CLIPS

Specifica Tecnica v3.0.0

Sommario
Descrizione dell'architettura ad alto livello per il *progetto*$_g$ CLIPS.

Nome del documento	Specifica Tecnica
Versione	3.0.0
Data di redazione	2016-09-11
Redazione	Viviana Alessio Matteo Franco Andrea Grendene Luca Soldera
Verifica	Enrico Bellio
Approvazione	Andrea Grendene
Uso	Esterno
Lista di distribuzione	prof. Tullio Vardanega prof. Riccardo Cardin Miriade SpA

Diario delle modifiche

Versione	Riepilogo	Autore	Ruolo	Data
3.0.0	Approvazione documento	Andrea Grendene	Responsabile	2016-09-11
2.1.0	Verifica documento	Viviana Alessio	Verificatore	2016-09-11
2.0.4	Sostituito MVC con MVP in appendice	Luca Soldera	Progettista	2016-09-03
2.0.3	Solstituita immagine singleton in appendice	Luca Soldera	Progettista	2016-09-02
2.0.2	Aggiunte descrizioni macro-caratteristiche dell'architettura	Luca Soldera	Progettista	2016-09-02
2.0.1	Modificate descrizioni delle tecnologie e aggiunta della sezione utilizzo	Luca Soldera	Progettista	2016-09-02
2.0.0	Approvazione documento	Tommaso Panozzo	Responsabile	2016-08-17
1.3.0	Verifica documento	Enrico Bellio	Verificatore	2016-07-21
1.2.1	Correzione riferimenti immagini	Viviana Alessio	Progettista	2016-07-20
1.2.0	Verifica documento	Enrico Bellio	Verificatore	2016-07-16
1.1.0	Aggiunta sezione Architettura del Database	Luca Soldera	Progettista	2016-07-12
1.0.0	Approvazione documento	Matteo Franco	Responsabile	2016-06-10
0.2.0	Correzione errori classi	Viviana Alessio	Progettista	2016-06-09
0.1.9	Verifica classi inserite	Tommaso Panozzo	Verificatore	2016-06-09
0.1.9	Inserimento classi package server	Andrea Grendene	Progettista	2016-06-07
0.1.9	Inserimento classi package server–urlrequesthandler	Andrea Grendene	Progettista	2016-06-07
0.1.9	Inserimento classi package server–data	Andrea Grendene	Progettista	2016-06-07
0.1.9	Correzione errori ortografici e sintattici	Viviana Alessio	Progettista	2016-06-05
0.1.8	Verifica classi inserite	Enrico Bellio	Verificatore	2016-06-05
0.1.8	Inserimento classi package client–data	Luca Soldera	Progettista	2016-06-04
0.1.8	Inserimento classi package client–datamanager	Luca Soldera	Progettista	2016-06-04
0.1.8	Inserimento classi package client–location	Luca Soldera	Progettista	2016-06-04

Versione	Riepilogo	Autore	Ruolo	Data
0.1.8	Inserimento classi package client–pathprogress	Luca Soldera	Progettista	2016-06-04
0.1.8	Inserimento classi package client–urlrequest	Luca Soldera	Progettista	2016-06-04
0.1.8	Inserimento classi package client	Luca Soldera	Progettista	2016-06-04
0.1.8	Correzione errori classi	Viviana Alessio	Progettista	2016-06-04
0.1.7	Verifica classi inserite	Enrico Bellio	Verificatore	2016-06-04
0.1.7	Inserimento classi package client–viewcontroller	Matteo Franco	Progettista	2016-05-30
0.1.7	Inserimento classi package client–viewcontroller–utility	Matteo Franco	Progettista	2016-05-30
0.1.7	Inserimento classi package client–viewcontroller–savedresults	Matteo Franco	Progettista	2016-05-30
0.1.7	Inserimento classi package client–viewcontroller–games	Matteo Franco	Progettista	2016-05-30
0.1.7	Inserimento classi package client–viewcontroller–building	Matteo Franco	Progettista	2016-05-30
0.1.7	Inserimento classi package client–viewcontroller–authentication	Matteo Franco	Progettista	2016-05-30
0.1.7	Correzione errori ortografici e sintattici	Viviana Alessio	Progettista	2016-05-28
0.1.6	Verifica packages inseriti	Tommaso Panozzo	Verificatore	2016-05-27
0.1.6	Inserimento package server	Andrea Grendene	Progettista	2016-05-26
0.1.6	Inserimento package server–urlrequesthandler	Andrea Grendene	Progettista	2016-05-26
0.1.6	Inserimento package server–data	Andrea Grendene	Progettista	2016-05-26
0.1.6	Inserimento package client–data	Luca Soldera	Progettista	2016-05-26
0.1.5	Inserimento package client–datamanager	Luca Soldera	Progettista	2016-05-26
0.1.5	Inserimento package client–location	Luca Soldera	Progettista	2016-05-26
0.1.5	Inserimento package client–pathprogress	Luca Soldera	Progettista	2016-05-26

Versione	Riepilogo	Autore	Ruolo	Data
0.1.5	Inserimento package client–urlrequest	Luca Soldera	Progettista	2016-05-26
0.1.5	Inserimento package client	Luca Soldera	Progettista	2016-05-26
0.1.5	Inserimento package client–viewcontroller	Matteo Franco	Progettista	2016-05-26
0.1.5	Inserimento package client–viewcontroller–utility	Matteo Franco	Progettista	2016-05-26
0.1.4	Inserimento package client–viewcontroller–savedresults	Matteo Franco	Progettista	2016-05-26
0.1.4	Inserimento package client–viewcontroller–games	Matteo Franco	Progettista	2016-05-26
0.1.4	Inserimento package client–viewcontroller–building	Matteo Franco	Progettista	2016-05-26
0.1.4	Inserimento package client–viewcontroller–authentication	Matteo Franco	Progettista	2016-05-26
0.1.4	Correzione errori ortografici e sintattici	Viviana Alessio	Progettista	2016-05-25
0.1.3	Verifica sezioni concluse	Tommaso Panozzo	Verificatore	2016-05-24
0.1.3	Stesura sezione Descrizione architettura	Viviana Alessio	Progettista	2016-05-23
0.1.3	Stesura sezione Diagrammi delle attività	Viviana Alessio	Progettista	2016-05-22
0.1.2	Stesura appendice Design pattern utilizzati	Viviana Alessio	Progettista	2016-05-21
0.1.1	Stesura sezione Tecnologie	Viviana Alessio	Progettista	2016-05-13
0.1.0	Stesura sezione Introduzione	Viviana Alessio	Progettista	2016-05-12
0.0.2	Stesura indice	Viviana Alessio	Progettista	2016-05-10
0.0.1	Creazione documento	Matteo Franco	Progettista	2016-05-09

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di spiegare dettagliatamente le strategie secondo cui il gruppo Beacon Strips intende condurre il *progetto*$_g$ didattico.

1.2 Scopo del *prodotto*$_g$

Il prodotto finale consisterà di un'applicazione mobile che, interagendo con dei beacons sparsi nell'area designata, guiderà l'utente attraverso un percorso. L'utente potrà completare il percorso superando tutte le prove che gli si presenteranno nelle diverse tappe. Le prove potranno essere degli indovinelli o dei semplici giochi inerenti all'area in cui si svolge il percorso.

1.3 Glossario

Al fine di evitare ogni ambiguità nel linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, gli acronimi e le abbreviazioni che necessitano di definizione sono riportati nel documento "*Glossario v3.0.0*".

Inoltre ogni occorrenza di un vocabolo presente nel Glossario sarà posta in corsivo e seguita da una 'g' minuscola a pedice (p.es. *Glossario*$_g$).

1.4 Riferimenti

1.4.1 Normativi

- **Capitolato d'appalto C2 - CLIPS:** Communication & Localisation with Indoor Positioning Systems.
  http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C2.pdf
- **Norme di Progetto**
  "*Norme di Progetto v4.0.0*"
- **Analisi dei Requisiti**
  "*Analisi dei Requisiti v4.0.0*"
- **Definizione di Prodotto**
  "*Definizione di Prodotto v2.0.0*"

1.4.2 Informativi

- **Software Engineering (10th edition**)
  Ian Sommerville
  Pearson Education — Addison-Wesley
- **Guide to the Software Engineering Body of Knowledge** IEEE Computer Society. Software Engineering Coordinating Committee
- **Slides del prof. Tullio Vardanega**
  riguardo la progettazione
- **Slides del prof. Riccardo Cardin**
  riguardo UML
  diagrammi delle classi

diagrammi dei package
diagrammi di attività
diagrammi di sequenza
riguardo i design pattern
design pattern strutturali
design pattern creazionali
design pattern comportamentali

- Documentazione per lo sviluppo di applicazioni per Android
- Documentazione java
- Documentazione JSON
- Documentazione PHP
- Documentazione MongoDB
- Docuentazione Android Beacon Library
- Documentazione AltBeacon

2 Tecnologie

Per lo sviluppo del progetto abbiamo la necessità di utlizzare delle tecnologie specifiche che sono state scelte dopo attenta analisi. In questa sezione del documento vengono riportate quelle che verranno utilizzate principalmente.

2.1 Java

Java è virtualmente alla base di qualsiasi tipo di applicazione in rete ed è lo standard globale per lo sviluppo e la distribuzione di applicazioni incorporate e per sistemi portatili, giochi, contenuto basato su Web e software aziendale. Con oltre 9 milioni di sviluppatori in tutto il mondo, Java consente di sviluppare, distribuire e utilizzare applicazioni e servizi in modo efficiente.

- **Vantaggi:**
  - è un linguaggio molto diffuso;
  - è un linguaggio libero e gratuito;
  - è un linguaggio multipiattaforma, ovvero dallo stesso codice si ottiene lo stesso programma, a prescindere dal sistema operativo usato e dalle caratteristiche specifiche dell'elaboratore;
  - contiene già di base molti design pattern e strutture varie, semplificando la loro implementazione;
  - sono disponibili numerose librerie utilizzabili con questo linguaggio, di conseguenza sono presenti molte funzioni già definite e validate;
  - è disponibile online una documentazione molto dettagliata;
  - utilizza una sintassi molto simile al C++, un linguaggio che tutti i componenti del $team_{\text{g}}$ conoscono;
  - rispetto al C++ è più semplice perché gestisce automaticamente molte funzionalità, come ad esempio la memoria e quindi l'allocazione e la deallocazione degli oggetti;
- **Svantaggi:**
  - essendo un linguaggio interpretato è più lento in fase di esecuzione rispetto ad uno nativo, sebbene questo svantaggio fosse più importante in passato, perché ormai la potenza che hanno raggiunto gli elaboratori è di gran lunga maggiore di quella necessaria per superare questo problema, di fatto al giorno d'oggi questo svantaggio si presenta solo con applicazioni molto pesanti, come i programmi grafici, o in supporti hardware più limitati, come ad esempio gli smartphone e i tablet;
  - richiede un interprete, e quindi un'ulteriore applicazione, comunque già presente in Android visto che tutte le sue applicazioni sono scritte in Java.

2.1.1 Utilizzo

Java viene utilizzato come linguaggio di programmazione per creare applicazioni mobile per il sistema operativo Android per questo verrà utilizzato dal team.

2.2 Android

Android è il sistema operativo più diffuso al mondo, usato per smartphone, tablet e altri dispositivi, come Android TV e Google Glass, sviluppati da Google, la proprietaria del sistema operativo.

Strutturalmente deriva da Linux, ma usa Java per le applicazioni che interagiscono con l'utente. In pratica il linguaggio di programmazione principale previsto è Java, più supportato e ricco di funzionalità già fornite di base, mentre in genere si usano C e C++ per ottenere prestazioni migliori, sebbene siano presenti meno librerie rispetto al primo linguaggio. Esiste comunque la possibilità di usare entrambi per la stessa applicazione, ad esempio utilizzando Java per la struttura principale e C++ per le parti più pesanti da eseguire. Il $team_{\mathrm{g}}$ userà solo Java, dato che il programma sviluppato dovrebbe essere abbastanza leggero da eseguire.
Android inoltre fornisce delle proprie librerie, che, insieme alla struttura del sistema operativo, caratterizzano la stesura del codice dell'applicazione. Ad esempio ogni programma viene eseguito in una macchina virtuale propria, così la sua esecuzione non può modificare né il sistema operativo né le altre applicazioni, mentre la comunicazione con gli altri programmi può avvenire soltanto tramite dei pattern specifici. La struttura del programma sfrutta altri pattern già definiti, come le Activity, che caratterizzano lo sviluppo dell'applicazione. Android quindi presenta vantaggi e svantaggi differenti rispetto a Java.

- **Vantaggi:**
  - è il sistema operativo più usato al mondo;
  - è libero e gratuito;
  - è disponibile online una documentazione molto dettagliata sulle librerie di Android;
  - per sviluppare un'applicazione Android bisogna usare dei pattern già definiti, semplificando il codice da scrivere e lo studio della loro applicazione;
  - dato che Android è stato pensato per i dispositivi mobile la connessione ad Internet, il GPS e le altre interazioni con sistemi di comunicazione esterni all'applicazione sono automatici, quindi il programmatore non deve conoscere come siano implementatoi ma soltanto come interagirci tramite il codice;
  - l'utilizzo di altri linguaggi come complemento a Java, ad esempio l'XML per le GUI, semplifica il lavoro da svolgere e lo rende più intuitivo;
  - dato che il programma viene eseguito in una macchina virtuale non può in alcun modo danneggiare né il sistema operativo né le altre applicazioni;
  - Android non viene usato solo per dispositivi mobile, di conseguenza è possibile adattare o creare un programma per gli altri prodotti Google che usano questo sistema operativo;
  - sono presenti numerose librerie, di cui molte sono ufficiali di Android.
- **Svantaggi:**
  - dato che Android è open source ogni azienda che lo usa tende a personalizzarlo, quindi la stessa applicazione può dare risultati leggermente diversi cambiando dispositivo, di conseguenza la programmazione potrebbe dover variare per rispondere a queste differenze;
  - l'utilizzo di linguaggi complementari, come l'XML e il C/C++, può richiedere una maggiore conoscenza per poterli sfruttare, aumentando quindi il carico di lavoro necessario per poter scrivere l'applicazione.

2.2.1 Utilizzo

Il team ha deciso di utilizzare Android come linguaggio per la creazione dell'applicazione del progetto, applicazione che costituisce la parte Client del progetto.

2.3 XML

XML è un metalinguaggio per la definizione di linguaggi di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.

- **Vantaggi:**
  - è un linguaggio molto diffuso;
  - è un linguaggio libero e gratuito;
  - è disponibile online una documentazione molto dettagliata;
  - permette di definire dei propri tag e di fissare il loro contenuto, evitando così problemi di interpretazione dei termini, anche se nel caso della nostra applicazione questo vantaggio è irrilevante, visto che siamo noi ad usare dei tag già definiti da Google;
  - i file XML vengono gestiti da quasi tutti i linguaggi di programmazione, permettendo così una completa integrazione con le applicazioni.
- **Svantaggi:**
  - l'utilizzo degli schemi per fissare il tipo di contenuto dei tag può rendere il controllo dei contenuti molto pesante;
  - anche il file XML stesso può aumentare parecchio le dimensioni, arrivando addirittura ad un incremento esponenziale, sebbene esistano delle tecniche di compressione che permettono di ridurre notevolmente la grandezza del file;
  - il file XML è poco significativo finché non viene usato un programma per interpretare i tag previsti e di conseguenza eseguire delle azioni ben precise.

2.3.1 Utilizzo

In particolare nello sviluppo di applicazioni Android la parte grafica viene scritta proprio attraverso file XML, quindi questo linguaggio è essenziale per lo sviluppo del nostro progetto.

2.4 JSON

JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati. È scritto in JavaScript, un linguaggio molto usato nel mondo del Web che ha permesso a JSON di diffondersi rapidamente. Viene usato al posto di XML, anche se quest'ultimo nasce come linguaggio di markup e non specificatamente per lo scambio di dati. Per le persone è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi.

- **Vantaggi:**
  - è un linguaggio molto diffuso;
  - è un linguaggio libero e gratuito;
  - è disponibile online una documentazione molto dettagliata;
  - è conciso e facile da leggere;
  - è leggero, garantendo delle prestazioni migliori per le richieste e le risposte tra client e server;
  - è supportato da tutti i browser che implementano Javascript, dato che è scritto con questo linguaggio;

  - permette di distinguere i tipi di dato inviati, ad esempio può specificare se viene inviato un booleano, un numero intero o una stringa;
  - molti linguaggi di programmazione possono gestire JSON e le relative comunicazione tra client e server tramite delle librerie apposite.

- **Svantaggi:**
  - è un linguaggio efficiente solo per lo scambio di dati tra client e server, quando serve in locale esistono sistemi più efficienti;
  - rispetto a XML ha funzionalità più limitate, anche se generalmente le caratteristiche mancanti non sono necessarie per svolgere il lavoro previsto;
  - non è validato, quindi se il mittente omette campi o sbaglia il formato dei dati, client diversi potrebbero interpretare diversamente il JSON.

2.4.1 Utilizzo

Nel progetto il team utilizzerà questo formato per lo scambio dei dati tra Server e Client.

2.5 JavaScript

Javascript è un linguaggio di scripting orientato agli oggetti e agli eventi, utilizzato soprattutto nella programmazione Web. Un linguaggio di scripting è un tipo di linguaggio che viene integrato all'interno di un altro programma, nel caso del Web ad esempio l'interprete Javascript viene ospitato dal browser, che esegue il codice delle pagine quando esse vengono chiamate.

- **Vantaggi:**
  - è un linguaggio molto diffuso;
  - è un linguaggio libero e gratuito;
  - è disponibile online una documentazione molto dettagliata;
  - dato che il codice viene eseguito in locale dal client il trasferimento dei dati risulta leggero e al server basta eseguire poche istruzioni;
  - è semplice e meno restrittivo di altri linguaggi famosi, come C++ e Java, perché ad esempio le variabili non sono tipizzate;
  - molti linguaggi di programmazione integrano e gestiscono Javascript tramite delle librerie apposite.

- **Svantaggi:**
  - non è un linguaggio sicuro, perché ad esempio permette di eseguire azioni malevole con un semplice script, anche se al giorno d'oggi questo problema è stato risolto in parte dai browser, usando accorgimenti come il blocco dei pop-up;
  - essendo un linguaggio meno restrittivo la stesura del codice Javascript è più esposto agli errori di programmazione, ad esempio il cast automatico di variabili può provocare dei comportamenti inaspettati dello script in certe condizioni;
  - nonostante esista uno standard universale di Javascript, le istruzioni implementate dai browser possono essere leggermente diverse a seconda del tipo o addirittura della versione del browser stesso.

2.5.1 Utilizzo

Nell'applicazione sviluppata dal $team_{\text{g}}$ questo linguaggio viene usato per impostare le risposte del server da inviare al client, utilizzando JSON per il trasferimenti dei dati e per effettuare le query per la gestione dei dati del database.

2.6 SQL

SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:

- creare e modificare schemi di database;
- inserire, modificare e gestire dati memorizzati;
- interrogare i dati memorizzati;
- creare e gestire strumenti di controllo ed accesso ai dati.

È costruito per essere semplice, sia in scrittura sia in lettura, e poco verboso, e per permettere automaticamente una gestione ottimale dei dati.

- **Vantaggi:**
  - è un linguaggio molto diffuso;
  - è un linguaggio libero e gratuito;
  - è disponibile online una documentazione molto dettagliata;
  - è semplice da utilizzare per eseguire qualsiasi delle sue operazioni;
  - è efficiente, tutte le azioni che si possono eseguire sono leggere e veloci;
  - molti linguaggi di programmazione permettono l'interrogazione dei database SQL tramite delle apposite librerie;
  - è un linguaggio molto restrittivo, garantendo così un'alta solidità del database.
- **Svantaggi:**
  - essendo un linguaggio molto restrittivo SQL non permette di eseguire alcune operazioni avanzate, e inoltre un cambiamento, anche piccolo, del database può risultare più difficile del previsto, ad esempio perché crea temporaneamente un'incongruenza delle chiavi primarie;
  - il database SQL non può essere partizionato e suddiviso in più parti, perché risulterebbe molto difficile o addirittura impossibile effettuare i controlli di integrità del database, di conseguenza diventa problematico ingrandirlo quando è necessario memorizzare moltissimi dati.

2.6.1 Utilizzo

Nell'applicazione SQL verrà utilizzato per la creazione del data base e per la gestione, tramite query, dei dati.

2.7 Librerie

2.7.1 Kontakt.io Android Proximity SDK

Kontakt.io Android Proximity SDK è una libreria per Android che supportai formati iBeacon e Eddystone.
Fornisce gli strumenti per la gestione e la creazione di applicazioni che utilizzino i beacon kontakt.io

- **Vantaggi:**
  - semplici da utilizzare;
  - sviluppata da un'azienda giovane che garantisce molti aggiornamenti;.
- **Svantaggi:**
  - utilizza due formati proprietari;
  - supportano solo i beacon kontakt.io.

2.7.1.1 Utilizzo

Nel nostro progetto l'interazione coi beacon verrà utilizzata durante lo svolgimento dei percorsi, per questo la libreria Kontakt verrà utilizzata dalle classi all'interno del package `pathprogress`, componente che controlla l'avanzamento dell'utente nel percorso che sta svolgendo.

2.7.2 Volley

Volley è una libreria Android che permette e facilita la comunicazione tra client e server tramite HTTP. Effettua automaticamente le chiamate, asincrone o sincrone, al server, in base alle impostazioni fornite dal programmatore. È nata per sopperire alla mancanza di librerie simili in Java, dove le uniche classi presenti per effettuare chiamate REST sono obsolete e non esenti da errori. Il risultato ottenuto è rappresentato da una libreria molto veloce, leggera e soprattutto con un'eccellente gestione della memoria.

- **Vantaggi:**
  - è una libreria libera e gratuita;
  - è una libreria ufficiale di Android, quindi il suo funzionamento e il suo mantenimento vengono garantiti;
  - sono disponibili online delle spiegazioni su come farla funzionare ed usare;
  - imposta e gestisce automaticamente le chiamate da effettuare al server, quindi il programmatore deve solo fornire i dati necessari;
  - le chiamate che effettua possono essere già asincrone, quindi il programmatore non deve preoccuparsi di gestire i thread;
  - permette di inviare vari tipi di oggetti, tra cui quelli scritti in JSON;
  - gestisce molto bene la memoria in modo automatico, è veloce e leggera.
- **Svantaggi:**
  - le spiegazioni online a volte sono scarse o poco chiare;
  - Volley è adatto per le piccole comunicazioni con il server, ma quando è richiesto, ad esempio, di fare un upload abbastanza grande, la libreria mostra delle lacune, come la

mancanza di una barra di avanzamento. Nella nostra applicazione non dovrebbero esserci problemi del genere, dato che i dati scambiati sono abbastanza piccoli e le chiamate risultano essere semplici;

– non è la libreria che garantisce il minor tempo di attesa per inviare e ricevere una risposta, altre come Retrofit sono più performanti.

