter::AttackPhasePresenter

- **Descrizione**

Interfaccia che gestisce l'application logic_G legata alla fase di attacco;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la fase di attacco;

- **Sottoclassi:**

- AttackPhasePresenterImpl

- **Relazioni con altre classi:**

- *IN BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *IN BattleshipManager*

Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;

- *IN AttackPhaseView*

Interfaccia che gestisce la GUI per dare la possibilità all'utente di attaccare gli avversari;

- *OUT BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *OUT AttackPhaseView*

Interfaccia che gestisce la GUI per dare la possibilità all'utente di attaccare gli avversari;



5 Componenti e classi

5.13.2.2 Client::Games::Battleship::Presenter::AttackPhasePresenterImpl

- **Descrizione**

Implementa l'interfaccia che gestisce l'application logic_G legata alla fase di attacco;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la fase di attacco;

- **Classi ereditate:**

- AttackPhasePresenter

5.13.2.3 Client::Games::Battleship::Presenter::BattleshipManager

- **Descrizione**

Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;

- **Utilizzo**

Utilizzata da UtilityManager per gestire il gioco di battaglia navale;

- **Sottoclassi:**

- BattleshipManagerImpl

- **Relazioni con altre classi:**

- *IN BaseFunctionsManager*

Gestisce quale presenter deve essere in esecuzione; classe statica;

- *IN BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *OUT BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *OUT AttackPhasePresenter*

Interfaccia che gestisce l'application logic_G legata alla fase di attacco;

- *OUT CreateTeamPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di creare una squadra;

- *OUT EnemyAttackPhasePresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il risultato degli attacchi nemici;

- *OUT FinalScorePresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il punteggio finale;

- *OUT JoinTeamPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

- *OUT SelectShipPositionPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di posizionare le sue navi nel campo di battaglia;

- *OUT StartGamePresenter*

Interfaccia che gestisce l'application logic_G legata alla fase iniziale dove si può visualizzare la classifica del locale o iniziare a giocare;

- *OUT TeamManagementPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'capitano di gestire la propria squadra;



5 Componenti e classi

5.13.2.4 Client::Games::Battleship::Presenter::BattleshipManagerImpl

- **Descrizione**

Implementa l’interfaccia che gestisce quale presenter di Battleship deve entrare in esecuzione;

- **Utilizzo**

Utilizzata da UtilityManager per gestire il gioco di battaglia navale;

- **Classi ereditate:**

- BattleshipManager

5.13.2.5 Client::Games::Battleship::Presenter::CreateTeamPresenter

- **Descrizione**

Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di creare una squadra;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la fase di creazione del team;

- **Sottoclassi:**

- CreateTeamPresenterImpl

- **Relazioni con altre classi:**

- *IN* BattleshipModel

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *IN* BattleshipManager

Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;

- *IN* CreateTeamView

Interfaccia che gestisce la GUI per dare la possibilità al capitano di creare una squadra;

- *OUT* BattleshipModel

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *OUT* CreateTeamView

Interfaccia che gestisce la GUI per dare la possibilità al capitano di creare una squadra;

5.13.2.6 Client::Games::Battleship::Presenter::CreateTeamPresenterImpl

- **Descrizione**

Implementa l’interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di creare una squadra;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la fase di creazione del team;

- **Classi ereditate:**

- CreateTeamPresenter

5.13.2.7 Client::Games::Battleship::Presenter::EnemyAttackPhasePresenter

- **Descrizione**

Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la fase di attacco del team avversario;



5 Componenti e classi

- **Sottoclassi:**
 - `EnemyAttackPhasePresenterImpl`
- **Relazioni con altre classi:**
 - *IN BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
 - *IN BattleshipManager*
Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;
 - *IN EnemyAttackPhaseView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;
 - *OUT BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
 - *OUT EnemyAttackPhaseView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;

5.13.2.8 Client::Games::Battleship::Presenter::EnemyAttackPhasePresenterImpl

- **Descrizione**
Implementa l’interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;
- **Utilizzo**
Utilizzato da BattleshipManager per gestire la fase di attacco del team avversario;
- **Classi ereditate:**
 - `EnemyAttackPhasePresenter`

5.13.2.9 Client::Games::Battleship::Presenter::FinalScorePresenter

- **Descrizione**
Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di visualizzare il punteggio finale;
- **Utilizzo**
Utilizzato da BattleshipManager per gestire la visualizzazione del punteggio finale;
- **Sottoclassi:**
 - `FinalScorePresenterImpl`
- **Relazioni con altre classi:**
 - *IN BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
 - *IN BattleshipManager*
Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;
 - *IN FinalScoreView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il punteggio finale;
 - *OUT BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
 - *OUT FinalScoreView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il punteggio finale;
 - *OUT Score*
Interfaccia che gestisce le informazioni riguardanti i punteggi di una squadra in classifica;



5 Componenti e classi

5.13.2.10 Client::Games::Battleship::Presenter::FinalScorePresenterImpl

- **Descrizione**

Implementa l'interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il punteggio finale;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la visualizzazione del punteggio finale;

- **Classi ereditate:**

- FinalScorePresenter

5.13.2.11 Client::Games::Battleship::Presenter::JoinTeamPresenter

- **Descrizione**

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la creazione del team da parte di un membro del team che non sia il capitano;

- **Sottoclassi:**

- JoinTeamPresenterImpl

- **Relazioni con altre classi:**

- *IN BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *IN BattleshipManager*

Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;

- *IN JoinTeamView*

Interfaccia che gestisce la GUI per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

- *OUT BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *OUT JoinTeamView*

Interfaccia che gestisce la GUI per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

5.13.2.12 Client::Games::Battleship::Presenter::JoinTeamPresenterImpl

- **Descrizione**

Implementa l'interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la creazione del team da parte di un membro del team che non sia il capitano;

- **Classi ereditate:**

- JoinTeamPresenter

- **Relazioni con altre classi:**

- *OUT MembersAdapter*

Vista per la personalizzazione della lista dei membri di un team.



5 Componenti e classi

5.13.2.13 Client::Games::Battleship::Presenter::LeaderboardPresenter

- **Descrizione**

Interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all’utente di aprire la classifica del gioco desiderato;

- **Utilizzo**

Utilizzata da UtilityManager per gestire la visualizzazione della classifica;

- **Relazioni con altre classi:**

- *IN* UtilityManager

Interfaccia che gestisce quale presenter del gestore dei giochi deve essere in esecuzione;

- *IN* LeaderboardView

Interfaccia che gestisce la GUI per dare la possibilità al utente di visualizzare la classifica relativa al gioco selezionato;

- *OUT* LeaderboardModel

Interfaccia che gestisce la bussines logic_G legata alla gestione della classifica;

- *OUT* Score

Interfaccia che gestisce le informazioni riguardanti i punteggi di una squadra in classifica;

- *OUT* LeaderboardView

Interfaccia che gestisce la GUI per dare la possibilità al utente di visualizzare la classifica relativa al gioco selezionato;

5.13.2.14 Client::Games::Battleship::Presenter::LeaderboardPresenterImpl

- **Descrizione**

Implementa l’interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all’utente di aprire la classifica del gioco desiderato;

- **Utilizzo**

Utilizzata da UtilityManager per gestire la visualizzazione della classifica;

- **Relazioni con altre classi:**

- *OUT* ScoreAdapter

Classe per adattare e personalizzare la lista di migliori punteggi.

5.13.2.15 Client::Games::Battleship::Presenter::SelectShipPositionPresenter

- **Descrizione**

Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire il posizionamento delle navi da parte dell’utente;

- **Sottoclassi:**

- SelectShipPositionPresenterImpl

- **Relazioni con altre classi:**

- *IN* BattleshipModel

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;



5 Componenti e classi

- *IN BattleshipManager*
Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;
- *IN SelectShipPositionView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;
- *OUT BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
- *OUT SelectShipPositionView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;

5.13.2.16 Client::Games::Battleship::Presenter::SelectShipPositionPresenterImpl

- **Descrizione**

Implementa l’interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire il posizionamento delle navi da parte dell’utente;

- **Classi ereditate:**

- SelectShipPositionPresenter

5.13.2.17 Client::Games::Battleship::Presenter::SenderShotPresenter

- **Descrizione**

Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di far inserire all’utente la posizione del colpo da sparare;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire l’inserimento del colpo da sparare;

5.13.2.18 Client::Games::Battleship::Presenter::SenderShotPresenterImpl

- **Descrizione**

Implementa l’interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di far inserire all’utente la posizione del colpo da sparare;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire l’inserimento del colpo da sparare;

5.13.2.19 Client::Games::Battleship::Presenter::StartGamePresenter

- **Descrizione**

Interfaccia che gestisce l’application logic_G legata alla fase iniziale dove si può visualizzare la classifica del locale o iniziare a giocare;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la fase di scelta se iniziare il gioco o visualizzare la classifica del locale;

- **Sottoclassi:**

- StartGamePresenterImpl

- **Relazioni con altre classi:**

- *IN BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;



5 Componenti e classi

- *IN BattleshipManager*
Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;
- *IN ChooseRoleView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di scegliere che ruolo avere all’interno della squadra;
- *OUT BattleshipModel*
Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;
- *OUT ChooseRoleView*
Interfaccia che gestisce la GUI per dare la possibilità all’utente di scegliere che ruolo avere all’interno della squadra;

5.13.2.20 Client::Games::Battleship::Presenter::StartGamePresenterImpl

- **Descrizione**

Implementa l’interfaccia che gestisce l’application logic_G legata alla fase iniziale dove si può visualizzare la classifica del locale o iniziare a giocare;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la fase di scelta se iniziare il gioco o visualizzare la classifica del locale;

- **Classi ereditate:**

- StartGamePresenter

5.13.2.21 Client::Games::Battleship::Presenter::TeamManagementPresenter

- **Descrizione**

Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’capitano di gestire la propria squadra;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la gestione della squadra da parte del capitano;

- **Sottoclassi:**

- TeamManagementPresenterImpl

- **Relazioni con altre classi:**

- *IN BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *IN BattleshipManager*

Interfacce che gestisce quale presenter di Battleship deve entrare in esecuzione;

- *IN TeamManagementView*

Interfaccia che gestisce la GUI per dare la possibilità all’utente di gestire la propria squadra;

- *OUT BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *OUT TeamManagementView*

Interfaccia che gestisce la GUI per dare la possibilità all’utente di gestire la propria squadra;

- *OUT Team*

Interfaccia che gestisce le informazioni riguardanti un team;



5 Componenti e classi

5.13.2.22 Client::Games::Battleship::Presenter::TeamManagementPresenterImpl

- **Descrizione**

Implementa l’interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’capitano di gestire la propria squadra;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la gestione della squadra da parte del capitano;

- **Classi ereditate:**

- TeamManagementPresenter

- **Relazioni con altre classi:**

- OUT MembersAdapter

Vista per la personalizzazione del lista dei membri di un team.

5.13.2.23 Client::Games::Battleship::Presenter::WaitingOpponentPresenter

- **Descrizione**

Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità ricercare ed aspettare un opposente;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la ricerca e l’attesa di un opposente;

- **Sottoclassi:**

- WaitingOpponentPresenterImpl

5.13.2.24 Client::Games::Battleship::Presenter::WaitingOpponentPresenterImpl

- **Descrizione**

Implementa l’interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità ricercare ed aspettare un opposente;

- **Utilizzo**

Utilizzato da BattleshipManager per gestire la ricerca e l’attesa di un opposente;

- **Classi ereditate:**

- WaitingOpponentPresenter

5.14 Client::Games::Battleship::Types

5.14.1 Informazioni generali

- **Descrizione**

Package contenente i tipi.

- **Padre: Battleship**

5.14.2 Classi

5.14.2.1 Client::Games::Battleship::Types::Cell

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti un settore del campo di battaglia;

- **Utilizzo**

Utilizzato da Grid per gestire lo stato del campo da gioco;

- **Sottoclassi:**



5 Componenti e classi

- CellImpl
- Relazioni con altre classi:
 - IN Grid
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia;
 - IN AttackPhaseView
Interfaccia che gestisce la GUI per dare la possibilità all’utente di attaccare gli avversari;
 - IN EnemyAttackPhaseView
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;
 - IN SelectShipPositionView
Interfaccia che gestisce la GUI per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;

5.14.2.2 Client::Games::Battleship::Types::CellImpl

- Descrizione
Implementa l’interfaccia che gestisce le informazioni riguardanti un settore del campo di battaglia;
- Utilizzo
Utilizzato da Grid per gestire lo stato del campo da gioco;
- Classi ereditate:
 - Cell

5.14.2.3 Client::Games::Battleship::Types::Field

- Descrizione
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;
- Utilizzo
Utilizzato dal package Battleship per gestire lo stato del campo da gioco;
- Sottoclassi:
 - FieldImpl
- Relazioni con altre classi:
 - IN AttackPhaseView
Interfaccia che gestisce la GUI per dare la possibilità all’utente di attaccare gli avversari;
 - IN EnemyAttackPhaseView
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;
 - IN SelectShipPositionView
Interfaccia che gestisce la GUI per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;
 - OUT Grid
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia;



5 Componenti e classi

5.14.2.4 Client::Games::Battleship::Types::FieldImpl

- **Descrizione**

Implementa l’interfaccia che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;

- **Utilizzo**

Utilizzato dal package Battleship per gestire lo stato del campo da gioco;

- **Classi ereditate:**

- Field

5.14.2.5 Client::Games::Battleship::Types::Grid

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti il campo di battaglia;

- **Utilizzo**

Utilizzato da Field per gestire lo stato del campo da gioco;

- **Sottoclassi:**

- GridImpl

- **Relazioni con altre classi:**

- *IN* Field

Interfaccia che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;

- *IN* AttackPhaseView

Interfaccia che gestisce la GUI per dare la possibilità all’utente di attaccare gli avversari;

- *IN* EnemyAttackPhaseView

Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;

- *IN* SelectShipPositionView

Interfaccia che gestisce la GUI per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;

- *OUT* Cell

Interfaccia che gestisce le informazioni riguardanti un settore del campo di battaglia;

5.14.2.6 Client::Games::Battleship::Types::GridImpl

- **Descrizione**

Implementa l’interfaccia che gestisce le informazioni riguardanti il campo di battaglia;

- **Utilizzo**

Utilizzato da Field per gestire lo stato del campo da gioco;

- **Classi ereditate:**

- Grid



5 Componenti e classi

5.14.2.7 Client::Games::Battleship::Types::Position

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti la posizione di un oggetto nel campo di battaglia;

- **Utilizzo**

Utilizzato da Battleship per gestire le informazioni riguardanti la posizione di un oggetto nel campo di battaglia (es. colpo, nave);

- **Sottoclassi:**

- PositionImpl

- **Relazioni con altre classi:**

- *IN Ship*

Interfaccia che gestisce le informazioni riguardanti una nave;

- *IN Shoot*

Interfaccia che gestisce le informazioni riguardanti un colpo sparato da un giocatore;

- *IN Shoot*

Interfaccia che gestisce le informazioni riguardanti un colpo sparato da un giocatore;

5.14.2.8 Client::Games::Battleship::Types::PositionImpl

- **Descrizione**

Implementa l'interfaccia che gestisce le informazioni riguardanti la posizione di un oggetto nel campo di battaglia;

- **Utilizzo**

Utilizzato da Battleship per gestire le informazioni riguardanti la posizione di un oggetto nel campo di battaglia (es. colpo, nave);

- **Classi ereditate:**

- Position

5.14.2.9 Client::Games::Battleship::Types::Ship

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Utilizzato da Battleship per gestire le informazioni riguardanti una nave;

- **Classi ereditate:**

- ShipType

- **Sottoclassi:**

- ShipImpl

- **Relazioni con altre classi:**

- *IN AttackPhaseView*

Interfaccia che gestisce la GUI per dare la possibilità all'utente di attaccare gli avversari;

- *IN EnemyAttackPhaseView*

Interfaccia che gestisce la GUI per dare la possibilità all'utente di visualizzare il risultato degli attacchi nemici;



5 Componenti e classi

- *OUT Position*

Interfaccia che gestisce le informazioni riguardanti la posizione di un oggetto nel campo di battaglia;

5.14.2.10 Client::Games::Battleship::Types::ShipImpl

- **Descrizione**

Implementa l’interfaccia che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Utilizzato da Battleship per gestire le informazioni riguardanti una nave;

- **Classi ereditate:**

- Ship

- ShipType

5.14.2.11 Client::Games::Battleship::Types::ShipType

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Utilizzato da Battleship per gestire le informazioni riguardanti una nave;

- **Sottoclassi:**

- Ship

- ShipImpl

- ShipTypeImpl

5.14.2.12 Client::Games::Battleship::Types::ShipTypeImpl

- **Descrizione**

Implementa l’interfaccia che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Utilizzato da Battleship per gestire le informazioni riguardanti una nave;

- **Classi ereditate:**

- ShipType

5.14.2.13 Client::Games::Battleship::Types::Shoot

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti un colpo sparato da un giocatore;

- **Utilizzo**

Utilizzato da Battleship per gestire le informazioni riguardanti un colpo sparato da un giocatore;

- **Sottoclassi:**

- ShootImpl

- **Relazioni con altre classi:**

- *IN AttackPhaseView*

Interfaccia che gestisce la GUI per dare la possibilità all’utente di attaccare gli avversari;



5 Componenti e classi

- **IN EnemyAttackPhaseView**
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;
- **OUT Position**
Interfaccia che gestisce le informazioni riguardanti la posizione di un oggetto nel campo di battaglia;
- **OUT Position**
Interfaccia che gestisce le informazioni riguardanti la posizione di un oggetto nel campo di battaglia;

5.14.2.14 Client::Games::Battleship::Types::ShootImpl

- **Descrizione**
Implementa l’interfaccia che gestisce le informazioni riguardanti un colpo sparato da un giocatore;
- **Utilizzo**
Utilizzato da Battleship per gestire le informazioni riguardanti un colpo sparato da un giocatore;
- **Classi ereditate:**
 - Shoot

5.15 Client::Games::Battleship::View

5.15.1 Informazioni generali

- **Descrizione**
Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative al gioco Battleship.
- **Padre: Battleship**

5.15.2 Classi

5.15.2.1 Client::Games::Battleship::View::AttackPhaseView

- **Descrizione**
Interfaccia che gestisce la GUI per dare la possibilità all’utente di attaccare gli avversari;
- **Utilizzo**
Utilizzato da AttackPhasePresenter per gestire la fase di attacco dell’utente;
- **Sottoclassi:**
 - AttackPhaseViewImpl
- **Relazioni con altre classi:**
 - **IN AttackPhasePresenter**
Interfaccia che gestisce l’application logic_G legata alla fase di attacco;
 - **OUT AttackPhasePresenter**
Interfaccia che gestisce l’application logic_G legata alla fase di attacco;
 - **OUT Cell**
Interfaccia che gestisce le informazioni riguardanti un settore del campo di battaglia;
 - **OUT Field**
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;



5 Componenti e classi

- *OUT Grid*
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia;
- *OUT Ship*
Interfaccia che gestisce le informazioni riguardanti una nave;
- *OUT Shoot*
Interfaccia che gestisce le informazioni riguardanti un colpo sparato da un giocatore;

5.15.2.2 Client::Games::Battleship::View::AttackPhaseViewImpl

- **Descrizione**

Implementa l’interfaccia che gestisce la GUI per dare la possibilità all’utente di attaccare gli avversari;

- **Utilizzo**

Utilizzato da AttackPhasePresenter per gestire la fase di attacco dell’utente;

- **Classi ereditate:**

- AttackPhaseView

5.15.2.3 Client::Games::Battleship::View::ChooseRoleView

- **Descrizione**

Interfaccia che gestisce la GUI per dare la possibilità all’utente di scegliere che ruolo avere all’interno della squadra;

- **Utilizzo**

Utilizzato da ChooseRolePresenter per gestire la scelta del ruolo dell’utente;

- **Sottoclassi:**

- ChooseRoleViewImpl

- **Relazioni con altre classi:**

- *IN StartGamePresenter*

Interfaccia che gestisce l’application logic_G legata alla fase iniziale dove si può visualizzare la classifica del locale o iniziare a giocare;

- *OUT StartGamePresenter*

Interfaccia che gestisce l’application logic_G legata alla fase iniziale dove si può visualizzare la classifica del locale o iniziare a giocare;

5.15.2.4 Client::Games::Battleship::View::ChooseRoleViewImpl

- **Descrizione**

Implementa l’interfaccia che gestisce la GUI per dare la possibilità all’utente di scegliere che ruolo avere all’interno della squadra;

- **Utilizzo**

Utilizzato da ChooseRolePresenter per gestire la scelta del ruolo dell’utente;

- **Classi ereditate:**

- ChooseRoleView



5 Componenti e classi

5.15.2.5 Client::Games::Battleship::View::CreateTeamView

- **Descrizione**

Interfaccia che gestisce la GUI per dare la possibilità al capitano di creare una squadra;

- **Utilizzo**

Utilizzato da CreateTeamPresenter per gestire la creazione della squadra da parte del capitano;

- **Sottoclassi:**

- CreateTeamViewImpl

- **Relazioni con altre classi:**

- *IN* CreateTeamPresenter

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di creare una squadra;

- *OUT* CreateTeamPresenter

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di creare una squadra;

5.15.2.6 Client::Games::Battleship::View::CreateTeamViewImpl

- **Descrizione**

Implementa l'interfaccia che gestisce la GUI per dare la possibilità al capitano di creare una squadra;

- **Utilizzo**

Utilizzato da CreateTeamPresenter per gestire la creazione della squadra da parte del capitano;

- **Classi ereditate:**

- CreateTeamView

5.15.2.7 Client::Games::Battleship::View::EnemyAttackPhaseView

- **Descrizione**

Interfaccia che gestisce la GUI per dare la possibilità all'utente di visualizzare il risultato degli attacchi nemici;

- **Utilizzo**

Utilizzata da EnemyAttackPhasePresenter per far visualizzare all'utente il risultato degli attacchi nemici;

- **Relazioni con altre classi:**

- *IN* EnemyAttackPhasePresenter

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il risultato degli attacchi nemici;

- *OUT* EnemyAttackPhasePresenter

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il risultato degli attacchi nemici;

- *OUT* Cell

Interfaccia che gestisce le informazioni riguardanti un settore del campo di battaglia;

- *OUT* Field

Interfaccia che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;



5 Componenti e classi

- *OUT Grid*
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia;
- *OUT Ship*
Interfaccia che gestisce le informazioni riguardanti una nave;
- *OUT Shoot*
Interfaccia che gestisce le informazioni riguardanti un colpo sparato da un giocatore;

5.15.2.8 Client::Games::Battleship::View::EnemyAttackPhaseViewImpl

- **Descrizione**
Implementa l’interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il risultato degli attacchi nemici;
- **Utilizzo**
Utilizzata da EnemyAttackPhasePresenter per far visualizzare all’utente il risultato degli attacchi nemici;

5.15.2.9 Client::Games::Battleship::View::FinalScoreView

- **Descrizione**
Interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il punteggio finale;
- **Utilizzo**
Utilizzata da FinalScorePresenter per far visualizzare all’utente il punteggio finale;
- **Relazioni con altre classi:**
 - *IN FinalScorePresenter*
Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di visualizzare il punteggio finale;
 - *OUT FinalScorePresenter*
Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di visualizzare il punteggio finale;

5.15.2.10 Client::Games::Battleship::View::FinalScoreViewImpl

- **Descrizione**
Implementa l’interfaccia che gestisce la GUI per dare la possibilità all’utente di visualizzare il punteggio finale;
- **Utilizzo**
Utilizzata da FinalScorePresenter per far visualizzare all’utente il punteggio finale;

5.15.2.11 Client::Games::Battleship::View::JoinTeamView

- **Descrizione**
Interfaccia che gestisce la GUI per dare la possibilità all’utente di vedere com’è composta la squadra e eventualmente dissociarsi;
- **Utilizzo**
Utilizzata da JoinTeamPresenter per dare la possibilità all’utente di vedere com’è composta la squadra e eventualmente dissociarsi;
- **Relazioni con altre classi:**
 - *IN JoinTeamPresenter*
Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’utente di vedere com’è composta la squadra e eventualmente dissociarsi;



5 Componenti e classi

- *OUT JoinTeamPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

5.15.2.12 Client::Games::Battleship::View::JoinTeamViewImpl

- **Descrizione**

Implementa l'interfaccia che gestisce la GUI per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

- **Utilizzo**

Utilizzata da JoinTeamPresenter per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

5.15.2.13 Client::Games::Battleship::View::MembersAdapter

- **Descrizione**

Vista per la personalizzazione del lista dei membri di un team.

