
UNIVERSITÀ DEGLI STUDI DI SALERNO

Ingegneria del Software

# System Design Document

**ANNO ACCADEMICO 2016/2017**

**Versione 1.6**


## Partecipanti:

| NOME | MATRICOLA |
|---|---|
| **Iannone Ida** | **0512102520** |
| **Della Porta Raffaele** | **0512102538** |

# Revision History:

| DATA | VERSIONE | DESCRIZIONE | AUTORE |
|---|---|---|---|
| 1/12/16 | 1.0 | scopo del sistema, obiettivi di design, criteri di performance. | Raffaele Della Porta |
| 1/12/16 | 1.0 | criteri di affidabilità, manutenibilità e costo | Ida Iannone |
| 2/12/16 | 1.1 | criteri per l'utente finale | Membri del team |
| 2/12/16 | 1.2 | sistema corrente e panoramica sistema proposto | Raffaele Della Porta |
| 2/12/16 | 1.2 | decomposizione del sistema, mapping HW/SW, controllo degli accessi e della sicurezza. | Ida Iannone |
| 3/12/16 | 1.3 | Controllo del Software globale, gestione dati persistenti, condizioni di boundary. | Raffaele della Porta |
| 4/12/16 | 1.4 | Revisione del documento fino a questo punto | Membri del team |
| 5/12/16 | 1.5 | glossario e completamento del documento. | Ida Iannone |
| 7/12/16 | 1.6 | Revisione completa del documento. | Raffaele Della Porta |

## Indice



# 1. Introduzione

## 1.1. Scopo del sistema

Da qualche anno nelle scuole primarie gli insegnanti devono rapportarsi con bambini sempre meno inclini allo studio sui libri. Data la facilità con cui usano i dispositivi elettronici e informatici per reperire informazioni, è adeguato unire l'utile al dilettevole, quindi stimolare l'interesse dei bambini con l'uso di giochi istruttivi. Lo scopo è aiutare gli insegnanti nel percorso scolastico per facilitare l'apprendimento di materie noiose tramite giochi istruttivi.

## 1.2. Obiettivi di Design

In un software gli obiettivi di design rappresentano le basi per il successivo sviluppo del prodotto, perché su di esse si fondano le scelte prese durante la fase di implementazione. Il sistema avrà quindi una struttura chiara, completa ma semplice da utilizzare. L'utilizzo del sistema non prevede alcuna competenza informatica grazie all'utilizzo di un'interfaccia atta a rendere l'interazione dell'utente facile e intuitiva. Gli obiettivi del design servono per identificare le qualità su cui deve essere focalizzato il sistema. Ogni importante decisione di design viene presa seguendo dei criteri che sono organizzati in 4 gruppi : Performance, Affidabilità, Manutenibilità e Utente finale.

### 1.2.1 Criteri di Performance

| | |
|---|---|
| **Tempi di Risposta** | Il sistema sarà in grado di rispondere in modo veloce all'utente, per evitare che l'utente si annoi o si distragga durante l'interazione in particolare nell'esecuzione dei giochi. |
| **Throughput** | La capacità di elaborazione del sistema permetterà di gestire in modo ottimale anche molteplici accessi. |
| **Memoria** | Non può essere stimata la quantità di memoria che verrà utilizzata dal sistema PlayForLearn. Il sistema dovrà essere in grado di memorizzare tutti gli alunni delle varie classi, che vogliono usufruire dell'applicazione. |

### 1.2.2 Criteri di Affidabilità

| | |
|---|---|
| **Robustezza** | Il sistema dovrà offrire un buon grado di robustezza agli input invalidi forniti dagli utenti. Il sistema, quindi, elaborerà i dati forniti dall'utente e nel caso in cui gli stessi siano errati questo lancerà un messaggio d'errore per avvisare l'utente che i dati inseriti non sono validi. Il sistema permette di interagire con il gioco solo selezionando una delle risposte disponibili in modo da evitare input non validi. |
| **Affidabilità** | Il sistema dovrà rispondere agli input forniti dagli utenti producendo unicamente l'output atteso. |

