**­KEY-KNOWLEDGE**



**Object Design Document**

Version 1.0

**TABELLA REVISIONI**

|  |  |  |  |
| --- | --- | --- | --- |
| **DATA** | **VERSIONE** | **DESCRIZIONE** | **AUTORE** |
| **04/11/2020** | **1.0** | **Prima versione del documento Requisiti e casi d’uso** | **Nicola Fernando Franco**  **Crescenzo Mazzone**  **Giovanni Battista Mercurio** |
| **20/02/2020** | **1.5** | **Prima revisione** | **Giovanni Battista Mercurio** |
| **18/04/2020** | **2.0** | **Revisione finale** | **Giovanni Battista Mercurio** |

***Package***

***VIEW***

Il package delle view all’interno della nostra app è chiamato layout , contiene all’interno varie pagine XML, come prima pagina xml principale abbiamo “Activity\_main.xm” che è la nostra home del sistema e anche pagina principale dell’app.  
da qui in poi ogni scelta ci porterà a nuove schermate quindi ad altre pagine xml.  
Un esempio può essere lo scorrimento verso sinistre per accedere alla schermata knowledge in questo modo passiamo dalla nostra “Activity\_main.xm” a “knowledge.xml”.  
ogni operatore sulle nostre pagine xml come bottoni, caselle di testo ,griglie ed altro vengono gestite dalle nostre classe java.

|  |  |
| --- | --- |
| Nome | Descrizione |
| Activity\_main.xml | Home del sistema |
| Home.xml | Home dell’Applicazione |
| Knowledge.xml | Schermata knowledge |
| Login.xml | Area di accesso all’APP |
| Match.xml | Schermata della partita |
| Search\_player.xml | Schermata di abbinamento giocatori |
| End\_Match.xml | Schermata di fine partita |
| Alert\_connection\_layout.xml | Schermata di avviso di connessione dispositivo |
| Drawer\_top.xml | Schermata superiore home |
| Item\_answer.xml | Schermata di quiz |
| Result\_layout.xml | Schermata di riassunto partita |
| Vignetta\_layout.xml | Schermata funzionamento |

***CONTROL***

Il package controller riceve, tramite il pacchetto View, i comandi dell’utente. Ogni classe control gestisce apposti metodi invocati dall’utente tramite le pagine xml durante l’uso dell’App come ad esempio sulla schermata di Login attraverso l’Apposito tasto genera un evento che invoca il metodo nella classe LoginControl.

|  |  |
| --- | --- |
| Nome | Descrizione |
| MainControl | Gestione della home |
| KnowledgeControl | Gestione dei dati riguardanti I giocatori |
| LoginControl | Gestione del Log-in |
| MatchControl | Gestione delle partite |
| PairingControl | Gestione degli abbinamenti |
| EndMatchControl | Gestione della fine di ogni partita |
| QuestionControl | Gestione delle domande |
| QuizControl | Gestioni dati in partita |
| UserControl | Gestione dati utente |

***MODEL***

Il package Model contiene tutte le classi dedite alla gestione dei dati persistenti. Esso si occupa di fare da tramite tra l’applicazione e il database sottostante. Ogni classe contenuta all’interno di questo pacchetto fornisce i metodi per accedere ai dati utili all’applicazione. Le classi contenute all’interno di questo package sono le seguenti:

## model.entity

|  |  |
| --- | --- |
| Nome | Descrizione |
| EndMatch | Rappresenta la fine della partita |
| IaModule | Rappresenta il modulo di inteligenza artificiale |
| Knowledge | Rappresenta la schermata Knowledge |
| Login | Rappresenta l’area di Login |
| MainActivity | Rappresenta l’Activity principale della nostra app |
| Pairing | Rappresenta l’abbinamento dei 2 giocatori |
| Question | Rappresenta le domande |
| Quiz | Rappresenta lo stato della partita |
| User | Rappresenta l’utente connesso |
| AdapterWrapper | Rappresenta la risposta dell’utente inerente a una domanda |
| InternetConnection | Rappresenta la connessione ad internet |
| Match | Rappresenta la partita |

## model.dao

|  |  |
| --- | --- |
| Nome | Descrizione |
| MatchManager | Modella le domande durante la partita |
| EndMatchManager | Modella le informazioni finali della partita col DataBase |
| KnowledgeManager | Modella le informazioni dell’area Knowledge col DataBase |
| LoginManager | Modella le informazioni utente col DataBase |
| PairingManager | Modella le interazioni dei 2 utenti col DataBase |
| QuestionManager | Modella le informazioni delle domande col DataBase |
| QuizManager | Modella le informazioni della partita col DataBase |
| UserManager | Modella le informazioni dell’utente col DataBase |