2.7.2.1 Utilizzo

Nel nostro progetto la libreria Volley è stata utilizzata per effettuare le chiamate dal Client al Server, infatti sono necessarie delle richieste per ottenere i dati dal database e per aggiornarli qual'ora un utente svolga dei percorsi e desideri salvarne il risultato. Nello specifico tutte le chiamate verranno effettuate dalle classi all'interno del package `urlrequest`.

2.7.3 Travis CI

Travis CI è un servizio di *Continuous Integration*$_g$ distribuita usato per compilare e testare le API server del progetto, ospitate da *GitHub*$_g$. La configurazione avviene attraverso un file `.travis.yaml` nella root directory del progetto che descrive le istruzioni da eseguire per compilare e testare il prodotto.

- **Vantaggi:**
  - è una servizio gratuito per le repository pubbliche;
  - ad ogni push di codice su GitHub l'intero progetto viene compilato e i test vengono eseguiti evidenziando subito se qualche commit ha introdotto errori o regressioni;
  - è ben integrato con *Slack*$_g$ cui invia le notifiche sul successo o fallimento dei test;
  - l'integrazione richiede solamente la stesura di un file *YAML* di piccole dimensioni.
- **Svantaggi:**
  - il linguaggio *YAML* per la definizione delle istruzioni non è conosciuto da alcun componente;
  - il linguaggio *YAML* è sì conciso ma allo stesso tempo può essere complesso.

2.7.3.1 Utilizzo

Nel nostro progetto, il file di configurazione di *Travis CI* si occupa di:

- installare *node* nella macchina ospite;
- installare le dipendenze del back-end;
- installare *mySQL*;
- creare la struttura del database;
- popolare il database con dei dati di test;
- avviare il server del backend;
- eseguire i test;
- notificare su Slack il successo o il fallimento delle operazioni precedenti.

2.7.4 Mocha

Mocha è un framework di test per *Node.js*$_g$. Attraverso *Mocha* è facile eseguire in serie dei test asincroni (caratteristica indispensabile per testare il backend) e si generano degli accurati report che aiutano ad identificare gli errori.

- **Vantaggi:**
  - è un framework molto diffuso per il test di progetti *JavaScript* e *Node.js* in particolare;
  - riduce notevolmente il codice necessario ad eseguire test asincroni;
  - non richiede l'integrazione di alcun pacchetto nel codice in produzione, solamente nelle classi che si occupano del test;
  - è perfettamente integrato con le Promise di JavaScript ampiamente utilizzate nel codice.
- **Svantaggi:**
  - è necessario leggere la documentazione presente nel sito ufficiale per comprenderne il funzionamento.

2.7.4.1 Utilizzo

Nel nostro progetto *Mocha* viene usato per i test del backend.

2.7.5 npm

npm è un package manager per *JavaScript*, utile per gestire le dipendenze e creare un pacchetto facilmente installabile

- **Vantaggi:**
  - permette di effettuare l'installazione di tutte le dipendenze del progetto con una sola istruzione;
  - è molto diffuso e comprende una vasta libreria di package.
- **Svantaggi:**
  - è necessario installare `npm` sul server.

2.7.5.1 Utilizzo

Il file `.package.json` nella root directory della repository nel server configura il pacchetto permettendo poi di usare i semplici comandi `npm install` per installare le dipendenze, `npm start` per avviare il server e `npm test` per lanciare la suite di test.

2.7.6 express

express è un package per il routing di chiamate del client verso le classi adibite a gestirle

- **Vantaggi:**
  - è facile da usare;
  - è molto diffuso ed è ben documentato.
- **Svantaggi:**
  - è necessario installare il pacchetto `express` sulla macchina.

2.7.6.1 Utilizzo

Ogni richiesta al server viene gestita da un'oggetto `server` di *express* che istanzia un oggetto della classe adibita a rispondere a tale richiesta.

3 Descrizione architettura

In questa sezione verrà descritta lo schema generale dell'architettura, nella sezione successiva verranno riportate tutti i packages e le classi dettagliatamente. Per i diagrammi delle classi e di attività è stato utilizzato il formalismo UML 2.0.
L'architettura descritta in questo documento è ad alto livello. Classi, sottoclassi e attributi verranno descritti più dettagliatamente nel periodo in cui attueremo la Progettazione di dettaglio.
Si procederà nella descrizione dell'architettura con un approccio top-down. Saranno descritte prima quindi le parti più generali per poi andare sempre più nello specifico. Si procederà quindi prima con la descizione dei packages e delle componenti, per poi passare alle singole classi. Successivamente verranno descritti i design pattern utilizzati.

3.1 Architettura generale

L'architettura generale dell'applicazione segue il modello client - server.
In particolare si utilizzerà lo stile architetturale REST (representational state transfer) per coordinare compomenti, connettori e dati attraverso un sistema ipermediale distribuito dove l'attenzione è data al ruolo delle componenti ed ai vincoli imposti dalle loro interazioni tra di essi.
Alcune caratteristiche e vincoli di REST sono i seguenti:

- REST offre una interfaccia che separa il client dal server. Questa separazione permette di avere ben chiaro quali siano i ruoli delle due componenti in modo che non ci siano conflitti tra le due. Questo porta beneficio anche alla portabilità e alla scalabilità del prodotto;
- la comunicazione tra client e server è stateless, il che significa che ogni richiesta del client sarà interpretabile dal server senza che esso utilizzi alcun contesto presente nel server;
- i risultati ottenuti da una richiesta client-server possono essere salvati in cache così che se il client avesse la necessità di riutilizzare dati già richiesti lo possa fare senza effettuare nuove richieste.

3.1.1 Client

Per la parte Client si è deciso di utilizzare il linguaggio Android per la creazione dell'applicazione con cui gli utenti interagiranno; verrà utilizzato il pattern architetturale Model View Presenter, spiegato in dettaglio nell'Appendice, per consentire la separazione logica delle varie componenti, permettendo di non dover modificare una componente logica qualora ne venga modificata un'altra, e facilitando lo sviluppo di unit test.
Il Client inoltre avrà un proprio database locale nel quale momorizzare temporaneamente i dati dell'utente e i dati richiesti al server, consentendo una maggiore fluidità limitando le chiamate al server.

3.2 Server

La parte server sarà costituita da un database in linguanggio SQL, per il salvataggio dei dati riguardanti l'utente, come salvataggi e dati per l'autenticazione, e tutte le informazioni sugli edifici che includono percorsi giocabili dall'utente con i relativi percorsi e classifiche.
Le interazioni tra Client e database verranno gestite da alcune classi progettate in Javascript, che gestiranno le richieste di dati e il loro aggiornamento.
Per il trasferimento di dati tra Server e Client verrà utilizzato il formato JSON.

3.3 Architettura del database

In particolare verrà utilizzato un database relazionale *MySQL*$_{\text{g}}$ con la possibilità di interagire con esso tramite l'applicazione web *phpMyAdmin*$_{\text{g}}$ in modo da semplificarne l'amministrazione. Le rappresentazione del database verrà illustrata nel documento "*Definizione di Prodotto v2.0.0*".

3.3.1 Componente `CLIPS`

Informazioni sul package

- **Descrizione**: package generale contenente il prodotto del progetto
- **Package contenuti**:

  CLIPS::client: componente globale per il front end del prodotto che utilizza il design pattern MVC_{g}. Si occupa di fornire un'interfaccia grafica dell'applicazione e di interagire con il lato server.

- – CLIPS::server: componente globale per il back end del prodotto

3.3.2 Componente `CLIPS::client`

Informazioni sul package

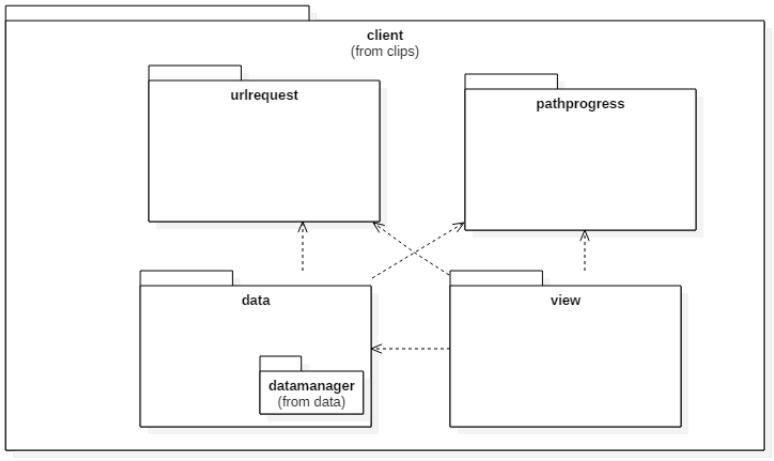

Figura 1: Schema package client

- **Descrizione**: componente globale per il front end del prodotto che utilizza il design pattern MVC_{g}. Si occupa di fornire un'interfaccia grafica dell'applicazione e di interagire con il lato server.
- **Padre**: CLIPS
- **Package contenuti**:

  CLIPS::client::data: package per la gestione in locale dei dati

- – CLIPS::client::pathprogress: componente che gestisce i dati del percorso e salva i risultati ottenuti nelle prove mentre si sta giocando
  – CLIPS::client::viewcontroller: componente che raggruppa tutte le view ed i controller relativi alle view

3.3.3 Componente `CLIPS::client::data`

Informazioni sul package

- **Descrizione**: package per la gestione in locale dei dati
- **Padre**: client

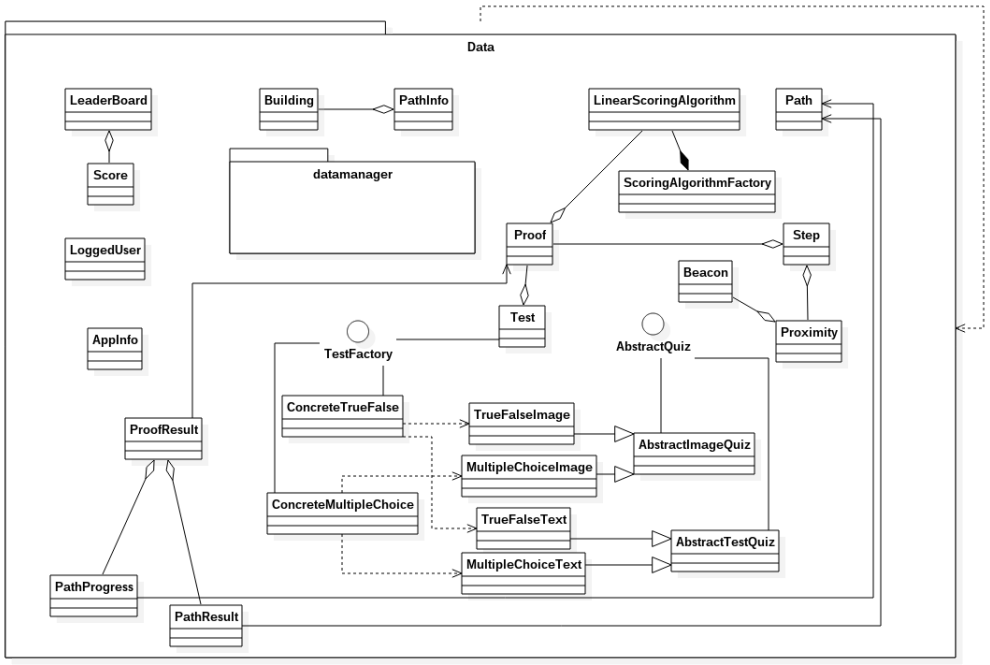

Figura 2: Schema sintetico package client::data

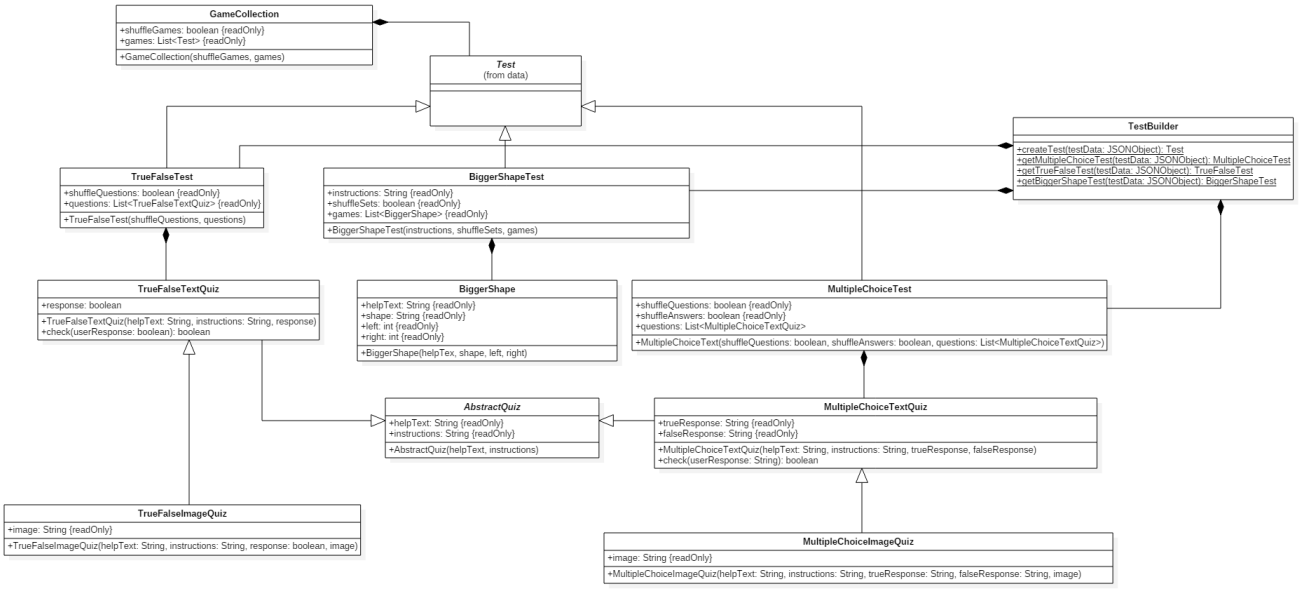

Figura 3: Prima parte schema package client::data

- **Package contenuti**:

  CLIPS::client::data::datamanager: componente che gestisce i dati in locale

- – CLIPS::client::data::urlrequest: componente che si occupa di effettuare le richieste al server

- **Classi**:

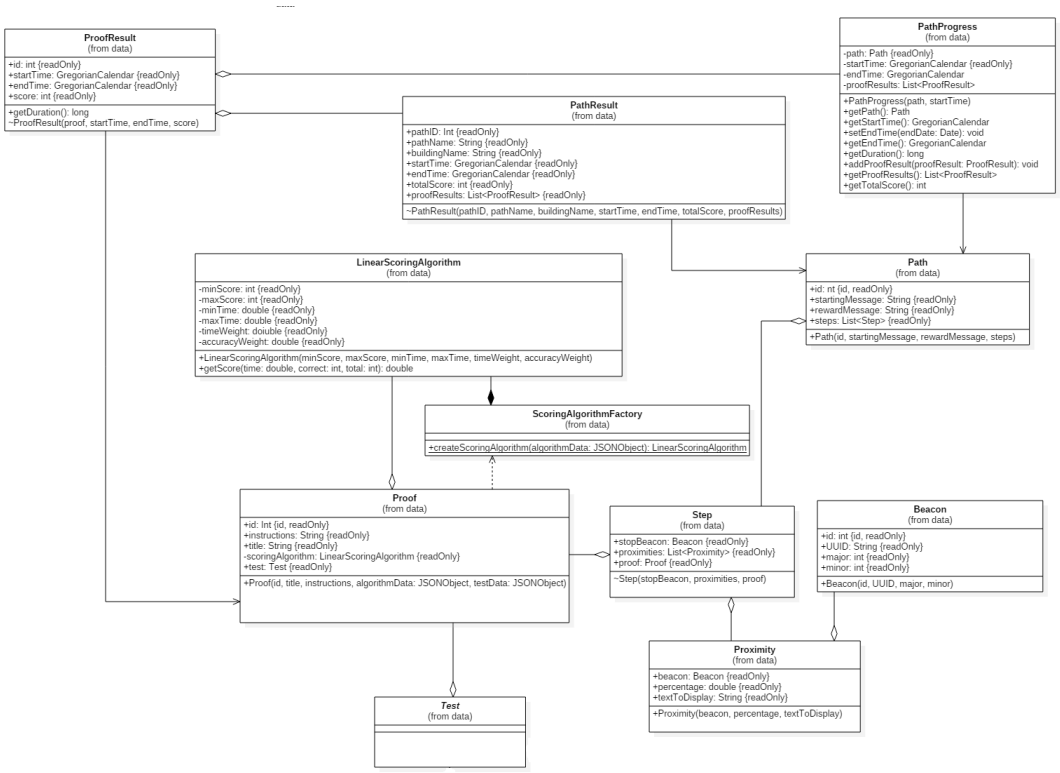

Figura 4: Seconda parte schema package client::data

`CLIPS::client::data::/textitAbstractQuiz`

- **Descrizione**: classe astratta per la definizione di un quiz da proporre all'utente
- **Utilizzo**: classe astratta da cui far derivare tutte le classi che rappresentano un quiz

`CLIPS::client::data::/textitTest`

- **Descrizione**: classe astratta che rappresenta la definizione di un test da proporre all'utente
- **Utilizzo**: classe astratta per rappresentare un test, ovvero un gruppo di giochi generalmente dello stesso tipo

`CLIPS::client::data::AppInfo`

- **Descrizione**: classe per la rappresentazione delle informazioni generali dell'applicazione
- **Utilizzo**: permette di salvare le informazioni generali relative all'applicazione

`CLIPS::client::data::Beacon`

- **Descrizione**: classe che rappresenta i dati di un beacon in locale

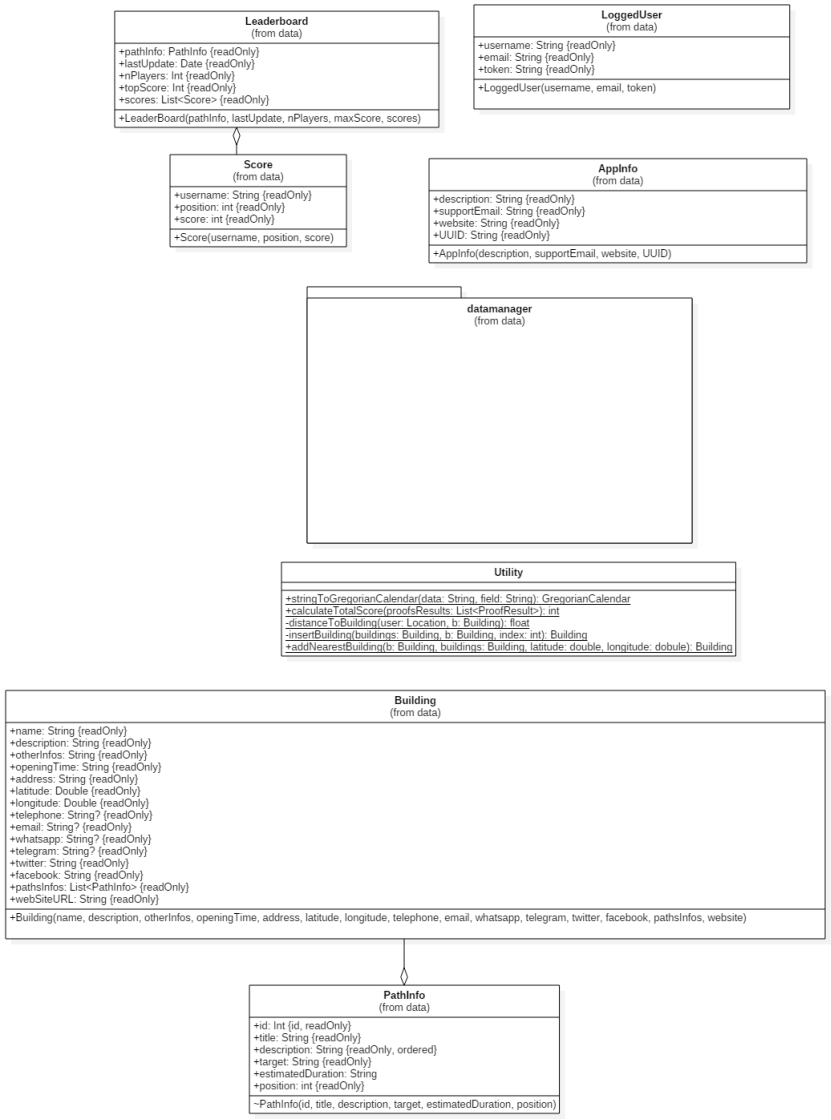

Figura 5: Terza parte schema package client::data

* **Utilizzo**: permette di salvere in locale le informazioni di un beacon

`CLIPS::client::data::BiggerShape`

* **Descrizione**: classe che rappresenta un gioco in cui bisogna selezionare la figura più grande tra le due che appaiono

* **Utilizzo**: classe per rappresentare un gioco in cui l'utente deve selezionare la figura più grande tra le due che compaiono

`CLIPS::client::data::BiggerShapeTest`

* **Descrizione**: classe che rappresenta un test di BiggerShape

* **Utilizzo**: classe per rappresentare un test di BiggerShape

* **Relazioni con altre classi**:
  - CLIPS::client::data::BiggerShape: classe che rappresenta un gioco in cui bisogna selezionare la figura più grande tra le due che appaiono

`CLIPS::client::data::Building`

* **Descrizione**: classe che rappresenta i dati di un edificio in cui è possibile utilizzare l'applicazione

* **Utilizzo**: permette di memorizzare i dati di un edificio in locale

* **Relazioni con altre classi**:
  - CLIPS::client::data::PathInfo: classe che si occupa di salvare in locale le informazioni generali di un percorso

`CLIPS::client::data::GameCollection`

* **Descrizione**: classe che rappresenta un insieme di test

* **Utilizzo**: classe per rappresentare un test in cui sono contenuti più tipi di giochi, di fatto è un test che contiene altri test

* **Relazioni con altre classi**:
  - CLIPS::client::data::/textitTest: classe astratta che rappresenta un test

`CLIPS::client::data::LeaderBoard`

* **Descrizione**: classe che rappresenta la classifica di un percorso di un determinato edificio

* **Utilizzo**: consente di salvare i dati della classifica in locale

* **Relazioni con altre classi**:
  - CLIPS::client::data::PathInfo: classe che si occupa di salvare in locale le informazioni generali di un percorso
  - CLIPS::client::data::Score: classe che rappresenta il risultato nella classifica di uno specifico percorso svolto dall'utente

`CLIPS::client::data::LinearScoringAlgorithm`

* **Descrizione**: classe per calcolare il punteggio di una prova

* **Utilizzo**: si occupa di calcolare il punteggio utilizzando un algoritmo lineare

`CLIPS::client::data::LoggedUser`

* **Descrizione**: classe che si occupa di memorizzare in locale i dati dell'utente che ha eseguito l'accesso

* **Utilizzo**: permette il salvataggio in locale dei dati di un utente loggato

`CLIPS::client::data::MultipleChoiceImageQuiz`

* **Descrizione**: classe che rappresenta una domanda a risposta multipla con immagine

* **Utilizzo**: classe per rappresentare una domanda a risposta multipla in cui viene usata un'immagine per la formulazione del quesito

`CLIPS::client::data::MultipleChoiceTest`

* **Descrizione**: classe che rappresenta un test di domande a risposta multiple

* **Utilizzo**: classe per rappresentare un test di domande a risposta multipla, con immagine o senza

* **Relazioni con altre classi**:
  - CLIPS::client::data::MultipleChoiceTextQuiz: classe che rappresenta una domanda a risposta multipla

`CLIPS::client::data::MultipleChoiceTextQuiz`

* **Descrizione**: classe che rappresenta una domanda a risposta multipla

* **Utilizzo**: classe per rappresentare una domanda a risposta multipla il cui contenuto è solo testo

`CLIPS::client::data::Path`

* **Descrizione**: classe che si occupa di salvare in locale i dati riguardanti un percorso di un determinato edificio

* **Utilizzo**: permette di salvare i dati di un percorso in locale

* **Relazioni con altre classi**:
  - CLIPS::client::data::Step: classe che rappresenta in locale lo spostamento da una prova a quella successiva

`CLIPS::client::data::PathInfo`

* **Descrizione**: classe che si occupa di salvare in locale le informazioni generali di un percorso

* **Utilizzo**: consente di salvare in locale le informazioni generali di un percorso

`CLIPS::client::data::PathProgress`

* **Descrizione**: classe per il salvataggio in locale del progresso di un percorso svolto dall'utente

* **Utilizzo**: permette di salvare in locale i risultati delle prove durante lo svolgimento del percorso

* **Relazioni con altre classi**:
  - CLIPS::client::data::Path: classe che si occupa di salvare in locale i dati riguardanti un percorso di un determinato edificio
  - CLIPS::client::data::ProofResult: classe che rappresenta il risultato di una prova svolta dall'utente

`CLIPS::client::data::PathResult`

* **Descrizione**: classe che rappresenta i risultati di un percorso svolto dall'utente

* **Utilizzo**: permette di salvare i dati di un percorso in locale

* **Relazioni con altre classi**:
  - CLIPS::client::data::ProofResult: classe che rappresenta il risultato di una prova svolta dall'utente

`CLIPS::client::data::Proof`

* **Descrizione**: classe che rappresenta una prova da proporre all'utente che sta svolgendo un percorso

* **Utilizzo**: salva in locale i dati della prova da far visualizzare all'utente
* **Relazioni con altre classi**:
  - CLIPS::client::data::/textitTest: classe astratta che rappresenta un test
  - CLIPS::client::data::LinearScoringAlgorithm: classe per calcolare il punteggio di una prova
  - CLIPS::client::data::ScoringAlgorithmFactory: classe base per la creazione degli algoritmi per il calcolo del punteggio di una determinata prova

`CLIPS::client::data::ProofResult`

* **Descrizione**: classe che rappresenta il risultato di una prova svolta dall'utente

* **Utilizzo**: permette di salvare in locale il risultato di una prova

* **Relazioni con altre classi**:
  - CLIPS::client::data::PathProgress: classe per il salvataggio in locale del progresso di un percorso svolto dall'utente
  - CLIPS::client::data::PathResult: classe che rappresenta i risultati di un percorso svolto dall'utente
  - CLIPS::client::data::Proof: classe che rappresenta una prova del percorso

`CLIPS::client::data::Proximity`

* **Descrizione**: classe che rappresenta in locale un beacon dedicato alla segnalazione della distanza dalla prossima prova

* **Utilizzo**: consente di salvare un beacon e la sua distanza dal beacon della prossima prova

* **Relazioni con altre classi**:
  - CLIPS::client::data::Beacon: classe che rappresenta un beacon in locale

`CLIPS::client::data::Score`

* **Descrizione**: classe che rappresenta il risultato nella classifica dell'utente

* **Utilizzo**: permette di memorizzare il punteggio e la posizione in classifica dell'utente

`CLIPS::client::data::ScoringAlgorithmFactory`

* **Descrizione**: classe base per la creazione degli algoritmi di calcolo del punteggio di una determinata prova

* **Utilizzo**: fornisce il metodo per la creazione di algoritmi per il calcolo dei punteggi

* **Relazioni con altre classi**:
  - CLIPS::client::data::LinearScoringAlgorithm: classe per calcolare il punteggio di una prova

`CLIPS::client::data::Step`

* **Descrizione**: classe che rappresenta in locale una frazione del percorso da svolgere

* **Utilizzo**: permette di salvare in locale le informazioni che rappresentano la ricerca della nuova prova

* **Relazioni con altre classi**:
  - CLIPS::client::data::Beacon: classe che rappresenta un beacon in locale
  - CLIPS::client::data::Proof: classe che rappresenta una prova del percorso
  - CLIPS::client::data::Proximity: classe che rappresenta in locale un beacon dedicato alla segnalazione della distanza dalla prossima prova

`CLIPS::client::data::TestBuilder`

* **Descrizione**: classe per la creazione degli oggetti Test partendo dagli oggetti JSON ricevuti dal server

* **Utilizzo**: permette di costruire gli oggetti Test partendo dagli oggetti JSON ricevuti dal server

* **Relazioni con altre classi**:

- CLIPS::client::data::BiggerShape: classe che rappresenta un gioco in cui bisogna selezionare la figura più grande tra le due che appaiono
- CLIPS::client::data::BiggerShapeTest: classe che rappresenta un test di BiggerShape
- CLIPS::client::data::GameCollection: classe che rappresenta un insieme di test
- CLIPS::client::data::GameCollection: classe che rappresenta un insieme di test
- CLIPS::client::data::MultipleChoiceImageQuiz: classe che rappresenta una domanda a risposta multipla con immagine
- CLIPS::client::data::MultipleChoiceImageQuiz: classe che rappresenta una domanda a risposta multipla con immagine
- CLIPS::client::data::MultipleChoiceTest: classe che rappresenta un test di domande a risposta multiple
- CLIPS::client::data::MultipleChoiceTest: classe che rappresenta un test di domande a risposta multiple
- CLIPS::client::data::MultipleChoiceTextQuiz: classe che rappresenta una domanda a risposta multipla
- CLIPS::client::data::MultipleChoiceTextQuiz: classe che rappresenta una domanda a risposta multipla
- CLIPS::client::data::TrueFalseImageQuiz: classe che rappresenta una domanda vero o falso con immagine
- CLIPS::client::data::TrueFalseImageQuiz: classe che rappresenta una domanda vero o falso con immagine
- CLIPS::client::data::TrueFalseTest: classe che rappresenta un test di domande vero o falso
- CLIPS::client::data::TrueFalseTest: classe che rappresenta un test di domande vero o falso
- CLIPS::client::data::TrueFalseTextQuiz: classe controller che si occupa di interagire con TrueFalseView
- CLIPS::client::data::TrueFalseTextQuiz: classe controller che si occupa di interagire con TrueFalseView

`CLIPS::client::data::TrueFalseImageQuiz`

* **Descrizione**: classe che rappresenta una domanda vero o falso con immagine
* **Utilizzo**: classe per rappresentare una domanda vero o falso in cui viene usata un'immagine per la formulazione del quesito

`CLIPS::client::data::TrueFalseTest`

* **Descrizione**: classe che rappresenta un test di domande vero o falso
* **Utilizzo**: classe per rappresentare un test di domande vero o falso, con immagine o senza
* **Relazioni con altre classi**:

· CLIPS::client::data::TrueFalseTextQuiz: classe controller che si occupa di interagire con TrueFalseView

`CLIPS::client::data::TrueFalseTextQuiz`

* **Descrizione**: classe controller che si occupa di interagire con TrueFalseView

* **Utilizzo**: si occupa di gestire le interazioni dell'utente con TrueFalseView

`CLIPS::client::data::Utility`

* **Descrizione**: classe contente vari metodi di utilità per altre classi

* **Utilizzo**: classe contente vari metodi di utilità per altre classi

* **Relazioni con altre classi**:
  · CLIPS::client::data::Building: classe che rappresenta un edificio
  · CLIPS::client::data::ProofResult: classe che rappresenta il risultato di una prova

3.3.4 Componente `CLIPS::client::data::datamanager`

Informazioni sul package

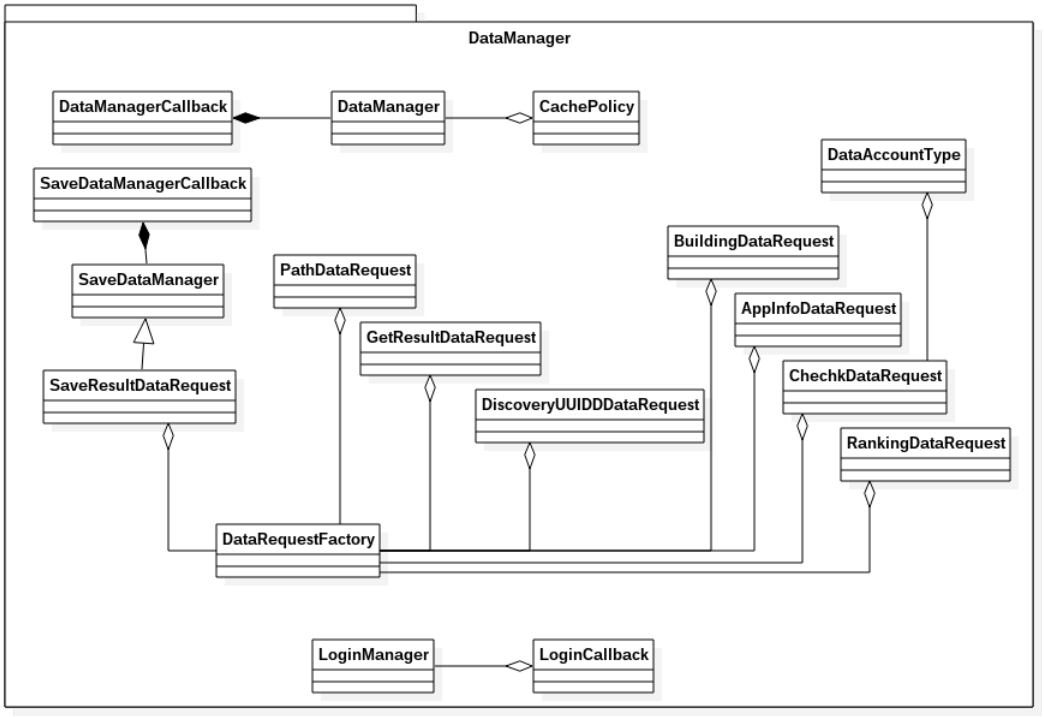

Figura 6: Schema sintetico package client::data::datamanager

- **Descrizione**: componente che gestisce i dati in locale
- **Padre**: data

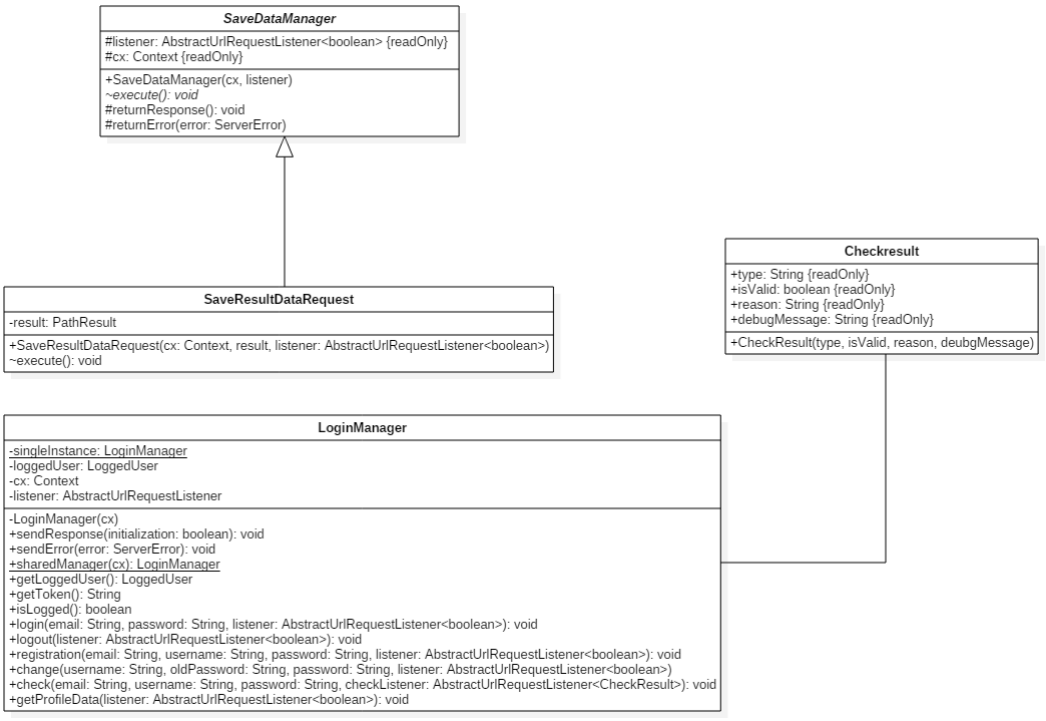

Figura 7: Prima parte schema package client::data::datamanager

- **Classi**:

  `CLIPS::client::data::datamanager::/textitSaveDataManager`

  * **Descrizione**: classe astratta per il salvataggio di dati nel server

  * **Utilizzo**: permette di effettuare il salvataggio dei dati nel server

  * **Relazioni con altre classi**:

    · CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager

    · CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

  `CLIPS::client::data::datamanager::AbstractDataManagerListener`

  * **Descrizione**: interfaccia che definisce la struttura del listener astratto usato nel package datamanager

  * **Utilizzo**: permette alle classi che interagiscono con il datamanager di definire un proprio listener concreto da poter usare per ottenere le informazioni richieste

  * **Relazioni con altre classi**:

    · CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

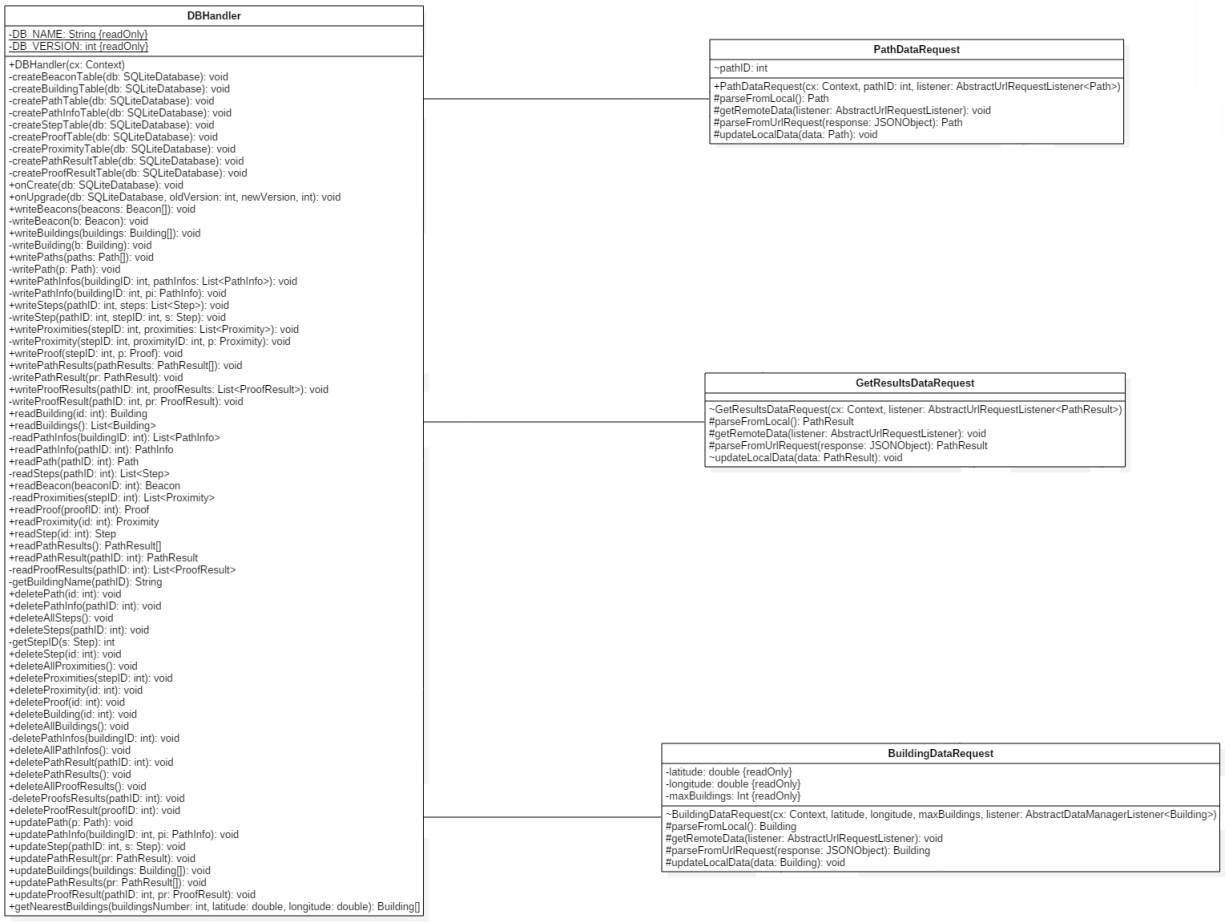

Figura 8: Seconda parte schema package client::data::datamanager

`CLIPS::client::data::datamanager::AppInfoDataRequest`

* **Descrizione**: classe concreta che gestisce le richieste dei dati sulle info dell'app
* **Utilizzo**: permette di ottenere i dati riguardanti le informazioni generali dell'app
* **Relazioni con altre classi**:
  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::AppInfo: classe per la rappresentazione delle informazioni dell'applicazione
  - CLIPS::client::data::urlrequest::RequestMaker: classe per la costruzione e l'invio di richieste al server, funziona come un'interfaccia dell'urlrequest per il resto dell'applicazione e quindi regola l'interazione con il package

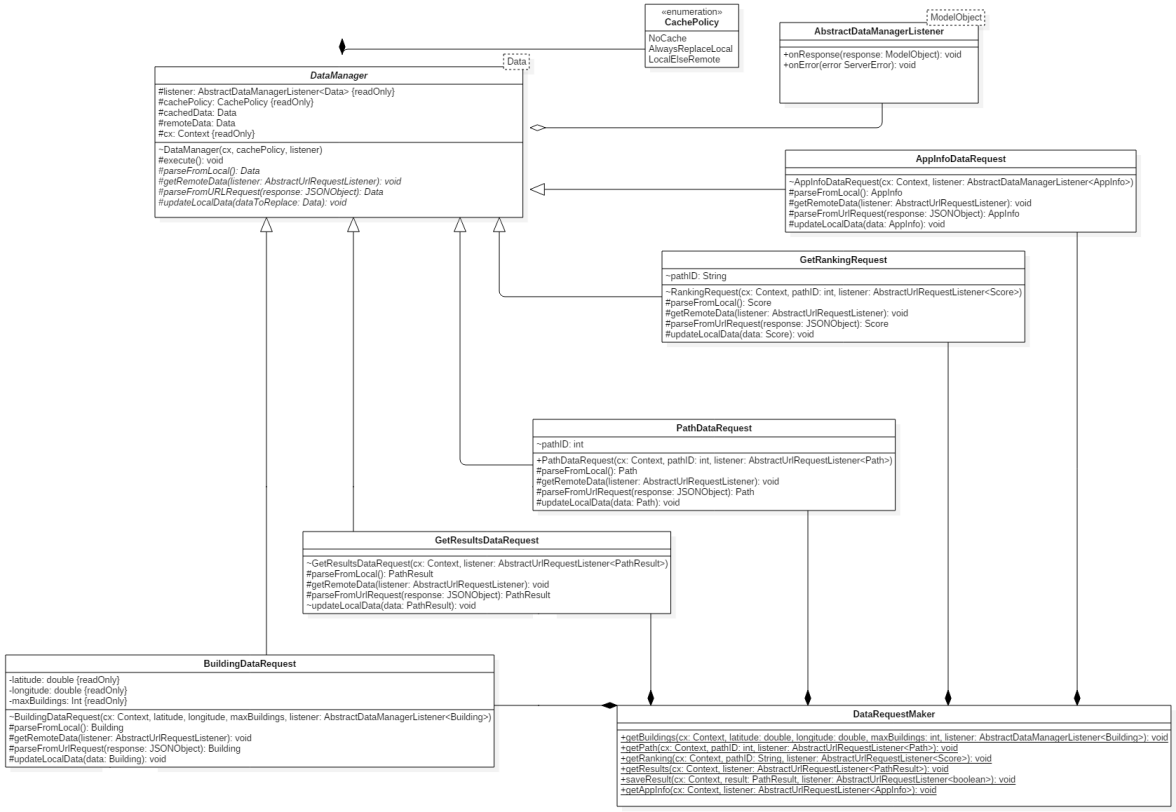

Figura 9: Terza parte schema package client::data::datamanager

- CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::data::datamanager::BuildingsDataRequest`

* **Descrizione**: classe concreta che gestisce le richieste al server per ottenere i dati riguardanti gli edifici abilitati ad utilizzare l'applicazione

* **Utilizzo**: permette di ottenere i dati sugli edifici

* **Relazioni con altre classi**:
  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::Building: classe che rappresenta un edificio
  - CLIPS::server::urlrequesthandler::DBHandler: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex
  - CLIPS::client::data::PathInfo: classe che si occupa di salvare in locale le informazioni generali di un percorso
  - CLIPS::client::data::urlrequest::RequestMaker: classe per la costruzione e l'invio di richieste al server, funziona come un'interfaccia dell'urlrequest per il resto dell'applicazione e quindi regola l'interazione con il package

- CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::data::datamanager::CheckResult`

* **Descrizione**: classe che contiene il risultato del check per il dato specificato