| | |
|---|---|
| **Disponibilità** | Il sistema deve permettere l'utilizzo di tutte le funzionalità anche ad una classe intera che si collega contemporaneamente al sistema, senza conseguenze sul carico di lavoro. Il sistema deve essere disponibile al 100% dalle 8:00 alle 20:00**.** |
| **Tolleranza ai Fault** | Verranno gestire in modo trasparente eventuali eccezioni generate dall'utilizzo e dall'esecuzione del sistema. |
| **Sicurezza e Fidatezza** | Il sistema di gestione della sicurezza prevede un login, per poter accedere all'applicazione; ad ogni tipologia di utente vengono assegnati dei permessi. Verranno gestiti i casi di errore in modo da non perdere informazioni dell'utente. |

### 1.2.3 Criteri di Manutenibilità

| | |
|---|---|
| **Estendibilità** | Bisogna modellare l'inserimento di nuove funzionalità sulla base di quelle esistenti, senza sostanziali modifiche. |
| **Modificabilità** | Deve essere possibile effettuare delle modifiche al sistema, nel caso di eventuali errori di sistema, o in caso di bug di qualche gioco. Per ottenere queste modifiche, il codice deve essere ben strutturato per poter essere riutilizzato. |
| **Leggibilità** | Il codice sarà facilmente leggibile e dotato di documentazione specifica per ogni costrutto utilizzato. |
| **Tracciabilità dei requisiti** | Tramite una buona documentazione, relativa al codice che verrà scritto, sarà possibile risalire ai rispettivi requisiti funzionali, use case e qualsiasi altro artefatto prodotto nelle fasi antecedenti l'implementazione. |
| **Portabilità** | il sistema sarà utilizzabile su qualunque dispositivo Android. |

### 1.2.4 Criteri di Costo

| | |
|---|---|
| **Manutenzione** | Un sistema ben organizzato non richiede particolari sprechi di tempi per la manutenzione. |
| **Produzione** | |

### 1.2.5 Criteri per l'Utente finale

| | |
|---|---|
| **Usabilità** | Il sistema deve essere semplice e intuitivo, e deve garantire una user-experience rilassante e piacevole, senza mai stancare l'utente. |
| **Utilità** | Play For Lear deve effettivamente insegnare qualcosa ai bambini che ne fanno uso. |

## 1.3 Definizioni, acronimi, abbreviazioni

In questa sezione si specificano gli acronimi e le abbreviazioni utilizzate nel seguito. Essi, infatti, pur essendo di uso comune, potrebbero indurre a interpretazioni personali, quindi potenzialmente diverse da quelle sottintese in questa trattazione.

- PFL: PlayForLearn, nome del sistema che verrà sviluppato.
- Alunno: Utente del sistema.
- Insegnante: Amministratore del sistema.
- UI: user interface.
- SDD: System Design Document.
- HW/SW: Hardware/Software.
- ODD: Object Design Document.
- DB: Database.
- AM: matrice degli accessi.

## 1.4 Riferimenti

Per la realizzazione del sistema sono stati utilizzati i seguenti materiali di riferimento:

- Android guida: https://developer.android.com/index.html
- Android Programming: The Big Nerd Ranch Guide: http://www.bignerdranch.com/we-write/android-programming.html
- B.Bruegge, A.H. Dutoit, Object Oriented Software Engineering - Using UML, Patterns and Java, Prentice Hall, 3rd edition, 2009.

## 1.5 Panoramica

Al secondo punto verrà presentato il sistema corrente, se esistente. Al terzo punto verrà presentata l'architettura del sistema proposto in cui gestiremo la decomposizione in sottosistemi, il mapping hardware/software, i dati persistenti, il controllo degli accessi e sicurezza, il controllo del flusso globale del sistema e le condizioni limite. Di seguito verranno presentati i servizi del sottosistema.

# 2. Architettura del software corrente

L’architettura proposta nel seguito non andrà a rimpiazzare nessuna struttura pre-esistente. Infatti la progettazione e lo sviluppo di essa segue, essenzialmente, i criteri della Green Field Engineering.