- **Utilizzo**

Utilizzata da TeamManagementPresenter e JoinTeamPresenter per visualizzare i membri di un team.

- **Relazioni con altre classi:**

- *IN JoinTeamPresenterImpl*

Implementa l'interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di vedere com'è composta la squadra e eventualmente dissociarsi;

- *IN TeamManagementPresenterImpl*

Implementa l'interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'capitano di gestire la propria squadra;

5.15.2.14 Client::Games::Battleship::View::SelectShipPositionView

- **Descrizione**

Interfaccia che gestisce la GUI per dare la possibilità all'utente di posizionare le sue navi nel campo di battaglia;

- **Utilizzo**

Utilizzata da SelectShipPositionPresenter per gestire la GUI per dare la possibilità all'utente di posizionare le sue navi nel campo di battaglia;

- **Sottoclassi:**

- *SelectShipPositionViewImpl*

- **Relazioni con altre classi:**

- *IN SelectShipPositionPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di posizionare le sue navi nel campo di battaglia;

- *OUT SelectShipPositionPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di posizionare le sue navi nel campo di battaglia;

- *OUT Cell*

Interfaccia che gestisce le informazioni riguardanti un settore del campo di battaglia;



5 Componenti e classi

- **OUT Field**
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;
- **OUT Grid**
Interfaccia che gestisce le informazioni riguardanti il campo di battaglia;

5.15.2.15 Client::Games::Battleship::View::SelectShipPositionViewImpl

- **Descrizione**
Implementa l’interfaccia che gestisce la GUI per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;
- **Utilizzo**
Utilizzata da SelectShipPositionPresenter per gestire la GUI per dare la possibilità all’utente di posizionare le sue navi nel campo di battaglia;
- **Classi ereditate:**
 - SelectShipPositionView

5.15.2.16 Client::Games::Battleship::View::TeamManagementView

- **Descrizione**
Interfaccia che gestisce la GUI per dare la possibilità all’utente di gestire la propria squadra;
- **Utilizzo**
Utilizzata da TeamManagementPresenter per gestire la GUI per dare la possibilità all’utente di gestire la propria squadra;
- **Sottoclassi:**
 - TeamManagementViewImpl
- **Relazioni con altre classi:**
 - **IN TeamManagementPresenter**
Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’capitano di gestire la propria squadra;
 - **OUT TeamManagementPresenter**
Interfaccia che gestisce l’application logic_G e la presentation logic_G per dare la possibilità all’capitano di gestire la propria squadra;

5.15.2.17 Client::Games::Battleship::View::TeamManagementViewImpl

- **Descrizione**
Implementa l’interfaccia che gestisce la GUI per dare la possibilità all’utente di gestire la propria squadra;
- **Utilizzo**
Utilizzata da TeamManagementPresenter per gestire la GUI per dare la possibilità all’utente di gestire la propria squadra;
- **Classi ereditate:**
 - TeamManagementView



5 Componenti e classi

5.15.2.18 Client::Games::Battleship::View::WaitingOpponentView

- **Descrizione**

Interfaccia che gestisce la GUI di attesa dell'opponente;

- **Utilizzo**

Utilizzata da WaitingOpponentPresenter per gestire la GUI di attesa dell'opponente;

- **Sottoclassi:**

- WaitingOpponentViewImpl

5.15.2.19 Client::Games::Battleship::View::WaitingOpponentViewImpl

- **Descrizione**

Implementa l'interfaccia che gestisce la GUI di attesa dell'opponente;

- **Utilizzo**

Utilizzata da WaitingOpponentPresenter per gestire la GUI di attesa dell'opponente;

- **Classi ereditate:**

- WaitingOpponentView

5.16 Client::Games::Utility

5.16.1 Informazioni generali

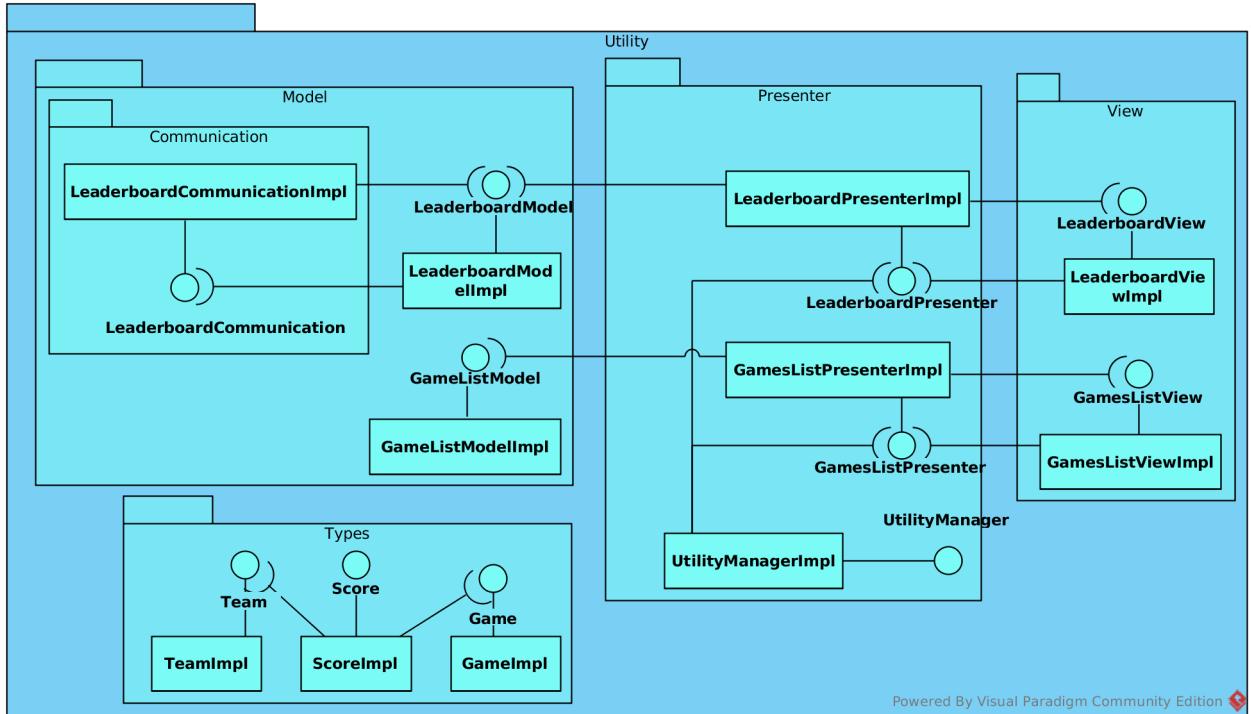


Figura 13: Client::Games::Utility

- **Descrizione**

Package che contiene le componenti relative alle funzionalità di selezione del gioco e di visualizzazione delle classifiche relative ai giochi.

- **Padre:** Games

- **Package contenuti:**



5 Componenti e classi

- **Model**
Package che contiene le componenti che hanno il compito di fornire il modello dei dati.
- **Presenter**
Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.
- **Types**
Package contenente i tipi.
- **View**
Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative ai giochi.

5.17 Client::Games::Utility::Model

5.17.1 Informazioni generali

- **Descrizione**
Package che contiene le componenti che hanno il compito di fornire il modello dei dati.
- **Padre: Utility**

5.17.2 Classi

5.17.2.1 Client::Games::Utility::Model::GamesListModel

- **Descrizione**
Interfaccia che gestisce la bussines logic_G legata alla gestione dei giochi avviabili;
- **Utilizzo**
Utilizzata da GamesListPresenter per gestire la bussines logic_G legata alla gestione dei giochi avviabili;
- **Relazioni con altre classi:**
 - *IN GamesListPresenter*
Interfaccia che gestisce la application logic_G e la presentation logic_G per fornire l'elenco dei giochi che l'applicazione mette a disposizione;

5.17.2.2 Client::Games::Utility::Model::GamesListModelImpl

- **Descrizione**
Implementa l'interfaccia che gestisce la bussines logic_G legata alla gestione dei giochi avviabili;
- **Utilizzo**
Utilizzata da GamesListPresenter per gestire la bussines logic_G legata alla gestione dei giochi avviabili;

5.18 Client::Games::Utility::Presenter

5.18.1 Informazioni generali

- **Descrizione**
Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.
- **Padre: Utility**



5 Componenti e classi

5.18.2 Classi

5.18.2.1 Client::Games::Utility::Presenter::GamesListPresenter

- **Descrizione**

Interfaccia che gestisce la application logic_G e la presentation logic_G per fornire l'elenco dei giochi che l'applicazione mette a disposizione;

- **Utilizzo**

Utilizzata da UtilityManager per gestire la selezione e l'avvio di un gioco;

- **Sottoclassi:**

- GamesListPresenterImpl

- **Relazioni con altre classi:**

- *IN UtilityManager*

Interfaccia che gestisce quale presenter del gestore dei giochi deve essere in esecuzione;

- *IN GamesListView*

Interfaccia che gestisce la GUI riguardante la visualizzazione della lista dei giochi disponibili;

- *OUT GamesListModel*

Interfaccia che gestisce la bussines logic_G legata alla gestione dei giochi avviabili;

- *OUT GamesListView*

Interfaccia che gestisce la GUI riguardante la visualizzazione della lista dei giochi disponibili;

5.18.2.2 Client::Games::Utility::Presenter::GamesListPresenterImpl

- **Descrizione**

Implementa l'interfaccia che gestisce la application logic_G e la presentation logic_G per fornire l'elenco dei giochi che l'applicazione mette a disposizione;

- **Utilizzo**

Utilizzata da UtilityManager per gestire la selezione e l'avvio di un gioco;

- **Classi ereditate:**

- GamesListPresenter

- **Relazioni con altre classi:**

- *OUT GamesAdapter*

Implementa l'interfaccia personalizzata della lista di giochi.

5.18.2.3 Client::Games::Utility::Presenter::UtilityManager

- **Descrizione**

Interfaccia che gestisce quale presenter del gestore dei giochi deve essere in esecuzione;

- **Utilizzo**

Utilizzata da BaseFunctionManager per gestire l'avvio e l'esecuzione dei giochi;

- **Sottoclassi:**

- UtilityManagerImpl

- **Relazioni con altre classi:**

- *IN BaseFunctionsManager*

Gestisce quale presenter deve essere in esecuzione; classe statica;



5 Componenti e classi

- *OUT LeaderboardPresenter*
Interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all’utente di aprire la classifica del gioco desiderato;
- *OUT GamesListPresenter*
Interfaccia che gestisce la application logic_G e la presentation logic_G per fornire l’elenco dei giochi che l’applicazione mette a disposizione;

5.18.2.4 Client::Games::Utility::Presenter::UtilityManagerImpl

- **Descrizione**
Implementa l’interfaccia che gestisce quale presenter del gestore dei giochi deve essere in esecuzione;
- **Utilizzo**
Utilizzata da BaseFunctionManager per gestire l’avvio e l’esecuzione dei giochi;
- **Classi ereditate:**
 - UtilityManager

5.19 Client::Games::Utility::Types

5.19.1 Informazioni generali

- **Descrizione**
Package contenente i tipi.
- **Padre: Utility**

5.19.2 Classi

5.19.2.1 Client::Games::Utility::Types::Game

- **Descrizione**
Interfaccia che gestisce l’identificazione di un particolare gioco;
- **Utilizzo**
Utilizzato da UtilityManager per avviare un particolare gioco;
- **Sottoclassi:**
 - GameImpl

5.19.2.2 Client::Games::Utility::Types::GameImpl

- **Descrizione**
Implementa l’interfaccia che gestisce l’identificazione di un particolare gioco;
- **Utilizzo**
Utilizzato da UtilityManager per avviare un particolare gioco;
- **Classi ereditate:**
 - Game



5 Componenti e classi

5.19.2.3 Client::Games::Utility::Types::Score

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti i punteggi di una squadra in classifica;

- **Utilizzo**

Utilizzata dal package Games per gestire i punteggi;

- **Sottoclassi:**

- ScoreImpl

- **Relazioni con altre classi:**

- *IN LeaderboardModel*

Interfaccia che gestisce la bussines logic_G legata alla gestione della classifica;

- *IN FinalScorePresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'utente di visualizzare il punteggio finale;

- *IN LeaderboardPresenter*

Interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all'utente di aprire la classifica del gioco desiderato;

- *OUT GamesListView*

Interfaccia che gestisce la GUI riguardante la visualizzazione della lista dei giochi disponibili;

5.19.2.4 Client::Games::Utility::Types::ScoreImpl

- **Descrizione**

Implementa l'interfaccia che gestisce le informazioni riguardanti i punteggi di una squadra in classifica;

- **Utilizzo**

Utilizzata dal package Games per gestire i punteggi;

- **Classi ereditate:**

- Score

5.19.2.5 Client::Games::Utility::Types::Team

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti un team;

- **Utilizzo**

Utilizzato dai giochi per gestire le informazioni riguardanti un team;

- **Sottoclassi:**

- TeamImpl

- **Relazioni con altre classi:**

- *IN BattleshipModel*

Interfaccia che gestisce la bussines logic_G legata al gioco di Battleship;

- *IN BattleshipCommunication*

Interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;

- *IN TeamManagementPresenter*

Interfaccia che gestisce l'application logic_G e la presentation logic_G per dare la possibilità all'capitano di gestire la propria squadra;



5 Componenti e classi

5.19.2.6 Client::Games::Utility::Types::TeamImpl

- **Descrizione**

Implementa l'interfaccia che gestisce le informazioni riguardanti un team;

- **Utilizzo**

Utilizzato dai giochi per gestire le informazioni riguardanti un team;

- **Classi ereditate:**

- Team

5.20 Client::Games::Utility::View

5.20.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative ai giochi.

- **Padre: Utility**

5.20.2 Classi

5.20.2.1 Client::Games::Utility::View::GamesAdapter

- **Descrizione**

Implementa l'interfaccia personalizzata della lista di giochi.

- **Utilizzo**

Utilizzata da GamesListPresenter per presentare la lista di giochi personalizzata.

- **Relazioni con altre classi:**

- *IN GamesListPresenterImpl*

Implementa l'interfaccia che gestisce la application logic_G e la presentation logic_G per fornire l'elenco dei giochi che l'applicazione mette a disposizione;

5.20.2.2 Client::Games::Utility::View::GamesListView

- **Descrizione**

Interfaccia che gestisce la GUI riguardante la visualizzazione della lista dei giochi disponibili;

- **Utilizzo**

Utilizzato da GamesListPresenter per gestire la GUI riguardante la visualizzazione della lista dei giochi disponibili;

- **Relazioni con altre classi:**

- *IN GamesListPresenter*

Interfaccia che gestisce la application logic_G e la presentation logic_G per fornire l'elenco dei giochi che l'applicazione mette a disposizione;

- *IN Score*

Interfaccia che gestisce le informazioni riguardanti i punteggi di una squadra in classifica;

- *OUT GamesListPresenter*

Interfaccia che gestisce la application logic_G e la presentation logic_G per fornire l'elenco dei giochi che l'applicazione mette a disposizione;



5 Componenti e classi

5.20.2.3 Client::Games::Utility::View::GamesListViewImpl

- **Descrizione**

Implementa l'interfaccia che gestisce la GUI riguardante la visualizzazione della lista dei giochi disponibili;

- **Utilizzo**

Utilizzato da GamesListPresenter per gestire la GUI riguardante la visualizzazione della lista dei giochi disponibili;

5.20.2.4 Client::Games::Utility::View::LeaderboardView

- **Descrizione**

Interfaccia che gestisce la GUI per dare la possibilità al utente di visualizzare la classifica relativa al gioco selezionato;

- **Utilizzo**

Utilizzato da LeaderboardPresenter per gestire la GUI per dare la possibilità al utente di visualizzare la classifica relativa al gioco selezionato;

- **Relazioni con altre classi:**

- *IN* LeaderboardPresenter

Interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all'utente di aprire la classifica del gioco desiderato;

- *OUT* LeaderboardPresenter

Interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all'utente di aprire la classifica del gioco desiderato;

5.20.2.5 Client::Games::Utility::View::LeaderboardViewImpl

- **Descrizione**

5.20.2.6 Client::Games::Utility::View::ScoreAdapter

- **Descrizione**

Classe per adattare e personalizzare la lista di migliori punteggi.

- **Utilizzo**

Utilizzata da LeaderboardPresenterImpl per personalizzare la visualizzazione della classifica.

- **Relazioni con altre classi:**

- *IN* LeaderboardPresenterImpl

Implementa l'interfaccia che gestisce Model:Leaderboard e View::Leaderboard implementando la application logic_G e la presentation logic_G per dare la possibilità all'utente di aprire la classifica del gioco desiderato;



5 Componenti e classi

5.21 Client::Profile

5.21.1 Informazioni generali

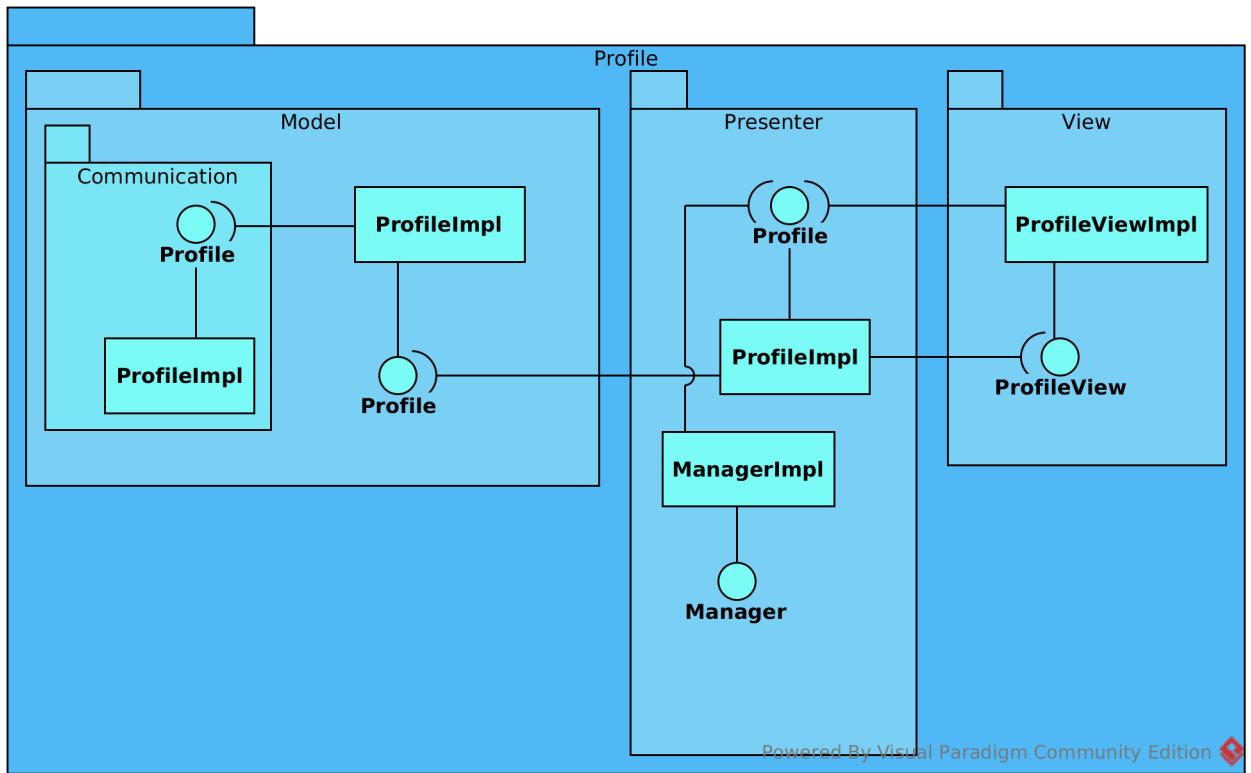


Figura 14: Client::Profile

- **Descrizione**
Package che contiene le componenti che realizzano la gestione del profilo.
- **Padre: Client**
- **Package contenuti:**
 - **Model**
Package che contiene le componenti che hanno il compito di fornire il modello dei dati.
 - **Presenter**
Contiene tutte le componenti che gestiscono l'application logic_G e permettono la comunicazione tra model e view.
 - **View**
Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative al profilo.