* **Utilizzo**: contiene tutte le informazioni sul dato controllato dal server, ovvero se è valido e il messaggio d'errore quando ne viene rilevato uno

* **Relazioni con altre classi**:
  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::data::datamanager::DataRequestMaker`

* **Descrizione**: classe che effettua le operazioni di ricerca e salvataggio dei dati

* **Utilizzo**: permette di effettuare operazioni di salvataggio dei dati nel server e di ricerca dei dati, richiedendoli al server o cercandoli in locale

* **Relazioni con altre classi**:
  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::AppInfo: classe per la rappresentazione delle informazioni dell'applicazione
  - CLIPS::client::data::datamanager::AppInfoDataRequest: classe concreta che gestisce le richieste dei dati sulle info dell'app
  - CLIPS::client::data::Building: classe che rappresenta un edificio
  - CLIPS::client::data::datamanager::BuildingsDataRequest: classe concreta che gestisce le richieste dei dati sugli edifici
  - CLIPS::client::data::datamanager::GetRankingDataRequest: classe concreta che gestisce le richieste dei dati della classifica
  - CLIPS::client::data::datamanager::GetResultsDataRequest: classe concreta che gestisce le richieste dei dati dei risultati
  - CLIPS::client::data::Path: classe che si occupa di salvare in locale i dati riguardanti un percorso
  - CLIPS::client::data::datamanager::PathDataRequest: classe concreta che gestisce le richieste dei dati sui percorsi
  - CLIPS::client::data::PathResult: classe che rappresenta i risultati di un percorso
  - CLIPS::client::data::datamanager::SaveResultDataRequest: classe concreta che effettua le richieste per salvare il risultato del percorso appena concluso

- CLIPS::client::data::Score: classe che rappresenta il risultato nella classifica dell'utente

`CLIPS::client::data::datamanager::DBHandler`

* **Descrizione**: classe che si occupa della gestione del database locale

* **Utilizzo**: utilizzando questa classe è possibile interagire con il database locale per ottenere e salvare i dati desiderati

`CLIPS::client::data::datamanager::GetRankingDataRequest`

* **Descrizione**: classe concreta che gestisce le richieste dei dati della classifica

* **Utilizzo**: permette di ottenere i dati della classifica

* **Relazioni con altre classi**:

  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::RequestMaker: classe per la costruzione e l'invio di richieste al server, funziona come un'interfaccia dell'urlrequest per il resto dell'applicazione e quindi regola l'interazione con il package
  - CLIPS::client::data::Score: classe che rappresenta il risultato nella classifica dell'utente
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::data::datamanager::GetResultsDataRequest`

* **Descrizione**: classe concreta che gestisce le richieste dei dati dei risultati delle prove

* **Utilizzo**: permette di ottenere i dati dei risultati

* **Relazioni con altre classi**:

  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::server::urlrequesthandler::DBHandler: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex
  - CLIPS::client::data::PathResult: classe che rappresenta i risultati di un percorso
  - CLIPS::client::data::ProofResult: classe che rappresenta il risultato di una prova
  - CLIPS::client::data::urlrequest::RequestMaker: classe per la costruzione e l'invio di richieste al server, funziona come un'interfaccia dell'urlrequest per il resto dell'applicazione e regola l'interazione con il package

- CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::data::datamanager::LoginManager`

* **Descrizione**: classe che gestisce le operazioni del profilo dell'utente

* **Utilizzo**: permette di effettuare il login, il logout, la registrazione, il check dei dati inseriti, l'aggiornamento dei dati salvati in locale e la modifica dei dati del profilo

* **Relazioni con altre classi**:
  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::datamanager::CheckResult: classe che contiene il risultato del check per il dato specificato
  - CLIPS::client::data::LoggedUser: classe che si occupa di memorizzare in locale i dati dell'utente loggato
  - CLIPS::client::data::urlrequest::RequestMaker: classe per la costruzione e l'invio di richieste al server, funziona come un'interfaccia dell'urlrequest per il resto dell'applicazione e regola l'interazione con il package
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::data::datamanager::PathDataRequest`

* **Descrizione**: classe concreta che gestisce le richieste dei dati sui percorsi di un edificio

* **Utilizzo**: permette di ottenere i dati sui percorsi

* **Relazioni con altre classi**:
  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::Beacon: classe che rappresenta un beacon in locale
  - CLIPS::server::urlrequesthandler::DBHandler: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex
  - CLIPS::client::data::Path: classe che si occupa di salvare in locale i dati riguardanti un percorso
  - CLIPS::client::data::Proof: classe che rappresenta una prova del percorso
  - CLIPS::client::data::Proximity: classe che rappresenta in locale un beacon indicato alla segnalazione della distanza dalla prossima prova
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

- CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo
- CLIPS::client::data::Step: classe che rappresenta in locale lo spostamento da una prova a quella successiva

`CLIPS::client::data::datamanager::SaveResultDataRequest`

* **Descrizione**: classe concreta che effettua le richieste per salvare il risultato del percorso appena concluso
* **Utilizzo**: permette di salvare in remoto il risultato del percorso appena svolto
* **Relazioni con altre classi**:
  - CLIPS::client::data::datamanager::AbstractDataManagerListener: interfaccia che definisce la struttura del listener astratto usato nel package datamanager
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::PathResult: classe che rappresenta i risultati di un percorso
  - CLIPS::client::data::urlrequest::RequestMaker: classe per la costruzione e l'invio di richieste al server, funziona come un'interfaccia dell'urlrequest per il resto dell'applicazione e quindi regola l'interazione con il package
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

3.3.5 Componente `CLIPS::client::data::urlrequest`

Informazioni sul package

- **Descrizione**: componente che si occupa di effettuare le richieste al server
- **Padre**: data
- **Classi**:

`CLIPS::client::data::urlrequest::AbstractUrlRequestListener`

* **Descrizione**: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
* **Utilizzo**: permette alle classi che interagiscono con l'urlrequest di definire un proprio listener concreto da poter usare per le chiamate al server
* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::data::urlrequest::AppInfoRequest`

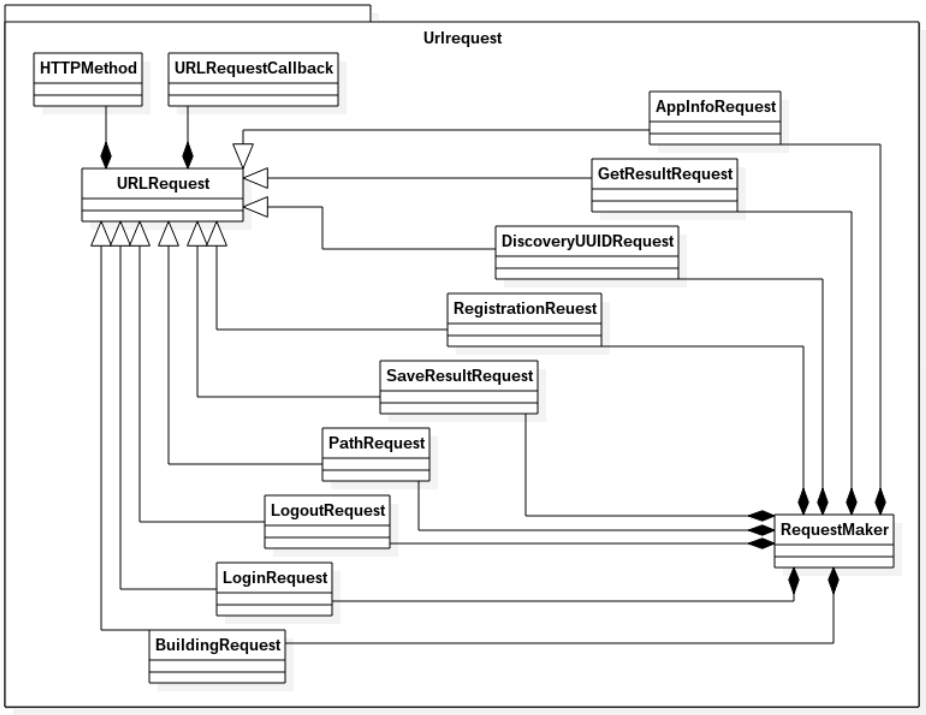

Figura 10: Schema sintetico package client::data::urlrequest

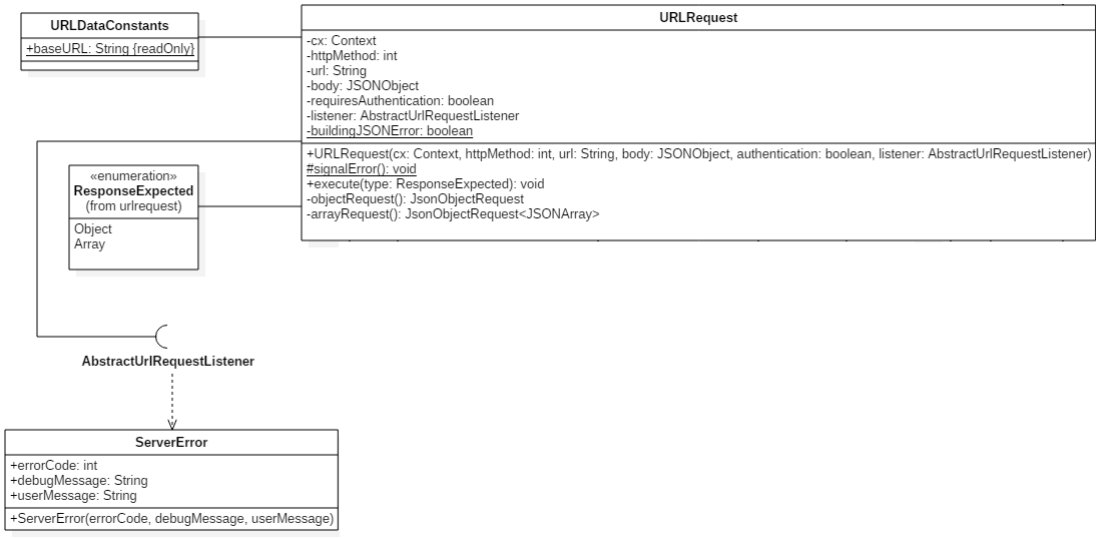

Figura 11: Prima parte schema package client::data::urlrequest

* **Descrizione**: classe per la richiesta al server di dati sulle info dell'app, tra cui lo UUID dei beacon usati per l'applicazione e le informazioni sull'applicazione

* **Utilizzo**: permette di effettuare una richiesta al server di dati sulle info dell'app

* **Relazioni con altre classi**:

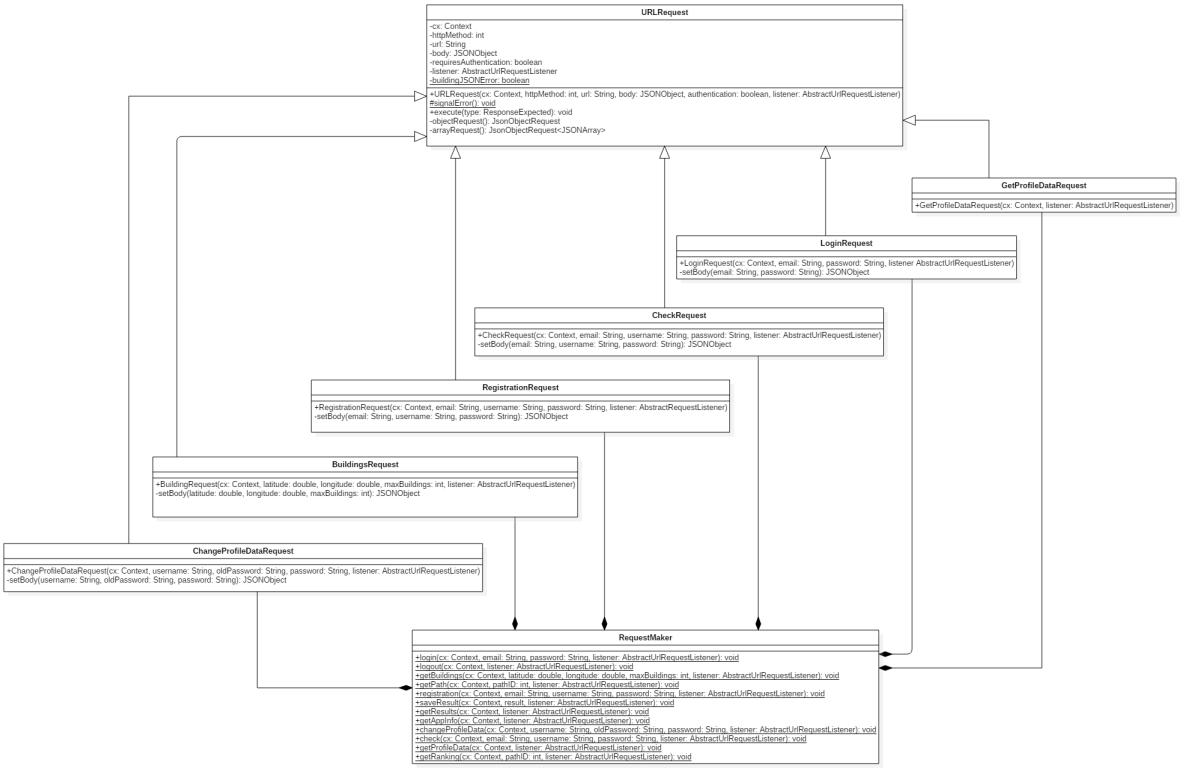

Figura 12: Seconda parte schema package client::data::urlrequest

- CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
- CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::BuildingsRequest`

* **Descrizione**: classe per la richiesta di dati sugli edifici
* **Utilizzo**: permette di effettuare una richiesta al server di dati sugli edifici
* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::ChangeProfileDataRequest`

* **Descrizione**: classe per la richiesta al server di cambiare i dati dell'utente
* **Utilizzo**: permette di inviare la richiesta al server di cambiare i dati dell'utente, in particolare possono essere modificati solo lo username e la password

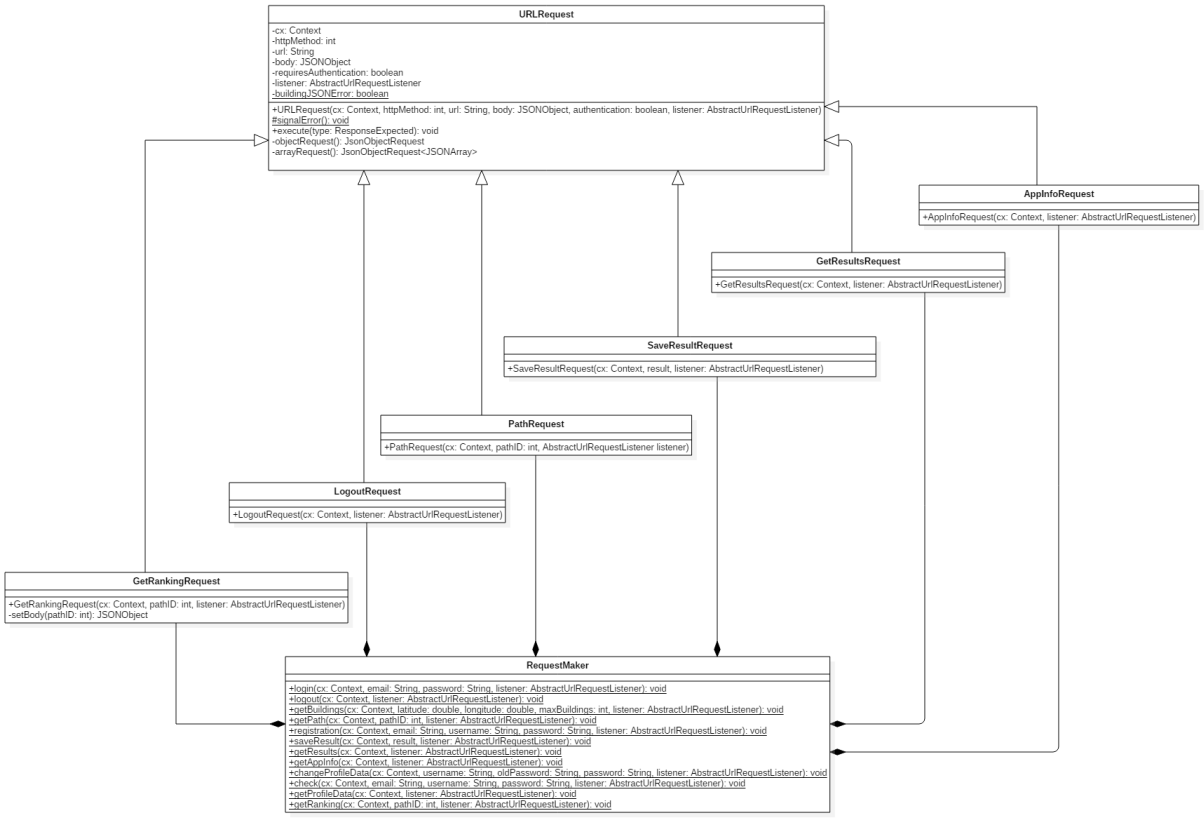

Figura 13: Terza parte schema package client::data::urlrequest

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::CheckRequest`

* **Descrizione**: classe per la richiesta al server del controllo dei dati inseriti dall'utente

* **Utilizzo**: permette di inviare la richiesta al server per controllare se l'email, lo username o la password sono corretti

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::ForgotPasswordRequest`

* **Descrizione**: Classe che effettua la richiesta di cambiare la password dimenticata dell'utente

* **Utilizzo**: Quando l'utente non ricorda la password può usare questa chiamata per ottenerne una generata casualmente dopo aver inserito l'indirizzo email dell'account da cambiare

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo
  - CLIPS::client::data::urlrequest::URLRequest: classe che si occupa delle richieste da inviare al server, fornisce un'implementazione delle richieste in base ai dati forniti alle sue classi derivate

`CLIPS::client::data::urlrequest::GetProfileDataRequest`

* **Descrizione**: classe per la richiesta al server di aggiornare in locale i dati dell'utente

* **Utilizzo**: permette di inviare la richiesta al server per aggiornare l'email e lo username dell'utente nel caso non fossero uguali a quelli salvati nel server

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::GetRankingRequest`

* **Descrizione**: classe per la richiesta al server dei dati sulla classifica di un determinato percorso

* **Utilizzo**: permette di inviare la richiesta al server per ottenere i dati sulla classifica del percorso selezionato

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::GetResultsRequest`

* **Descrizione**: classe per la richiesta al server dei risultati salvati dall'utente in precedenza

* **Utilizzo**: permette di effettuare la richiesta al server dei risultati salvati dall'utente in precedenza

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::LoginRequest`

* **Descrizione**: classe per la richiesta di login al server

* **Utilizzo**: permette di effettuare una richiesta di login al server

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::LogoutRequest`

* **Descrizione**: classe per la richiesta di logout al server

* **Utilizzo**: permette di effettuare una richiesta di logout al server

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::PathRequest`

* **Descrizione**: classe per la richiesta al server dei dati sul percorso selezionato, in particolare fornisce tutte le informazioni necessarie per poter svolgere il percorso

* **Utilizzo**: permette di richiedere al server i dati sul percorso selezionato

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::RegistrationRequest`

* **Descrizione**: classe per la richiesta di registrazione al server

* **Utilizzo**: permette di effettuare una richiesta di registrazione al server

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::URLDataConstants: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

`CLIPS::client::data::urlrequest::RequestMaker`

* **Descrizione**: classe per la costruzione e l'invio di richieste al server, funziona come un'interfaccia dell'urlrequest per il resto dell'applicazione e quindi regola l'interazione con il package

* **Utilizzo**: permette la costruzione e l'invio di richieste al server

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::urlrequest::AppInfoRequest: classe per la richiesta al server di dati sulle info dell'app, tra cui lo UUID dei beacon usati per l'applicazione e le informazioni sull'applicazione
  - CLIPS::client::data::urlrequest::BuildingsRequest: classe per la richiesta di dati sugli edifici
  - CLIPS::client::data::urlrequest::ChangeProfileDataRequest: classe per la richiesta al server di cambiare i dati dell'utente
  - CLIPS::client::data::urlrequest::CheckRequest: classe per la richiesta al server del controllo dei dati inseriti dall'utente
  - CLIPS::client::data::urlrequest::GetProfileDataRequest: classe per la richiesta al server di aggiornare in locale i dati dell'utente
  - CLIPS::client::data::urlrequest::GetRankingRequest: classe per la richiesta al server dei dati sulla classifica di un determinato percorso
  - CLIPS::client::data::urlrequest::GetResultsRequest: classe per la richiesta al server dei risultati salvati dall'utente in precedenza
  - CLIPS::client::data::urlrequest::LoginRequest: classe per la richiesta di login al server
  - CLIPS::client::data::urlrequest::LogoutRequest: classe per la richiesta di logout al server
  - CLIPS::client::data::urlrequest::PathRequest: classe per la richiesta al server dei dati sul percorso selezionato, in particolare fornisce tutte le informazioni necessarie per poter svolgere il percorso
  - CLIPS::client::data::PathResult: classe che rappresenta i risultati di un percorso
  - CLIPS::client::data::urlrequest::RegistrationRequest: classe per la richiesta di registrazione al server

- CLIPS::client::data::urlrequest::SaveResultRequest: classe per la richiesta al server di salvataggio del risultato del percorso appena finito

`CLIPS::client::data::urlrequest::SaveResultRequest`

* **Descrizione**: classe per la richiesta al server di salvataggio del risultato del percorso appena finito

* **Utilizzo**: permette di effettuare una richiesta al server di salvataggio del risultato del percorso appena finito

`CLIPS::client::data::urlrequest::ServerError`

* **Descrizione**: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

* **Utilizzo**: permette di rappresentare l'errore ricevuto in una forma più semplice da usare nel resto dell'applicazione

`CLIPS::client::data::urlrequest::URLDataConstants`

* **Descrizione**: classe che contiene i dati per effettuare le richieste al server che potrebbero variare nel tempo

* **Utilizzo**: permette di gestire facilmente eventuali cambiamenti futuri dei dati necessari ad effettuare le richieste al server

`CLIPS::client::data::urlrequest::URLRequest`

* **Descrizione**: classe che si occupa delle richieste da inviare al server, fornisce un'implementazione delle richieste in base ai dati forniti alle sue classi derivate

* **Utilizzo**: permette di effettuare tutte le richieste al server necessarie per far funzionare l'applicazione

* **Relazioni con altre classi**:

  - CLIPS::client::data::urlrequest::AbstractUrlRequestListener: interfaccia che definisce la struttura del listener astratto usato nel package urlrequest
  - CLIPS::client::data::datamanager::LoginManager: classe che gestisce le operazioni del profilo dell'utente
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

3.3.6 Componente `CLIPS::client::pathprogress`

Informazioni sul package

- **Descrizione**: componente che gestisce i dati del percorso e salva i risultati ottenuti nelle prove mentre si sta giocando
- **Padre**: client

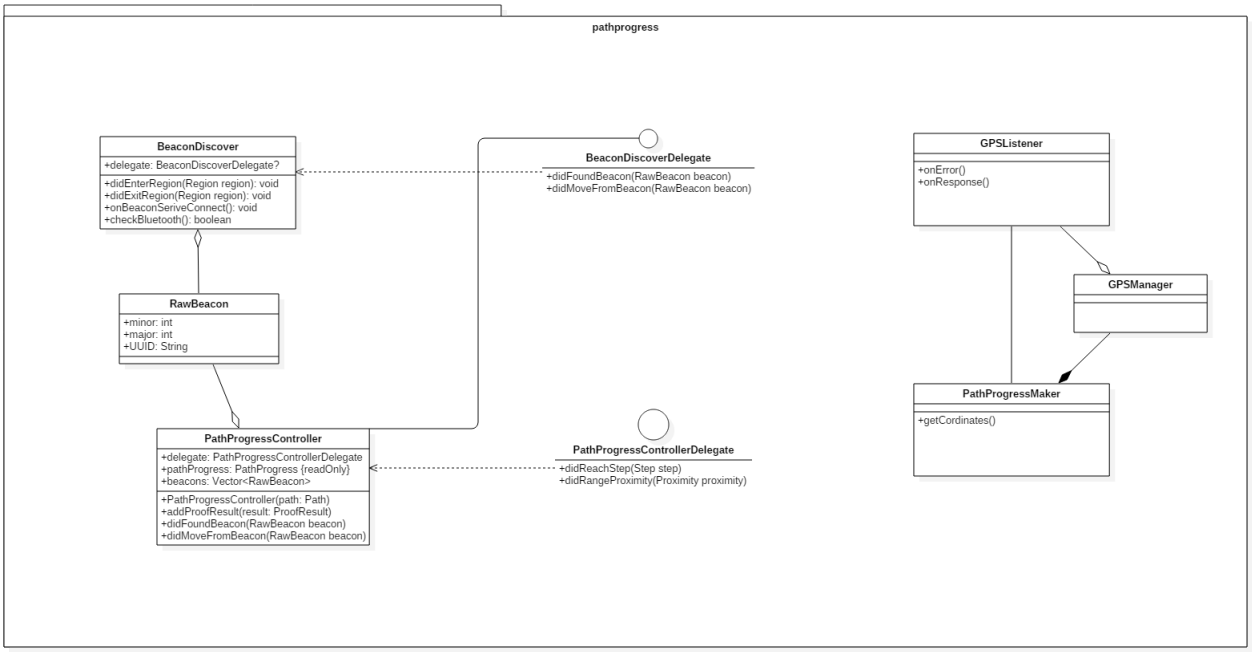

Figura 14: Schema package client::pathprogress

- **Classi**:

  `CLIPS::client::pathprogress::BeaconDiscover`

  - **Descrizione**: classe che si interfaccia direttamente con i beacon fisici
  - **Utilizzo**: permette di rilevare i beacon

  `CLIPS::client::pathprogress::BeaconDiscoverDelegate`

  - **Descrizione**: interfaccia utilizzata per notificare se il dispositivo è entrato o uscito dal raggio di un beacon
  - **Utilizzo**: permette di notificare se si è entrati o usciti dal raggio di un beacon
  - **Relazioni con altre classi**:
    - CLIPS::client::pathprogress::PathProgressController: classe che gestisce l'avanzamento di un percorso

  `CLIPS::client::pathprogress::GPSListener`

  - **Descrizione**: interfaccia che definisce la struttura del listener astratto usato nel package pathprogress
  - **Utilizzo**: permette alle classi che interagiscono con il pathprogress di definire un proprio listener concreto da poter usare per ottenere le informazioni richieste

* **Relazioni con altre classi**:
  - CLIPS::client::data::urlrequest::ServerError: classe che rappresenta l'errore che il client riceve quando la richiesta al server non ha successo

`CLIPS::client::pathprogress::GPSManager`

* **Descrizione**: classe che rileva la posizione dell'utente tramite il GPS

* **Utilizzo**: appena viene creato l'oggetto comunica con i satelliti GPS tramite i Google Play Services, appena riceve una posizione sufficientemente precisa restituisce le coordinate ricevute

* **Relazioni con altre classi**:
  - CLIPS::client::pathprogress::GPSListener: interfaccia che definisce la struttura del listener astratto usato nel package pathprogress

`CLIPS::client::pathprogress::PathProgressController`

* **Descrizione**: classe che gestisce l'avanzamento di un percorso

* **Utilizzo**: permette di gestire i dati dell'avanzamento di un percorso

* **Relazioni con altre classi**:
  - CLIPS::client::pathprogress::BeaconDiscoverDelegate: interfaccia utilizzata per notificare se il dispositivo è entrato o uscito dal raggio di un beacon
  - CLIPS::client::data::PathProgress: classe per il salvataggio in locale del progresso di un percorso
  - CLIPS::client::pathprogress::PathProgressControllerDelegate: interfaccia utilizzata per notificare se si è in presenza del beacon del prossimo step o di uno del proximity
  - CLIPS::client::pathprogress::RawBeacon: classe che rappresenta un beacon semplice

`CLIPS::client::pathprogress::PathProgressControllerDelegate`

* **Descrizione**: interfaccia utilizzata per notificare se si è in presenza del beacon del prossimo step o di uno del proximity

* **Utilizzo**: notifica se si nel raggio del beacon del prossimo step o di uno di proximity

* **Relazioni con altre classi**:
  - CLIPS::client::pathprogress::GPSListener: interfaccia che definisce la struttura del listener astratto usato nel package pathprogress
  - CLIPS::client::pathprogress::GPSManager: classe che rileva la posizione dell'utente tramite il GPS

`CLIPS::client::pathprogress::RawBeacon`

* **Descrizione**: classe che rappresenta un beacon semplice

* **Utilizzo**: utilizzato per rappresentare un beacon fisico

3.3.7 Componente `CLIPS::client::viewcontroller`

Informazioni sul package

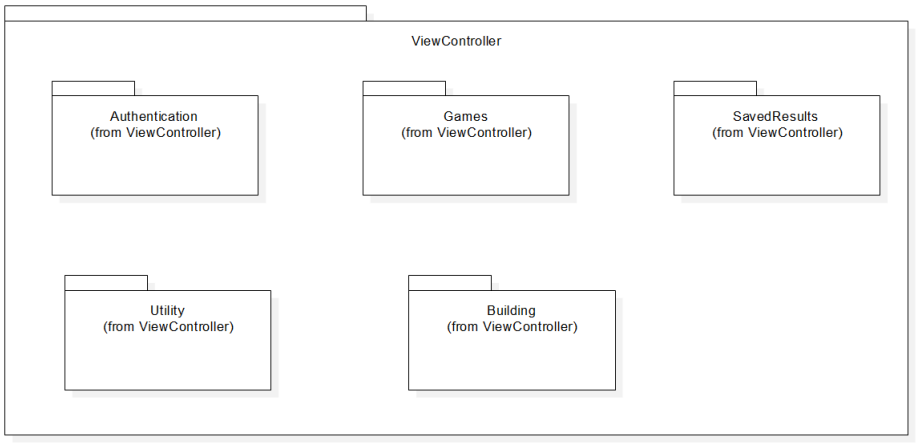

Figura 15: Schema package client::viewcontroller

- **Descrizione**: componente che raggruppa tutte le view ed i controller relativi alle view
- **Padre**: client
- **Package contenuti**:

  CLIPS::client::viewcontroller::authentication: componente che si occupa di gestire l'autenticazione dell'utente

- – CLIPS::client::viewcontroller::building: componente che gestisce le informazioni e le interazioni dell'utente con gli edifici abilitati
  – CLIPS::client::viewcontroller::games: componente che gestisce le prove che l'utente deve completare all'interno di un percorso
  – CLIPS::client::viewcontroller::savedresults: componente che raggruppa le le view e i controller dei risultati salvati e delle classifiche
  – CLIPS::client::viewcontroller::utility: componente che raggruppa le view generali dell'app
- **Classi**:

  `CLIPS::client::viewcontroller::ResultAdapter`

  * **Descrizione**: classe che permette la corretta visualizzazione dei risultati di un utente

  * **Utilizzo**: viene usata da SavedResultActivity

3.3.8 Componente `CLIPS::client::viewcontroller::authentication`

Informazioni sul package

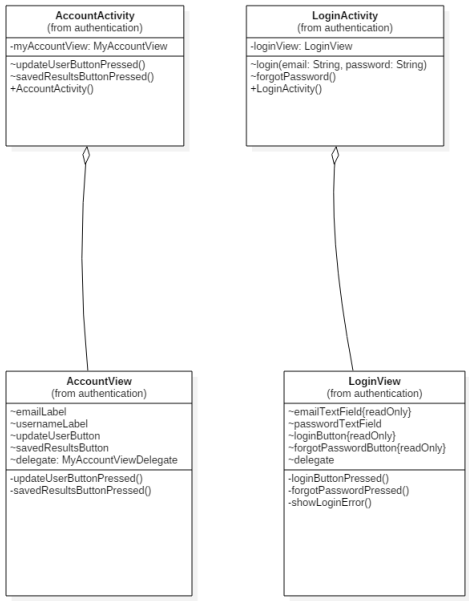

Figura 16: Prima parte schema package client::viewcontroller::authentication

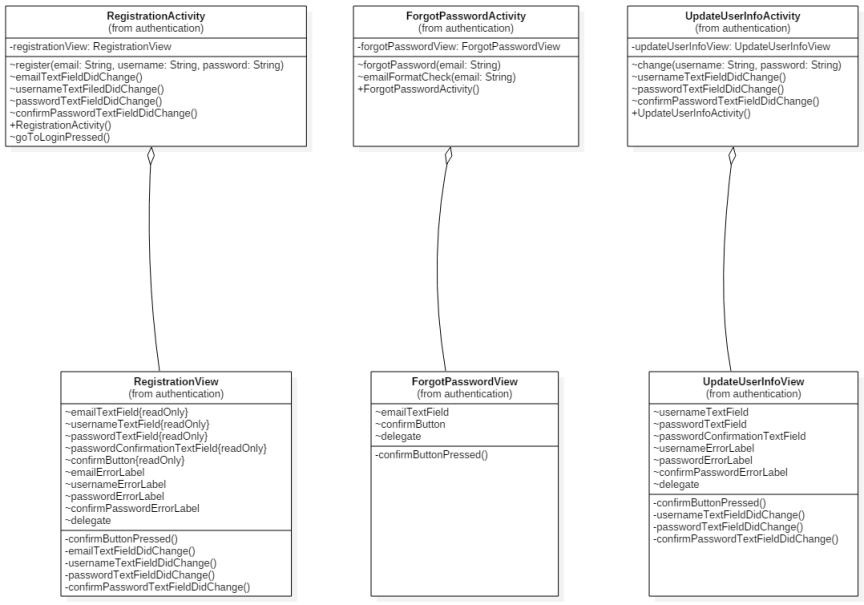

Figura 17: Seconda parte schema package client::viewcontroller::authentication

- **Descrizione**: componente che si occupa di gestire l'autenticazione dell'utente
- **Padre**: viewcontroller
- **Classi**:

`CLIPS::client::viewcontroller::authentication::AccountActivity`

* **Descrizione**: classe controller che si occupa di interagire con AccountView

* **Utilizzo**: si occupa di gestire le interazioni dell'utente con AccountView

`CLIPS::client::viewcontroller::authentication::AccountView`

* **Descrizione**: classe che rappresenta la schermata delle informazioni dell'utente

* **Utilizzo**: permette all'utente di visualizzare le informazioni del proprio profilo

* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::authentication::AccountActivity: classe controller che si occupa di interagire con AccountView

`CLIPS::client::viewcontroller::authentication::ChangeProfileActivity`

* **Descrizione**: permette all'utente di cambiare le proprie credenziali

* **Utilizzo**: l'utente grazie a ChangeProfileView potrà cambiare le proprie credenziali

`CLIPS::client::viewcontroller::authentication::ConfirmRegistration`

* **Descrizione**: classe che interagisce con RegistrationActivity e permette all'utente di visualizzare i propri dati di registrazione

* **Utilizzo**: l'utente può visualizzare i dati inseriti in precedenza

* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::authentication::RegistrationActivity: classe controller che si occupa di interagire con RegistrationView

`CLIPS::client::viewcontroller::authentication::ForgotPasswordActivity`

* **Descrizione**: classe controller che si occupa di interagire con ForgotPasswordView

* **Utilizzo**: si occupa di gestire le interazioni dell'utente con ForgotPasswordView

* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::authentication::ForgotPasswordView: classe che si occupa della visualizzazione della schermata per la richiesta di una nuova password

`CLIPS::client::viewcontroller::authentication::ForgotPasswordView`

* **Descrizione**: classe che si occupa della visualizzazione della schermata per la richiesta di una nuova password

* **Utilizzo**: consente all'utente di inserire la mail per ricevere una nuova password

`CLIPS::client::viewcontroller::authentication::LoginActivity`

* **Descrizione**: classe controller che si occupa di interagire con LoginView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con LoginView
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::authentication::LoginView: classe che si occupa della visualizzazione della schermata per il login

`CLIPS::client::viewcontroller::authentication::LoginView`

* **Descrizione**: classe che si occupa della visualizzazione della schermata per il login
* **Utilizzo**: consente all'utente di inserire i propri dati per effettuare il login

`CLIPS::client::viewcontroller::authentication::RecoverPassword`

* **Descrizione**: classe che rappresenta l'avvenuto recupero della password di un utente
* **Utilizzo**: view che permette all'utente di sapere che la sua password è stata recuperata correttamente

`CLIPS::client::viewcontroller::authentication::RegistrationActivity`

* **Descrizione**: classe controller che si occupa di interagire con RegistrationView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con RegistrationView
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::authentication::RegistrationView: classe che si occupa della visualizzazione della schermata per la registrazione

`CLIPS::client::viewcontroller::authentication::RegistrationView`

* **Descrizione**: classe che si occupa della visualizzazione della schermata per la registrazione
* **Utilizzo**: consente all'utente di inserire i propri dati per effettuare la registrazione

`CLIPS::client::viewcontroller::authentication::UpdateUserInfoActivity`

* **Descrizione**: classe controller che si occupa di interagire con UpdateUserInfoView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con UpdateUserInfoView
* **Relazioni con altre classi**:

· CLIPS::client::viewcontroller::authentication::UpdateUserInfoView: classe che si occupa della visualizzazione della schermata per il cambio delle credenziali

`CLIPS::client::viewcontroller::authentication::UpdateUserInfoView`

∗ **Descrizione**: classe che si occupa della visualizzazione della schermata per il cambio delle credenziali

∗ **Utilizzo**: consente all'utente di inserire i nuovi dati per cambiare le sue credenziali

3.3.9 Componente `CLIPS::client::viewcontroller::building`

Informazioni sul package

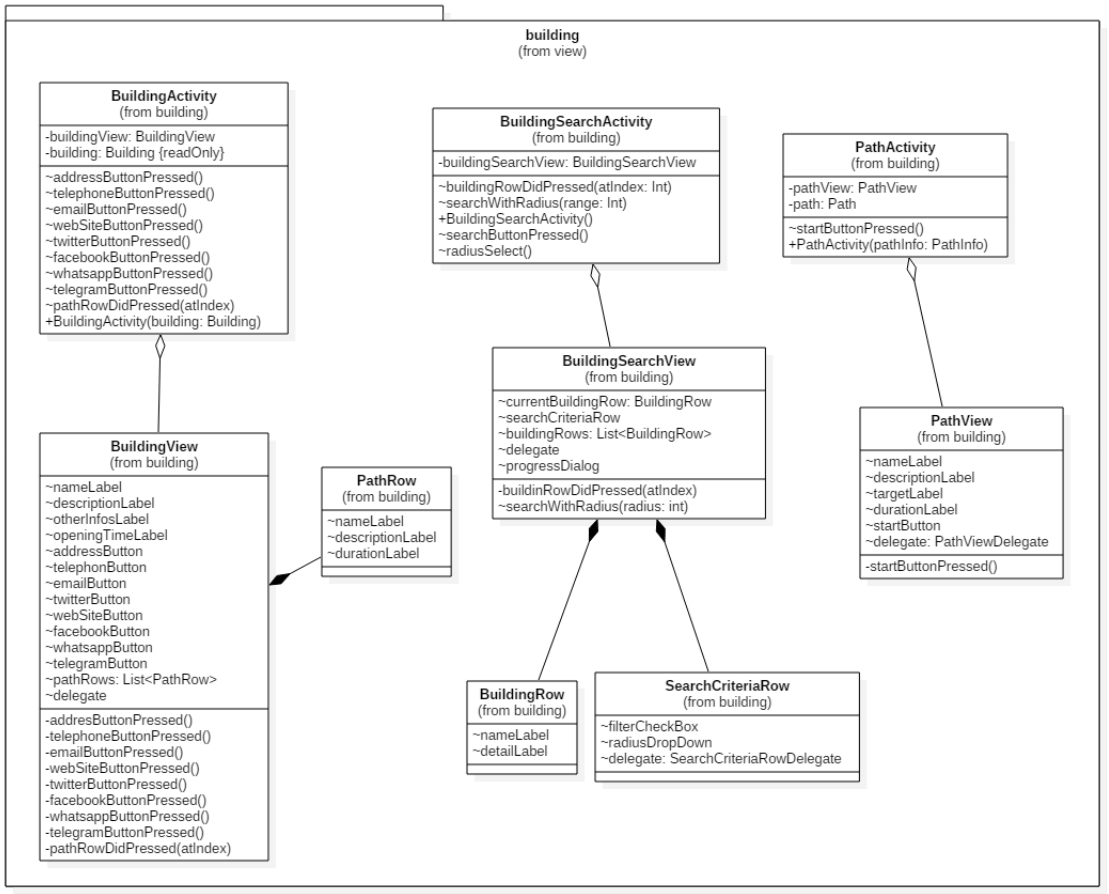

Figura 18: Schema package client::viewcontroller::building

- **Descrizione**: componente che gestisce le informazioni e le interazioni dell'utente con gli edifici abilitati
- **Padre**: viewcontroller
- **Classi**:

`CLIPS::client::viewcontroller::building::BuildingActivity`

* **Descrizione**: classe controller che si occupa di interagire con BuildingView
* **Utilizzo**: classe controller che si occupa di interagire con BuildingView
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::building::BuildingView: classe per la visualizzazione di un edificio

`CLIPS::client::viewcontroller::building::BuildingRow`

* **Descrizione**: classe per la rappresentazione di un edificio nella ricerca
* **Utilizzo**: permette di visualizzare le informazioni di un edificio nella lista dei risultati di ricerca

`CLIPS::client::viewcontroller::building::BuildingSearchActivity`

* **Descrizione**: classe controller che si occupa di interagire con BuildingSearchView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con BuildingSearchView
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::building::BuildingSearchView: classe per la ricerca degli edifici

`CLIPS::client::viewcontroller::building::BuildingSearchView`

* **Descrizione**: classe per la ricerca degli edifici
* **Utilizzo**: permette all'utente di cercare gli edifici vicini per visualizzare le informazioni e/o i percorsi disponibili
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::building::BuildingRow: classe per la rappresentazione di un edificio nella ricerca
  - CLIPS::client::viewcontroller::building::SearchCriteriaRow: classe utilizzata per determinare i criteri di ricerca degli edifici

`CLIPS::client::viewcontroller::building::BuildingView`

* **Descrizione**: classe per la visualizzazione di un edificio
* **Utilizzo**: permette all'utente di visualizzare le informazioni relative all'edificio selezionato
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::building::PathRow: classe per la rappresentazione di un percorso nella ricerca

`CLIPS::client::viewcontroller::building::PathActivity`

- **Descrizione**: classe controller che si occupa di interagire con PathView
- **Utilizzo**: si occupa di gestire le interazioni dell'utente con PathView
- **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::building::PathView: classe che si occupa della visualizzazione della schermata riguardante un percorso

`CLIPS::client::viewcontroller::building::PathRow`

- **Descrizione**: classe per la rappresentazione di un percorso nella ricerca
- **Utilizzo**: permette un percorso disponibile nella lista dei risultati di ricerca

`CLIPS::client::viewcontroller::building::PathView`

- **Descrizione**: classe che si occupa della visualizzazione della schermata riguardante un percorso
- **Utilizzo**: consente all'utente di visualizzare le informazioni riguardanti un percorso e se l'utente si trova nell'edificio del percorso consente di iniziarlo

`CLIPS::client::viewcontroller::building::SearchCriteriaRow`

- **Descrizione**: classe utilizzata per determinare i criteri di ricerca degli edifici
- **Utilizzo**: permette all'utente di visualizzare i criteri di ricerca disponibili

3.3.10 Componente `CLIPS::client::viewcontroller::games`

Informazioni sul package

- **Descrizione**: componente che gestisce le prove che l'utente deve completare all'interno di un percorso
- **Padre**: viewcontroller
- **Classi**:

`CLIPS::client::viewcontroller::games::MultipleChoiceQuizActivity`

- **Descrizione**: classe controller che si occupa di interagire con MultipleChoiceView
- **Utilizzo**: si occupa di gestire le interazioni dell'utente con MultipleChoiceView
- **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::games::QuizActivity: interfaccia per segnalare gli eventi della classe QuizView

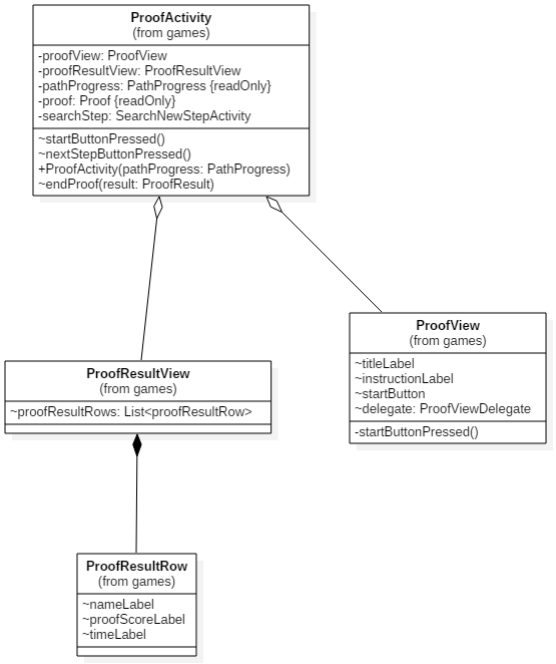

Figura 19: Prima parte schema package client::viewcontroller::games

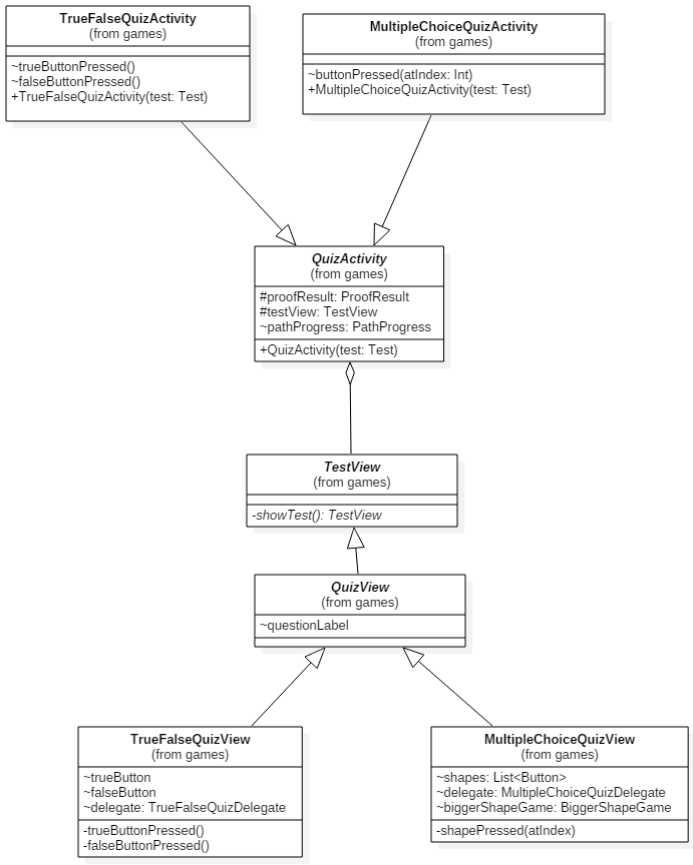

Figura 20: Seconda parte schema package client::viewcontroller::games

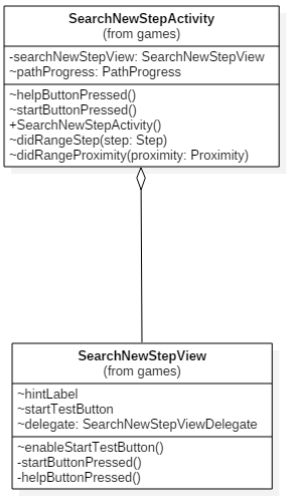

Figura 21: Terza parte schema package client::viewcontroller::games

`CLIPS::client::viewcontroller::games::MultipleChoiceQuizView`

* **Descrizione**: classe per il quiz a risposta multipla
* **Utilizzo**: si occupa di fornire un'interfaccia per il quiz a risposta multipla
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::games::QuizView: classe base per i quiz

`CLIPS::client::viewcontroller::games::ProofActivity`

* **Descrizione**: classe controller che si occupa di interagire con ProofView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con ProofView
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::games::ProofResultView: classe che rappresenta la schermata dei risultati

`CLIPS::client::viewcontroller::games::ProofResultActivity`

* **Descrizione**: classe che permette di visualizzare il risultato di una prova
* **Utilizzo**: l'utente può vedere il risultato della prova che ha appena svolto

`CLIPS::client::viewcontroller::games::ProofResultRow`

* **Descrizione**: classe che rappresenta una risultato rappresentato all'interno di una riga che può essere cliccata
* **Utilizzo**: permette all'utente di visualizzare le informazioni generali di un risultato di una prova e di cliccarci per visualizzarne le informazioni dettagliate

`CLIPS::client::viewcontroller::games::ProofResultView`

* **Descrizione**: classe che rappresenta la schermata dei risultati
* **Utilizzo**: permette all'utente di visualizzare le informazioni generali dei risultati delle prove e cliccare sulle prove delle quali vuole visualizzare le informazioni dettagliate
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::games::ProofResultRow: classe che rappresenta una risultato rappresentato all'interno di una riga che può essere cliccata

`CLIPS::client::viewcontroller::games::ProofView`

* **Descrizione**: classe che rappresenta la schermata della prova
* **Utilizzo**: permette all'utente di visualizzare la prova da giocare

`CLIPS::client::viewcontroller::games::QuizActivity`

* **Descrizione**: interfaccia per segnalare gli eventi della classe QuizView
* **Utilizzo**: si occupa di fornire i metodi necessari alla classe QuizView per notificare gli eventi

`CLIPS::client::viewcontroller::games::QuizResultView`

* **Descrizione**: classe per la visualizzazione del risultato di un quiz
* **Utilizzo**: fornisce all'utente un'interfaccia affinché visualizzi il risultato del quiz

`CLIPS::client::viewcontroller::games::QuizView`

* **Descrizione**: classe base per i quiz
* **Utilizzo**: fornisce una base per i vari tipi di test da istanziare
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::games::TestView: classe che fornisce una base dalla quale è possibile creare vari tipi di giochi

`CLIPS::client::viewcontroller::games::SearchNewStepActivity`

* **Descrizione**: classe controller che si occupa di interagire con SearchNewStepView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con SearchNewStepView
* **Relazioni con altre classi**:

- CLIPS::client::viewcontroller::games::SearchNewStepView: classe che rappresenta la schermata per la ricerca della prossima prova del percorso

`CLIPS::client::viewcontroller::games::SearchNewStepView`

* **Descrizione**: classe che rappresenta la schermata per la ricerca della prossima prova del percorso
* **Utilizzo**: permette all'utente di cercare in modo semplificato la ricerca della prossima prova del percorso

`CLIPS::client::viewcontroller::games::TestResultView`

* **Descrizione**: classe che fornisce una base per la visualizzazione del risultato della prova
* **Utilizzo**: permette all'utente di visualizzare il risultato della prova

`CLIPS::client::viewcontroller::games::TestView`

* **Descrizione**: classe che fornisce una base dalla quale è possibile creare vari tipi di giochi
* **Utilizzo**: viene utilizzata per visualizzare un'interfaccia di gioco all'utente

`CLIPS::client::viewcontroller::games::TrueFalseQuizActivity`

* **Descrizione**: classe controller che si occupa di interagire con TrueFalseQuizView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con TrueFalseQuizView
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::games::QuizActivity: interfaccia per segnalare gli eventi della classe QuizView

`CLIPS::client::viewcontroller::games::TrueFalseQuizView`

* **Descrizione**: classe per il quiz vero/falso
* **Utilizzo**: si occupa di fornire un'interfaccia per la prova di tipo vero/falso
* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::games::QuizView: classe base per i quiz

3.3.11 Componente `CLIPS::client::viewcontroller::savedresults`

Informazioni sul package

- **Descrizione**: componente che raggruppa le le view e i controller dei risultati salvati e delle classifiche
- **Padre**: viewcontroller

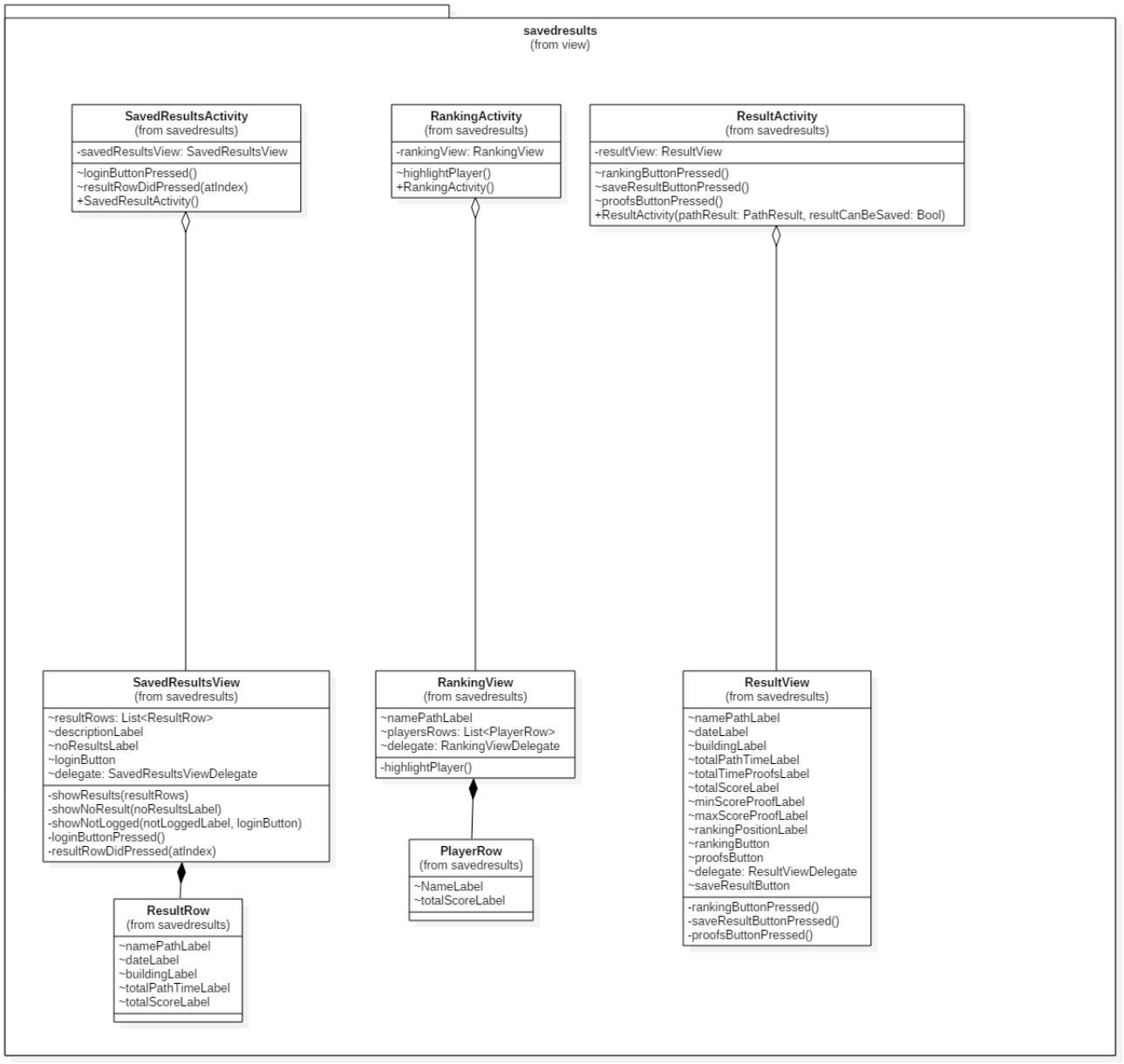

Figura 22: Schema package client::viewcontroller::savedresults

- **Classi**:

  `CLIPS::client::viewcontroller::savedresults::PlayerRow`

  - **Descrizione**: classe che rappresenta una riga all'interno di una classifica
  - **Utilizzo**: permette all'utente di visualizzare il nome del giocatore e il suo punteggio nel percorso d'interesse

  `CLIPS::client::viewcontroller::savedresults::RankingActivity`

* **Descrizione**: classe controller che si occupa di interagire con RankingView

* **Utilizzo**: si occupa di gestire le interazioni dell'utente con RankingView

`CLIPS::client::viewcontroller::savedresults::RankingView`

* **Descrizione**: classe per rappresentare la classifica di un percorso

* **Utilizzo**: permette all'utente di visualizzare la classifica del percorso

* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::savedresults::RankingActivity: classe controller che si occupa di interagire con RankingView

`CLIPS::client::viewcontroller::savedresults::ResultActivity`

* **Descrizione**: classe controller che si occupa di interagire con ResultView

* **Utilizzo**: si occupa di gestire le interazioni dell'utente con ResultView

* **Relazioni con altre classi**:
  - CLIPS::client::viewcontroller::savedresults::ResultView: classe che rappresenta la schermata nel quali si possono visualizzare i risultati di un percorso
  - CLIPS::client::viewcontroller::savedresults::ResultView: classe che rappresenta la schermata nel quali si possono visualizzare i risultati di un percorso

`CLIPS::client::viewcontroller::savedresults::ResultRow`

* **Descrizione**: classe che rappresenta una riga di un risultato

* **Utilizzo**: permette all'utente di visualizzare le informazioni generali di un risultato e di cliccarci per visualizzare quelle dettagliate

`CLIPS::client::viewcontroller::savedresults::ResultView`

* **Descrizione**: classe che rappresenta la schermata nel quali si possono visualizzare i risultati di un percorso

* **Utilizzo**: permette all'utente di visualizzare le informazioni dettagliate del risultato di un percorso

`CLIPS::client::viewcontroller::savedresults::SavedResultsActivity`

* **Descrizione**: classe controller che si occupa di interagire con SavedResultView

* **Utilizzo**: si occupa di gestire le interazioni dell'utente con SavedResultView

* **Relazioni con altre classi**:

· CLIPS::client::viewcontroller::savedresults::SavedResultView: classe che rappresenta la schermata dei risultati salvati di un utente

`CLIPS::client::viewcontroller::savedresults::SavedResultView`

* **Descrizione**: classe che rappresenta la schermata dei risultati salvati di un utente
* **Utilizzo**: permette all'utente di visualizzare i propri risultati salvati
* **Relazioni con altre classi**:
  · CLIPS::client::viewcontroller::savedresults::ResultRow: classe che rappresenta una riga di un risultato

3.3.12 Componente `CLIPS::client::viewcontroller::utility`

Informazioni sul package

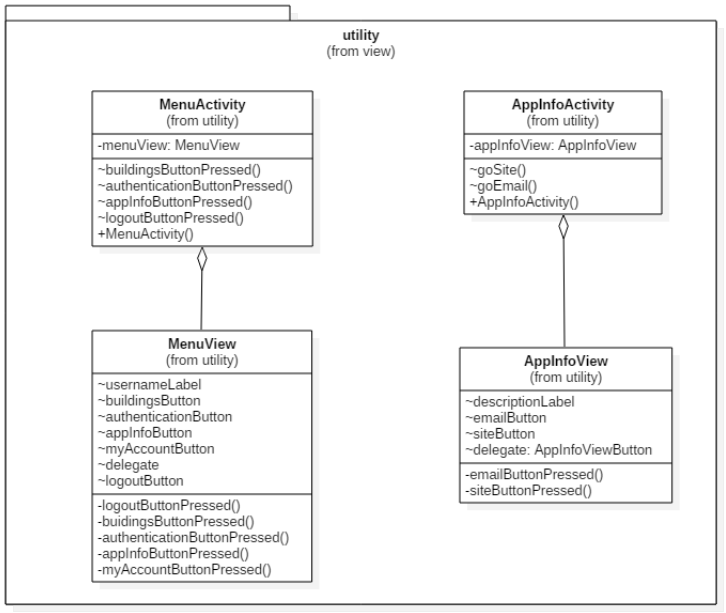

Figura 23: Schema package client::viewcontroller::utility

- **Descrizione**: componente che raggruppa le view generali dell'app
- **Padre**: viewcontroller
- **Classi**:

`CLIPS::client::viewcontroller::utility::AppInfoActivity`

* **Descrizione**: classe controller che si occupa di interagire con AppInfoView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con AppInfoView
* **Relazioni con altre classi**:

· CLIPS::client::viewcontroller::utility::AppInfoView: classe che si occupa delle informazioni generali dell'app

`CLIPS::client::viewcontroller::utility::AppInfoView`

* **Descrizione**: classe che si occupa delle informazioni generali dell'app
* **Utilizzo**: permette all'utente di visualizzare le informazioni generali dell'app

`CLIPS::client::viewcontroller::utility::MenuActivity`

* **Descrizione**: classe controller che si occupa di interagire con MenuView
* **Utilizzo**: si occupa di gestire le interazioni dell'utente con MenuView
* **Relazioni con altre classi**:
  · CLIPS::client::viewcontroller::utility::MenuView: classe che si occupa di far visualizzare il menu

`CLIPS::client::viewcontroller::utility::MenuView`

* **Descrizione**: classe che si occupa di far visualizzare il menu
* **Utilizzo**: consente all'utente di navigare nell'app tramite il menu

3.3.13 Componente `CLIPS::server`

Informazioni sul package

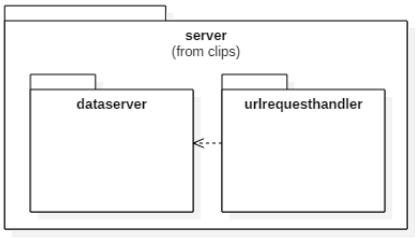

Figura 24: Schema package server

- **Descrizione**: componente globale per il back end del prodotto
- **Padre**: CLIPS
- **Package contenuti**:

  CLIPS::server::dataserver: package per la gestione dei dati sul server

- – CLIPS::server::urlrequesthandler: componente che gestisce le richieste inviate al server e le risposte da inviare al client

3.3.14 Componente `CLIPS::server::dataserver`

Informazioni sul package

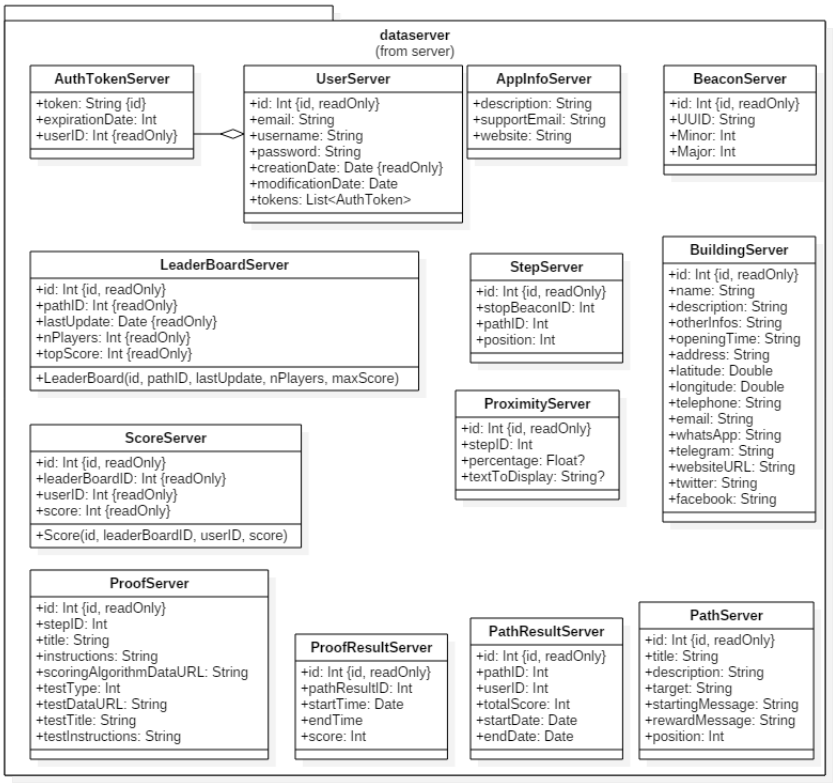

Figura 25: Schema package server::dataserver

- **Descrizione**: package per la gestione dei dati sul server
- **Padre**: server
- **Classi**:

  `CLIPS::server::dataserver::AppInfoServer`

  * **Descrizione**: classe che rappresenta le informazioni dell'app sul server

  * **Utilizzo**: permette di salvare sul server le informazioni dell'app

  `CLIPS::server::dataserver::AuthTokenServer`

  * **Descrizione**: Classe che rappresenta un token che si riferisce ad un utente loggato

  * **Utilizzo**: permette utilizzare un token per riferirsi ad utente loggato

  `CLIPS::server::dataserver::BeaconServer`

  * **Descrizione**: classe che rappresenta un beacon nel server

  * **Utilizzo**: permette di salvare sul server un beacon

  `CLIPS::server::dataserver::BuildingServer`

* **Descrizione**: classe che rappresenta un edificio sul server

* **Utilizzo**: permette di salvare e modificare i dati di un edificio sul server

`CLIPS::server::dataserver::LeaderBoardServer`

* **Descrizione**: classe che rappresenta la classifica sul server

* **Utilizzo**: permette di salvare i dati della classifica sul server

`CLIPS::server::dataserver::PathResultServer`

* **Descrizione**: classe che rappresenta il risultato di un percorso sul server

* **Utilizzo**: consente di salvare i risultati di un percorso sul server

`CLIPS::server::dataserver::PathServer`

* **Descrizione**: classe che rappresenta un percorso sul server

* **Utilizzo**: permette di creare, modificare ed eliminare un percorso sul server

`CLIPS::server::dataserver::ProofResultServer`

* **Descrizione**: classe che rappresenta i risultati di una prova sul server

* **Utilizzo**: consente di salvare il risultato di una prova sul server

`CLIPS::server::dataserver::ProofServer`

* **Descrizione**: classe che rappresenta una prova sul server

* **Utilizzo**: permette di creare, modificare ed eliminare una prova sul server

`CLIPS::server::dataserver::ProximityServer`

* **Descrizione**: classe che rappresenta sul server un beacon indicato alla segnalazione della distanza dalla prossima prova

* **Utilizzo**: permette di salvare i beacon di segnalazione sul server

`CLIPS::server::dataserver::ScoreManager`

* **Descrizione**: classe che rappresenta un risultato sul server

* **Utilizzo**: permette si salvare un risultato nel server

`CLIPS::server::dataserver::StepServer`

* **Descrizione**: classe che rappresenta uno step di un percorso nel server

* **Utilizzo**: permette di salvare sul server uno step di un percorso

`CLIPS::server::dataserver::UserServer`

* **Descrizione**: classe che rappresenta un utente nel server

* **Utilizzo**: permette di salvare sul server un utente

* **Relazioni con altre classi**:
  - CLIPS::server::dataserver::AuthTokenServer: Classe che rappresenta un token che si riferisce ad un utente loggato