# 3. Architettura del software proposto

## 3.1. Panoramica

Play For Learn è un sistema Greenfield Engineering.
E’ un software che nasce per dare supporto agli insegnanti, proponendo dei giochi integrativi e didatticamente utili.
Lo stile architetturale che implementeremo sarà quello Model/View/Controller. In questa architettura i sottosistemi sono classificati in 3 tipi differenti:
Sottosistema Model: mantiene la conoscenza del dominio applicativo.
Sottosistema View: visualizza all’utente gli oggetti del dominio applicativo.
Sottosistema Controller: responsabile della sequenza di interazioni con l’utente e di notificare ai View i cambiamenti nel modello. L'MCV è di fondamentale importanza nell’ambito della programmazione Object-Oriented, poiché separa la logica di presentazione dei dati dalla logica applicativa. Nello specifico:

**Model:**

Il model mantiene conoscenza del dominio di applicazione, fornisce le operazioni di accedere ai dati utili dell’applicazione. Contiene tutti gli oggetti entity del sistema.

**View:**

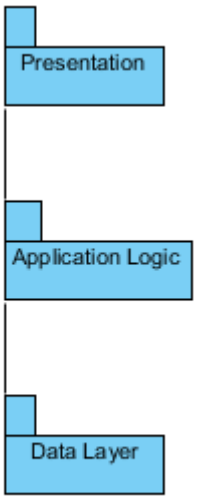
La logica di presentazione dei dati viene gestita solo e solamente dalla View. Ciò implica che questa deve fondamentalmente gestire la costruzione dell'interfaccia grafica. La nostra view comprenderà l’interfaccia grafica della home page dell’applicazione, con i possibili comandi che l’utente potrà cliccare. Fra questi sarà possibile scegliere una materia per poter giocare ad un gioco, poter visualizzare la top-10, poter visualizzare i propri record personali e visualizzare il proprio profilo. L’interfaccia grafica può essere costituita da schermate diverse che presentano più modi di interagire con i dati dell'applicazione.

**Controller:**

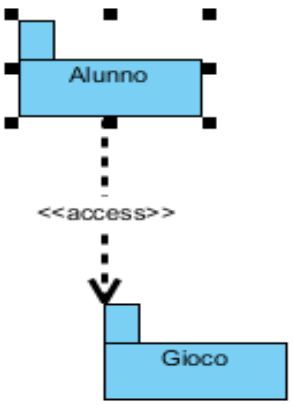
Questo componente ha la responsabilità di trasformare le interazioni dell'utente della View in azioni eseguite dal Model. Nel nostro sistema i controller saranno tutti i vari gestori per poter fare l’autenticazione(login e

logout); la gestione utenti (inserimento, eliminazione, accettazione e rifiuto utente); gestione della guida(visualizzare guida), gestione dei record e la gestione gioco(avvio e interruzione del gioco).

## 3.2. Decomposizione del Sistema

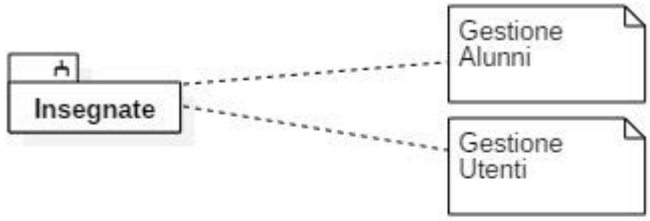


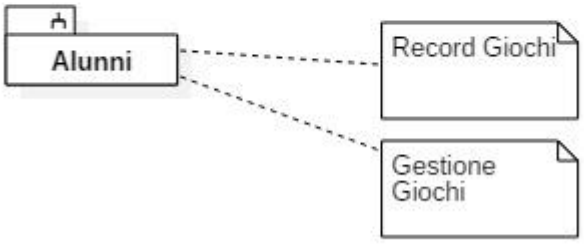

Per rendere il sistema più semplice da progettare e aumentare la manutenibilità si è deciso di decomporlo in sottosistemi, al fine di rendere la progettazione il più semplice possibile. Per decomporre il sistema in sottosistemi bisogna rendere minimo l'accoppiamento fra i sottosistemi appartenenti a layers diversi, e rendere massima la coesione delle componenti all'interno dello stesso layer.
Il sistema è stato diviso orizzontalmente in tre stati (layer), in riferimento al pattern architetturale da utilizzare. Questa separazione dell'interfaccia dalla logica applicativa consente di modificare e/o sviluppare diverse interfacce utente per la stessa logica applicativa.
L'organizzazione dei sottosistemi segue la logica three-tier. Questa permette di ottenere la separazione dell'interfaccia dalla logica applicativa consentendo di modificare e/o sviluppare diverse interfacce utente per la stessa logica applicativa.