***Interfaccia delle classi***

***USER***

|  |  |
| --- | --- |
| Nome | User |
| Descrizione | Rappresenta un utente registrato nel sistema |
| Attributi | * String nickname * String email * String password * String stato * String ruolo * Int numPartiteVinte * int numPartiteGiocate |
| Signature dei metodi | * getNickname() * setNickname(String x) * getEmail() * setEmai(String x) * getPassword * SetPassword(String x) * getStato() * setStato(String x) * getRuolo() * setRuolo(String x) * getNumPartiteGiocate() * setNumPartiteGiocate(int x) * getNumPartiteVinte() * setNumPartiteVinte(int x) |
| Pre-condizioni | **context** User::setNickName(n)  **pre**: n.matches(‘^[A-Za-z0-9]{​​6,25}​​$’) |
| **context** User:setEmail(e)  **pre**: e.matches**[a-zA-Z0-9.\_%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4})** |
| **context** User::setPassword(p)  **pre**: p.matches **((?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[@#$%]).{8,20})** |
|  |
|  |
| Post-condizioni |  |
| Invarianti |  |

***QUIZ***

|  |  |
| --- | --- |
| Nome | Quiz |
| Descrizione | Rappresentano i quiz durante le partite |
| Attributi | * int id * String mode * int numQuesiti * user user1 * user user2 * Int punteggioG1 * int punteggioG2 * String status |
| Signature dei metodi | * Quiz(int a, String b, int c, String d, String e) * getid() * setId(int id) * getMode() * setMode(String mode) * getUser1() * setUser1(String user 1) * getUser2() * setUser2(String user2) * getNumQuesiti() * setNumQuesiti(int numQuesiti) * getPunteggioG1 () * setPunteggioG1 (int punteggioG1) * getPunteggioG2 () * setPunteggioG2 (int punteggioG2) * getStatus() * setStatus(String status) |
| Pre-condizioni | **Context** Quiz:: setId(i) **pre**: i.matches (‘^[A-Za-z0-9]+$’) |
| **Context** Quiz:: setMode(m) **pre**: m.matches (‘^[A-Za-z]+$’) |
| **Context** Quiz:: setNumQuesiti(nm) **pre**: nm.matches (‘^[0-9]+$’) |
| **Context** Quiz:: setPunteggiog1(p1) **pre**: p1.matches (‘^[0-9]+$’) |
| **Context** Quiz:: setPunteggiog2(p2) **pre**: p2.matches (‘^[0-9]+$’) |
| Post-condizioni |  |
| Invarianti |  |

***QUESTION***

|  |  |
| --- | --- |
| Nome | Question |
| Descrizione | Rappresentano tutte le domande |
| Attributi | * String id * String testo * String risposta1 * String risposta2 * String risposta3 * String risposta4 * String categoria * Int risposta\_esatta * Int livello |
| Signature dei metodi | * getId() * setId(String id) * getTesto() * setTesto(String testo) * getRisposta1() * setRisposta1(String risposta1) * getRisposta2() * setRisposta2(String risposta2) * getRisposta3() * setRisposta3(String risposta3) * getRisposta4() * setRisposta4(String risposta4) * getCtegoria() * setCategoria(String categoria) * getRisposta\_esatta() * setRisposta\_esatta(String risposta\_corretta) * getLivello() * setLivello(int livello) |
| Pre-condizioni | **context** Question::setTesto(t)  **pre**: t.matches **[a-zA-Z0-9]** |
| **context** Question::setRisposta1(r1)  **pre**: r1.matches **[a-zA-Z0-9]** |
| **context** Question::setRisposta2(r2)  **pre**: r2.matches **[a-zA-Z0-9]** |
| **context** Question::setRisposta3(r3)  **pre**: r3.matches **[a-zA-Z0-9]** |
| **context** Question::setRisposta4(r4)  **pre**: r4.matches **[a-zA-Z0-9]** |
| Post-condizioni |  |
| Invarianti |  |

***LOGIN***

|  |  |
| --- | --- |
| Nome | Login |
| Descrizione | Rappresenta l’area di accesso all’APP |
| Attributi | * SharedPreferences pref * EditText us * EditText pass * LoginControl control |
| Signature dei metodi | * onCreate(Bundle saveInstanceState) * message(String x) * access(View view) * register(View view) * saveUser(User user) |
| Pre-condizioni |  |
|  |
|  |
|  |
|  |
| Post-condizioni |  |
| Invarianti |  |

***PAIRING***

|  |  |
| --- | --- |
| Nome | Pairing |
| Descrizione | Rappresenta l’abbinamento tra 2 giocatori |
| Attributi | * User user * String mode * Quiz quiz |
| Signature dei metodi | * onCreate(Bundle saveInstanceState) * message(String x) * setQuiz(Quiz x) * onBackPressed() |
| Pre-condizioni |  |
|  |
|  |
|  |
|  |
| Post-condizioni |  |
| Invarianti |  |

***KNOWLEDGE***

|  |  |
| --- | --- |
| Nome | Knowledge |
| Descrizione | Rappresenta la schermata Knowledge |
| Attributi | * User user |
| Signature dei metodi | * onCreate(Bundle saveInstanceState) |
| Pre-condizioni |  |
|  |
| Post-condizioni |  |
| Invarianti |  |