5.22 Client::Profile::Model

5.22.1 Informazioni generali

- **Descrizione**
Package che contiene le componenti che hanno il compito di fornire il modello dei dati.
- **Padre: Profile**
- **Package contenuti:**



5 Componenti e classi

- **Communication**

Package che contiene le componenti che hanno il compito di mettere in comunicazione client e server per le funzionalità riguardanti la gestione del profilo.

5.22.2 Classi

5.22.2.1 Client::Profile::Model::ProfileModel

- **Descrizione**

Gestisce tutta la business logic_G riguardante la gestione del profilo dell’utente;

- **Utilizzo**

Viene utilizzata per gestire tutta la business logic_G riguardante la gestione del profilo utente;

- **Sottoclassi:**

- ProfileModelImpl

- **Relazioni con altre classi:**

- *IN* ProfilePresenter

Gestisce l’application logic_G di tutte le funzionalità legate al profilo;

- *OUT* ProfileCommunication

Gestisce tutta la comunicazione con il server riguardante la gestione del profilo dell’utente;

- *OUT* Profile

Gestisce le informazioni che rappresentano un profilo;

5.22.2.2 Client::Profile::Model::ProfileModelImpl

- **Descrizione**

Gestisce tutta la business logic_G riguardante la gestione del profilo dell’utente;

- **Utilizzo**

Viene utilizzata per gestire tutta la business logic_G riguardante la gestione del profilo utente;

- **Classi ereditate:**

- ProfileModel

5.23 Client::Profile::Model::Communication

5.23.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di mettere in comunicazione client e server per le funzionalità riguardanti la gestione del profilo.

- **Padre: Model**

5.23.2 Classi

5.23.2.1 Client::Profile::Model::Communication::ProfileCommunication

- **Descrizione**

Gestisce tutta la comunicazione con il server riguardante la gestione del profilo dell’utente;

- **Utilizzo**

Viene utilizzata per comunicare con il server;



5 Componenti e classi

- **Sottoclassi:**
 - ProfileCommunicationImpl
- **Relazioni con altre classi:**
 - *IN ProfileModel*
Gestisce tutta la business logic_G riguardante la gestione del profilo dell’utente;
 - *OUT Profile*
Gestisce le informazioni che rappresentano un profilo;
 - *OUT ProfileCommunication*
Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al profilo dell’utente, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

5.23.2.2 Client::Profile::Model::Communication::ProfileCommunicationImpl

- **Descrizione**
Gestisce tutta la comunicazione con il server riguardante la gestione del profilo dell’utente;
- **Utilizzo**
Viene utilizzata per comunicare con il server;
- **Classi ereditate:**
 - ProfileCommunication

5.24 Client::Profile::Presenter

5.24.1 Informazioni generali

- **Descrizione**
Contiene tutte le componenti che gestiscono l’application logic_G e permettono la comunicazione tra model e view.
- **Padre: Profile**

5.24.2 Classi

5.24.2.1 Client::Profile::Presenter::ProfileManager

- **Descrizione**
Gestisce quale presenter del gestore dei giochi deve essere in esecuzione;
- **Utilizzo**
Viene utilizzata per la gestione del profilo;
- **Sottoclassi:**
 - ProfileManagerImpl
- **Relazioni con altre classi:**
 - *IN BaseFunctionsManager*
Gestisce quale presenter deve essere in esecuzione; classe statica;
 - *OUT ProfilePresenter*
Gestisce l’application logic_G di tutte le funzionalità legate al profilo;
 - *OUT Profile*
Gestisce le informazioni che rappresentano un profilo;



5 Componenti e classi

5.24.2.2 Client::Profile::Presenter::ProfileManagerImpl

- **Descrizione**

La classe gestisce le funzionalità del profilo;

- **Utilizzo**

Viene utilizzata per la gestione del profilo;

- **Classi ereditate:**

- ProfileManager

5.24.2.3 Client::Profile::Presenter::ProfilePresenter

- **Descrizione**

Gestisce l'application logic_G di tutte le funzionalità legate al profilo;

- **Utilizzo**

Viene utilizzata per gestire l'application logic_G di tutte le funzionalità legate al profilo;

- **Sottoclassi:**

- ProfilePresenterImpl

- **Relazioni con altre classi:**

- IN ProfileManager

Gestisce quale presenter del gestore dei giochi deve essere in esecuzione;

- IN ProfileView

Gestisce le funzionalità di un view passiva_G legate alla gestione del profilo;

- OUT ProfileModel

Gestisce tutta la business logic_G riguardante la gestione del profilo dell'utente;

- OUT ProfileView

Gestisce le funzionalità di un view passiva_G legate alla gestione del profilo;

- OUT Profile

Gestisce le informazioni che rappresentano un profilo;

5.24.2.4 Client::Profile::Presenter::ProfilePresenterImpl

- **Descrizione**

La classe rappresenta l'insieme delle funzionalità legate al profilo;

- **Utilizzo**

Viene utilizzata modificare le informazioni relative al profilo;

- **Classi ereditate:**

- ProfilePresenter

5.25 Client::Profile::View

5.25.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che hanno il compito di visualizzare nella GUI_G le informazioni relative al profilo.

- **Padre: Profile**



5 Componenti e classi

5.25.2 Classi

5.25.2.1 Client::Profile::View::ProfileView

- **Descrizione**

Gestisce le funzionalità di un view passiva_G legate alla gestione del profilo;

- **Utilizzo**

Viene utilizzata per visualizzare una GUI per la gestione del profilo;

- **Sottoclassi:**

- ProfileViewImpl

- **Relazioni con altre classi:**

- *IN* ProfilePresenter

Gestisce l'application logic_G di tutte le funzionalità legate al profilo;

- *OUT* ProfilePresenter

Gestisce l'application logic_G di tutte le funzionalità legate al profilo;

- *OUT* Profile

Gestisce le informazioni che rappresentano un profilo;

5.25.2.2 Client::Profile::View::ProfileViewImpl

- **Descrizione**

Gestisce le funzionalità di un view passiva_G legate alla gestione del profilo;

- **Utilizzo**

Viene utilizzata per visualizzare una GUI per la gestione del profilo;

- **Classi ereditate:**

- ProfileView

5.26 Client::Types

5.26.1 Informazioni generali

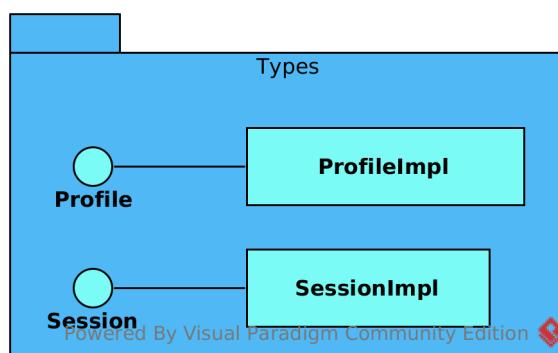


Figura 15: Client::Types

- **Descrizione**

Package che contiene i tipi comuni usati dalle componenti dell'applicazione.

- **Padre: Client**



5 Componenti e classi

5.26.2 Classi

5.26.2.1 Client::Types::Profile

- **Descrizione**

Gestisce le informazioni che rappresentano un profilo;

- **Utilizzo**

Viene utilizzata per gestire le informazioni che rappresentano un profilo;

- **Sottoclassi:**

- ProfileImpl

- **Relazioni con altre classi:**

- *IN ProfileCommunication*

Gestisce tutta la comunicazione con il server riguardante la gestione del profilo dell’utente;

- *IN ProfileModel*

Gestisce tutta la business logic_G riguardante la gestione del profilo dell’utente;

- *IN ProfileManager*

Gestisce quale presenter del gestore dei giochi deve essere in esecuzione;

- *IN ProfilePresenter*

Gestisce l’application logic_G di tutte le funzionalità legate al profilo;

- *IN ProfileView*

Gestisce le funzionalità di un view passiva_G legate alla gestione del profilo;

5.26.2.2 Client::Types::ProfileImpl

- **Descrizione**

Gestisce le informazioni che rappresentano un profilo;

- **Utilizzo**

Viene utilizzata per gestire le informazioni che rappresentano un profilo;

- **Classi ereditate:**

- Profile

5.26.2.3 Client::Types::Session

- **Descrizione**

Rappresenta una sessione di un utente;

- **Utilizzo**

Viene utilizzata per rappresentare una sessione utente;

- **Sottoclassi:**

- SessionImpl

5.26.2.4 Client::Types::SessionImpl

- **Descrizione**

Rappresenta una sessione di un utente;

- **Utilizzo**

Viene utilizzata per rappresentare una sessione utente;

- **Classi ereditate:**

- Session



5.27 Server

5.27.1 Informazioni generali

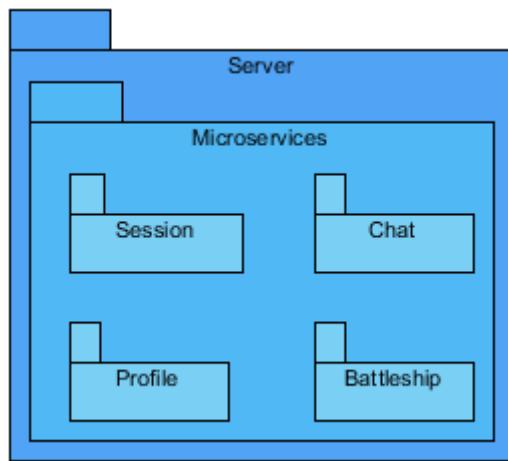


Figura 16: Server

- **Descrizione**
Package che contiene il back end del progetto.
- **Package contenuti:**
 - **Microservices**
Packege che contiene i micro servizi.

5.28 Server::Microservices

5.28.1 Informazioni generali

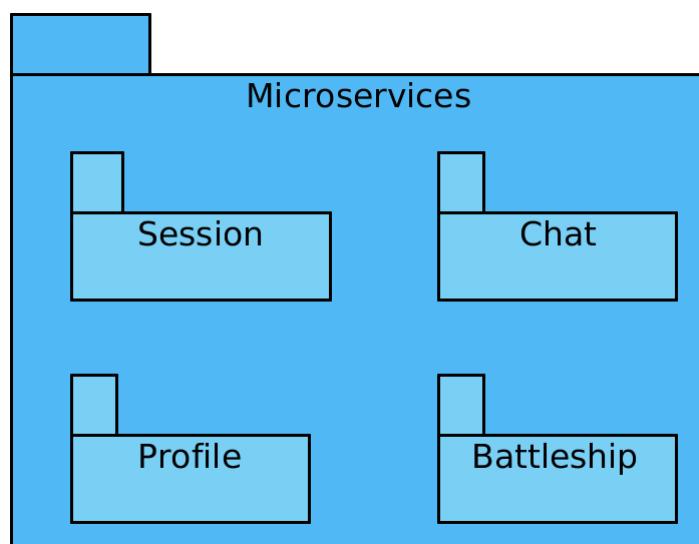


Figura 17: Server::Microservices

- **Descrizione**
Packege che contiene i micro servizi.



5 Componenti e classi

- **Padre:** Server
- **Package contenuti:**
 - Battleship
Package che contiene le componenti che realizzano il micro servizio che si occupa della gestione del gioco Battleship.
 - Profile
Package che contiene le componenti che realizzano il micro servizio che si occupa della gestione dei profili degli utenti.

5.29 Server::Microservices::Battleship

5.29.1 Informazioni generali

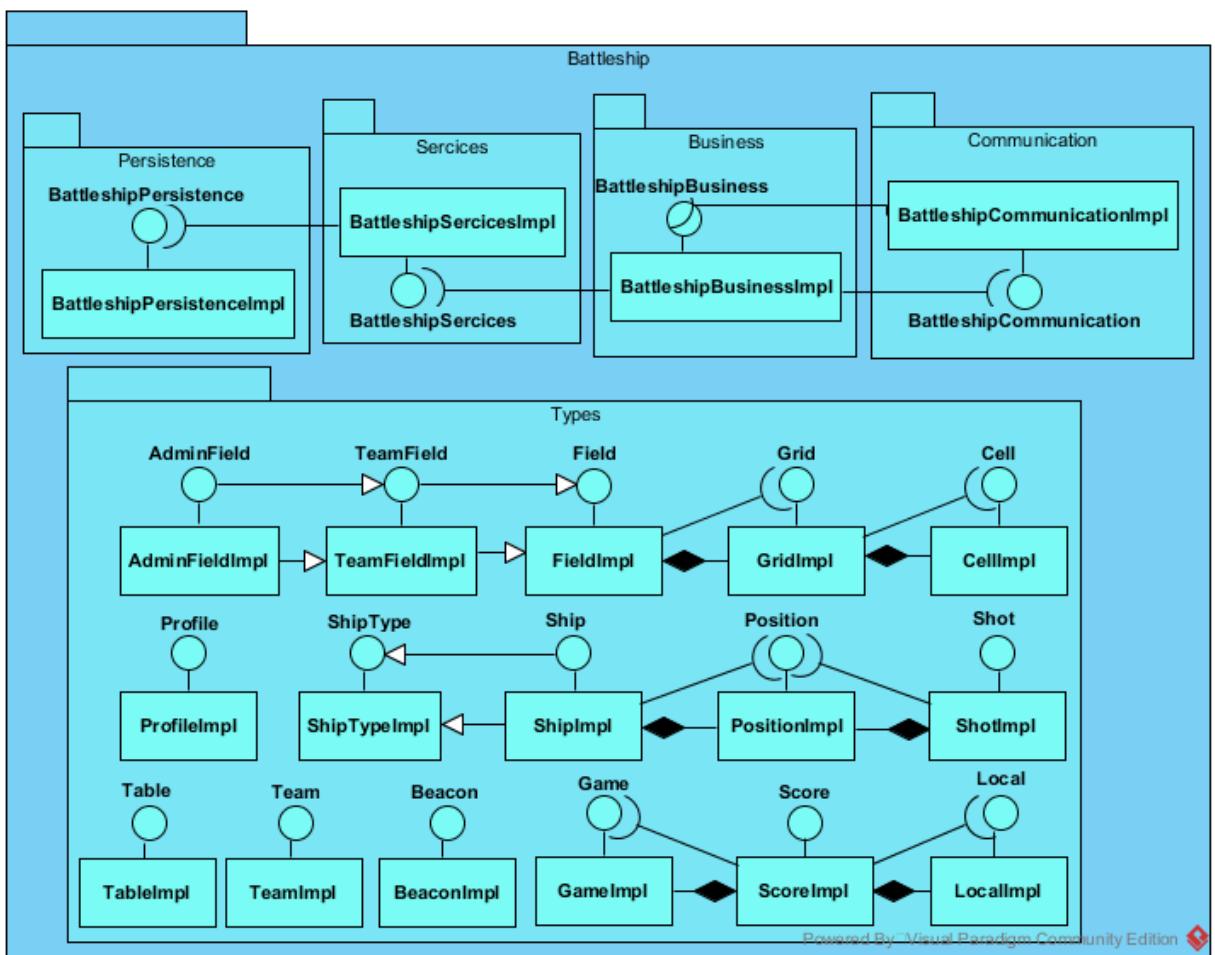


Figura 18: Server::Microservices::Battleship

- **Descrizione**
Package che contiene le componenti che realizzano il micro servizio che si occupa della gestione del gioco Battleship.
- **Padre: Microservices**
- **Package contenuti:**
 - **Business**
Package che contiene la logica di business.



5 Componenti e classi

– Communication

Package che contiene i componenti che hanno il compito di: esporre API del micro servizio; inviare delle informazioni ad altri componenti del sistema.

– Persistence

Package che contiene le componenti che realizzano lo strato di persistenza dei dati.

– Services

Package che contiene le componenti che realizzano lo strato di servizio per l'accesso ai dati.

– Types

Package contenente i tipi.

5.30 Server::Microservices::Battleship::Business

5.30.1 Informazioni generali

- **Descrizione**

Package che contiene la logica di business.

- **Padre: Battleship**

5.30.2 Classi

5.30.2.1 Server::Microservices::Battleship::Business::BattleshipBusiness

- **Descrizione**

Interfaccia che gestisce la logica di funzionamento del servizio e risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il gioco Battleship;

- **Utilizzo**

Viene utilizzata per gestire la logica del gioco Battleship e quindi la risposta alle richieste provenienti dall'esterno;

- **Sottoclassi:**

- BattleshipBusinessImpl

- **Relazioni con altre classi:**

- *IN BattleshipCommunication*

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al gioco Battleship, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- *OUT BattleshipCommunication*

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al gioco Battleship, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- *OUT BattleshipServices*

Interfaccia che gestisce i dati relativi al gioco Battleship;

5.30.2.2 Server::Microservices::Battleship::Business::BattleshipBusinessImpl

- **Descrizione**

Classe che gestisce la logica di funzionamento del servizio e risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il gioco Battleship;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia BattleshipBusiness, utilizzati per gestire la logica del gioco Battleship;



5 Componenti e classi

- **Classi ereditate:**

- BattleshipBusiness

5.31 Server::Microservices::Battleship::Communication

5.31.1 Informazioni generali

- **Descrizione**

Package che contiene i componenti che hanno il compito di: esporre API del micro servizio; inviare delle informazioni ad altri componenti del sistema.

- **Padre:** Battleship

5.31.2 Classi

5.31.2.1 Server::Microservices::Battleship::Communication::BattleshipCommunication

- **Descrizione**

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al gioco Battleship, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- **Utilizzo**

Viene utilizzata per gestire la comunicazione tra server e client riguardante le funzionalità legate al gioco Battleship;

- **Sottoclassi:**

- BattleshipCommunicationImpl

- **Relazioni con altre classi:**

- *IN* BattleshipCommunication

Interfaccia per la gestione della comunicazione tra client e server per le funzionalità che riguardano il gioco Battleship;

- *IN* BattleshipBusiness

Interfaccia che gestisce la logica di funzionamento del servizio e risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il gioco Battleship;

- *OUT* BattleshipBusiness

Interfaccia che gestisce la logica di funzionamento del servizio e risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il gioco Battleship;

5.31.2.2 Server::Microservices::Battleship::Communication

::BattleshipCommunicationImpl

- **Descrizione**

Classe che gestisce la comunicazione tra server e client riguardante le funzionalità legate al gioco Battleship, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia BattleshipCommunication, utilizzati per gestire la comunicazione tra server e client riguardante le funzionalità legate al gioco Battleship;

- **Classi ereditate:**

- BattleshipCommunication



5 Componenti e classi

5.32 Server::Microservices::Battleship::Persistence

5.32.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano lo strato di persistenza dei dati.

- **Padre: Battleship**

- **Package contenuti:**

- **Dao**

Package che contiene le componenti che realizzano il DAO.

5.32.2 Classi

5.32.2.1 Server::Microservices::Battleship::Persistence::BattleshipPersistence

- **Descrizione**

Interfaccia che gestisce i dati elementari relativi al gioco Battleship, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **Utilizzo**

Viene utilizzata per interfacciarsi e comunicare direttamente con il database utilizzato;

- **Sottoclassi:**

- **BattleshipPersistenceImpl**

- **Relazioni con altre classi:**

- **IN BattleshipServices**

Interfaccia che gestisce i dati relativi al gioco Battleship;

5.32.2.2 Server::Microservices::Battleship::Persistence::BattleshipPersistenceImpl

- **Descrizione**

Classe che gestisce i dati elementari relativi al gioco Battleship, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia BattleshipPersistence, utilizzati per interfacciarsi e comunicare direttamente con il database utilizzato;

- **Classi ereditate:**

- **BattleshipPersistence**

5.33 Server::Microservices::Battleship::Persistence::Dao

5.33.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano il DAO.

- **Padre: Persistence**

- **Package contenuti:**

- **Sql**

Package che contiene le componenti che realizzano la parte di DAO relativa ai database relazionali.



5 Componenti e classi

5.33.2 Classi

5.33.2.1 Server::Microservices::Battleship::Persistence::Dao::BattleshipDao

- **Descrizione**

Gestisce il salvataggio e il recupero di informazioni relative al gioco Battleship;

- **Utilizzo**

Viene usata per esporre le possibili operazioni relative al gioco Battleship;

- **Sottoclassi:**

- SqlBattleshipDaoImpl

5.33.2.2 Server::Microservices::Battleship::Persistence::Dao::DaoFactory

- **Descrizione**

Fornisce una interfaccia per l'accesso ai dati elementari legati al gioco battleship, indipendente dal linguaggio usato per memorizzarli;

- **Utilizzo**

Viene usata per creare la tipologia di factory richiesta e per fornire i dao relativi a tale tipologia;

- **Sottoclassi:**

- SqlDaoFactoryImpl

5.34 Server::Microservices::Battleship::Persistence::Dao::Sql

5.34.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano la parte di DAO relativa ai database relazionali.

- **Padre: Dao**

5.34.2 Classi

5.34.2.1 Server::Microservices::Battleship::Persistence::Dao::Sql::SqlBattleshipDaoImpl

- **Descrizione**

Gestisce il salvataggio e il recupero di informazioni relative al gioco Battleship mediante SQL;

- **Utilizzo**

Viene usata per salvare e recuperare le informazioni relative al gioco Battleship mediante SQL;

- **Classi ereditate:**

- BattleshipDao

5.34.2.2 Server::Microservices::Battleship::Persistence::Dao::Sql::SqlDaoFactoryImpl

- **Descrizione**

Gestisce le funzionalità per la creazione di specifici oggetti che manipolano i dati relativi al gioco battleship;

- **Utilizzo**

Viene usata per creare specifici oggetti che manipolano i dati, in un database di tipo sql, relativi al gioco Battleship;



5 Componenti e classi

- **Classi ereditate:**

- DaoFactory

5.35 Server::Microservices::Battleship::Services

5.35.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano lo strato di servizio per l'accesso ai dati.

- **Padre:** Battleship

5.35.2 Classi

5.35.2.1 Server::Microservices::Battleship::Services::BattleshipServices

- **Descrizione**

Interfaccia che gestisce i dati relativi al gioco Battleship;

- **Utilizzo**

Viene utilizzata per raggruppare e suddividere in maniera logica le componenti grezze fornite dallo strato sottostante (Persistence) e renderle quindi utili per la parte di Business;

- **Sottoclassi:**

- BattleshipServicesImpl

- **Relazioni con altre classi:**

- *IN* BattleshipBusiness

Interfaccia che gestisce la logica di funzionamento del servizio e risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il gioco Battleship;

- *OUT* BattleshipPersistence

Interfaccia che gestisce i dati elementari relativi al gioco Battleship, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

5.35.2.2 Server::Microservices::Battleship::Services::BattleshipServicesImpl

- **Descrizione**

Classe che gestisce i dati relativi al gioco Battleship;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia BattleshipServices, utilizzati per raggruppare e suddividere in maniera logica le componenti grezze fornite dallo strato sottostante (Persistence) e renderle quindi utili per la parte di Business;

- **Classi ereditate:**

- BattleshipServices

5.36 Server::Microservices::Battleship::Types

5.36.1 Informazioni generali

- **Descrizione**

Package contenente i tipi.