In base alla tipologia di utenza cambia la tipologia di visualizzazione delle informazioni e delle funzionalità.

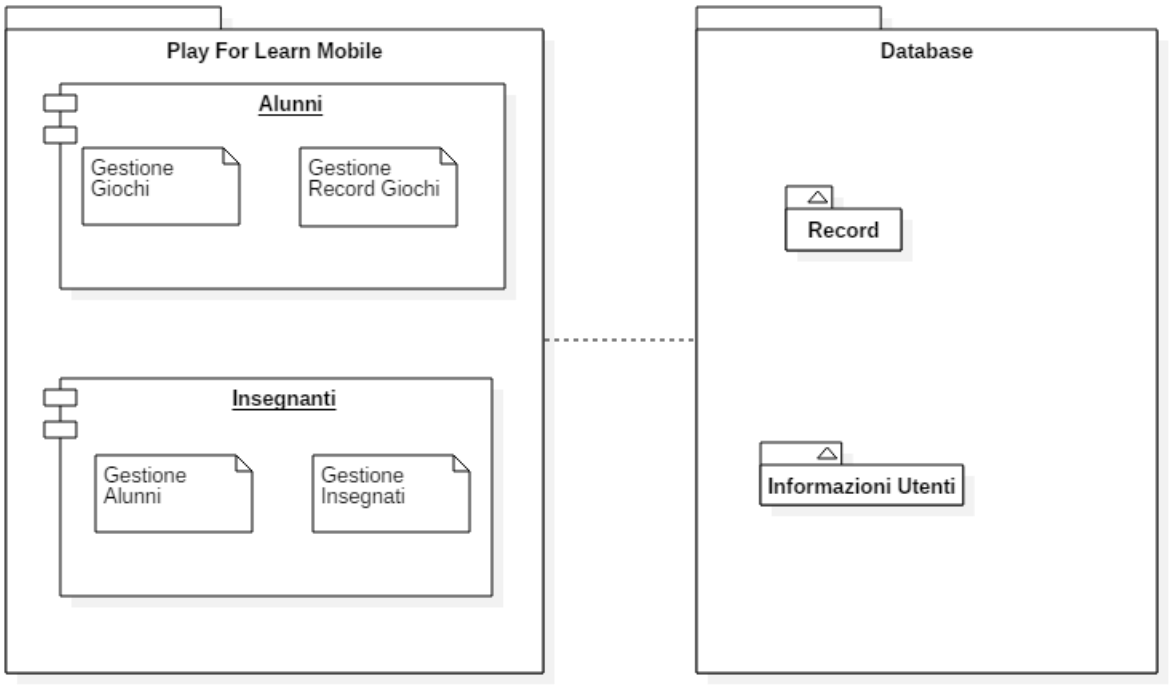
Analizzando i vari sottosistemi si ottiene il seguente diagramma:


Insegnate
Gestione
Alunni
Gestione
Utenti


Alunni
Record Giochi
Gestione
Giochi

## 3.3. Mapping HW/SW

Il sistema è connesso al Database tramite una connessione web.



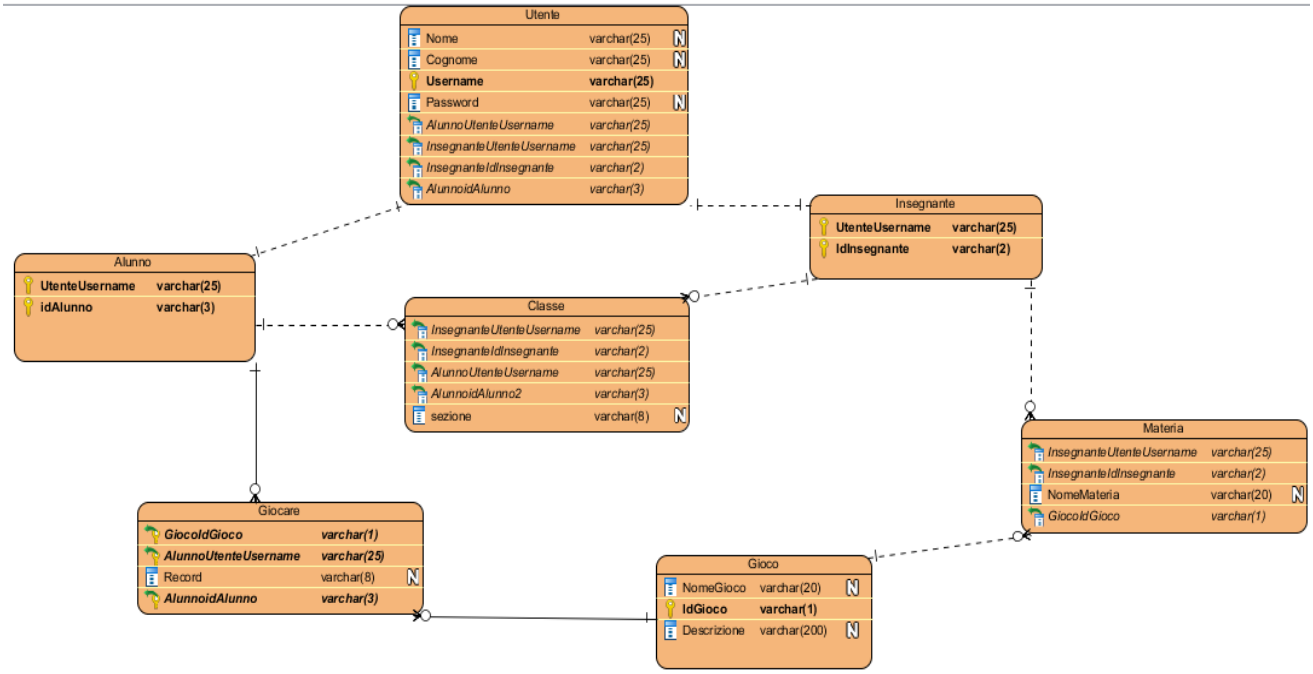

Il sottosistema Insegnanti si occupa della gestione dei dati degli utenti presenti sul database e differenzia quelli degli alunni da quelli degli insegnanti.
Il sottosistema alunni si occupa della gestione dei dati del singolo alunno e anche della gestione dei giochi, quindi tutto quello che riguarda l'avvio o l'interruzione dei giochi; inoltre salva e aggiorna i dati relativi ai record sul database.

## 3.4. Gestione dei dati persistenti

I dati relativi agli utenti vengono salvati sul database, mentre i dati relativi al singolo utente loggato vengono salvati sul dispositivo. Lo stesso criterio viene usato per i record. Mentre i giochi vengono scaricati sul dispositivo alla prima esecuzione del gioco, in base alla classe di appartenenza del bambino. I giochi scaricati dal database vengono salvati in locale.

### 3.4.1 Modello E/R

Di seguito è riportato il diagramma Entità/Relazioni del sistema PlayForLearn. Verrà presentato il diagramma E/R già ristrutturato, in modo da poter definire le entità del sistema.




## 3.4.2 Dizionario dei dati

| Entità | Descrizione | Attributi | Identificatore |
|---|---|---|---|
| **Utente** | Contiene tutte le informazioni anagrafiche dei vari utenti: alunno e insegnante. | Nome, Cognome, Username, Password. | **Username** |
| **Alunno** | Un semplice utente del sistema. | fk_Username, IdAlunno. | **IdAlunno** |
| **Insegnante** | Un particolare utente del sistema, nonchè l'amministratore del sistema. | IdInsegnante, fk_Username. | **IdInsegnante** |
| **Gioco** | Contiene tutte le informazioni riguardanti un gioco. | Nome, Classe, IdGioco. | **IdGioco** |
| **Classe** | Contiene le 5 classi | Sezione, | |

| | | | |
|---|---|---|---|
| | di appartenenza degli alunni. | fk_idAlunno, fk_IdInsegnante. | |
| **Materia** | Contiene alcune materie che è possibile scegliere dalla home principale. | NomeMateria, fk_IdInsegnante, fkIdGioco, fk_IdAlunno. | |

**Tabella: Utente**

| **Attributo** | **Tipo** | **Chiave** | **Descrizione** | **Opzionale** |
|---|---|---|---|---|
| Username | varchar(25) | Primaria | Campo username nell'autenticazione del sistema. | No |
| Nome | varchar(25) | | Nome del proprietario dell'account. | Si |
| Cognome | varchar(25) | | Cognome del proprietario dell'account. | Si |
| Password | varchar(25) | | Campo Password nell'autenticazione del sistema. | No |

**Tabella: Alunno**

| **Attributo** | **Tipo** | **Chiave** | **Descrizione** | **Opzionale** |
|---|---|---|---|---|
| IdAlunno | varchar(3) | Primaria | Identifica un alunno del | No |

| | | | | |
|---|---|---|---|---|
| | | | sistema. | |
| fk_Username | varchar(25) | esterna(Utente) | Identifica un utente del sistema ovvero l'alunno. | Si |

**Tabella: Insegnante**

| **Attributo** | **Tipo** | **Chiave** | **Descrizione** | **Opzionale** |
|---|---|---|---|---|
| fk_Username | varchar(25) | esterna(Utente) | Identifica un utente del sistema ovvero l'insegnante. | Si |
| IdInsegnante | varchar(2) | Primaria | Identifica un insegnante, nonché amministratore del sistema. | No |

**Tabella: Giocare**

| **Attributo** | **Tipo** | **Chiave** | **Descrizione** | **Opzionale** |
|---|---|---|---|---|
| IdAlunno | varchar(3) | primaria/esterna(Utente) | Identifica un utente che vuole fare un gioco. | No |
| IdGioco | varchar(1) | primaria/esterna(Gioco) | Quel particolare gioco è stato scelto da un utente. | No |
| Record | varchar(10) | | Record personale | Si |

| | | | | |
|---|---|---|---|---|
| | | | dell’utente. | |

**Tabella: Gioco**

| Attributo | Tipo | Chiave | Descrizione | Opzionale |
|---|---|---|---|---|
| IdGioco | varchar(1) | primaria | Identifica un gioco. | No |
| Nome | varchar(20) | | Assegnato un nome ad ogni gioco. | No |
| Descrizione | text | | Una piccola descrizione del gioco che è stato selezionato. | Si |

**Tabella: Classe**

| Attributo | Tipo | Chiave | Descrizione | Opzionale |
|---|---|---|---|---|
| fk_IdAlunno | varchar(3) | | Rappresenta la classe in cui l’alunno è stato assegnato. | Si |
| fk_IdInsegnante | varchar(2) | | Rappresenta la classe in cui l’insegnante insegna. | Si |
| Sezione | varchar(8) | | Rappresenta la sezione della classe. | No |

**Tabella: Materia**

| Attributo | Tipo | Chiave | Descrizione | Opzionale |
|---|---|---|---|---|
| fk_IdGioco | varchar(1) | | Rappresenta un determinato gioco in base alla materia scelta. | No |
| fk_IdInsegnante | varchar(2) | | Rappresenta la materia per cui l'insegnante insegna. | Si |
| NomeMateria | varchar(20) | | Rappresenta il nome della materia | No |

## 3.5. Controllo degli accessi e della sicurezza

Possono accedere alla applicazione solo gli utenti registrati e loggati con username e password; la registrazione di ogni utente(insegnante o alunno) deve essere confermata da un amministratore (insegnante).
Utilizzeremo la matrice degli accessi presentata nel sottoparagrafo successivo al fine di presentare in maniera schematica le operazioni consentite agli attori sulle diverse entità.