3.3.15 Componente `CLIPS::server::urlrequesthandler`

Informazioni sul package

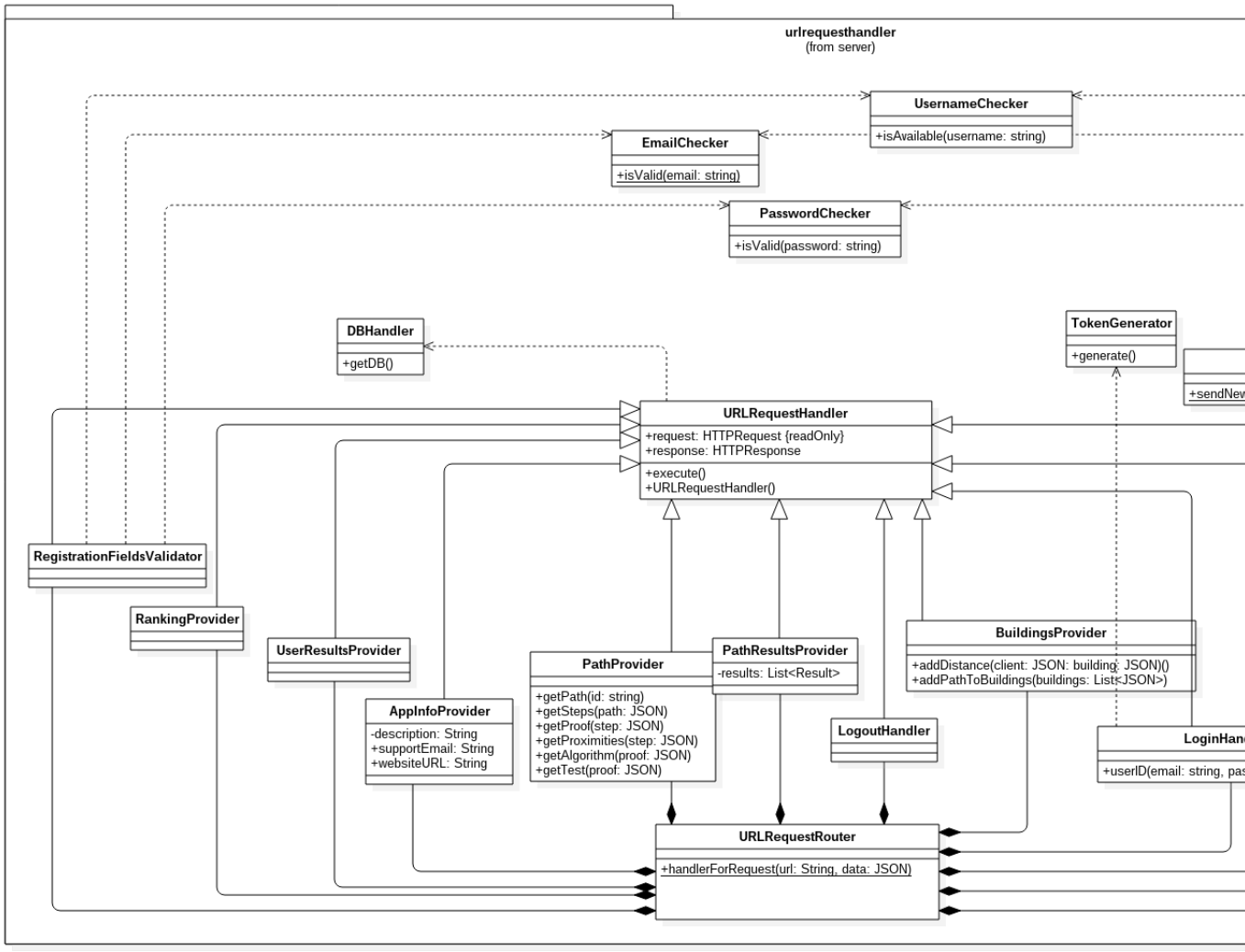

Figura 26: Schema package server::urlrequesthandler

- **Descrizione**: componente che gestisce le richieste inviate al server e le risposte da inviare al client

- **Padre**: server
- **Classi**:

`CLIPS::server::urlrequesthandler::AppInfoProvider`

  * **Descrizione**: classe che gestisce la richiesta di informazioni sull'app (come la descrizione dell'app, l'indirizzo del sito di supporto, l'email del supporto tecnico ed altro)

  * **Utilizzo**: si occupa di restituire le informazioni dell'app richieste

`CLIPS::server::urlrequesthandler::BuildingsProvider`

  * **Descrizione**: classe che gestisce la richiesta degli edifici dal client al server

  * **Utilizzo**: si occupa di restituire le informazioni sugli edifici richieste

  * **Relazioni con altre classi**:

    · CLIPS::server::urlrequesthandler::DBHandler: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex

`CLIPS::server::urlrequesthandler::DBHandler`

  * **Descrizione**: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex

  * **Utilizzo**: creando un oggetto di questa classe è possibile creare query per interagire con il database

`CLIPS::server::urlrequesthandler::EmailChecker`

  * **Descrizione**: fornisce i metodi per la validazione di un indirizzo email

  * **Utilizzo**: attraverso il metodo isValid(string) è possibile controllare se l'indirizzo email passato in input è valido

`CLIPS::server::urlrequesthandler::EmailSender`

  * **Descrizione**: classe usata dal server per inviare un email

  * **Utilizzo**: si occupa di inviare un'email da beaconstrips.swe@gmail.com

`CLIPS::server::urlrequesthandler::GetUserData`

  * **Descrizione**: classe del server che gestisce la richiesta dei dati utente

  * **Utilizzo**: si occupa di tornare al client i dati sull'utente che ha effettuato la chiamata

`CLIPS::server::urlrequesthandler::LoginHandler`

* **Descrizione**: classe che gestisce le richieste di login da parte del client
* **Utilizzo**: si occupa di gestire il login lato server
* **Relazioni con altre classi**:
  - CLIPS::server::urlrequesthandler::DBHandler: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex

`CLIPS::server::urlrequesthandler::LogoutHandler`

* **Descrizione**: classe che gestisce la richiesta di logout da parte del client (eliminando il token associato al client)
* **Utilizzo**: si occupa di effettuare il logout lato server

`CLIPS::server::urlrequesthandler::PasswordChecker`

* **Descrizione**: verifica che la password soddisfi i requisiti minimi di sicurezza
* **Utilizzo**: utilizzare il metodo isValid(string) per verificare la sicurezza sufficiente della password inserita

`CLIPS::server::urlrequesthandler::PasswordResetEmailSender`

* **Descrizione**: classe che invia una email contenente i dettagli per il reset di una password dimenticata
* **Utilizzo**: si occupa di notificare l'utente della nuova password impostata dopo averne chiesto il ripristino

`CLIPS::server::urlrequesthandler::PasswordResetHandler`

* **Descrizione**: classe del server per gestire la richiesta di reset di una password dimenticata
* **Utilizzo**: si occupa di verificare se l'indirizzo email fornito corrisponde con quello di qualche account ed eventualmente di resecare la password ed inviare una mail di avvenuto ripristino

`CLIPS::server::urlrequesthandler::PathProvider`

* **Descrizione**: classe che gestisce la richiesta del percorso dal client
* **Utilizzo**: si occupa di restituire le informazioni sul percorso richieste
* **Relazioni con altre classi**:
  - CLIPS::server::urlrequesthandler::DBHandler: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex

`CLIPS::server::urlrequesthandler::PathResultsProvider`

* **Descrizione**: classe che gestisce la richiesta del risultato del percorso dal client

* **Utilizzo**: si occupa di restituire le informazioni del risultato sul percorso richieste

* **Relazioni con altre classi**:
  - CLIPS::server::urlrequesthandler::DBHandler: classe che si occupa di gestire il DB e di ritornare un oggetto configurato della libreria knex

`CLIPS::server::urlrequesthandler::PostUserData`

* **Descrizione**: classe del server che si occupa della modifica dei dati utente

* **Utilizzo**: si occupa di modificare nel database i dati dell'utente quando un client chiede che vengano modificati

`CLIPS::server::urlrequesthandler::RankingProvider`

* **Descrizione**: provider della classifica per il percorso specificato

* **Utilizzo**: si occupa di restituire al richiedente la classifica del percorso specificato

`CLIPS::server::urlrequesthandler::RegistrationFieldsValidator`

* **Descrizione**: classe del server che si occupa di validare i dati di registrazione

* **Utilizzo**: la classe presenta un metodo execute che verifica la richiesta effettuata dal client e quali sono i dati forniti e risponde con le indicazioni di quali sono campi validi

* **Relazioni con altre classi**:
  - CLIPS::server::urlrequesthandler::EmailChecker: fornisce i metodi per la validazione di un indirizzo email
  - CLIPS::server::urlrequesthandler::PasswordChecker: verifica che la password soddisfi i requisiti minimi di sicurezza
  - CLIPS::server::urlrequesthandler::UsernameChecker: si occupa di verificare la validità dell'username (in particolare se è già in uso da un utente)

`CLIPS::server::urlrequesthandler::RegistrationHandler`

* **Descrizione**: classe che si occupa di registrare i nuovi utenti

* **Utilizzo**: è necessario impostare gli attributi response e request e successivamente chiamare il metodo execute()

`CLIPS::server::urlrequesthandler::TokenGenerator`

* **Descrizione**: classe per la creazione di stringhe casuali univoche

* **Utilizzo**: si occupa di creare un token casuale

`CLIPS::server::urlrequesthandler::URLRequestHandler`

* **Descrizione**: classe astratta che rappresenta un gestore di richieste di dati fatte al server

* **Utilizzo**: si occupa di gestire le richieste ricevute dal client

`CLIPS::server::urlrequesthandler::URLRequestRouter`

* **Descrizione**: classe che si occupa di creare il corretto URLRequestHandler per gestire la richiesta HTTP rivolta al server

* **Utilizzo**: crea la classe URLRequestHandler appropriata

* **Relazioni con altre classi**:
  - CLIPS::server::urlrequesthandler::AppInfoProvider: classe che gestisce la richiesta di informazioni sull'app (come la descrizione dell'app, l'indirizzo del sito di supporto, l'email del supporto tecnico ed altro)
  - CLIPS::server::urlrequesthandler::BuildingsProvider: classe che gestisce la richiesta degli edifici dal client al server
  - CLIPS::server::urlrequesthandler::LoginHandler: classe che gestisce le richieste di login da parte del client
  - CLIPS::server::urlrequesthandler::LogoutHandler: classe che gestisce la richiesta di logout da parte del client (eliminando il token associato al client)
  - CLIPS::server::urlrequesthandler::PathProvider: classe che gestisce la richiesta del percorso dal client
  - CLIPS::server::urlrequesthandler::PathResultsProvider: classe che gestisce la richiesta del risultato del percorso dal client
  - CLIPS::server::urlrequesthandler::RegistrationFieldsValidator: classe del server che si occupa di validare i dati di registrazione
  - CLIPS::server::urlrequesthandler::UserResultProvider: classe che gestisce la richiesta di un utente

`CLIPS::server::urlrequesthandler::UserDataRequest`

* **Descrizione**: classe che si occupa di gestire una richiesta che opera sui dati dell'utente

* **Utilizzo**: la classe mette a disposizioni delle sottoclassi una varietà di metodi per gestire agevolmente i dati degli utenti

`CLIPS::server::urlrequesthandler::UsernameChecker`

* **Descrizione**: si occupa di verificare la validità dell'username (in particolare se è già in uso da un utente)

* **Utilizzo**: è utile per verificare che un username non sia ancora stato utilizzato

`CLIPS::server::urlrequesthandler::UserResultProvider`

* **Descrizione**: classe che gestisce la richiesta di un utente

* **Utilizzo**: si occupa di restituire gli utenti richiesti dal client

4 Tabella tracciamento Componenti-Requisiti

Componenti	Requisiti
CLIPS::client::authentication	R0F1.1
	R0F1.1.1
	R0F1.1.2
	R0F1.1.3
	R0F1.1.4
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.2
	R0F1.1.5.3
	R0F1.1.5.4
	R0F1.1.5.4.1
	R0F1.1.5.4.2
	R0F1.2
	R0F1.2.1
	R0F1.2.2
	R0F1.2.3
	R0F1.2.4
	R0F1.2.5
	R0F1.2.6
	R0F1.2.7
	R0F1.2.7.1
	R0F1.2.7.2
	R0F2
	R0F5.3
	R0F7
	R0F7.1
	R0F7.2
	R0F7.3

Componenti	Requisiti
CLIPS::client::viewcontroller::building	R0F3
	R0F3.1
	R0F3.1.1
	R0F3.1.1.1
	R0F3.1.1.2
	R0F3.1.2.5
	R0F3.1.2.5.2
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F5.2.1
	R0F5.3.1
	R0F6
	R0F6.1
	R0F6.1.1
	R0F6.1.1.1
	R0F6.2
	R0F6.2.1
	R0F6.2.1.1
	R0F6.2.1.2
	R0F6.2.1.3
	R0F6.2.2
	R0F6.2.3
	R0F6.2.3.1
	R0F6.2.3.2
	R0F6.2.3.3
	R0F6.2.3.4
	R0F6.2.3.5
	R0F6.2.3.6

Componenti	Requisiti
CLIPS::client	R0F1.1
	R0F1.1.1
	R0F1.1.2
	R0F1.1.3
	R0F1.1.4
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.2
	R0F1.1.5.3
	R0F1.1.5.4.1
	R0F1.1.5.4.2
	R0F1.2
	R0F1.2.1
	R0F1.2.2
	R0F1.2.3
	R0F1.2.4
	R0F1.2.5
	R0F1.2.6
	R0F1.2.6.1
	R0F1.2.7
	R0F1.2.7.1
	R0F1.2.7.2
	R0F2
	R0F3
	R0F3.1
	R0F3.1.1
	R0F3.1.1.1
	R0F3.1.1.2
	R0F3.1.2
	R0F3.1.2.2
	R0F3.1.2.3
	R0F3.1.2.4
	R0F3.1.2.5
	R0F3.1.2.5.1
	R0F3.1.2.5.1.1

Componenti	Requisiti
CLIPS::client	R0F3.1.2.5.2
	R0F3.2
	R0F3.2.1
	R0F3.2.2
	R0F3.2.2.1
	R0F3.2.2.1.1
	R0F3.2.3
	R0F3.2.3.1
	R0F3.2.4
	R0F3.2.4.1
	R0F3.2.5
	R0F3.2.6
	R0F3.3
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F4.3.1
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F5.1.1.6
	R0F5.2
	R0F5.2.1
	R0F5.3
	R0F5.3.1
	R0F6
	R0F6.1
	R0F6.1.1
	R0F6.1.1.1

Componenti	Requisiti
CLIPS::client	R0F6.2
	R0F6.2.1
	R0F6.2.1.1
	R0F6.2.1.2
	R0F6.2.1.3
	R0F6.2.2
	R0F6.2.3
	R0F6.2.3.1
	R0F6.2.3.2
	R0F6.2.3.3
	R0F6.2.3.4
	R0F6.2.3.5
	R0F6.2.3.6
	R0F7
	R0F7.1
	R0F7.2
	R0F7.3
	R1F3.2.3.2

Componenti	Requisiti
CLIPS::client::data	R0F1.1
	R0F1.1.4
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.2
	R0F1.2
	R0F1.2.6
	R0F1.2.6.1
	R0F1.2.7.2
	R0F2
	R0F3
	R0F3.1
	R0F3.2.2.1
	R0F3.2.2.1.1
	R0F3.2.3.1
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F6.1
	R0F6.2
	R0F6.2.1
	R0F6.2.1.1
	R0F6.2.1.2
	R0F6.2.1.3
	R0F6.2.2
	R0F6.2.3

Componenti	Requisiti
CLIPS::client::data	R0F6.2.3.1
	R0F6.2.3.2
	R0F6.2.3.3
	R0F6.2.3.4
	R0F6.2.3.5
	R0F6.2.3.6
	R0F7
	R0F7.1
	R0F7.2

Componenti	Requisiti
CLIPS::client::data::datamanager	R0F1.1
	R0F1.1.4
	R0F1.2.6.1
	R0F2
	R0F3
	R0F3.1
	R0F3.3
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F6
	R0F6.1
	R0F6.1.1.1
	R0F6.2
	R0F6.2.1
	R0F6.2.1.1
	R0F1
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.4.3
	R0F1.2
	R0F1.2.6
	R0F1.2.6.1
	R0F1.2.7
	R0F1.2.7.1
	R0F1.2.7.1.1

Componenti	Requisiti
CLIPS::client::data::datamanager	R0F1.2.7.1.2
	R0F1.2.7.2
	R0F1.2.7.2.1
	R0F1.2.7.2.2
	R0F1.2.7.3
	R0F1.2.7.4
	R0F3
	R0F3.1.2.1
	R0F3.2.5
	R0F3.3
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F6
	R0F6.1
	R0F6.1.1.1
	R0F6.2.1
	R0F6.2.1.1
	R0F7
	R0F7.1
	R0F7.1.1
	R0F7.2
	R0F7.2.1
	R0F7.3

Componenti	Requisiti
CLIPS::server::data	R0F1
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.4.3
	R0F1.2
	R0F1.2.6
	R0F1.2.6.1
	R0F1.2.7
	R0F1.2.7.1
	R0F1.2.7.1.1
	R0F1.2.7.1.2
	R0F1.2.7.2
	R0F1.2.7.2.1
	R0F1.2.7.2.2
	R0F1.2.7.3
	R0F1.2.7.4
	R0F3
	R0F3.1.2.1
	R0F3.2.5
	R0F3.3
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F6
	R0F6.1
	R0F6.1.1.1
	R0F6.2.1
	R0F6.2.1.1
	R0F7
	R0F7.1
	R0F7.1.1
	R0F7.2

Componenti	Requisiti
CLIPS::client::viewcontroller::games	R0F3 R0F3.1 R0F3.1.1 R0F3.1.1.1 R0F3.1.1.2 R0F3.1.2 R0F3.1.2.2 R0F3.1.2.3 R0F3.1.2.4 R0F3.1.2.5 R0F3.1.2.5.1 R0F3.1.2.5.1.1 R0F3.3
CLIPS::client::pathprogress	R0F3 R0F3.1 R0F3.1.1 R0F3.1.1.1 R0F3.1.2.5 R0F3.1.2.5.2 R0F6 R0F6.1.1.1
CLIPS::viewcontroller::savedresults	R0F3.2 R0F3.2.1 R0F3.2.2 R0F3.2.3 R0F3.2.4 R0F3.2.4.1 R0F3.2.6 R0F5.1.1.6 R0F5.2 R1F3.2.3.2

Componenti	Requisiti
CLIPS::server	R0F1
	R0F1.1
	R0F1.1.4
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.2
	R0F1.1.5.4.3
	R0F1.2
	R0F1.2.6
	R0F1.2.6.1
	R0F1.2.7
	R0F1.2.7.1
	R0F1.2.7.1.1
	R0F1.2.7.1.2
	R0F1.2.7.2
	R0F1.2.7.2.1
	R0F1.2.7.2.2
	R0F1.2.7.3
	R0F1.2.7.4
	R0F2
	R0F3.1.2.1
	R0F3.3
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F5.1.1.6

Componenti	Requisiti
CLIPS::server	R0F6
	R0F6.1
	R0F6.1.1.1
	R0F6.2
	R0F6.2.1
	R0F6.2.1.1
	R0F7
	R0F7.1
	R0F7.1.1
	R0F7.2
	R0F7.2.1
	R0F7.3

Componenti	Requisiti
CLIPS::client::data:urlrequest	R0F1.1 R0F1.1.4
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.2
	R0F1.2
	R0F1.2.6
	R0F1.2.6.1
	R0F1.2.7
	R0F1.2.7.1
	R0F1.2.7.2
	R0F2
	R0F3
	R0F3.3
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F5.1.1.6
	R0F6
	R0F6.1
	R0F6.1.1.1
	R0F6.2
	R0F6.2.1
	R0F6.2.1.1
	R0F7
	R0F7.1
	R0F7.2
	R0F7.3

Componenti	Requisiti
CLIPS::server::urlrequesthandler	R0F1.1
	R0F1.1.4
	R0F1.1.5
	R0F1.1.5.1
	R0F1.1.5.2
	R0F1.2
	R0F1.2.6
	R0F1.2.6.1
	R0F1.2.7
	R0F1.2.7.1
	R0F1.2.7.2
	R0F2
	R0F3
	R0F3.3
	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.3
	R0F5.1.1.4
	R0F5.1.1.5
	R0F5.1.1.6
	R0F6
	R0F6.1
	R0F6.1.1.1
	R0F6.2
	R0F6.2.1
	R0F6.2.1.1
	R0F7
	R0F7.1
	R0F7.2
	R0F7.3

Componenti	Requisiti
CLIPS::client::viewcontroller::utility	R0F4
	R0F4.1
	R0F4.2
	R0F4.3
	R0F4.3.1

Componenti	Requisiti
CLIPS::client::viewcontroller	R0F3
	R0F3.1
	R0F3.1.1
	R0F3.1.1.1
	R0F3.1.1.2
	R0F3.1.2
	R0F3.1.2.2
	R0F3.1.2.3
	R0F3.1.2.4
	R0F3.1.2.5
	R0F3.1.2.5.1
	R0F3.1.2.5.1.1
	R0F3.1.2.5.2
	R0F3.2
	R0F3.2.1
	R0F3.2.2
	R0F3.2.3
	R0F3.2.4
	R0F3.2.4.1
	R0F3.2.5
	R0F3.2.6
	R0F3.3
	R0F5
	R0F5.1
	R0F5.1.1
	R0F5.1.1.1
	R0F5.1.1.2
	R0F5.1.1.4
	R0F5.1.1.5
	R0F5.1.1.6
	R0F5.2
	R0F5.2.1
	R0F5.3.1
	R0F6

Componenti	Requisiti
CLIPS::client::viewcontroller	R0F6.1.1 R0F6.1.1.1 R0F6.2 R0F6.2.1 R0F6.2.1.1 R1F3.2.3.2

Tabella 1: Tracciamento componenti-requisiti

5 Tabella tracciamento Requisiti-Componenti

Requisiti	Descrizion	Componenti
R0F1	L'utente deve poter possedere un account registrato nel database	CLIPS::server::data CLIPS::server
R0F1.1	Il sistema deve permettere all'utente di autenticarsi	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.1.1	Il sistema deve chiedere all'utente l'email per la procedura di autenticazione	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.1.2	Il sistema deve chiedere all'utente la password per la procedura di autenticazione	CLIPS::CLIPS::client ::authentication CLIPS::client

Requisiti	Descrizion	Componenti
R0F1.1.3	Il sistema deve permettere all'utente di confermare i dati inseriti per la procedura di autenticazione	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.1.4	Se i dati confermati dall'utente sono validi, egli viene autenticato	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.1.5	Il sistema deve segnalare all'utente ogni eventuale errore durante la procedura di autenticazione	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler

Requisiti	Descrizion	Componenti
R0F1.1.5.1	Il sistema interrompe la procedura di autenticazione se l'email non corrisponde a nessuna di quelle registrate	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.1.5.2	Il sistema interrompe la procedura di autenticazione se la password non corrisponde a quella associata all'email	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.1.5.3	L'applicazi informa l'utente dell'errore nell'inserimento dei dati	CLIPS::CLIPS::client ::authentication CLIPS::client

Requisiti	Descrizion	Componenti
R0F1.1.5.4	L'utente può chiedere una nuova password inserendo l'email	CLIPS::CLIPS::client ::authentication
R0F1.1.5.4.1	L'utente deve inserire l'email del proprio account per proseguire al recupero delle credenziali	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.1.5.4.2	L'utente riceve un'email all'indirizzo inserito con una password casuale.	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.1.5.4.3	La password dell'utente viene sostituita con la password inviata	CLIPS::server::data CLIPS::server
R0F1.2	L'utente deve poter creare un proprio account	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler

Requisiti	Descrizion	Componenti
R0F1.2.1	La registrazione richiede l'email all'utente	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.2.2	La registrazione richiede l'username all'utente	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.2.3	La registrazione richiede la password all'utente	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.2.4	L'utente reinserisce la nuova password	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.2.5	Il sistema chiede all'utente di confermare i dati della propria registrazione	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F1.2.6	Se i dati confermati dall'utente sono validi, il suo account viene registrato	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler

Requisiti	Descrizion	Componenti
R0F1.2.6.1	L'utente viene automaticamente autenticato	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.2.7	Il sistema deve segnalare all'utente i vari problemi di registrazione	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.2.7.1	L'utente ha inserito un'email non valida	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.2.7.1.1	L'email deve avere un formato valido: deve contenere una @ preceduto da altri caratteri seguito da un dominio valido	CLIPS::server::data CLIPS::server

Requisiti	Descrizion	Componenti
R0F1.2.7.1.2	L'email non deve essere già in uso	CLIPS::server::data CLIPS::server
R0F1.2.7.2	L'utente ha inserito un username non valido	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F1.2.7.2.1	L'username deve avere un formato valido: deve contenere solo caratteri alfanu-merici	CLIPS::server::data CLIPS::server
R0F1.2.7.2.2	L'username non deve essere già in uso da un altro utente	CLIPS::server::data CLIPS::server
R0F1.2.7.3	La password deve contenere un minimo di 6 carattere ed un massimo di 16	CLIPS::server::data CLIPS::server

Requisiti	Descrizion	Componenti
R0F1.2.7.4	L'utente ha reinserito una password che non corrisponde a quella inserita in precedenza	CLIPS::server::data CLIPS::server
R0F2	L'utente deve poter fare il logout	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F3	L'utente deve poter effettuare un percorso tra quelli disponibili nel luogo in cui si trova	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::pathprogress CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F3.1	L'utente gioca il percorso selezionato fino alla sua conclusione	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::pathprogress CLIPS::client::viewcontroller
R0F3.1.1	L'utente cerca il beacon corrispondente alla prima stazione	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::pathprogress CLIPS::client::viewcontroller
R0F3.1.1.1	Se il dispositivo dell'utente trova il beacon corrispondente alla stazione corretta l'utente può cominciare a giocare	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::pathprogress CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F3.1.1.2	I beacon che il dispositivo rileva non inerenti con la stazione cercata devono essere ignorati dall'app	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::viewcontroller
R0F3.1.2	L'utente gioca la prova relativa a quella stazione	CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::viewcontroller
R0F3.1.2.1	La prova proposta all'utente è una fra quelle disponibili	CLIPS::server::data CLIPS::server
R0F3.1.2.2	L'app mostra le istruzioni della prova	CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::viewcontroller
R0F3.1.2.3	L'utente deve poter svolgere la prova	CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F3.1.2.4	Finita la prova il dispositivo deve mostrare il risultato ottenuto	CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::viewcontroller
R0F3.1.2.5	Finita la prova l'utente può proseguire il percorso o averlo terminato	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::pathprogress CLIPS::client::viewcontroller
R0F3.1.2.5.1	Se la prova non è l'ultima l'app indica qual'è la prossima stazione	CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::viewcontroller
R0F3.1.2.5.1.1	L'app mostra all'utente le informazioni per trovare la prossima stazione	CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::client::viewcontroller

Requisiti	Descrizio	Componenti
R0F3.1.2.5.2	Se la prova è l'ultima l'utente ha finito il percorso	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::pathprogress CLIPS::client::viewcontroller
R0F3.2	Quando il percorso è finito il dispositivo mostra all'utente il riepilogo dei dati suoi e generali	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller
R0F3.2.1	Il dispositivo mostra il tempo totale del percorso	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller
R0F3.2.2	Il dispositivo mostra il tempo totale impiegato per eseguire le prove	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F3.2.2.1	Il tempo totale per eseguire le prove viene calcolato sommando i tempi impiegati per eseguire ogni prova (è il tempo totale senza considerare gli spostamenti effettuati per cambiare stazione)	CLIPS::client CLIPS::client::data
R0F3.2.2.1.1	La durata della prova viene misurata da quando si accetta di affrontarla fino alla conferma della soluzione	CLIPS::client CLIPS::client::data
R0F3.2.3	Il dispositivo mostra il punteggio totale ottenuto	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F3.2.3.1	Il punteggio totale viene calcolato sommando il numero di punti ottenuto in ogni singola prova	CLIPS::client CLIPS::client::data
R0F3.2.4	Il dispositivo mostra la posizione in classifica che l'utente ha raggiunto con il punteggio ottenuto	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller
R0F3.2.4.1	L'utente deve poter visualizzare la classifica generale CLIPS::clie	CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller
R0F3.2.5	Il dispositivo visualizza la prova con il maggior numero di punti realizzati	CLIPS::client CLIPS::server::data CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F3.2.6	Il dispositivo visualizza la prova con il minor numero di punti realizzati	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller
R0F3.3	L'utente autenticato può salvare il risultato ottenuto	CLIPS::client CLIPS::client::data::datamanager CLIPS::server::data CLIPS::client::CLIPS::client::viewcontroller::games CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F4	L'utente deve poter visualizzare alcune informazioni sull'app	CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::CLIPS::client::viewcontroller::utility

Requisiti	Descrizion	Componenti
R0F4.1	L'utente deve poter visualizzare una schermata con una descrizione generale dell'app	CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::CLIPS::client::viewcontroller::utility
R0F4.2	L'utente deve poter accedere alla pagina web relativa all'app tramite un link presente all'interno dell'app	CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::CLIPS::client::viewcontroller::utility
R0F4.3	L'utente deve poter inviare una segnalazione tramite mail in caso di errore	CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::CLIPS::client::viewcontroller::utility

Requisiti	Descrizion	Componenti
R0F4.3.1	Quando l'utente clicca il pulsante per inviare la segnalazione viene aperta una nuova email da scrivere tramite il gestore di email predefinito. Nell'email il destinatario viene impostato con l'email destinata alle segnalazioni	CLIPS::client CLIPS::client::CLIPS::client::viewcontroller::utility
R0F5	L'utente autenticato deve poter visualizzare i risultati dei percorsi effettuati precedentemente	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F5.1	Se l'utente è autenticato e ha percorsi salvati deve poterne vedere l'elenco	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F5.1.1	Se l'utente clicca su un percorso l'app deve fornire tutte le informazioni salvate	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F5.1.1.1	Il dispositivo deve visualizzare il nome del percorso	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F5.1.1.2	Il dispositivo deve visualizzare il nome del luogo dove si è svolto il percorso	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F5.1.1.3	Il dispositivo deve visualizzare la CLIPS::clie in cui si è svolto il percorso	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F5.1.1.4	Il dispositivo deve visualizzare il punteggio totale del percorso	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F5.1.1.5	Il dispositivo deve visualizzare il tempo totale per lo svolgimento del percorso	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F5.1.1.6	Il dispositivo può visualizzare la posizione attuale nella classifica generale del risultato ottenuto	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F5.2	Se l'utente è autenticato ma non ha percorsi svolti salvati viene mostrato un invito a svolgere un percorso.	CLIPS::client CLIPS::CLIPS::client::viewcontroller::savedresults CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F5.2.1	Nella spiegazione si informa l'utente che in quella schermata è possibile visualizzare i percorsi svolti quando si salveranno.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::viewcontroller
R0F5.3	Se l'utente non è autenticato viene invitato ad autenticarsi.	CLIPS::CLIPS::client ::authentication CLIPS::client
R0F5.3.1	L'app deve spiegare che l'utente, quando è autenticato, può vedere l'elenco dei percorsi salvati in quella schermata.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F6	L'utente deve poter visualizzare dove si trovano gli edifici con percorsi più vicini alla sua posizione.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F6.1	L'utente deve poter cercare gli edifici con percorsi più vicini inserendo un raggio massimo.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F6.1.1	L'utente deve poter inserire il raggio in chilometri per poter cercare gli edifici con percorsi più vicini.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F6.1.1.1	Se la ricerca ha esito positivo vengono elencati tutti gli edifici con percorsi presenti nell'area cercata.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F6.2	Cliccando su un edificio della lista l'utente deve poter visualizzare tutte le informazioni su di esso.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller

Requisiti	Descrizion	Componenti
R0F6.2.1	L'utente deve poter visualizzare delle informazioni specifiche sull'edificio.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F6.2.1.1	L'utente deve poter visualizzare il nome dell'edificio.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data CLIPS::client::data::datamanager CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler CLIPS::client::viewcontroller
R0F6.2.1.2	L'utente deve poter visualizzare la destinazione d'uso dell'edificio (ad esempio museo di storia egizia).	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data

Requisiti	Descrizion	Componenti
R0F6.2.1.3	L'utente deve poter visualizzare l'indirizzo dell'edificio.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data
R0F6.2.2	L'utente deve poter visualizzare il link al sito dell'edificio.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data
R0F6.2.3	L'utente deve poter visualizzare alcuni contatti dell' edificio.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data
R0F6.2.3.1	L'utente deve poter visualizzare il contatto telefonico dell'edificio.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data
R0F6.2.3.2	L'utente deve poter contattare l'edificio per email.	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data

Requisiti	Descrizion	Componenti
R0F6.2.3.3	L'utente deve poter contattare l'edificio tramite Facebook (se la struttura possiede un account Facebook).	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data
R0F6.2.3.4	L'utente deve poter contattare l'edificio tramite Twitter (se la struttura possiede un account Twitter)	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data
R0F6.2.3.5	L'utente deve poter cotattare l'edificio tramite WhatsApp (se la struttura possiede un contatto pubblico di WhatsApp).	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data

Requisiti	Descrizion	Componenti
R0F6.2.3.6	L'utente deve poter contattare l'edificio tramite Telegram (se la struttura ha un contatto pubblico di Telegram).	CLIPS::CLIPS::client ::CLIPS::client::viewcontroller::building CLIPS::client CLIPS::client::data
R0F7	L'utente autenticato deve poter modificare le proprie credenziali d'accesso.	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F7.1	L'utente deve poter modificare la propria password.	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F7.1.1	La password deve rispettare il requisito R0F1.2.7.3	CLIPS::server::data CLIPS::server

Requisiti	Descrizion	Componenti
R0F7.2	L'utente deve poter modificare il proprio username.	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::client::data CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler
R0F7.2.1	L'username deve rispettare i requisiti R0F1.2.7.2 e R0F1.2.7.2	CLIPS::server::data CLIPS::server
R0F7.3	In caso di errore su qualche campo l'app deve informare l'utente	CLIPS::CLIPS::client ::authentication CLIPS::client CLIPS::server::data CLIPS::server CLIPS::client::data:urlrequest CLIPS::server::urlrequesthandler

Tabella 2: Tracciamento requisiti-componenti

A Design pattern utilizzati

A.1 Pattern architetturali

A.1.1 Model View Presenter

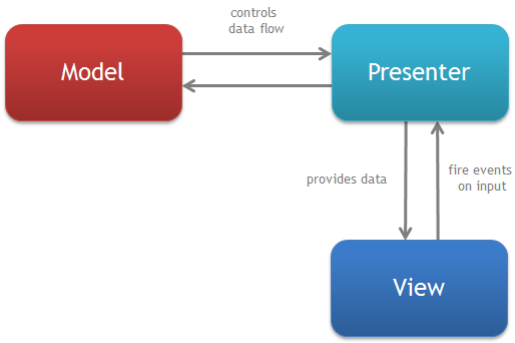

Figura 27: Struttura del pattern MVP

- **Descrizione**
  Model View Presenter è un design pattern architetturale simile al Model View Controller nel quale il presenter è posizionato tra model e view. Permette di dividere l'architettura del sistema che si intende sviluppare in tre blocchi:
  - **il model** che contiene le classi con i metodi di accesso ai dati;
  - **la view**, completamente pasiva, contiene le classi che permettono all'utente di visualizzare i dati e segnala tramite gli eventi le interazioni dell'utente;
  - **il presenter** che si occupa di fare da tramite tra vista e modello, ovvero riceve i comandi dell'utente attraverso la vista e va a cambiare lo stato del modello di conseguenza, successivamente aggiorna la vista.
- **Vantaggi**
  Questo pattern architetturale permette il riuso del codice in quanto molte parti di lavoro sono indipendenti. Ad esempio il modello creato potrà essere utilizzato con diverse viste.
  Grazie all'indipendenza di alcune parti è semplice dividere il lavoro tra più componenti di un team e sarà quindi più facile anche la manutenzione.
- **Utilizzo nel progetto**
  Nel nostro progetto il pattern MVP è usato nella parte client dell'architettura, nello specifico il model è rappresentato dalle classi del database locale, ciè quelle del package **Data**, le view sono rappresentate dai **file XML** di Android che vengono utilizzati per la gestione del layout mentre il presenter è costituito dalle classi **activity** e dalle classi all'interno del package **Datamanager**.

A.2 Pattern creazionali

A.2.1 Abstract Factory

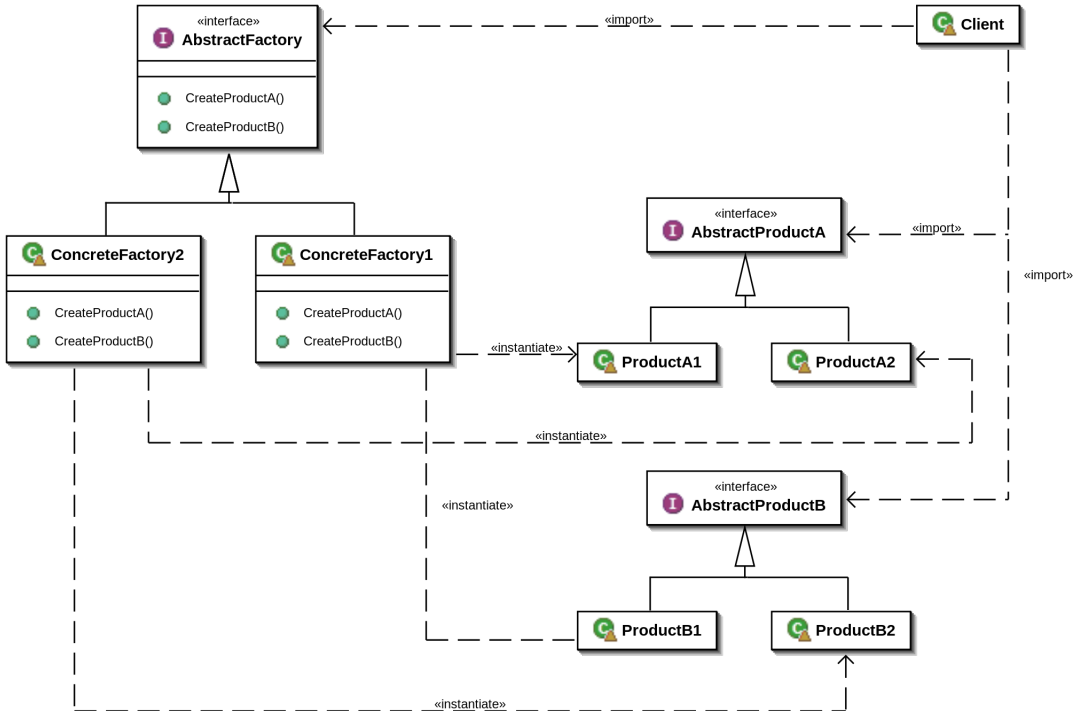

Figura 28: Struttura del pattern Abstract Factory

- **Descrizione**
  Il design pattern Abstract Factory fornisce un'interfaccia per creare famiglie di prodotti senza specificare classi concrete. Ogni famiglia di prodotti ha una classe base astratta da cui derivano delle classi concrete. Queste classi concrete sono istanziate dalle classi concrete della factory corrispondenti.
- **Vantaggi**
  Questo design pattern offre vantaggi quando si vogliono modellare famiglie di prodotti che potranno essere ampliate nel futuro. Le modifiche necessarie ad aggiungere nuovi elementi alle famiglie saranno essenzialmente due, aggiungere una classe in ogni famiglia di prodotti ed un solo metodo nella factory.
- **Utilizzo nel progetto**
  Nel nostro progetto abbiamo utilizzato questo design pattern per modellare i tipi di quiz che vengono proposti agli utenti. Abbiamo infatti deciso di utilizzare solo due famiglie di quiz, quindi vi è la necessità di avere modo in futuro di ampliare le tipologie di quiz disponibili in modo semplice ed efficace.

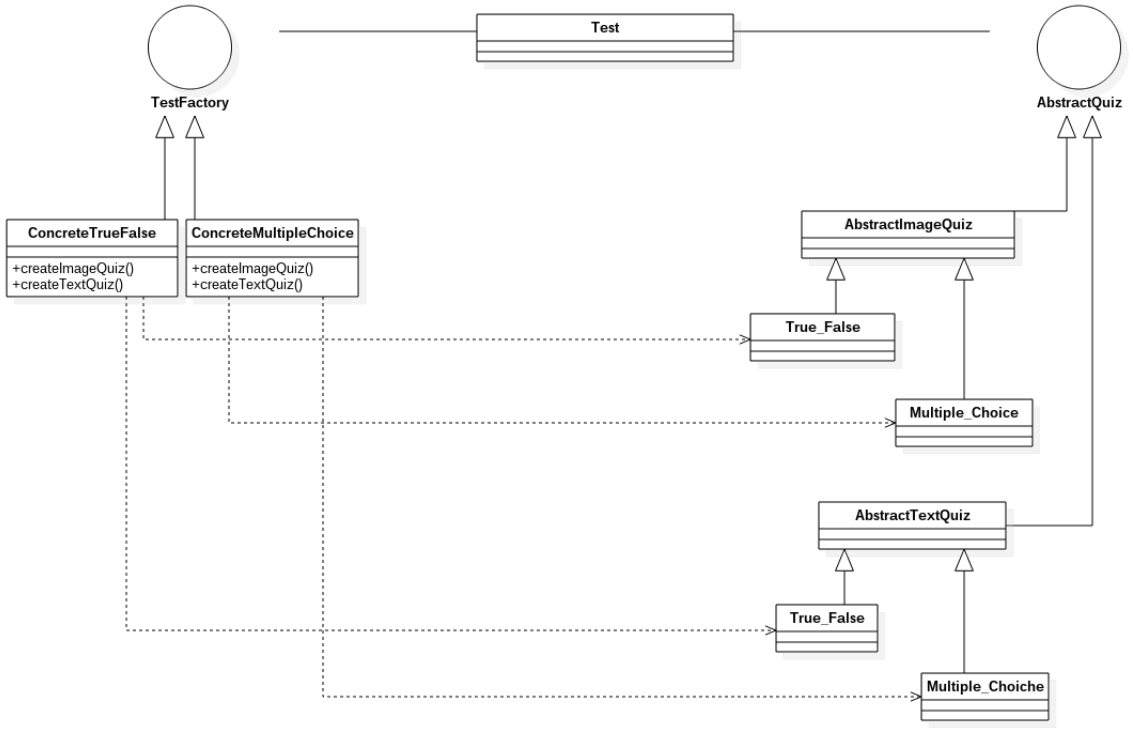

Figura 29: Utilizzo di Abstract Factory nel progetto

A.2.2 Singleton

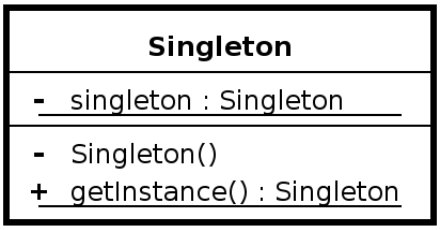

Figura 30: Struttura del pattern Singleton

- **Descrizione**
  Assicura l'esistenza di un'unica istanza di una classe e permette di avere un punto di accesso globale a questa. Per rendere possibile ciò si mette il costruttore protetto o privato e si crea un metodo statico, chiamato factory, che fornisce l'accesso all'unica copia dell'oggetto (contiene un puntatore all'unica istanza).
  È stata valutata come alternativa la *Dependency Injection*$_g$, un altro *design pattern*$_g$. Essa prevede la costruzione di una classe le cui dipendenze vengono ricevute dall'esterno. Di base non ha le stesse proprietà del *Singleton*$_g$, ma può acquisirle tramite degli appositi accorgimenti, come l'utilizzo di una *Factory*$_g$ per la costruzione delle dipendenze. Questo *design pattern*$_g$ ha come vantaggi una maggiore facilità nel testare la classe, perché si possono creare delle dipendenze false da inviare alla *Dependency Injection*$_g$, e la separazione tra il comportamento della componente dalla risoluzione delle sue dipendenze; al contrario il *Singleton*$_g$ rende privato tutto quello che le occorre per costruire la classe, mantenendolo quindi al pro-

prio interno. Lo svantaggio invece risiede nella maggiore complessità di costruzione, in quanto la *Dependency Injection*$_{\text{g}}$ richiede l'utilizzo di almeno due classi con una forte dipendenza tra loro, anziché un'unica classe come il *Singleton*$_{\text{g}}$. Quest'ultimo è stato scelto alla fine perché nel nostro caso le dipendenze sono poche e il risultato prodotto è facilmente verificabile tramite degli appositi metodi, perciò lo svantaggio della *Dependency Injection*$_{\text{g}}$ peserebbe molto di più rispetto a vantaggi.

- **Vantaggi**
  Se, al contrario di quanto avviene utilizzando Singleton, venisse reso visibile il costruttore della classe non si potrebbe garantire che esista un solo esemplare della classe. Un altro modo di procedere potrebbe essere quello di dichiarare una variabile globale, ma in questo modo si "ruberebbe" un nome al namespace globale. Sigleton permette inoltre di dichiarare sottoclassi.

- **Utilizzo nel progetto**
  Nel nostro progetto abbiamo creato un singleton per "LoginManager" che contiene i dati relativi a i dati di login dell'utente che sta utilizzando l'applicazione.

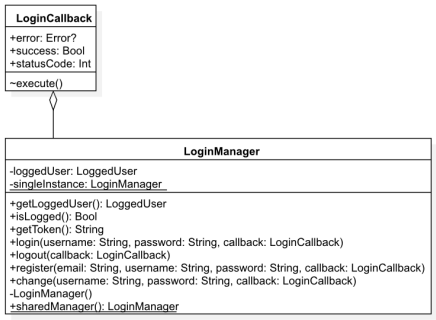

Figura 31: Utilizzo di Abstract Factory nel progetto