- **Padre:** Battleship



5 Componenti e classi

5.36.2 Classi

5.36.2.1 Server::Microservices::Battleship::Types::AdminField

- **Descrizione**

Interfaccia che rappresenta un campo da gioco sul quale si possono aggiungere nuovi colpi;

- **Utilizzo**

Viene utilizzata per rappresentare un campo da gioco sul quale si possono aggiungere nuovi colpi;

- **Classi ereditate:**

- TeamField

5.36.2.2 Server::Microservices::Battleship::Types::AdminFieldImpl

- **Descrizione**

Classe che rappresenta un campo da gioco sul quale si possono aggiungere nuovi colpi;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia AdminField, utilizzati per rappresentare un campo da gioco sul quale si possono aggiungere nuovi colpi;

- **Classi ereditate:**

- TeamFieldImpl

5.36.2.3 Server::Microservices::Battleship::Types::Beacon

- **Descrizione**

Interfaccia che gestisce le informazioni associate ad un beacon;

- **Utilizzo**

Viene utilizzata per gestire le informazioni che rappresentano un beacon;

- **Sottoclassi:**

- BeaconImpl

5.36.2.4 Server::Microservices::Battleship::Types::BeaconImpl

- **Descrizione**

Classe che gestisce la logica di funzionamento riguardanti i beacon;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Beacon, utilizzati per gestire i dispositivi beacon;

- **Classi ereditate:**

- Beacon

5.36.2.5 Server::Microservices::Battleship::Types::Cell

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti un settore del campo di battaglia del gioco Battleship;

- **Utilizzo**

Viene utilizzata per rappresentare una cella del campo di battaglia del gioco Battleship;

- **Sottoclassi:**

- CellImpl



5 Componenti e classi

5.36.2.6 Server::Microservices::Battleship::Types::CellImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti un settore del campo di battaglia del gioco Battleship;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Cell, utilizzati per rappresentare lo stato di una cella del campo di battaglia del gioco Battleship;

- **Classi ereditate:**

 - Cell

5.36.2.7 Server::Microservices::Battleship::Types::Field

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti il campo di battaglia del gioco Battleship;

- **Sottoclassi:**

 - FieldImpl
 - TeamField

5.36.2.8 Server::Microservices::Battleship::Types::FieldImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti il campo di battaglia del gioco Battleship;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Field, utilizzati per gestire le informazioni riguardanti il campo di battaglia del gioco Battleship;

- **Classi ereditate:**

 - Field

- **Sottoclassi:**

 - TeamFieldImpl

5.36.2.9 Server::Microservices::Battleship::Types::Game

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti un gioco;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti un gioco;

- **Sottoclassi:**

 - GameImpl

- **Relazioni con altre classi:**

 - IN Score

Interfaccia che gestisce le informazioni che rappresentano un punteggio effettuato da una squadra in un determinato locale in una partita ad un gioco.



5 Componenti e classi

5.36.2.10 Server::Microservices::Battleship::Types::GameImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti un gioco;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia Game, utilizzati per la gestione delle informazioni riguardanti un gioco;

- **Classi ereditate:**

- Game

5.36.2.11 Server::Microservices::Battleship::Types::Grid

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti il campo di battaglia di una squadra;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti il campo di battaglia di una squadra;

- **Sottoclassi:**

- GridImpl

5.36.2.12 Server::Microservices::Battleship::Types::GridImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti il campo di battaglia di una squadra;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia Grid, utilizzati per gestire le informazioni riguardanti il campo di battaglia di una squadra;

- **Classi ereditate:**

- Grid

5.36.2.13 Server::Microservices::Battleship::Types::Local

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti un locale specifico;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti un locale specifico;

- **Sottoclassi:**

- LocalImpl

- **Relazioni con altre classi:**

- IN Score

Interfaccia che gestisce le informazioni che rappresentano un punteggio effettuato da una squadra in un determinato locale in una partita ad un gioco.



5 Componenti e classi

5.36.2.14 Server::Microservices::Battleship::Types::LocalImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti un locale specifico;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Profile, utilizzati per gestire le informazioni riguardanti un locale specifico;

- **Classi ereditate:**

 - Local

5.36.2.15 Server::Microservices::Battleship::Types::Position

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti la posizione di un oggetto nel campo di battaglia del gioco Battleship;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti la posizione di un oggetto nel campo di battaglia del gioco Battleship;

- **Sottoclassi:**

 - PositionImpl

5.36.2.16 Server::Microservices::Battleship::Types::PositionImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti la posizione di un oggetto nel campo di battaglia del gioco Battleship;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Position, utilizzati per gestire le informazioni riguardanti la posizione di un oggetto nel campo di battaglia del gioco Battleship;

- **Classi ereditate:**

 - Position

5.36.2.17 Server::Microservices::Battleship::Types::Profile

- **Descrizione**

Interfaccia che gestisce le informazioni che rappresentano un profilo;

- **Utilizzo**

Viene utilizzata per gestire le informazioni che rappresentano un profilo;

- **Sottoclassi:**

 - ProfileImpl

5.36.2.18 Server::Microservices::Battleship::Types::ProfileImpl

- **Descrizione**

Classe che gestisce le informazioni che rappresentano un profilo;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Profile, utilizzati per la gestione del profilo di un utente;

- **Classi ereditate:**

 - Profile



5 Componenti e classi

5.36.2.19 Server::Microservices::Battleship::Types::Score

- **Descrizione**

Interfaccia che gestisce le informazioni che rappresentano un punteggio effettuato da una squadra in un determinato locale in una partita ad un gioco;

- **Utilizzo**

Viene utilizzata per gestire le informazioni che rappresentano un punteggio effettuato da una squadra in un determinato locale in una partita ad un gioco;

- **Sottoclassi:**

 - ScoreImpl

- **Relazioni con altre classi:**

 - OUT Game

Interfaccia che gestisce le informazioni riguardanti un gioco;

 - OUT Local

Interfaccia che gestisce le informazioni riguardanti un locale specifico;

5.36.2.20 Server::Microservices::Battleship::Types::ScoreImpl

- **Descrizione**

Classe che gestisce le informazioni che rappresentano un punteggio effettuato da una squadra in un determinato locale in una partita ad un gioco;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Score, utilizzati per la gestione delle informazioni che rappresentano un punteggio effettuato da una squadra in un determinato locale in una partita ad un gioco;

- **Classi ereditate:**

 - Score

5.36.2.21 Server::Microservices::Battleship::Types::Ship

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti una nave;

- **Classi ereditate:**

 - ShipType

- **Sottoclassi:**

 - ShipImpl

5.36.2.22 Server::Microservices::Battleship::Types::ShipImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Ship, utilizzati per gestire le informazioni riguardanti una nave;

- **Classi ereditate:**

 - Ship

 - ShipType



5 Componenti e classi

5.36.2.23 Server::Microservices::Battleship::Types::ShipType

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti una nave;

- **Sottoclassi:**

- Ship
- ShipImpl
- ShipTypeImpl

5.36.2.24 Server::Microservices::Battleship::Types::ShipTypeImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti una nave;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Ship, utilizzati per gestire le informazioni riguardanti una nave;

- **Classi ereditate:**

- ShipType

5.36.2.25 Server::Microservices::Battleship::Types::Shot

- **Descrizione**

Interfaccia che gestisce le informazioni riguardanti un colpo sparato da un giocatore;

- **Utilizzo**

Viene utilizzata per gestire le informazioni riguardanti un colpo sparato da un giocatore;

- **Sottoclassi:**

- ShotImpl

5.36.2.26 Server::Microservices::Battleship::Types::ShotImpl

- **Descrizione**

Classe che gestisce le informazioni riguardanti un colpo sparato da un giocatore;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Shoot, utilizzati per gestire le informazioni riguardanti un colpo sparato da un giocatore;

- **Classi ereditate:**

- Shot

5.36.2.27 Server::Microservices::Battleship::Types::Table

- **Descrizione**

Interfaccia che gestisce le informazioni associate ad un tavolo;

- **Utilizzo**

Utilizzato da BattleshipBusiness per controllare se ci sono team presenti nel tavolo;



5 Componenti e classi

5.36.2.28 Server::Microservices::Battleship::Types::TableImpl

- **Descrizione**

Classe che implementa l’interfaccia che gestisce le informazioni associate ad un tavolo;

- **Utilizzo**

Utilizzato da BattleshipBusiness per controllare se ci sono team presenti nel tavolo;

5.36.2.29 Server::Microservices::Battleship::Types::Team

- **Descrizione**

Interfaccia che gestisce le informazioni che rappresentano un team;

- **Utilizzo**

Viene utilizzata per gestire le informazioni che rappresentano un team;

- **Sottoclassi:**

- TeamImpl

- **Relazioni con altre classi:**

5.36.2.30 Server::Microservices::Battleship::Types::TeamField

- **Descrizione**

Interfaccia che rappresenta il campo avversario del team di cui si fa parte;

- **Utilizzo**

Viene utilizzata per rappresentare il campo avversario del team di cui si fa parte;

- **Classi ereditate:**

- Field

- **Sottoclassi:**

- AdminField

- TeamFieldImpl

5.36.2.31 Server::Microservices::Battleship::Types::TeamFieldImpl

- **Descrizione**

Classe che rappresenta il campo avversario del team di cui si fa parte;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia TeamField, utilizzati per rappresentare il campo avversario del team di cui si fa parte;

- **Classi ereditate:**

- FieldImpl

- TeamField

- **Sottoclassi:**

- AdminFieldImpl



5 Componenti e classi

5.36.2.32 Server::Microservices::Battleship::Types::TeamImpl

- **Descrizione**

Classe che gestisce le informazioni che rappresentano un team;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia Team, utilizzati per la gestione del profilo di un team;

- **Classi ereditate:**

- Team

5.37 Server::Microservices::Profile

5.37.1 Informazioni generali

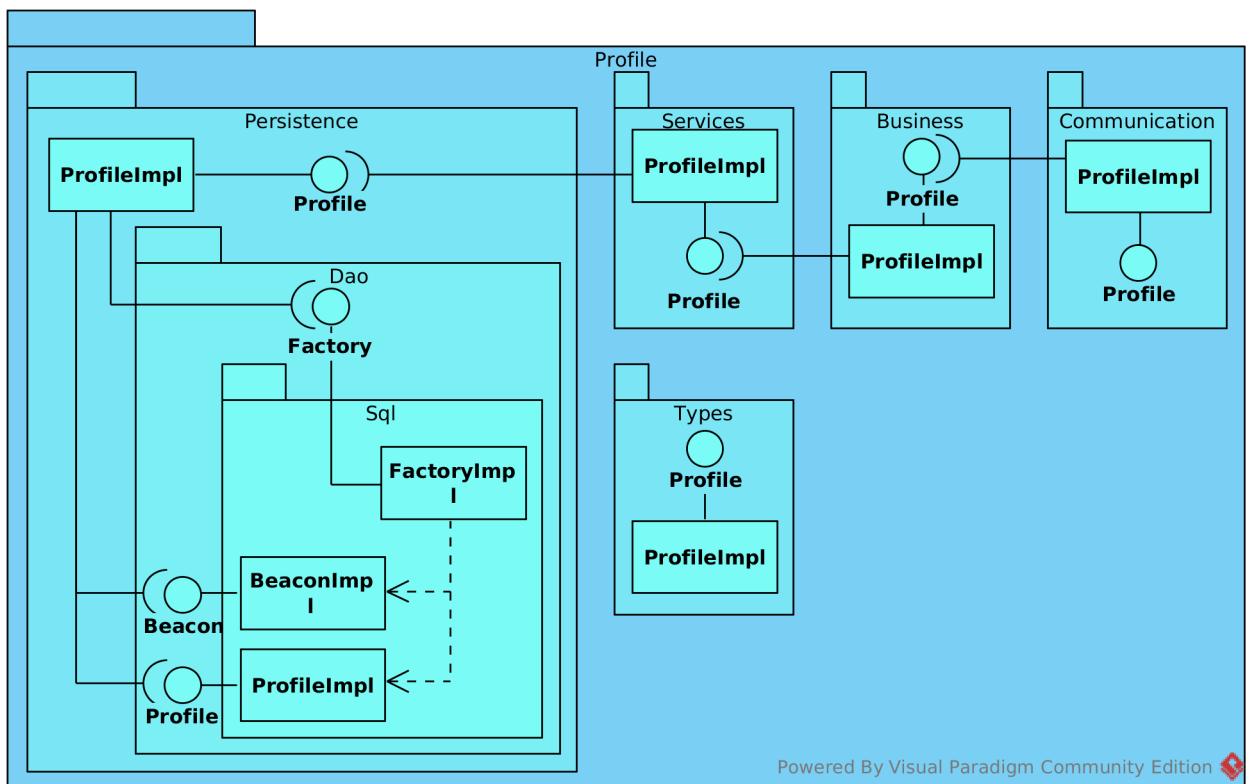


Figura 19: Server::Microservices::Profile

- **Descrizione**

Package che contiene le componenti che realizzano il micro servizio che si occupa della gestione dei profili degli utenti.

- **Padre: Microservices**

- **Package contenuti:**

- **Business**

Package che contiene la logica di business.

- **Communication**

Package che contiene i componenti che hanno il compito di: esporre API del micro servizio; inviare delle informazioni ad altri componenti del sistema.



5 Componenti e classi

- **Persistence**
Package che contiene le componenti che realizzano lo strato di persistenza dei dati.
- **Services**
Package che contiene le componenti che realizzano lo strato di servizio per l'accesso ai dati.
- **Types**
Package contenente i tipi.

5.38 Server::Microservices::Profile::Business

5.38.1 Informazioni generali

- **Descrizione**
Package che contiene la logica di business.
- **Padre: Profile**

5.38.2 Classi

5.38.2.1 Server::Microservices::Profile::Business::ProfileBusiness

- **Descrizione**
Interfaccia che gestisce la logica di funzionamento del profilo, quindi risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il profilo dell'utente;
- **Utilizzo**
Viene utilizzata per gestire la logica del profilo dell'utente e gestisce quindi la risposta alle richieste provenienti dall'esterno;
- **Sottoclassi:**
 - **ProfileBusinessImpl**
- **Relazioni con altre classi:**
 - **IN ProfileCommunication**
Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al profilo dell'utente, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;
 - **OUT ProfileServices**
Interfaccia che gestisce i dati relativi al profilo dell'utente;
 - **OUT Beacon**
Interfaccia che gestisce le informazioni associate ad un beacon;
 - **OUT Profile**
Interfaccia che gestisce le informazioni che rappresentano un profilo;

5.38.2.2 Server::Microservices::Profile::Business::ProfileBusinessImpl

- **Descrizione**
Classe che gestisce la logica di funzionamento del servizio e risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il profilo dell'utente;
- **Utilizzo**
Implementa i metodi offerti dall'interfaccia ProfileBusiness, utilizzati per gestire la logica del profilo dell'utente;
- **Classi ereditate:**
 - **ProfileBusiness**



5 Componenti e classi

5.39 Server::Microservices::Profile::Communication

5.39.1 Informazioni generali

- **Descrizione**

Package che contiene i componenti che hanno il compito di: esporre API del micro servizio; inviare delle informazioni ad altri componenti del sistema.

- **Padre: Profile**

5.39.2 Classi

5.39.2.1 Server::Microservices::Profile::Communication::ProfileCommunication

- **Descrizione**

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al profilo dell'utente, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- **Utilizzo**

Viene utilizzata per gestire la comunicazione tra server e client riguardante le funzionalità legate al profilo dell'utente;

- **Sottoclassi:**

- `ProfileCommunicationImpl`

- **Relazioni con altre classi:**

- *IN ProfileCommunication*

Gestisce tutta la comunicazione con il server riguardante la gestione del profilo dell'utente;

- *OUT ProfileBusiness*

Interfaccia che gestisce la logica di funzionamento del profilo, quindi risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il profilo dell'utente;

- *OUT Beacon*

Interfaccia che gestisce le informazioni associate ad un beacon;

- *OUT Profile*

Interfaccia che gestisce le informazioni che rappresentano un profilo;

5.39.2.2 Server::Microservices::Profile::Communication::ProfileCommunicationImpl

- **Descrizione**

Classe che gestisce la comunicazione tra server e client riguardante le funzionalità legate al profilo dell'utente, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia `ProfileCommunication`, utilizzati per gestire la comunicazione tra server e client riguardante le funzionalità legate al profilo dell'utente;

- **Classi ereditate:**

- `ProfileCommunication`



5 Componenti e classi

5.40 Server::Microservices::Profile::Persistence

5.40.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano lo strato di persistenza dei dati.

- **Padre: Profile**

- **Package contenuti:**

- **Dao**

Package che contiene le componenti che realizzano il DAO.

5.40.2 Classi

5.40.2.1 Server::Microservices::Profile::Persistence::ProfilePersistence

- **Descrizione**

Interfaccia che gestisce i dati elementari relativi al profilo dell'utente, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **Utilizzo**

Viene utilizzata per interfacciarsi e comunicare direttamente con il database utilizzato;

- **Sottoclassi:**

- **ProfilePersistenceImpl**

- **Relazioni con altre classi:**

- **IN ProfileServices**

Interfaccia che gestisce i dati relativi al profilo dell'utente;

- **OUT BeaconDao**

Gestisce il salvataggio e il recupero di informazioni relative ad un beacon;

- **OUT DaoFactory**

Fornisce una interfaccia per l'accesso ai dati elementari legati al profilo, indipendente dal linguaggio usato per memorizzarli;

- **OUT ProfileDao**

Gestisce il salvataggio e il recupero di informazioni relative ad un profilo;

5.40.2.2 Server::Microservices::Profile::Persistence::ProfilePersistenceImpl

- **Descrizione**

Classe che gestisce i dati elementari relativi al profilo dell'utente, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia ProfilePersistence, utilizzati per interfacciarsi e comunicare direttamente con il database utilizzato;

- **Classi ereditate:**

- **ProfilePersistence**



5 Componenti e classi

5.41 Server::Microservices::Profile::Persistence::Dao

5.41.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano il DAO.

- **Padre: Persistence**

- **Package contenuti:**

- **Sql**

Package che contiene le componenti che realizzano la parte di DAO relativa ai database relazionali.

5.41.2 Classi

5.41.2.1 Server::Microservices::Profile::Persistence::Dao::BeaconDao

- **Descrizione**

Gestisce il salvataggio e il recupero di informazioni relative ad un beacon;

- **Utilizzo**

Viene usata per esporre le possibili operazioni relative ad un beacon;

- **Sottoclassi:**

- **SqlBeaconDaoImpl**

- **Relazioni con altre classi:**

- **IN SqlDaoFactoryImpl**

Gestisce le funzionalità per la creazione di specifici oggetti che manipolano i dati relativi al profilo;

- **IN ProfilePersistence**

Interfaccia che gestisce i dati elementari relativi al profilo dell'utente, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **OUT Beacon**

Interfaccia che gestisce le informazioni associate ad un beacon;

5.41.2.2 Server::Microservices::Profile::Persistence::Dao::DaoFactory

- **Descrizione**

Fornisce una interfaccia per l'accesso ai dati elementari legati al profilo, indipendente dal linguaggio usato per memorizzarli;

- **Utilizzo**

Viene usata per creare la tipologia di factory richiesta e per fornire i dao relativi a tale tipologia;

- **Sottoclassi:**

- **SqlDaoFactoryImpl**

- **Relazioni con altre classi:**

- **IN ProfilePersistence**

Interfaccia che gestisce i dati elementari relativi al profilo dell'utente, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **OUT SqlDaoFactoryImpl**

Gestisce le funzionalità per la creazione di specifici oggetti che manipolano i dati relativi al profilo;



5 Componenti e classi

5.41.2.3 Server::Microservices::Profile::Persistence::Dao::ProfileDao

- **Descrizione**

Gestisce il salvataggio e il recupero di informazioni relative ad un profilo;

- **Utilizzo**

Viene usata per esporre le possibili operazioni relative ad un profilo;

- **Sottoclassi:**

- SqlProfileDaoImpl

- **Relazioni con altre classi:**

- *IN* SqlDaoFactoryImpl

Gestisce le funzionalità per la creazione di specifici oggetti che manipolano i dati relativi al profilo;

- *IN* ProfilePersistence

Interfaccia che gestisce i dati elementari relativi al profilo dell’utente, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- *OUT* Beacon

Interfaccia che gestisce le informazioni associate ad un beacon;

- *OUT* Profile

Interfaccia che gestisce le informazioni che rappresentano un profilo;

5.42 Server::Microservices::Profile::Persistence::Dao::Sql

5.42.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano la parte di DAO relativa ai database relazionali.

- **Padre: Dao**

5.42.2 Classi

5.42.2.1 Server::Microservices::Profile::Persistence::Dao::Sql::SqlBeaconDaoImpl

- **Descrizione**

Gestisce il salvataggio e il recupero di informazioni relative ad un beacon mediante SQL;

- **Utilizzo**

Viene usata per salvare e recuperare le informazioni relative ad un beacon mediante SQL;

- **Classi ereditate:**

- BeaconDao

5.42.2.2 Server::Microservices::Profile::Persistence::Dao::Sql::SqlDaoFactoryImpl

- **Descrizione**

Gestisce le funzionalità per la creazione di specifici oggetti che manipolano i dati relativi al profilo;

- **Utilizzo**

Viene usata per creare specifici oggetti che manipolano i dati, in un database di tipo sql, relativi al profilo;

- **Classi ereditate:**



5 Componenti e classi

- DaoFactory
- Relazioni con altre classi:
 - IN DaoFactory
Fornisce una interfaccia per l'accesso ai dati elementari legati al profilo, indipendente dal linguaggio usato per memorizzarli;
 - OUT BeaconDao
Gestisce il salvataggio e il recupero di informazioni relative ad un beacon;
 - OUT ProfileDao
Gestisce il salvataggio e il recupero di informazioni relative ad un profilo;

5.42.2.3 Server::Microservices::Profile::Persistence::Dao::Sql::SqlProfileDaoImpl

- Descrizione
Gestisce il salvataggio e il recupero di informazioni relative ad un profilo mediante SQL;
- Utilizzo
Viene usata per salvare e recuperare le informazioni relative ad un profilo mediante SQL;
- Classi ereditate:
 - ProfileDao

5.43 Server::Microservices::Profile::Services

5.43.1 Informazioni generali

- Descrizione
Package che contiene le componenti che realizzano lo strato di servizio per l'accesso ai dati.
- Padre: Profile

5.43.2 Classi

5.43.2.1 Server::Microservices::Profile::Services::ProfileServices

- Descrizione
Interfaccia che gestisce i dati relativi al profilo dell'utente;
- Utilizzo
Viene utilizzata per raggruppare e suddividere in maniera logica le componenti grezze fornite dallo strato sottostante (Persistence) e renderle quindi utili per la parte di Business;
- Sottoclassi:
 - ProfileServicesImpl
- Relazioni con altre classi:
 - IN ProfileBusiness
Interfaccia che gestisce la logica di funzionamento del profilo, quindi risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il profilo dell'utente;
 - OUT ProfilePersistence
Interfaccia che gestisce i dati elementari relativi al profilo dell'utente, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;



5 Componenti e classi

- *OUT Beacon*
Interfaccia che gestisce le informazioni associate ad un beacon;
- *OUT Profile*
Interfaccia che gestisce le informazioni che rappresentano un profilo;

5.43.2.2 Server::Microservices::Profile::Services::ProfileServicesImpl

- **Descrizione**
Classe che gestisce i dati relativi al profilo dell’utente;
- **Utilizzo**
Implementa i metodi offerti dall’interfaccia ProfileServices, utilizzati per raggruppare e suddividere in maniera logica le componenti grezze fornite dallo strato sottostante (Persistence) e renderle quindi utili per la parte di Business;
- **Classi ereditate:**
 - ProfileServices

5.44 Server::Microservices::Profile::Types

5.44.1 Informazioni generali

- **Descrizione**
Package contenente i tipi.
- **Padre: Profile**

5.44.2 Classi

5.44.2.1 Server::Microservices::Profile::Types::Beacon

- **Descrizione**
Interfaccia che gestisce le informazioni associate ad un beacon;
- **Utilizzo**
Viene utilizzata per gestire le informazioni che rappresentano un beacon;
- **Sottoclassi:**
 - BeaconImpl
- **Relazioni con altre classi:**
 - *IN ProfileBusiness*
Interfaccia che gestisce la logica di funzionamento del profilo, quindi risponde agli input e alle richieste che arrivano dall’esterno per quanto riguarda il profilo dell’utente;
 - *IN ProfileCommunication*
Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al profilo dell’utente, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;
 - *IN BeaconDao*
Gestisce il salvataggio e il recupero di informazioni relative ad un beacon;
 - *IN ProfileDao*
Gestisce il salvataggio e il recupero di informazioni relative ad un profilo;
 - *IN ProfileServices*
Interfaccia che gestisce i dati relativi al profilo dell’utente;



5 Componenti e classi

5.44.2.2 Server::Microservices::Profile::Types::BeaconImpl

- **Descrizione**

Classe che gestisce la logica di funzionamento riguardanti i beacon;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Beacon, utilizzati per gestire i dispositivi beacon;

- **Classi ereditate:**

 - Beacon

5.44.2.3 Server::Microservices::Profile::Types::Profile

- **Descrizione**

Interfaccia che gestisce le informazioni che rappresentano un profilo;

- **Utilizzo**

Viene utilizzata per gestire le informazioni che rappresentano un profilo;

- **Sottoclassi:**

 - ProfileImpl

- **Relazioni con altre classi:**

 - *IN ProfileBusiness*

Interfaccia che gestisce la logica di funzionamento del profilo, quindi risponde agli input e alle richieste che arrivano dall'esterno per quanto riguarda il profilo dell’utente;

 - *IN ProfileCommunication*

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate al profilo dell’utente, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

 - *IN ProfileDao*

Gestisce il salvataggio e il recupero di informazioni relative ad un profilo;

 - *IN ProfileServices*

Interfaccia che gestisce i dati relativi al profilo dell’utente;

5.44.2.4 Server::Microservices::Profile::Types::ProfileImpl

- **Descrizione**

Classe che gestisce le informazioni che rappresentano un profilo di un utente;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia Profile, utilizzati per la gestione del profilo di un utente;

- **Classi ereditate:**

 - Profile



5.45 Server::Microservices::Session

5.45.1 Informazioni generali

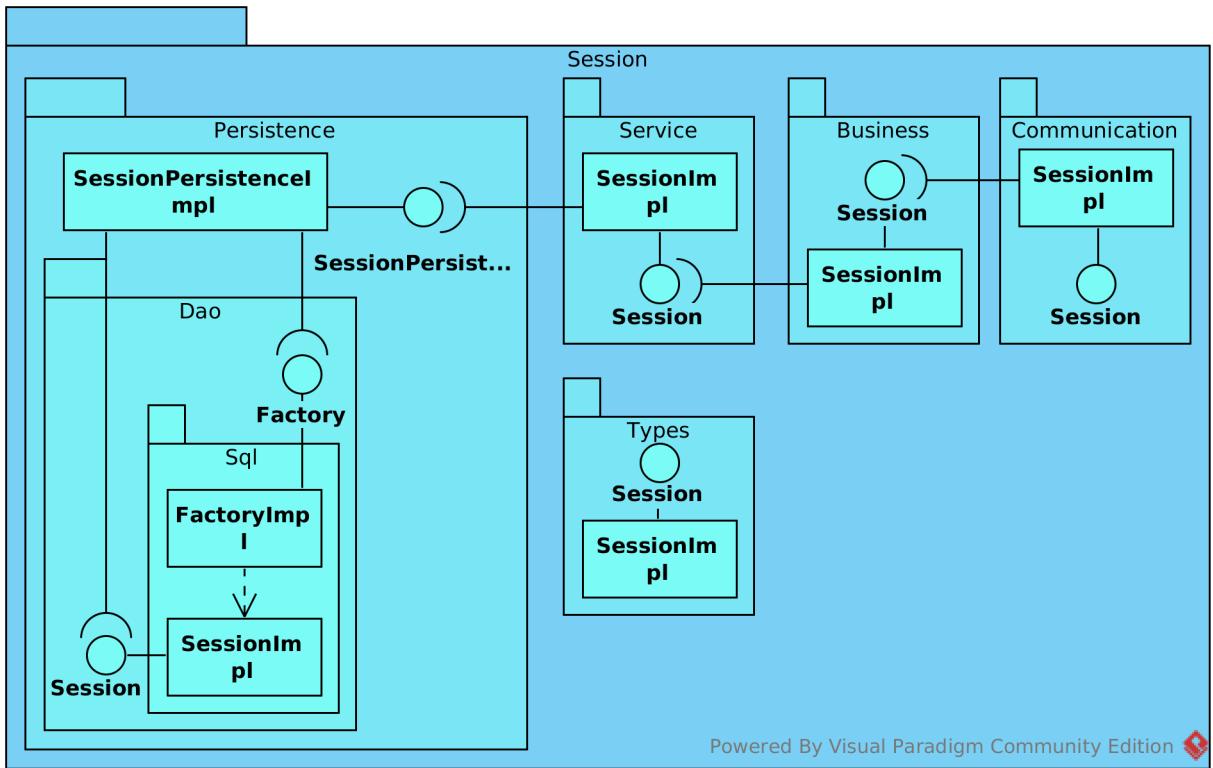


Figura 20: Server::Microservices::Session

- **Descrizione**

Package che contiene le componenti che realizzano il micro servizio che si occupa della gestione delle sessioni.

- **Padre: Microservices**

- **Package contenuti:**

- **Persistence**

Package che contiene le componenti che realizzano lo strato di persistenza dei dati.

- **Communication**

Package che contiene i componenti che hanno il compito di: esporre API del micro servizio; inviare delle informazioni ad altri componenti del sistema.

- **Business**

Package che contiene le componenti che realizzano lo strato di servizio per l'accesso ai dati.

- **Services**

Package che contiene le componenti che realizzano lo strato di servizio per l'accesso ai dati.

- **Types**

Package contenente i tipi.



5 Componenti e classi

5.46 Server::Microservices::Session::Business

5.46.1 Informazioni generali

- **Descrizione**

Package che contiene la logica di business.

- **Padre: Session**

5.46.2 Classi

5.46.2.1 Server::Microservices::Session::Business::SessionBusiness

- **Descrizione**

Interfaccia che gestisce la logica di funzionamento delle sessioni, quindi risponde agli input e alle richieste che arrivano dall'esterno per l'inizio o la distruzione delle sessioni;

- **Utilizzo**

Viene utilizzata per gestire la logica delle sessioni, quindi risponde agli input e alle richieste che arrivano dall'esterno per l'inizio o la distruzione delle sessioni;

- **Sottoclassi:**

- SessionBusinessImpl

- **Relazioni con altre classi:**

- *IN SessionCommunication*

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate alle sessioni, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- *OUT SessionServices*

Interfaccia che gestisce i dati relativi la sessione di un utente;

- *OUT Session*

Interfaccia che gestisce i dati e le procedure che identificano una sessione;

5.46.2.2 Server::Microservices::Session::Business::SessionBusinessImpl

- **Descrizione**

Classe che gestisce la logica di funzionamento delle sessioni, quindi risponde agli input e alle richieste che arrivano dall'esterno per l'inizio o la distruzione delle sessioni;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia SessionBusiness, utilizzati per gestire la logica di funzionamento delle sessioni;

- **Classi ereditate:**

- SessionBusiness

5.47 Server::Microservices::Session::Communication

5.47.1 Informazioni generali

- **Descrizione**

Package che contiene i componenti che hanno il compito di: esporre API del micro servizio; inviare delle informazioni ad altri componenti del sistema.

- **Padre: Session**



5 Componenti e classi

5.47.2 Classi

5.47.2.1 Server::Microservices::Session::Communication::SessionCommunication

- **Descrizione**

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate alle sessioni, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- **Utilizzo**

Viene utilizzata per gestire la comunicazione tra server e client riguardante le funzionalità legate alle sessioni;

- **Sottoclassi:**

- SessionCommunicationImpl

- **Relazioni con altre classi:**

- *IN* EnviromentCommunication

Gestisce la comunicazione tra client e server per le funzionalità di base;

- *OUT* SessionBusiness

Interfaccia che gestisce la logica di funzionamento delle sessioni, quindi risponde agli input e alle richieste che arrivano dall'esterno per l'inizio o la distruzione delle sessioni;

- *OUT* Session

Interfaccia che gestisce i dati e le procedure che identificano una sessione;

5.47.2.2 Server::Microservices::Session::Communication::SessionCommunicationImpl

- **Descrizione**

Classe che gestisce la comunicazione tra server e client riguardante le funzionalità legate alle sessioni, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia SessionCommunication, utilizzati per gestire la comunicazione tra server e client riguardante le funzionalità legate alle sessioni;

- **Classi ereditate:**

- SessionCommunication

5.48 Server::Microservices::Session::Persistence

5.48.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano lo strato di persistenza dei dati.

- **Padre: Session**

- **Package contenuti:**

- Dao

Package che contiene le componenti che realizzano il DAO.



5 Componenti e classi

5.48.2 Classi

5.48.2.1 Server::Microservices::Session::Persistence::SessionPersistence

- **Descrizione**

Interfaccia che gestisce i dati elementari relativi alla sessione, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **Utilizzo**

Viene utilizzata per interfacciarsi e comunicare direttamente con il database utilizzato;

- **Sottoclassi:**

- SessionPersistenceImpl

- **Relazioni con altre classi:**

- *IN* SessionServices

Interfaccia che gestisce i dati relativi la sessione di un utente;

- *OUT* SessionDao

Gestisce tutte le funzionalità per il salvataggio e il reperimento di informazioni riguardanti una sessione;

- *OUT* SqlDaoFactoryImpl

Gestisce le funzionalità per l'accesso ai dati elementari legati alla sessione, manipolati tramite SQL;

5.48.2.2 Server::Microservices::Session::Persistence::SessionPersistenceImpl

- **Descrizione**

Classe che gestisce i dati elementari relativi alla sessione, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia SessionPersistence, utilizzati per interfacciarsi e comunicare direttamente con il database utilizzato;

- **Classi ereditate:**

- SessionPersistence

5.49 Server::Microservices::Session::Persistence::Dao

5.49.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano il DAO.

- **Padre: Persistence**

- **Package contenuti:**

- Sql

Package che contiene le componenti che realizzano la parte di DAO relativa ai database relazionali.

5.49.2 Classi

5.49.2.1 Server::Microservices::Session::Persistence::Dao::DaoFactory

- **Descrizione**

Fornisce una interfaccia per l'accesso ai dati elementari legati alla sessione, indipendentemente dal linguaggio usato per memorizzarli;



5 Componenti e classi

- **Utilizzo**

Viene usata per creare la tipologia di factory richiesta e per fornire i dao relativi a tale tipologia;

- **Sottoclassi:**

- SqlDaoFactoryImpl

5.49.2.2 Server::Microservices::Session::Persistence::Dao::SessionDao

- **Descrizione**

Gestisce tutte le funzionalità per il salvataggio e il reperimento di informazioni riguardanti una sessione;

- **Utilizzo**

Viene usata per esporre le possibili operazioni relative ad una sessione;

- **Sottoclassi:**

- SqlSessionDaoImpl

- **Relazioni con altre classi:**

- *IN* SqlDaoFactoryImpl

Gestisce le funzionalità per l'accesso ai dati elementari legati alla sessione, manipolati tramite SQL;

- *IN* SessionPersistence

Interfaccia che gestisce i dati elementari relativi alla sessione, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- *OUT* Session

Interfaccia che gestisce i dati e le procedure che identificano una sessione;

5.50 Server::Microservices::Session::Persistence::Dao::Sql

5.50.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano la parte di DAO relativa ai database relazionali.

- **Padre: Dao**

5.50.2 Classi

5.50.2.1 Server::Microservices::Session::Persistence::Dao::Sql::SqlDaoFactoryImpl

- **Descrizione**

Gestisce le funzionalità per l'accesso ai dati elementari legati alla sessione, manipolati tramite SQL;

- **Utilizzo**

Viene usata per creare specifici oggetti che manipolano i dati, in un database di tipo sql, relativi alla sessione;

- **Classi ereditate:**

- DaoFactory

- **Relazioni con altre classi:**

- *IN* SessionPersistence

Interfaccia che gestisce i dati elementari relativi alla sessione, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;



5 Componenti e classi

- *OUT SessionDao*

Gestisce tutte le funzionalità per il salvataggio e il reperimento di informazioni riguardanti una sessione;

5.50.2.2 Server::Microservices::Session::Persistence::Dao::Sql::SqlSessionDaoImpl

- **Descrizione**

Gestisce il salvataggio e il recupero di informazioni relative ad una sessione mediante SQL;

- **Utilizzo**

Viene usata per salvare e recuperare le informazioni relative ad una sessione mediante SQL;

- **Classi ereditate:**

- *SessionDao*

5.51 Server::Microservices::Session::Services

5.51.1 Informazioni generali

- **Descrizione**

Package che contiene le componenti che realizzano lo strato di servizio per l'accesso ai dati.

- **Padre: Session**

5.51.2 Classi

5.51.2.1 Server::Microservices::Session::Services::SessionServices

- **Descrizione**

Interfaccia che gestisce i dati relativi la sessione di un utente;

- **Utilizzo**

Viene utilizzata per raggruppare e suddividere in maniera logica le componenti grezze fornite dallo strato sottostante (Persistence) e renderle quindi utili per la parte di Business;

- **Sottoclassi:**

- *SessionServicesImpl*

- **Relazioni con altre classi:**

- *IN SessionBusiness*

Interfaccia che gestisce la logica di funzionamento delle sessioni, quindi risponde agli input e alle richieste che arrivano dall'esterno per l'inizio o la distruzione delle sessioni;

- *OUT SessionPersistence*

Interfaccia che gestisce i dati elementari relativi alla sessione, quindi ha il compito di interfacciarsi e comunicare direttamente con il database utilizzato;

- *OUT Session*

Interfaccia che gestisce i dati e le procedure che identificano una sessione;



5 Componenti e classi

5.51.2.2 Server::Microservices::Session::Services::SessionServiceImpl

- **Descrizione**

Classe che gestisce i dati relativi la sessione di un utente;

- **Utilizzo**

Implementa i metodi offerti dall’interfaccia SessionServices, utilizzati per raggruppare e suddividere in maniera logica le componenti grezze fornite dallo strato sottostante (Persistence) e renderle quindi utili per la parte di Business;

- **Classi ereditate:**

 - SessionServices

5.52 Server::Microservices::Session::Types

5.52.1 Informazioni generali

- **Descrizione**

Package contenente i tipi.

- **Padre: Session**

5.52.2 Classi

5.52.2.1 Server::Microservices::Session::Types::Error

- **Descrizione**

Interfaccia che permette la creazione degli errori;

- **Utilizzo**

Viene usata per esporre i metodi che deve avere un classe di tipo Error;

- **Sottoclassi:**

 - ErrorImpl

5.52.2.2 Server::Microservices::Session::Types::ErrorConstants

- **Descrizione**

Classe che contiene le tipologie di errori;

- **Utilizzo**

Le classi che si occupano della gestione di errori fanno riferimento a questa classe per specificare la tipologia di errore;

5.52.2.3 Server::Microservices::Session::Types::ErrorImpl

- **Descrizione**

Classe che rappresenta un errore;

- **Utilizzo**

Viene usata per segnalare il fallimento di un operazione richiesta;

- **Classi ereditate:**

 - Error



5 Componenti e classi

5.52.2.4 Server::Microservices::Session::Types::Session

- **Descrizione**

Interfaccia che gestisce i dati e le procedure che identificano una sessione;

- **Utilizzo**

Viene utilizzata per gestire i dati e le procedure che identificano una sessione;

- **Sottoclassi:**

- SessionImpl

- **Relazioni con altre classi:**

- *IN SessionBusiness*

Interfaccia che gestisce la logica di funzionamento delle sessioni, quindi risponde agli input e alle richieste che arrivano dall'esterno per l'inizio o la distruzione delle sessioni;

- *IN SessionCommunication*

Interfaccia che gestisce la comunicazione tra server e client riguardante le funzionalità legate alle sessioni, quindi rimane in ascolto delle richieste che arrivano dal esterno del sistema server;

- *IN SessionDao*

Gestisce tutte le funzionalità per il salvataggio e il reperimento di informazioni riguardanti una sessione;

- *IN SessionServices*

Interfaccia che gestisce i dati relativi la sessione di un utente;

5.52.2.5 Server::Microservices::Session::Types::SessionImpl

- **Descrizione**

Classe che gestisce i dati e le procedure che identificano una sessione;

- **Utilizzo**

Implementa i metodi offerti dall'interfaccia Session, utilizzati per la gestione dei dati e le procedure che identificano una sessione;

- **Classi ereditate:**

- Session



6 Design Pattern

6 Design Pattern

I Design Pattern_G sono delle soluzioni progettuali a problemi ricorrenti. Essi semplificano di molto la progettazione e rendono possibile il riuso del codice, facendo diventare più manutenibile l'intero progetto. Diversi problemi da risolvere danno vita a diverse tipologie di Design Pattern_G che sono:

- **Pattern architetturali:** sono pattern che operano ad alto livello e gettano le basi per impostare l'organizzazione strutturale di un sistema.
- **Pattern creazionali:** la loro funzione è quella di nascondere i costruttori delle classi così facendo è possibile costruire oggetti senza sapere come essi siano implementati. Dei metodi vengono messi a disposizione per la creazione degli oggetti.
- **Pattern strutturali:** consentono il riutilizzo degli oggetti esistenti fornendo agli utilizzatori delle interfacce più adatte alle loro esigenze.
- **Pattern comportamentali:** forniscono delle soluzioni a problemi ricorrenti di interazione tra classi.



6 Design Pattern

6.1 Design Pattern Architetturali

6.1.1 Microservices Architecture

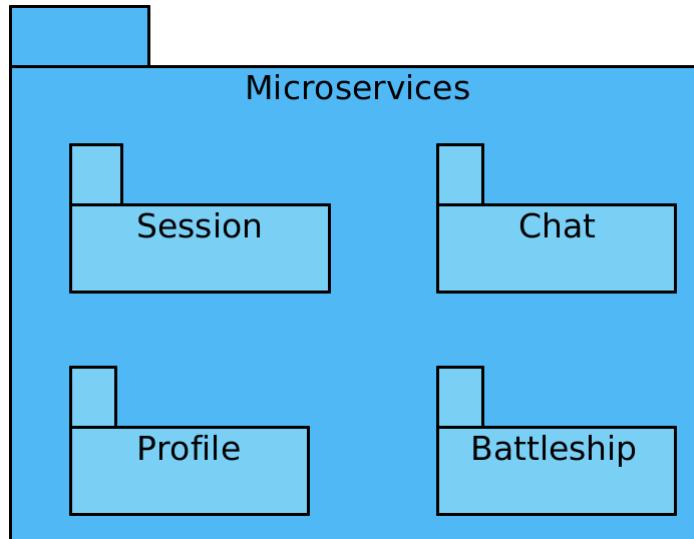


Figura 21: Diagramma della Microservices Architecture applicata al progetto CLIPS

- **Scopo d'utilizzo:** per rendere ottima la suddivisione delle componenti e la loro distribuzione è stato deciso di adottare la Microservice Architecture, in questo modo è stata aumentata anche la scalabilità del progetto. Così facendo c'è una netta suddivisione tra front-end_G, back-end_G e database_G, che rende molto più adatto lo sviluppo di una applicazione mobile.
- **Contesto d'utilizzo:** abbiamo deciso che vista la presenza di diversi servizi messi a disposizione dall'applicazione, la scelta che soddisfaceva al meglio le nostre esigenze fosse la Microservice Architecture. Per coadiuvare il tutto è stato scelto uno stile architettonico REST_G per gestire le comunicazioni tra front-end_G e back-end_G tramite richieste HTTP_G. Essendo la nostra applicazione dotata di servizi indipendenti l'uno dall'altro quali la battaglia navale o la chat, la scelta della Microservice Architecture è sembrata la più logica poiché permette di aggiungere o rimuovere servizi a piacimento senza intaccare l'integrità e la funzionalità dell'intero software. Ciò facendo abbiamo garantito anche un certo grado di modularità e scalabilità. In caso di malfunzionamento di un modulo sarà quindi possibile usufruire lo stesso del software eccetto il modulo difettoso. Inoltre così il software è più manutenibile. Adottando questa architettura è stato possibile affidare lo sviluppo di funzionalità a membri differenti del team, senza che essi fossero dipendenti dallo sviluppo di altre parti assegnate ad altri membri.



6 Design Pattern

6.1.2 Model View Presenter

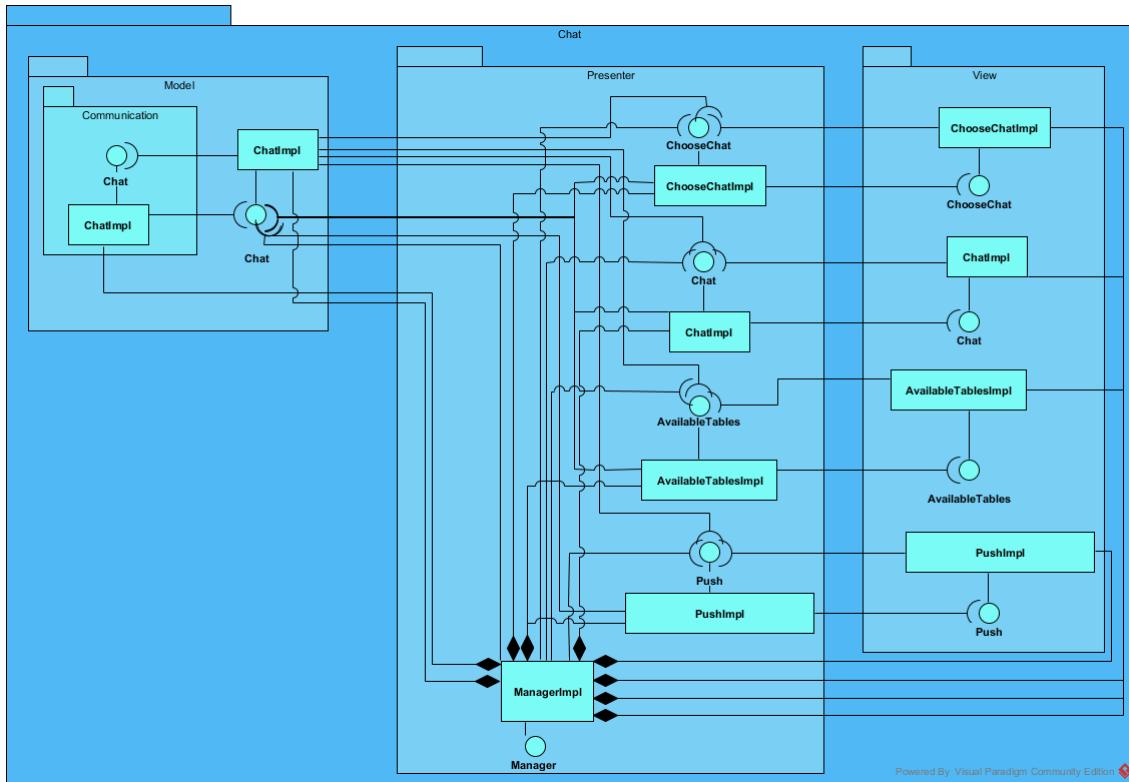


Figura 22: Pattern MVP applicato al progetto CLIPS

- **Scopo d'utilizzo:** per gestire al meglio il lato client dell'applicazione, avevamo bisogno di un modello architettonale che permettesse la separazione dei componenti che presentano i dati dai dati stessi. Trattandosi di un'applicazione che visualizza dati costantemente in aggiornamento è necessario visualizzare sempre i dati aggiornati.
- **Contesto d'utilizzo:** il pattern MVP verrà applicato ai servizi messi a disposizione dall'applicazione dal lato front-end, i quali avranno bisogno di mantenere aggiornati i dati visualizzati. Per fare ciò il meccanismo di aggiornamento delle View andrà a prelevare i dati aggiornati dal Model tramite i meccanismi messi a disposizione del Presenter.



6.2 Design Pattern Creazionali

6.2.1 Singleton

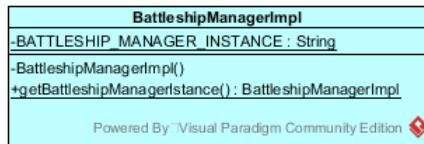


Figura 23: Pattern Abstract Factory applicato al progetto CLIPS

- **Scopo d'utilizzo:** permette di assicurare che l'istanza di una classe sia univoca all'interno del programma stesso e per fornire la possibilità di accesso globale a tutte le altri componenti che ne necessitino l'utilizzo.
- **Contesto d'utilizzo:** il seguente Design Pattern viene utilizzato sia da alcune componenti del Server, sia da elementi del Client. In particolare, nella parte Client, il suo utilizzo risulta molto utile per assicurare l'esistenza di una sola istanza di BattleshipManager, ovvero la classe responsabile di gestire tutto il flusso di esecuzione di una partita a Battaglia navale.

6.2.2 Abstract Factory

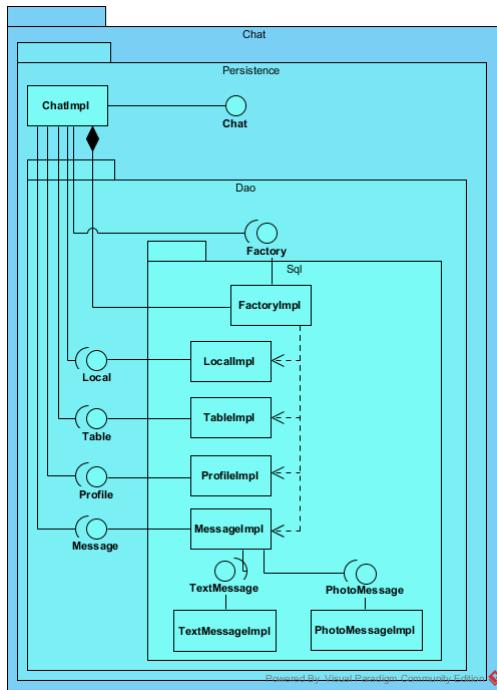


Figura 24: Pattern Abstract Factory applicato al progetto CLIPS

- **Scopo d'utilizzo:** fornisce un'interfaccia per creare famiglie di prodotti senza specificare classi concrete. Nasconde i nomi delle classi concrete e permette al client di lavorare unicamente con interfacce.
- **Contesto d'utilizzo:** vengono utilizzati per ciascun MicroServices, due livelli di Factory; il primo permette di essere indipendenti dal tipo di database mentre il secondo crea gli oggetti che rappresentano i record del database stesso.



6.3 Design Pattern Strutturali

6.3.1 Adapter Pattern

- **Scopo d'utilizzo:** permette che classi inizialmente incompatibili lavorino insieme convertendo l'interfaccia di una classe in un' interfaccia che si aspetta il client;
- **Contesto d'utilizzo:** viene utilizzato per adattare il tipo AltBeacon all'interfaccia Beacon.

6.3.2 Data Access Object DAO

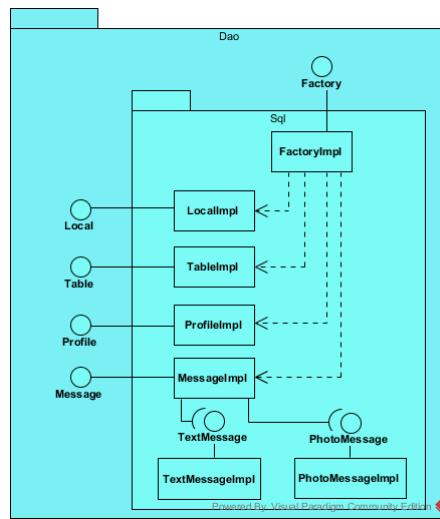


Figura 25: DAO applicato al progetto CLIPS

- **Scopo d'utilizzo:** disaccoppia la logica di business dalla logica di accesso ai dati; questo approccio garantisce che un eventuale cambiamento della base di dati non comporti modifiche sui componenti di business.
- **Contesto d'utilizzo:** viene utilizzato un approccio DAO lato server, all'interno di ciascun package Dao. Viene così disaccoppiata la logica di business dalla logica di accesso ai dati. In pratica si tratta di classi che rappresentano una specifica tabella nel database e ciò ci permette di isolare l'accesso a una tabella tramite query, inoltre aumenta la sua manutenibilità. Queste classi si occupano di fare da intermediario tra il database e l'applicazione occupandosi di inserire informazioni nel database o prelevarle per mandarle agli oggetti del software.



6.3.3 Facade

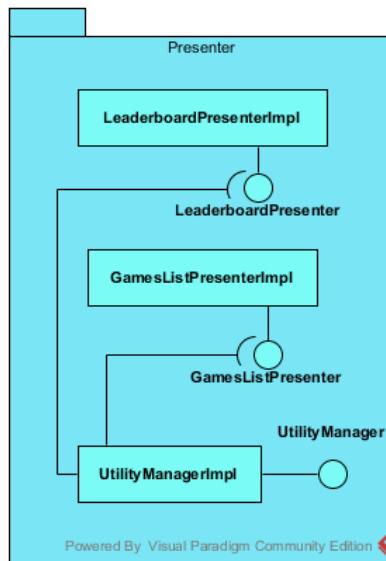


Figura 26: Facade applicato al progetto CLIPS

- **Scopo d'utilizzo:** fornisce un' interfaccia unica e semplice per un sottosistema complesso.
- **Contesto d'utilizzo:** le classi Manager all'interno dei package Presenter, risultano essere dei Facade per il sottosistema formato dalle restanti classi presenti nel package.



6.4 Pattern comportamentali

6.4.1 Observer Pattern

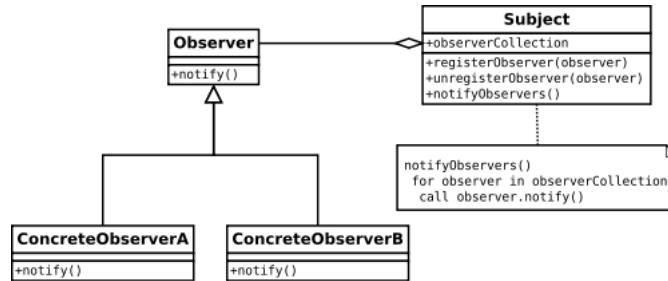


Figura 27: Pattern Abstract Factory applicato al progetto CLIPS

- **Scopo d'utilizzo:** è utilizzato nel caso in cui esista una relazione uno a molti tra oggetti, dove il cambiamento di un oggetto deve essere notificato agli altri oggetti dipendenti da esso.
- **Contesto d'utilizzo:** il suddetto design pattern viene utilizzato principalmente per permettere ai vari Client, in ascolto sullo stesso Server, di ricevere notifiche da quest'ultimo. In particolare, il Server grazie a questo design pattern può avvertire i vari Client designati di svariati eventi, ad esempio, un cambiamento nella composizione del Team, nel posizionamento di una nuova nave sul terreno di gioco o un nuovo colpo sparato dalla squadra avversaria. Sarà premura dei Client stessi poi, scaricare le nuove informazioni dal Server.



7 Stime di Fattibilità e Bisogno di Risorse

7 Stime di Fattibilità e Bisogno di Risorse

L'architettura definita precedentemente ha raggiunto un livello di dettaglio sufficiente per fornire una stima di fattibilità e di bisogno di risorse.

L'analisi dell'architettura progettata ha permesso di constatare che le tecnologie che si è scelto di adottare risultino sufficientemente adeguate per la realizzazione del prodotto e riescono a ricoprire le esigenze progettuali.

Gli strumenti scelti sono stati studiati dai componenti del team, la conoscenza generale è attualmente discreta ma si impegneranno ad approfondire le loro conoscenze soprattutto per quanto riguarda l'ambito Android_G e framework Play.

Gli strumenti utilizzati dal team sono:

- IntelliJ_G per la stesura del codice lato server;
- Android Studio per la stesura del codice lato client;
- Il framework Play! per la gestione del server e della comunicazione REST.

Tali strumenti potranno garantire una realizzazione efficace di tutti gli aspetti architetturali dell'applicazione.



8 Tracciamento

8.1 Tracciamento Classi-Requisiti

Classe	Requisiti
Client::BaseFunctions::Model::Communication::- Eviroment	RVO9
Client::BaseFunctions::Model::- Enviroment	RVO9
Client::BaseFunctions::Model::- EnviromentService	RVO9
Client::BaseFunctions::Presenter::- Enviroment	RFD1.3.1 RFD1.4.1 RFO1.5.1
Client::BaseFunctions::Presenter::- MainMenu	RFO1 RFO1.2 RFD1.3 RFD1.4 RFO1.5
Client::BaseFunctions::Presenter::- Manager	RFO6
Client::BaseFunctions::Types::Beacon	RFO7
Client::BaseFunctions::Types::Errors::- BluetoothError	RFO7.1
Client::BaseFunctions::Types::Errors::- Error	RFD1.3.2 RFD1.4.2 RFO1.5.2
Client::BaseFunctions::View::- EnviromentError	RFD1.3.2 RFD1.4.2 RFO1.5.2
Client::BaseFunctions::View::MainMenu	RFO1 RFO1.1 RFO1.2 RFD1.4 RFO1.5
Client::Chat::Model::Chat	RFD4 RFD4.1 RFD4.2 RFD4.3
Client::Chat::Model::Communication	RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4



8 Tracciamento

Classe	Requisiti
Client::Chat::Presenter::- AvailableTables	RFD4.1 RFD4.1.1 RFD4.1.2
Client::Chat::Presenter::Chat	RFD4 RFD4.3 RFD4.3.2 RFD4.3.3 RFD4.4 RFD4.4.1
Client::Chat::Presenter::ChooseChat	RFO2.2.3 RFD4.1 RFD4.1.2 RFD4.1.3 RFD4.2 RFD4.2.2
Client::Chat::Presenter::Manager	RFD4
Client::Chat::Presenter::Push	RFD4.1 RFD4.1.1 RFD4.1.2
Client::Chat::View::AvailableTables	RFD4.1 RFD4.1.1 RFD4.1.2
Client::Chat::View::Chat	RFD4.3.1 RFD4.3.4 RFD4.4
Client::Chat::View::ChooseChat	RFD4.1.1 RFD4.1.4 RFD4.1.5 RFD4.1.6 RFD4.2.1 RFD4.2.4 RFD4.2.5
Client::Chat::View::Push	RFD4.3 RFD4.3.1 RFD4.3.2 RFD4.3.3
Client::Games::Battleship::Model::- Battleship	RFO5 RFO5.1 RFO5.2 RFO5.3
Client::Games::Battleship::Model::- Communication::Battleship	RFO5.1 RFO5.2.1 RFO5.3
Client::Games::Battleship::Presenter::- AttackPhase	RFO5.3.4 RFO5.3.4.1



8 Tracciamento

Classe	Requisiti
	RFO5.3.4.3 RFO5.3.4.4 RFO5.3.4.6
Client::Games::Battleship::Presenter::- ChooseRole	RFO5.1.1 RFO5.1.1.1 RFO5.1.1.2 RFO5.1.1.3 RFO5.1.1.4 RFO5.1.1.5
Client::Games::Battleship::Presenter::- CreateTeam	RFO5.1.1.2 RFO5.1.1.3 RFO5.1.1.4
Client::Games::Battleship::Presenter::- EnemyAttackPhase	RFO5.3.4 RFO5.3.4.1 RFO5.3.4.3 RFO5.3.4.4 RFO5.3.4.6
Client::Games::Battleship::Presenter::- FinalScore	RFO5.3.5 RFO5.3.5.1 RFO5.3.5.5 RFO5.3.5.6 RFO5.3.5.7 RFO5.3.5.8
Client::Games::Battleship::Presenter::- JoinTeam	RFO5.1.2 RFO5.1.2.1 RFO5.1.2.2 RFO5.1.2.3 RFO5.1.3 RFO5.1.3.1 RFO5.1.3.2
Client::Games::Battleship::Presenter::- Manager	RFO5 RFO5.1 RFO5.2 RFO5.3 RFO5.3.1 RFO5.3.2 RFD5.4
Client::Games::Battleship::Presenter::- SelectShipPosition	RFO5.3.3 RFO5.3.3.1 RFO5.3.3.3 RFO5.3.3.4 RFO5.3.3.6



8 Tracciamento

Classe	Requisiti
	RFO5.3.3.8 RFO5.3.3.9 RFO5.3.5.7
Client::Games::Battleship::Presenter::- TeamManagement	RFO5.1.4 RFO5.1.4.1 RFO5.1.4.2 RFO5.1.4.3 RFO5.1.4.4
Client::Games::Battleship::Types::Cell	RFO5.3.3.3 RFO5.3.4.2
Client::Games::Battleship::Types::- Field	RFO5.3.3.2 RFO5.3.4.3
Client::Games::Battleship::Types::Grid	RFO5.3 RFO5.3.1 RFO5.3.2 RFO5.3.3
Client::Games::Battleship::Types::- Position	RFO5.3.3.1 RFO5.3.3.2 RFO5.3.3.3 RFO5.3.3.4
Client::Games::Battleship::Types::Ship	RFO5.3.3 RFO5.3.3.1 RFO5.3.3.2
Client::Games::Battleship::Types::- Shoot	RFO5.1.4 RFO5.1.4.1 RFO5.1.4.2
Client::Games::Battleship::View::- AttackPhase	RFO5.3.4.2 RFO5.3.4.5
Client::Games::Battleship::View::- ChooseRole	RFO5.1.1
Client::Games::Battleship::View::- CreateTeam	RFO5.1.1.1 RFO5.1.1.5
Client::Games::Battleship::View::- EnemyAttackPhase	RFO5.3.4.2 RFO5.3.4.5
Client::Games::Battleship::View::- FinalScore	RFO5.3.3.4 RFO5.3.5.2 RFO5.3.5.3
Client::Games::Battleship::View::- JoinTeam	RFO5.1.2.1 RFO5.1.2.4



8 Tracciamento

Classe	Requisiti
Client::Games::Battleship::View::- SelectShipPosition	RFO5.3.3.2 RFO5.3.3.5 RFO5.3.3.7 RFO5.3.3.10
Client::Games::Battleship::View::- TeamManagement	RFO5.1.4.1 RFO5.1.4.5
Client::Games::Utility::Model::- GamesList	RFO5
Client::Games::Utility::Model::- Leaderboard	RFD5.4 RFD5.4.1
Client::Games::Utility::Presenter::- GamesList	RFO5
Client::Games::Utility::Presenter::- Leaderboard	RFD5.4 RFD5.4.1
Client::Games::Utility::Presenter::- Manager	RFO5
Client::Games::Utility::Presenter::- Manager	RFO5
Client::Games::Utility::Types::Score	RFD5.4
Client::Games::Utility::Types::Team	RFO5.1 RFO5.1.1 RFO5.1.2 RFO5.1.3 RFO5.1.4
Client::Games::Utility::View::- GamesList	RFO5
Client::Games::Utility::View::- Leaderboard	RFD5.4 RFD5.4.1
Client::Profile::Model::Communication	RFO1.1 RFO1.2
Client::Profile::Model::Profile	RFO1.1 RFO1.2 RFO2 RFO2.1 RFO2.2
Client::Profile::Presenter::Profile	RFO1.1 RFO2 RFO2.2 RFO2.2.2 RFO2.2.3
Client::Profile::View::Profile	RFO2.1 RFO2.2.1 RFO2.2.4



8 Tracciamento

Classe	Requisiti
	RFO2.2.5
Server::Microservices::Battleship::- Business::Battleship	RFO5 RFO5.3 RFO5.3.1 RFO5.3.2 RFO5.3.3
Server::Microservices::Battleship::- Communication::Battleship	RFO5.1 RFO5.2 RFO5.3
Server::Microservices::Battleship::- Manager::Loader	RFO5 RFO5.3
Server::Microservices::Battleship::- Persistence::Battleship	RFO5.3 RFO5.3.1 RFO5.3.2 RFO5.3.3
Server::Microservices::Battleship::- Services::Battleship	RFO5
Server::Microservices::Battleship::Types::- Cell	RFO5.3.3 RFO5.3.4
Server::Microservices::Battleship::Types::- Position	RFO5.3 RFO5.3.1 RFO5.3.2 RFO5.3.3
Server::Microservices::Battleship::Types::- Field	RFO5.3.3 RFO5.3.3.1 RFO5.3.3.2 RFO5.3.3.3 RFO5.3.3.4 RFO5.3.3.5 RFO5.3.3.6 RFO5.3.3.7 RFO5.3.3.8 RFO5.3.3.9 RFO5.3.3.10
Server::Microservices::Battleship::Types::- Ship	RFO5.3.3 RFO5.3.3.1 RFO5.3.3.2
Server::Microservices::Battleship::Types::- Shoot	RFO5.3.4 RFO5.3.4.1 RFO5.3.4.2



8 Tracciamento

Classe	Requisiti
	RFO5.3.4.3 RFO5.3.4.4 RFO5.3.4.5
Server::Microservices::Chat::Business::- Chat	RFD4 RFD4.1 RFD4.2 RFD4.3 RFD4.4
Server::Microservices::Chat::Communication::- Chat	RFD4
Server::Microservices::Chat::Manager::- Loader	RFD4
Server::Microservices::Chat::Persistence::- Dao::Sql::Factory	RFD4 RFD4.3
Server::Microservices::Chat::Persistence::- Dao::Sql::Local	RFO1 RFO1.1
Server::Microservices::Chat::Persistence::- Dao::Sql::Message	RFD4.3 RFD4.3.1 RFD4.3.2
Server::Microservices::Chat::Persistence::- Dao::Sql::Profile	RFO1.1 RFO1.2 RFO2
Server::Microservices::Chat::Persistence::- Dao::Sql::Table	RFO1 RFD4 RFO5
Server::Microservices::Chat::Persistence::- Dao::Sql::TextMessage	RFD4 RFD4.3
Server::Microservices::Chat::Services::- Chat	RFD4 RFD4.1 RFD4.2 RFD4.3
Server::Microservices::Chat::Types::- Local	RFO1 RFO1.1
Server::Microservices::Chat::Types::- Message	RFD4 RFD4.3
Server::Microservices::Chat::Types::- Profile	RFO1.1 RFO1.2



8 Tracciamento

Classe	Requisiti
	RFO2
Server::Microservices::Chat::Types::-Table	RFD4.1 RFD4.1.1 RFD4.1.2 RFD4.1.3 RFD4.1.4 RFD4.1.5 RFD4.1.6
Server::Microservices::Chat::Types::-TextMessage	RFD4.3 RFD4.3.1 RFD4.3.2 RFD4.3.3 RFD4.3.4
Server::Microservices::MicroservicesLoader::-MainLoader	RFO1.2 RFD1.3 RFD1.4 RFO1.5
Server::Microservices::Profile::Business::-Profile	RFO2 RFO2.1 RFO2.2
Server::Microservices::Profile::-Communication::Profile	RFO2 RFO2.2
Server::Microservices::Profile::Manger::-Loader	RFO2 RFO2.2
Server::Microservices::Profile::Persistence::-Dao::Factory	RFO1.1 RFO2 RFO2.2 RFO2.2.2
Server::Microservices::Profile::Persistence::-Dao::Sql::Beacon	RFO7
Server::Microservices::Profile::Persistence::-Dao::Sql::Factory	RFO2 RFO2.2
Server::Microservices::Profile::Persistence::-Dao::Sql::Profile	RFO1.2 RFO2 RFO2.2
Server::Microservices::Profile::Persistence::-Profile	RFO1.2 RFO2 RFO2.2



8 Tracciamento

Classe	Requisiti
Server::Microservices::Profile::Services::-Profile	RFO1.2 RFO2 RFO2.2
Server::Microservices::Profile::Types::-Beacon	RFO7
Server::Microservices::Profile::Types::-Profile	RFO1.2 RFO2 RFO2.2 RFO2.2.2
Server::Microservices::Score::Business::-Score	RFO5.3.5 RFO5.3.5.3 RFD5.4 RFD5.4.1
Server::Microservices::Score::Communication::-Score	RFO5.3.5.4 RFD5.4
Server::Microservices::Score::Dao::Sql::-Factory	RFO5.3.5.4 RFD5.4
Server::Microservices::Score::Manager::-Loader	RFO5.3.5.4 RFD5.4
Server::Microservices::Score::Services::-Score	RFO5.3.5.4 RFD5.4
Server::Microservices::Score::Types::-Game	RFO5.3.5.4 RFD5.4 RFD5.4.1
Server::Microservices::Score::Types::-Local	RFO5.3.5.4 RFD5.4 RFD5.4.1
Server::Microservices::Score::Types::-Score	RFO5.3.5.4 RFD5.4 RFD5.4.1
Server::Microservices::Score::Types::-Team	RFD5.4 RFD5.4.1
Server::Microservices::Session::Business::-Session	RFO1.1
Server::Microservices::Session::Manager::-Loader	RFO1.1



8 Tracciamento

Classe	Requisiti
Server::Microservices::Session::Manager::- Loader	RFO1.1
Server::Microservices::Session::Persistence::- Dao::Sql::Factory	RFO1.1
Server::Microservices::Session::Persistence::- Dao::Sql::Session	RFO1.1
Server::Microservices::Session::Persistence::- Session	RFO1.1
Server::Microservices::Session::Services::- Session	RFO1.1
Server::Microservices::Session::Types::- Session	RFO1.1
Server::Microservices::Team::Business::- Team	RFO5.1 RFO5.1.1 RFO5.1.3 RFO5.1.4
Server::Microservices::Team::Communication::- Team	RFO5.1 RFO5.1.1 RFO5.1.1.2 RFO5.1.2 RFO5.1.2.2 RFO5.1.3 RFO5.1.4
Server::Microservices::Team::Manager::- Loader	RFO5.1 RFO5.1.1 RFO5.1.2 RFO5.1.3 RFO5.1.4
Server::Microservices::Team::Persistence::- Team	RFO5.1 RFO5.1.1 RFO5.1.2 RFO5.1.3 RFO5.1.4
Server::Microservices::Team::Services::- Services	RFO5.1 RFO5.1.1 RFO5.1.2 RFO5.1.3 RFO5.1.4
Server::Microservices::Team::Types::- Profile	RFO5.1 RFO5.1.1 RFO5.1.2 RFO5.1.3



8 Tracciamento

Classe	Requisiti
	RFO5.1.4
Server::Microservices::Team::Types::-Team	RFO5.1
	RFO5.1.1
	RFO5.1.3

Tabella 1: Tracciamento Classi-Requisiti



8.2 Tracciamento Requisiti-Classi

Requisito	Classi
RFO1	Client::BaseFunctions::Presenter::- MainMenu Client::BaseFunctions::View::MainMenu Server::Microservices::Chat::Persistence::- Dao::Sql::Local Server::Microservices::Chat::Persistence::- Dao::Sql::Table Server::Microservices::Chat::Types::- Local
RFO1.1	Client::BaseFunctions::View::MainMenu Client::Profile::Model::Communication Client::Profile::Model::Profile Client::Profile::Presenter::Profile Server::Microservices::Chat::Persistence::- Dao::Sql::Local Server::Microservices::Chat::Persistence::- Dao::Sql::Profile Server::Microservices::Chat::Types::- Local Server::Microservices::Chat::Types::- Profile Server::Microservices::Profile::Persistence::- Dao::Factory Server::Microservices::Session::Business::- Session Server::Microservices::Session::Manager::- Loader Server::Microservices::Session::Manager::- Loader Server::Microservices::Session::Persistence::- Dao::Sql::Factory Server::Microservices::Session::Persistence::- Dao::Sql::Session Server::Microservices::Session::Persistence::- Session Server::Microservices::Session::Services::- Session Server::Microservices::Session::Types::- Session
RFO1.2	Client::BaseFunctions::Presenter::- MainMenu Client::BaseFunctions::View::MainMenu Client::Profile::Model::Communication Client::Profile::Model::Profile Server::Microservices::Chat::Persistence::- Dao::Sql::Profile



8 Tracciamento

Requisito	Classi
	Server::Microservices::Chat::Types::- Profile Server::Microservices::MicroservicesLoader::- MainLoader Server::Microservices::Profile::Persistence::- Dao::Sql::Profile Server::Microservices::Profile::Persistence::- Profile Server::Microservices::Profile::Services::- Profile Server::Microservices::Profile::Types::- Profile
RFD1.3	Client::BaseFunctions::Presenter::- MainMenu Server::Microservices::MicroservicesLoader::- MainLoader
RFD1.3.1	Client::BaseFunctions::Presenter::- Enviroment
RFD1.3.2	Client::BaseFunctions::Types::Errors::- Error Client::BaseFunctions::View::- EnviromentError
RFD1.4	Client::BaseFunctions::Presenter::- MainMenu Client::BaseFunctions::View::MainMenu Server::Microservices::MicroservicesLoader::- MainLoader
RFD1.4.1	Client::BaseFunctions::Presenter::- Enviroment
RFD1.4.2	Client::BaseFunctions::Types::Errors::- Error Client::BaseFunctions::View::- EnviromentError
RFO1.5	Client::BaseFunctions::Presenter::- MainMenu Client::BaseFunctions::View::MainMenu Server::Microservices::MicroservicesLoader::- MainLoader
RFO1.5.1	Client::BaseFunctions::Presenter::- Enviroment
RFO1.5.2	Client::BaseFunctions::Types::Errors::- Error Client::BaseFunctions::View::- EnviromentError
RFO2	Client::Profile::Model::Profile Client::Profile::Presenter::Profile Server::Microservices::Chat::Persistence::- Dao::Sql::Profile



8 Tracciamento

Requisito	Classi
	Server:::Microservices:::Chat:::Types::- Profile Server:::Microservices:::Profile:::Business::- Profile Server:::Microservices:::Profile:::- Communication:::Profile Server:::Microservices:::Profile:::Manger::- Loader Server:::Microservices:::Profile:::Persistence::- Dao:::Factory Server:::Microservices:::Profile:::Persistence::- Dao:::Sql:::Factory Server:::Microservices:::Profile:::Persistence::- Dao:::Sql:::Profile Server:::Microservices:::Profile:::Persistence::- Profile Server:::Microservices:::Profile:::Services::- Profile Server:::Microservices:::Profile:::Types::- Profile
RFO2.1	Client:::Profile:::Model:::Profile Client:::Profile:::View:::Profile Server:::Microservices:::Profile:::Business::- Profile
RFO2.2	Client:::Profile:::Model:::Profile Client:::Profile:::Presenter:::Profile Server:::Microservices:::Profile:::Business::- Profile Server:::Microservices:::Profile:::- Communication:::Profile Server:::Microservices:::Profile:::Manger::- Loader Server:::Microservices:::Profile:::Persistence::- Dao:::Factory Server:::Microservices:::Profile:::Persistence::- Dao:::Sql:::Factory Server:::Microservices:::Profile:::Persistence::- Dao:::Sql:::Profile Server:::Microservices:::Profile:::Persistence::- Profile Server:::Microservices:::Profile:::Services::- Profile Server:::Microservices:::Profile:::Types::- Profile
RFO2.2.1	Client:::Profile:::View:::Profile
RFO2.2.2	Client:::Profile:::Presenter:::Profile Server:::Microservices:::Profile:::Persistence::- Dao:::Factory



8 Tracciamento

Requisito	Classi
	Server::Microservices::Profile::Types::- Profile
RFO2.2.3	Client::Chat::Presenter::ChooseChat Client::Profile::Presenter::Profile
RFO2.2.4	Client::Profile::View::Profile
RFO2.2.5	Client::Profile::View::Profile
RFD4	Client::Chat::Model::Chat Client::Chat::Model::Communication Client::Chat::Presenter::Chat Client::Chat::Presenter::Manager Server::Microservices::Chat::Business::- Chat Server::Microservices::Chat::Communication::- Chat Server::Microservices::Chat::Manager::- Loader Server::Microservices::Chat::Persistence::- Dao::Sql::Factory Server::Microservices::Chat::Persistence::- Dao::Sql::Table Server::Microservices::Chat::Persistence::- Dao::Sql::TextMessage Server::Microservices::Chat::Services::- Chat Server::Microservices::Chat::Types::- Message
RFD4.1	Client::Chat::Model::Chat Client::Chat::Model::Communication Client::Chat::Presenter::- AvailableTables Client::Chat::Presenter::ChooseChat Client::Chat::Presenter::Push Client::Chat::View::AvailableTables Server::Microservices::Chat::Business::- Chat Server::Microservices::Chat::Services::- Chat Server::Microservices::Chat::Types::- Table
RFD4.1.1	Client::Chat::Presenter::- AvailableTables Client::Chat::Presenter::Push Client::Chat::View::AvailableTables Client::Chat::View::ChooseChat Server::Microservices::Chat::Types::- Table
RFD4.1.2	Client::Chat::Presenter::- AvailableTables Client::Chat::Presenter::ChooseChat



8 Tracciamento

Requisito	Classi
	Client::Chat::Presenter::Push Client::Chat::View::AvailableTables Server::Microservices::Chat::Types::-Table
RFD4.1.3	Client::Chat::Presenter::ChooseChat Server::Microservices::Chat::Types::-Table
RFD4.1.4	Client::Chat::View::ChooseChat Server::Microservices::Chat::Types::-Table
RFD4.1.5	Client::Chat::View::ChooseChat Server::Microservices::Chat::Types::-Table
RFD4.1.6	Client::Chat::View::ChooseChat Server::Microservices::Chat::Types::-Table
RFD4.2	Client::Chat::Model::Chat Client::Chat::Model::Communication Client::Chat::Presenter::ChooseChat Server::Microservices::Chat::Business::-Chat Server::Microservices::Chat::Services::-Chat
RFD4.2.1	Client::Chat::View::ChooseChat
RFD4.2.2	Client::Chat::Presenter::ChooseChat
RFD4.2.4	Client::Chat::View::ChooseChat
RFD4.2.5	Client::Chat::View::ChooseChat
RFD4.3	Client::Chat::Model::Chat Client::Chat::Model::Communication Client::Chat::Presenter::Chat Client::Chat::View::Push Server::Microservices::Chat::Business::-Chat Server::Microservices::Chat::Persistence::-Dao::Sql::Factory Server::Microservices::Chat::Persistence::-Dao::Sql::Message Server::Microservices::Chat::Persistence::-Dao::Sql::TextMessage Server::Microservices::Chat::Services::-Chat Server::Microservices::Chat::Types::-Message Server::Microservices::Chat::Types::-TextMessage
RFD4.3.1	Client::Chat::View::Chat Client::Chat::View::Push Server::Microservices::Chat::Persistence::-Dao::Sql::Message



8 Tracciamento

Requisito	Classi
	Server::Microservices::Chat::Types::- TextMessage
RFD4.3.2	Client::Chat::Presenter::Chat Client::Chat::View::Push Server::Microservices::Chat::Persistence::- Dao::Sql::Message Server::Microservices::Chat::Types::- TextMessage
RFD4.3.3	Client::Chat::Presenter::Chat Client::Chat::View::Push Server::Microservices::Chat::Types::- TextMessage
RFD4.3.4	Client::Chat::View::Chat Server::Microservices::Chat::Types::- TextMessage
RFD4.4	Client::Chat::Model::Communication Client::Chat::Presenter::Chat Client::Chat::View::Chat Server::Microservices::Chat::Business::- Chat
RFD4.4.1	Client::Chat::Presenter::Chat
RFO5	Client::Games::Battleship::Model::- Battleship Client::Games::Battleship::Presenter::- Manager Client::Games::Utility::Model::- GamesList Client::Games::Utility::Presenter::- GamesList Client::Games::Utility::Presenter::- Manager Client::Games::Utility::Presenter::- Manager Client::Games::Utility::View::- GamesList Server::Microservices::Battleship::- Business::Battleship Server::Microservices::Battleship::- Manager::Loader Server::Microservices::Battleship::- Services::Battleship Server::Microservices::Chat::Persistence::- Dao::Sql::Table
RFO5.1	Client::Games::Battleship::Model::- Battleship Client::Games::Battleship::Model::- Communication::Battleship Client::Games::Battleship::Presenter::- Manager



8 Tracciamento

Requisito	Classi
	Client::Games::Utility::Types::Team Server::Microservices::Battleship::-Communication::Battleship Server::Microservices::Team::Business::-Team Server::Microservices::Team::Communication::-Team Server::Microservices::Team::Manager::-Loader Server::Microservices::Team::Persistence::-Team Server::Microservices::Team::Services::-Services Server::Microservices::Team::Types::-Profile Server::Microservices::Team::Types::-Team
RFO5.1.1	Client::Games::Battleship::Presenter::-ChooseRole Client::Games::Battleship::View::-ChooseRole Client::Games::Utility::Types::Team Server::Microservices::Team::Business::-Team Server::Microservices::Team::Communication::-Team Server::Microservices::Team::Manager::-Loader Server::Microservices::Team::Persistence::-Team Server::Microservices::Team::Services::-Services Server::Microservices::Team::Types::-Profile Server::Microservices::Team::Types::-Team
RFO5.1.1.1	Client::Games::Battleship::Presenter::-ChooseRole Client::Games::Battleship::View::-CreateTeam
RFO5.1.1.2	Client::Games::Battleship::Presenter::-ChooseRole Client::Games::Battleship::Presenter::-CreateTeam Server::Microservices::Team::Communication::-Team
RFO5.1.1.3	Client::Games::Battleship::Presenter::-ChooseRole



8 Tracciamento

Requisito	Classi
	Client::Games::Battleship::Presenter::- CreateTeam
RFO5.1.1.4	Client::Games::Battleship::Presenter::- ChooseRole Client::Games::Battleship::Presenter::- CreateTeam
RFO5.1.1.5	Client::Games::Battleship::Presenter::- ChooseRole Client::Games::Battleship::View::- CreateTeam
RFO5.1.2	Client::Games::Battleship::Presenter::- JoinTeam Client::Games::Utility::Types::Team Server::Microservices::Team::Communication::- Team Server::Microservices::Team::Manager::- Loader Server::Microservices::Team::Persistence::- Team Server::Microservices::Team::Services::- Services Server::Microservices::Team::Types::- Profile
RFO5.1.2.1	Client::Games::Battleship::Presenter::- JoinTeam Client::Games::Battleship::View::- JoinTeam
RFO5.1.2.2	Client::Games::Battleship::Presenter::- JoinTeam Server::Microservices::Team::Communication::- Team
RFO5.1.2.3	Client::Games::Battleship::Presenter::- JoinTeam
RFO5.1.2.4	Client::Games::Battleship::View::- JoinTeam
RFO5.1.3	Client::Games::Battleship::Presenter::- JoinTeam Client::Games::Utility::Types::Team Server::Microservices::Team::Business::- Team Server::Microservices::Team::Communication::- Team Server::Microservices::Team::Manager::- Loader Server::Microservices::Team::Persistence::- Team Server::Microservices::Team::Services::- Services



8 Tracciamento

Requisito	Classi
	<pre>Server:::Microservices:::Team:::Types::- Profile Server:::Microservices:::Team:::Types::- Team</pre>
RFO5.1.3.1	<pre>Client:::Games:::Battleship:::Presenter::- JoinTeam</pre>
RFO5.1.3.2	<pre>Client:::Games:::Battleship:::Presenter::- JoinTeam</pre>
RFO5.1.4	<pre>Client:::Games:::Battleship:::Presenter::- TeamManagement Client:::Games:::Battleship:::Types::- Shoot Client:::Games:::Utility:::Types:::Team Server:::Microservices:::Team:::Business::- Team Server:::Microservices:::Team:::Communication::- Team Server:::Microservices:::Team:::Manager::- Loader Server:::Microservices:::Team:::Persistence::- Team Server:::Microservices:::Team:::Services::- Services Server:::Microservices:::Team:::Types::- Profile</pre>
RFO5.1.4.1	<pre>Client:::Games:::Battleship:::Presenter::- TeamManagement Client:::Games:::Battleship:::Types::- Shoot Client:::Games:::Battleship:::View::- TeamManagement</pre>
RFO5.1.4.2	<pre>Client:::Games:::Battleship:::Presenter::- TeamManagement Client:::Games:::Battleship:::Types::- Shoot</pre>
RFO5.1.4.3	<pre>Client:::Games:::Battleship:::Presenter::- TeamManagement</pre>
RFO5.1.4.4	<pre>Client:::Games:::Battleship:::Presenter::- TeamManagement</pre>
RFO5.1.4.5	<pre>Client:::Games:::Battleship:::View::- TeamManagement</pre>
RFO5.2	<pre>Client:::Games:::Battleship:::Model::- Battleship Client:::Games:::Battleship:::Presenter::- Manager Server:::Microservices:::Battleship::- Communication:::Battleship</pre>



8 Tracciamento

Requisito	Classi
RFO5.2.1	<pre>Client:::Games:::Battleship:::Model::- Communication:::Battleship</pre>
RFO5.3	<pre>Client:::Games:::Battleship:::Model::- Battleship Client:::Games:::Battleship:::Model::- Communication:::Battleship Client:::Games:::Battleship:::Presenter::- Manager Client:::Games:::Battleship:::Types:::Grid Server:::Microservices:::Battleship::- Business:::Battleship Server:::Microservices:::Battleship::- Communication:::Battleship Server:::Microservices:::Battleship::- Manager:::Loader Server:::Microservices:::Battleship::- Persistence:::Battleship Server:::Microservices:::Battleship:::Types::- Position</pre>
RFO5.3.1	<pre>Client:::Games:::Battleship:::Presenter::- Manager Client:::Games:::Battleship:::Types:::Grid Server:::Microservices:::Battleship::- Business:::Battleship Server:::Microservices:::Battleship::- Persistence:::Battleship Server:::Microservices:::Battleship:::Types::- Position</pre>
RFO5.3.2	<pre>Client:::Games:::Battleship:::Presenter::- Manager Client:::Games:::Battleship:::Types:::Grid Server:::Microservices:::Battleship::- Business:::Battleship Server:::Microservices:::Battleship::- Persistence:::Battleship Server:::Microservices:::Battleship:::Types::- Position</pre>
RFO5.3.3	<pre>Client:::Games:::Battleship:::Presenter::- SelectShipPosition Client:::Games:::Battleship:::Types:::Grid Client:::Games:::Battleship:::Types:::Ship Server:::Microservices:::Battleship::- Business:::Battleship Server:::Microservices:::Battleship::- Persistence:::Battleship Server:::Microservices:::Battleship:::Types::- Cell Server:::Microservices:::Battleship:::Types::- Position</pre>



8 Tracciamento

Requisito	Classi
	<pre>Server::Microservices::Battleship::Types::- Field Server::Microservices::Battleship::Types::- Ship</pre>
RFO5.3.3.1	<pre>Client::Games::Battleship::Presenter::- SelectShipPosition Client::Games::Battleship::Types::- Position Client::Games::Battleship::Types::Ship Server::Microservices::Battleship::Types::- Field Server::Microservices::Battleship::Types::- Ship</pre>
RFO5.3.3.2	<pre>Client::Games::Battleship::Types::- Field Client::Games::Battleship::Types::- Position Client::Games::Battleship::Types::Ship Client::Games::Battleship::View::- SelectShipPosition Server::Microservices::Battleship::Types::- Field Server::Microservices::Battleship::Types::- Ship</pre>
RFO5.3.3.3	<pre>Client::Games::Battleship::Presenter::- SelectShipPosition Client::Games::Battleship::Types::Cell Client::Games::Battleship::Types::- Position Server::Microservices::Battleship::Types::- Field</pre>
RFO5.3.3.4	<pre>Client::Games::Battleship::Presenter::- SelectShipPosition Client::Games::Battleship::Types::- Position Client::Games::Battleship::View::- FinalScore Server::Microservices::Battleship::Types::- Field</pre>
RFO5.3.3.5	<pre>Client::Games::Battleship::View::- SelectShipPosition Server::Microservices::Battleship::Types::- Field</pre>
RFO5.3.3.6	<pre>Client::Games::Battleship::Presenter::- SelectShipPosition Server::Microservices::Battleship::Types::- Field</pre>
RFO5.3.3.7	<pre>Client::Games::Battleship::View::- SelectShipPosition</pre>



8 Tracciamento

Requisito	Classi
	Server::Microservices::Battleship::Types::- Field
RFO5.3.3.8	Client::Games::Battleship::Presenter::- SelectShipPosition Server::Microservices::Battleship::Types::- Field
RFO5.3.3.9	Client::Games::Battleship::Presenter::- SelectShipPosition Server::Microservices::Battleship::Types::- Field
RFO5.3.3.10	Client::Games::Battleship::View::- SelectShipPosition Server::Microservices::Battleship::Types::- Field
RFO5.3.4	Client::Games::Battleship::Presenter::- AttackPhase Client::Games::Battleship::Presenter::- EnemyAttackPhase Server::Microservices::Battleship::Types::- Cell Server::Microservices::Battleship::Types::- Shoot
RFO5.3.4.1	Client::Games::Battleship::Presenter::- AttackPhase Client::Games::Battleship::Presenter::- EnemyAttackPhase Server::Microservices::Battleship::Types::- Shoot
RFO5.3.4.2	Client::Games::Battleship::Types::Cell Client::Games::Battleship::View::- AttackPhase Client::Games::Battleship::View::- EnemyAttackPhase Server::Microservices::Battleship::Types::- Shoot
RFO5.3.4.3	Client::Games::Battleship::Presenter::- AttackPhase Client::Games::Battleship::Presenter::- EnemyAttackPhase Client::Games::Battleship::Types::- Field Server::Microservices::Battleship::Types::- Shoot
RFO5.3.4.4	Client::Games::Battleship::Presenter::- AttackPhase Client::Games::Battleship::Presenter::- EnemyAttackPhase



8 Tracciamento

Requisito	Classi
	Server::Microservices::Battleship::Types::- Shoot
RFO5.3.4.5	Client::Games::Battleship::View::- AttackPhase Client::Games::Battleship::View::- EnemyAttackPhase Server::Microservices::Battleship::Types::- Shoot
RFO5.3.4.6	Client::Games::Battleship::Presenter::- AttackPhase Client::Games::Battleship::Presenter::- EnemyAttackPhase
RFO5.3.5	Client::Games::Battleship::Presenter::- FinalScore Server::Microservices::Score::Business::- Score
RFO5.3.5.1	Client::Games::Battleship::Presenter::- FinalScore
RFO5.3.5.2	Client::Games::Battleship::View::- FinalScore
RFO5.3.5.3	Client::Games::Battleship::View::- FinalScore Server::Microservices::Score::Business::- Score
RFO5.3.5.4	Server::Microservices::Score::Communication::- Score Server::Microservices::Score::Dao::Sql::- Factory Server::Microservices::Score::Manager::- Loader Server::Microservices::Score::Services::- Score Server::Microservices::Score::Types::- Game Server::Microservices::Score::Types::- Local Server::Microservices::Score::Types::- Score
RFO5.3.5.5	Client::Games::Battleship::Presenter::- FinalScore
RFO5.3.5.6	Client::Games::Battleship::Presenter::- FinalScore
RFO5.3.5.7	Client::Games::Battleship::Presenter::- FinalScore Client::Games::Battleship::Presenter::- SelectShipPosition
RFO5.3.5.8	Client::Games::Battleship::Presenter::- FinalScore



8 Tracciamento

Requisito	Classi
RFD5.4	<pre> Client::Games::Battleship::Presenter::- Manager Client::Games::Utility::Model::- Leaderboard Client::Games::Utility::Presenter::- Leaderboard Client::Games::Utility::Types::Score Client::Games::Utility::View::- Leaderboard Server::Microservices::Score::Business::- Score Server::Microservices::Score::Communication::- Score Server::Microservices::Score::Dao::Sql::- Factory Server::Microservices::Score::Manager::- Loader Server::Microservices::Score::Services::- Score Server::Microservices::Score::Types::- Game Server::Microservices::Score::Types::- Local Server::Microservices::Score::Types::- Score Server::Microservices::Score::Types::- Team </pre>
RFD5.4.1	<pre> Client::Games::Utility::Model::- Leaderboard Client::Games::Utility::Presenter::- Leaderboard Client::Games::Utility::View::- Leaderboard Server::Microservices::Score::Business::- Score Server::Microservices::Score::Types::- Game Server::Microservices::Score::Types::- Local Server::Microservices::Score::Types::- Score Server::Microservices::Score::Types::- Team </pre>
RFO6	<pre> Client::BaseFunctions::Presenter::- Manager </pre>
RFO7	<pre> Client::BaseFunctions::Types::Beacon Server::Microservices::Profile::Persistence::- Dao::Sql::Beacon </pre>



8 Tracciamento

Requisito	Classi
	Server::Microservices::Profile::Types::- Beacon
RFO7.1	Client::BaseFunctions::Types::Errors::- BluetoothError
RVO9	Client::BaseFunctions::Model::Communication::- Eviroment Client::BaseFunctions::Model::- Enviroment Client::BaseFunctions::Model::- EnviromentService

Tabella 2: Tracciamento Requisiti-Classi



A Descrizione Design Pattern

A Descrizione Design Pattern

Sono illustrati di seguito i Design Pattern implementati nella costruzione dell'architettura di alto livello.

A.1 Design Pattern Architetturali

A.1.1 Microservice Architecture

Il Microservice Architecture pattern fornisce uno stile di architettura software dove applicazioni complesse sono composte da processi ridotti e indipendenti, che comunicano tra loro. Questi servizi sono disaccoppiati e svolgono piccoli task, facilitando un approccio modulare alla costruzione del sistema. In dettaglio il Microservice Architecture pattern permette che:

- i servizi siano facili da sostituire;
- i servizi possano essere implementati usando differenti linguaggi di programmazione, databases, componenti hardware e software;
- l'architettura risulti simmetrica piuttosto che gerarchica.

Un architettura di questo tipo è particolarmente adatta a uno sviluppo continuo del software. Di seguito un diagramma esplicativo di tale architettura:

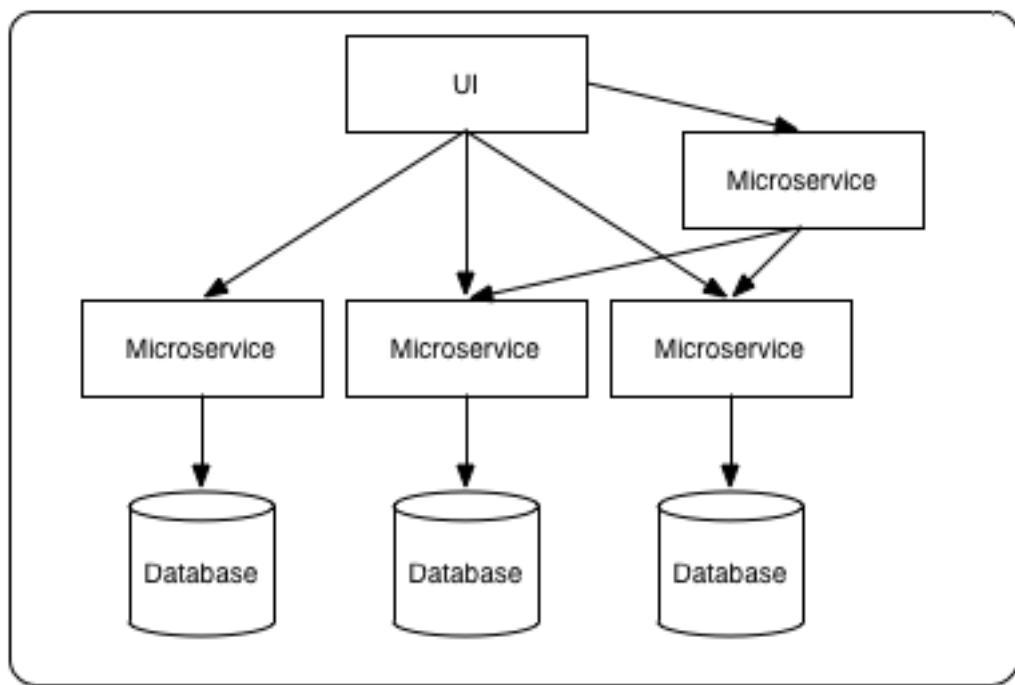


Figura 28: Diagramma della Microservices Architecture

A.1.2 Model View Presenter

Il Model View Presenter è un pattern architettonico, viene utilizzato maggiormente per sviluppare interfacce utente. Il pattern prevede:

- **Model:** un'interfaccia che definisce i dati che devono essere visualizzati.
- **View:** un template di visualizzazione e un'interfaccia di comunicazione.
- **Presenter:** opera sul model e sulla view; riceve dati dal model e li formatta per visualizzarli nella view.



A Descrizione Design Pattern

Di seguito un diagramma esplicativo di tale pattern:

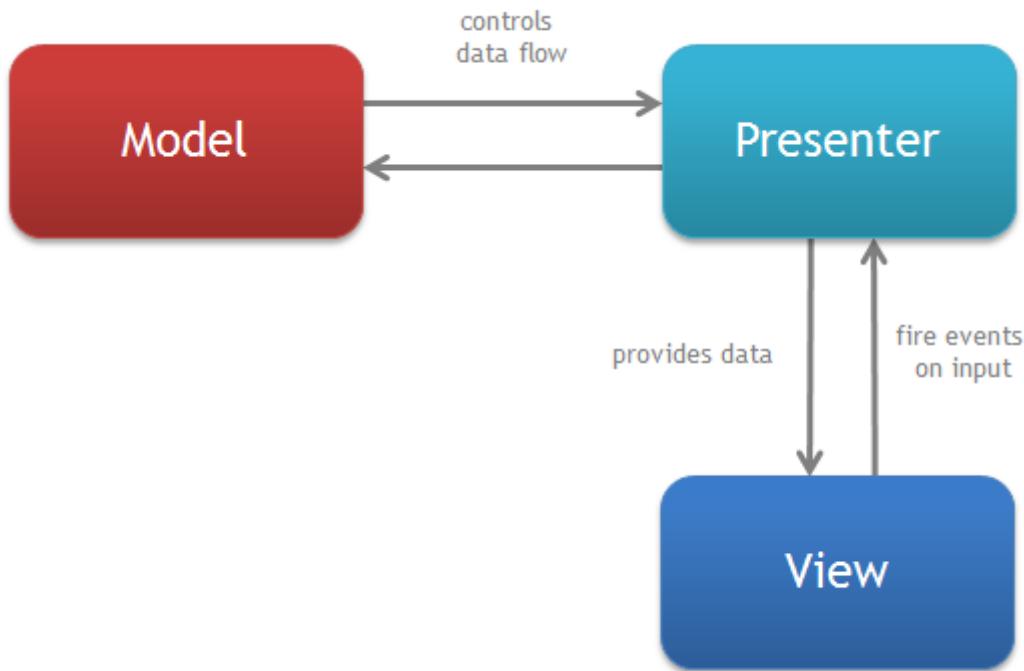


Figura 29: Diagramma del pattern Model View Presenter

A.2 Design Pattern Creazionali

A.2.1 Abstract Factory

L'Abstract Factory pattern fornisce una metodologia per raggruppare un gruppo di Factory connesse tra loro. Il client crea una implementazione concreta della abstract factory e utilizza l'interfaccia generica della factory per creare oggetti concreti. Il pattern separa la concreta implementazione di un gruppo di oggetti dal loro uso generale. Il sistema risulta quindi essere indipendente dall'implementazione degli oggetti concreti. Di seguito un diagramma esplicativo di tale pattern:



A Descrizione Design Pattern

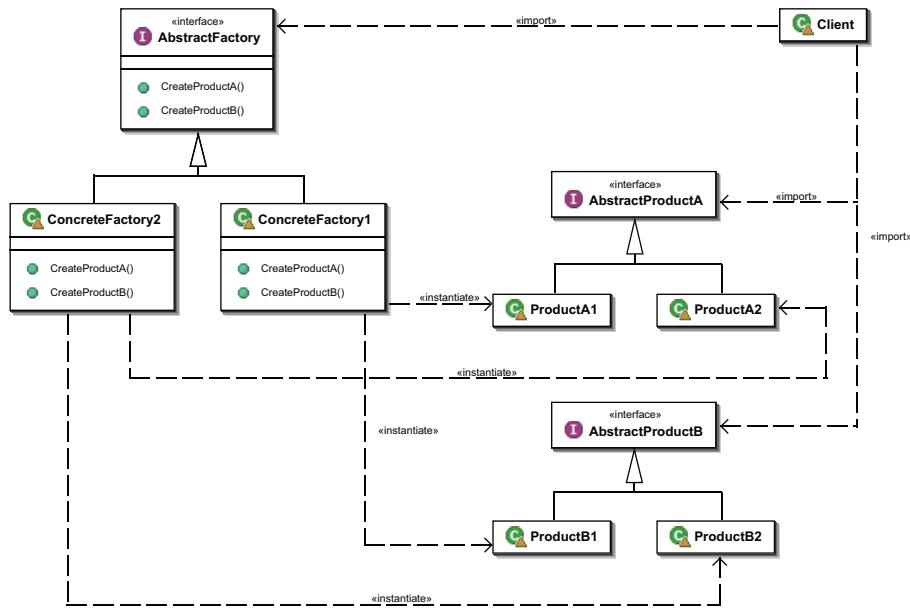


Figura 30: Diagramma del pattern Abstract Factory

A.3 Design Pattern Strutturali

A.3.1 Adapter Pattern

L'Adapter pattern è un ponte tra due interfacce incompatibili. Il pattern coinvolge un'unica classe la cui responsabilità è unire le differenti funzionalità di interfacce indipendenti o incompatibili tra loro. In seguito un diagramma esplicativo di tale pattern:

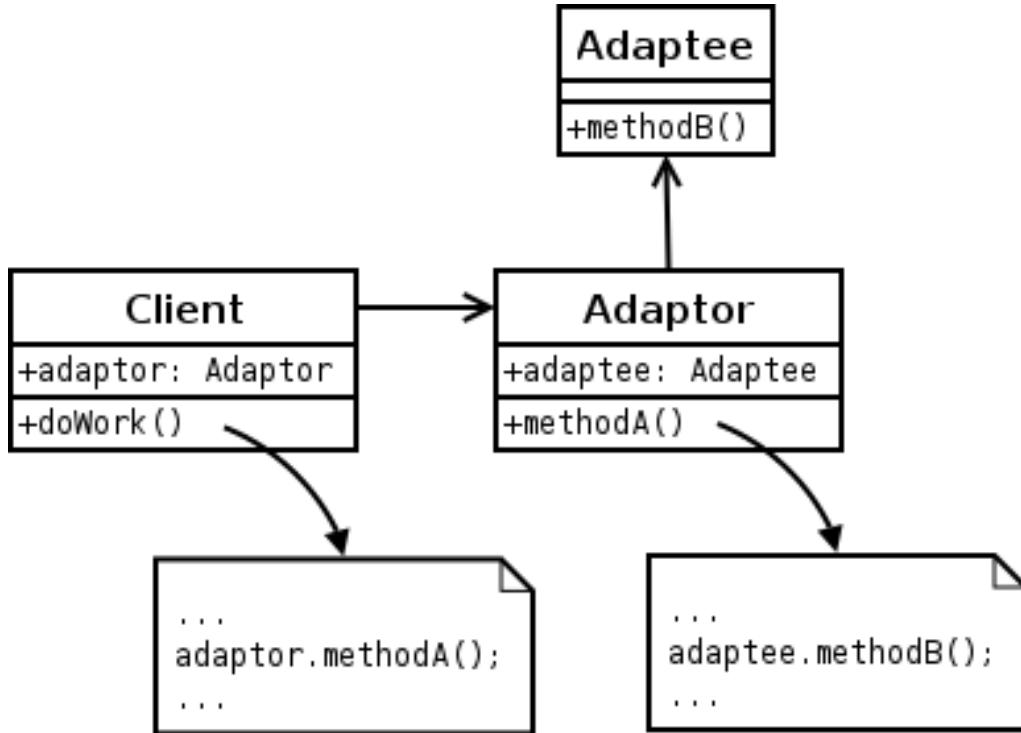


Figura 31: Diagramma del pattern Adapter



A Descrizione Design Pattern

A.3.2 Data Access Object DAO

Il DAO fornisce un’interfaccia astratta per qualche tipo di base di dati. Il pattern fornisce operazioni specifiche sui dati senza esporre i dettagli del database; separa la logica di accesso ai dati da come questi dati vengono rappresentati in una specifica base di dati. In questo modo sia la logica di accesso che la logica di rappresentazione possono evolvere frequentemente e indipendentemente la una dall’altra. In seguito un diagramma esplicativo di tale pattern:

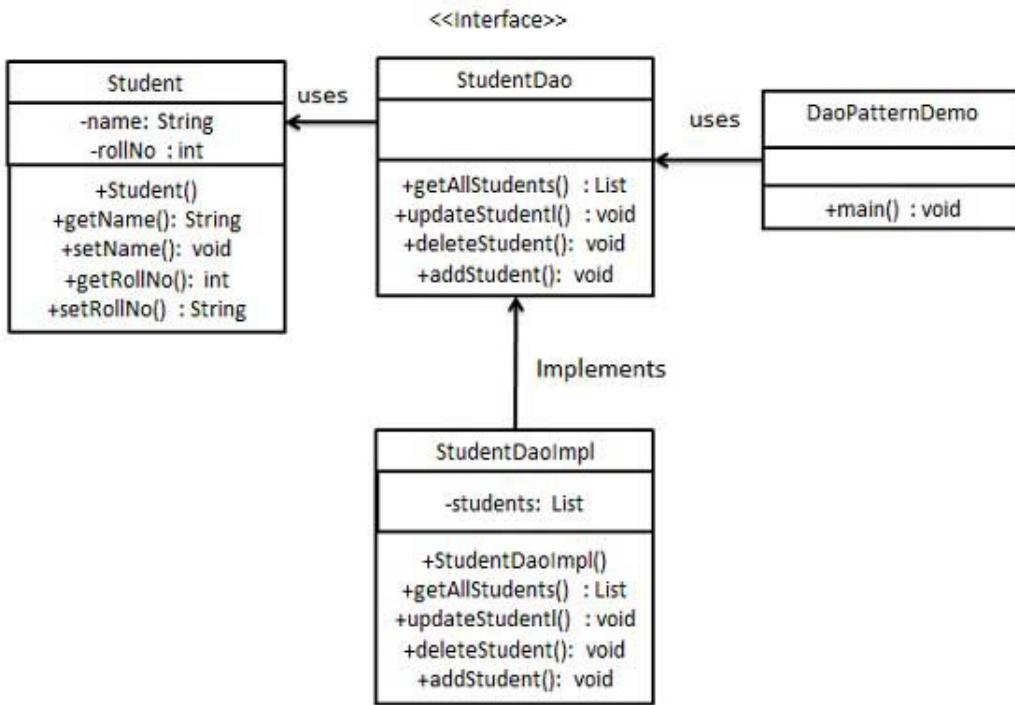


Figura 32: Diagramma del pattern DAO

A.3.3 Facade

Il pattern Facade nasconde la complessità di un sistema e fornisce al client un’interfaccia grazie alla quale potrà accedere alle funzionalità del sistema. Questo pattern si utilizza quando si ha la necessità di una singola interfaccia semplice; quando si vuole ottenere disaccoppiamento tra sottosistemi e il client; quando si vuole stratificare un sistema. L’utilizzo del Facade riduce il numero di classi del sottosistema con cui il client deve interagire; elimina le dipendenze circolari; riduce i tempi di compilazione e di building. Allo stesso tempo si può incorrere in un problema di sovrardimensionamento della classe Facade che quindi risulta essere un single point of failure. In seguito un diagramma esplicativo di tale pattern:



A Descrizione Design Pattern

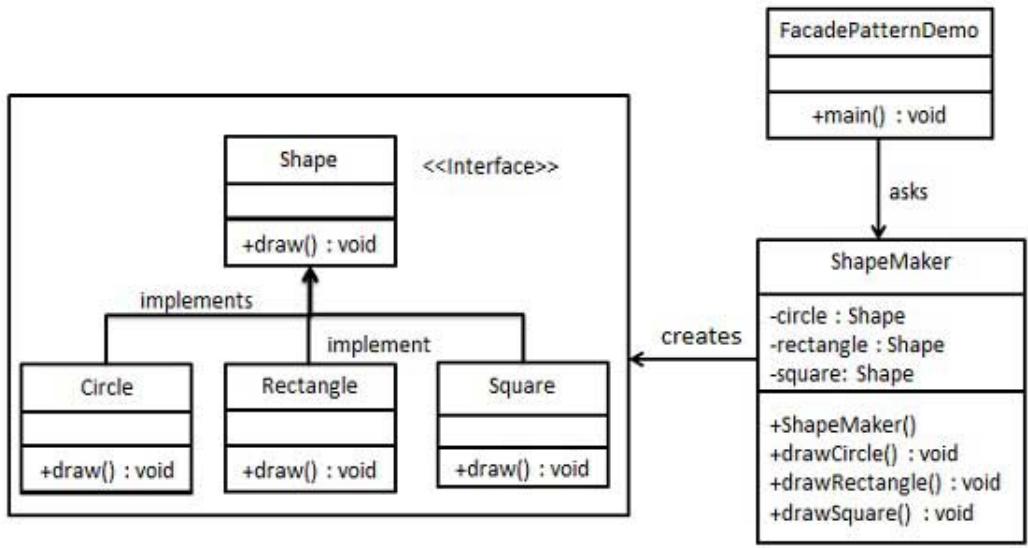


Figura 33: Diagramma del pattern Facade

A.4 Design Pattern Comportamentali

A.4.1 Observer Pattern

In questo pattern un oggetto, chiamato "subject", mantiene una lista delle sue dipendenze, chiamate "observers", e notifica automaticamente a quest'ultime qualsiasi cambiamento di stato, in genere chiamando uno dei loro metodi. Il pattern viene principalmente usato nei sistemi distribuiti di gestione di eventi. In seguito un diagramma esplicativo di tale pattern:

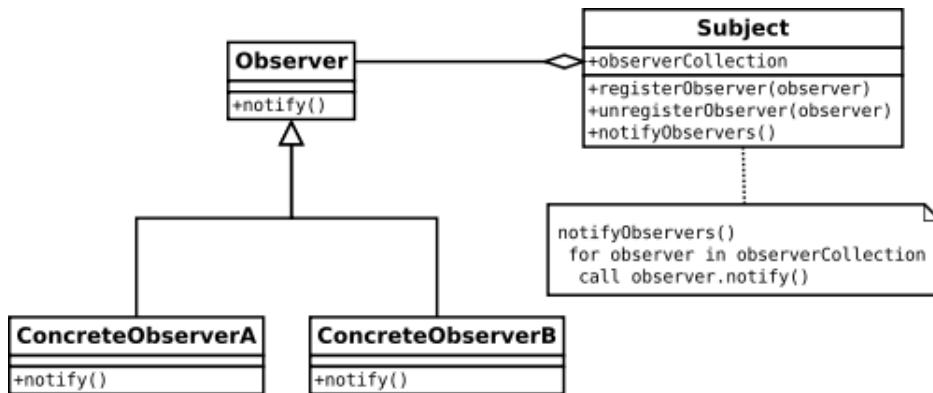


Figura 34: Diagramma del pattern Observer