## 3.5.1 AM_1 Gestione Utente

| Oggetto/Attore | Utente Registrato |
|---|---|
| Amministratore | RegistraAccount()<br>VisualizzaAccount()<br>EliminaAccount()<br>AccettazioneAccount()<br>RifiutoAccount() |

| | |
|---|---|
| Alunno | VisualizzaAccount()<br>EliminaAccount() |

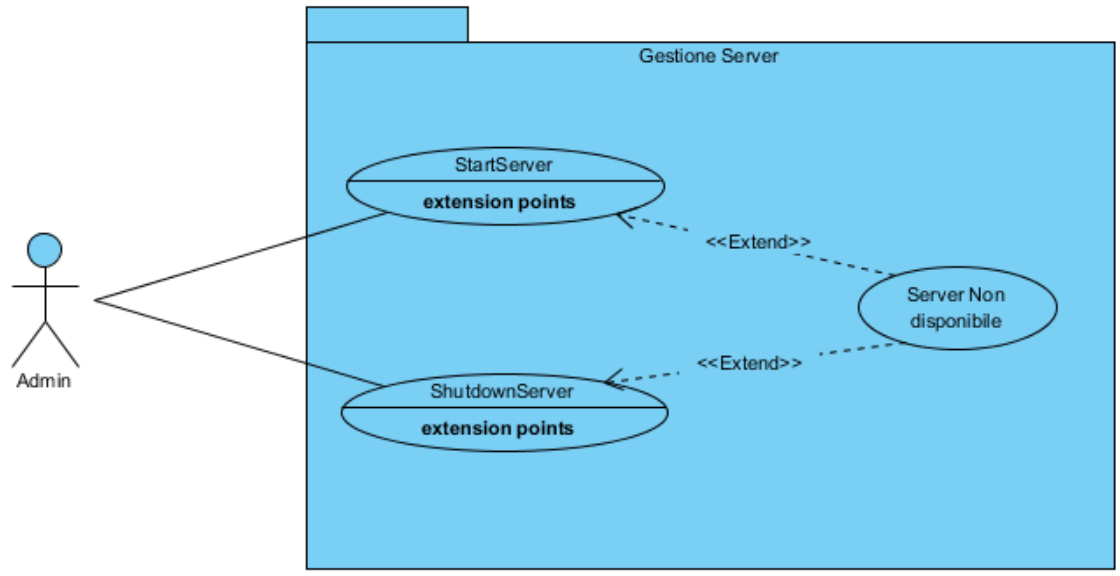
### 3.5.2 AM_1 Gestione Gioco

| **Oggetto/Attore** | **Gioco** |
|---|---|
| Alunno | AvviaGloco()<br>InterruzioneGioco() |

## 3.6. Controllo del Software globale

Il sistema PFL fornisce funzionalità che richiedono una continua interazione da parte dell'utente, per tale ragione abbiamo adottato un controllo di flusso globale del sistema di tipo event-driven. Ciò implica che gli eventi esterni a cui il sistema deve reagire possono essere rilevati in risposta ad un interrupt. Il sistema, quindi resta in attesa di un'azione da parte dell'utente, il quale interagendo con l'interfaccia grafica, scatena un evento che sarà gestito da un event handler  e da un dispatcher, che effettua materialmente la chiamata, spesso utilizzando una "coda degli eventi" che contiene l'elenco degli eventi già verificatisi, ma non ancora "processati".

## 3.7. Condizioni Boundary




| | |
|---|---|
| **ID** | UC_1.0 |

<table>
<tr><td>Nome Use Case</td><td>Start Server</td></tr>
<tr><td>Partecipanti</td><td>Admin</td></tr>
<tr><td>Condizione di Ingresso</td><td>L’amministratore avvia il server la mattina alle 8:00.</td></tr>
<tr><td>Flusso di Eventi</td><td>
<table>
<tr><td>Amministratore</td><td>Sistema</td></tr>
<tr><td>L’amministratore avvia il server attraverso la funzione per l’avvio del server</td><td></td></tr>
<tr><td></td><td>il server è online.</td></tr>
</table>
</td></tr>
<tr><td>Condizione d’uscita</td><td>Il server viene avviato con successo ed i suoi servizi vengono messi a disposizione del client remoto.</td></tr>
<tr><td>Eccezioni</td><td>ServerNonDisponibile</td></tr>
<tr><td>Requisiti di qualità</td><td></td></tr>
</table>

| | |
|---|---|
| **ID** | UC_2.0 |
| **Nome Use Case** | Shutdown Server |
| **Partecipanti** | Admin |
| **Condizione di Ingresso** | L’amministratore accede al server |

| | |
|---|---|
| **Flusso di Eventi** | <table><tr><td>Amministratore</td><td>Sistema</td></tr><tr><td>L’amministratore richiede l’arresto del server attraverso la funzione di spegnimento</td><td></td></tr><tr><td></td><td>il server è offline</td></tr></table> |
| **Condizione d’uscita** | Il server viene arrestato. |
| **Eccezioni** | ServerNonDisponibile |
| **Requisiti di qualità** | |

| | |
|---|---|
| **ID** | UC_3.0 |
| **Nome Use Case** | Server non disponibile |
| **Partecipanti** | Sistema PlayForLearn server |
| **Condizione di Ingresso** | L’amministratore accede al sistema. |
| **Flusso di Eventi** | <table><tr><td>Amministratore</td><td>Sistema</td></tr><tr><td></td><td>il sistema rileva un crash e mostra un messaggio di notifica.</td></tr></table> |
| **Condizione d’uscita** | il sistema mostra una notifica relativa al crash del sistema. |
| **Eccezioni** | ServerNonDisponibile |
| **Requisiti di qualità** | |

### 3.7.1 Start-up

Per il primo start-up del sistema “PFL” è necessario l’avvio di un web server che fornisca il servizio di un database per la gestione dei dati persistenti e

l'interpretazione ed esecuzione del codice. In seguito, tramite l'interfaccia di Login, sarà possibile autenticarsi tramite opportune credenziali(username e password) come utente a pieno accesso alle funzionalità del sistema.
Una volta effettuato l'accesso, "PFL" presenterà all'utente la home, dal quale si possono effettuare tutte le operazioni che il sistema fornisce.

### 3.7.2 Terminazione

Al momento della chiusura dell'applicazione, si ha la terminazione del sistema con un regolare Logout del sistema. Viene assicurata la consistenza dei dati, annullando eventuali operazioni che erano in esecuzione. Nel caso di terminazione di una parte del sottosistema, per esempio quella del gioco, il sistema potrà continuare a svolgere le operazioni richieste dall'utente.

### 3.7.3 Failure

Possono verificarsi diversi casi di fallimento del sistema:

1. Nel caso in cui si verifichi un'interruzione inaspettata dell'alimentazione, non sono previsti metodi che ripristinano lo stato del sistema a prima dello spegnimento inaspettato.
2. Nel caso di guasti dovuti al sovraccarico del database con successivo fallimento dello stesso, è prevista una procedura il salvataggio periodico dei dati sotto forma di codice SQL per la successiva rigenerazione del DB.
3. Nel caso in cui si verifica un bug nel client, il sistema potrà continuare a funzionare regolarmente.
4. Nel caso in cui ci sia un crash dovuto a un bug nel database server, il sistema risulta inutilizzabile; in quanto non è possibile accedere ai dati.
5. In caso di mancanza di rete per la connessione al database i dati verranno temporaneamente salvati solo in locale, quando si riavrà la connessione i dati verranno aggiornati, se necessario.

# 4. Glossario

- **Greenfield Engineering**: Tipologia di sviluppo che comincia da zero, non esiste alcun sistema a priori e i requisiti sono ottenuti dall'utente finale e dal cliente. Nasce perciò, a partire dai bisogni dell'utente.
- **MVC**: Model/View/Controller è l'architettura del software implementato in questo sistema.
- **Throughput**: è la quantità di dati trasmessi in una unità di tempo, mentre la capacità dipende esclusivamente da quanta informazione è disponibile sul canale nella trasmissione.
- **Tolleranza ai Fault:** capacità del sistema di resistere in casi di errore.we
- **event-driven**: tecnica di programmazione a eventi.
- **event handler**: gestore degli eventi dell' event-driven.
- **dispatcher**: è un modulo del sistema operativo che passa effettivamente il controllo della CPU ai processi scelti dallo scheduler a breve termine.
- **SQL**: linguaggio di programmazione nell'ambito dei database.