***MATCH***

|  |  |
| --- | --- |
| Nome | Match |
| Descrizione | Rappresenta la partita |
| Attributi | * LinearLayout grid * TextView text * TextView numero * TextView categoria * TextView livello * Button b1 * Button b2 * Button b3 * Button b4 * Button confirm * int currentQuestion * int risposta\_corretta * int player * Quiz quiz * Question question * MatchControl control |
| Signature dei metodi | * onCreate(Bundle saveInstanceState) * onClick(View v) * setQuestions(Question q) * message(String x) * next(View view) * onRestart() * onResume() * onStop() * onDestroy() |
| Pre-condizioni |  |
|  |
| Post-condizioni |  |
| Invarianti |  |

***MAIN ACTIVITY***

|  |  |
| --- | --- |
| Nome | Main\_Activity |
| Descrizione | Rappresenta il main |
| Attributi | * SharedPreferences pref * User user * MainControl control * TextView textView * ImageView logo * GestureDetector detector * Animation zoom |
| Signature dei metodi | * Oncreate(Bundle saveInstanceState) * onStart() * setContent(int x, int y) * message(String x) * login(View view) * logout(View view) * onTouchEvent(MotionEvent event) * startMatch1(View view) * startMatch2(View view) * startMatch3(View view) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***END MATCH***

|  |  |
| --- | --- |
| Nome | End Match |
| Descrizione | Rappresenta la fine di una partita |
| Attributi | * Quiz quiz * Int player * EndMatchControl control * TextView text * Button returnButton |
| Signature dei metodi | * Oncreate(Bundle saveInstanceState) * waitOpponent() * end(Quiz q) * setText(String x) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***LOGIN MANAGER***

|  |  |
| --- | --- |
| Nome | Login\_manager |
| Descrizione | Modella le interazioni riguardante l’area di log-in |
| Attributi | * String TABLE * String OFFLINE * String ONLINE |
| Signature dei metodi | * accessUser(String x, String pass, LoginControl control) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***MATCH MANAGER***

|  |  |
| --- | --- |
| Nome | Match\_manager |
| Descrizione | Modella le interazioni riguardante le partite |
| Attributi | * String table * int CATEGORIES * int LEVELS * int QUESTIONS * Sring[] categories * Sring[] levels * Sring[] questions * Quiz quiz * DataBaseReference mDatabase * MatchControl control * IaModule module * QuestionManager managerQuestion * ValueEventListner listner |
| Signature dei metodi | * MatchManager(Quiz q, MatchControl c) * getQuestion() * getQuestion(int current, Boolean resp) * setQuitListner(Quiz quiz, int player) * quit() * onCancelled() |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***PAIRING MANAGER***

|  |  |
| --- | --- |
| Nome | Pairing\_manager |
| Descrizione | Modella le interazioni tra i 2 giocatori |
| Attributi | * int MAX\_ROOMS * int LAST\_ROOM * String TABLE * DatabaseReference mBatabase * PairingControl control * MatchControl control2 |
| Signature dei metodi | * PairingManager(PairingControl c) * createMatch(User user, String mode) * resetMatch(Quiz quiz) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***Question Manager***

|  |  |
| --- | --- |
| Nome | Question\_manager |
| Descrizione | Modella le interazioni riguardo le domande |
| Attributi | * String TABLE * DatabaseReference mBatabase * QuestionControl control |
| Signature dei metodi | * QuestionManager(MatchControl c) * getQuestion(String categoria,String livello , String id) * onCancelled(DataBaseError error) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***Main Manager***

|  |  |
| --- | --- |
| Nome | Main\_manager |
| Descrizione | Modella le interazioni riguardante l’area di benvenuto |
| Attributi | * String TABLE * String ONLINE * String OFFLINE * DatabaseReference mBatabase |
| Signature dei metodi | * accessUser(String nick, MaInControl control) * onDataChange() * onCancelled() * logout(User user) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***EndMatch Manager***

|  |  |
| --- | --- |
| Nome | EndMatch\_manager |
| Descrizione | Modella le interazioni riguardante il termine della partita |
| Attributi | * EndMatchControl control * DatabaseReference mBatabase * String TABLE |
| Signature dei metodi | * EndMatchManger() * updateMatch(Quiz quiz, int player) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***Quiz Manager***

|  |  |
| --- | --- |
| Nome | QuizManager |
| Descrizione |  |
| Attributi | * int MAX\_ROOMS * String TABLE * DatabaseReference mBatabase * ValueEventListner listner * Quiz quiz * Quiz quizInEvent * QuizControl controller |
| Signature dei metodi | * QuizManager(Quiz q) * setQuitListner(Quiz quiz,int player,MatchControl control) * quit(Quiz quiz) * getQuizEvent() * setController(QuizController controller) * createQuiz(User user, String mode, PairingControl control) * onComplete(DatabaseError error, booleancommitted, DataSnapshot currentData) * resetQuiz(Quiz quiz) updateQuiz(Quiz quiz,intplayer,EndMatchControl control) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

***User Manager***

|  |  |
| --- | --- |
| Nome | UserManager |
| Descrizione |  |
| Attributi | * String TABLE * DatabaseReference mBatabase * UserControl controller * User user |
| Signature dei metodi | * setState(String state,String nick) * getUser(String nick,String pass,LoginManager l) * getUserInEvent() |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |