

SDD System Design Document

DInfBoard

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 2.0 |
| Data | 16/12/2018 |
| Destinatario | Filomena Ferrucci – Carmine Gravino |
| Presentato Da | Daniele Cerardi – Simone Pirozzi – Antonio Junior Citarella – Antonio Vitale – Francesco Di Lascio |
| Approvato Da |  |

|  |  |
| --- | --- |
| Nome | Matricola |
| Daniele Cerardi | 0512104555 |
| Antonio Junior Citarella | 0512104543 |
| Francesco Di Lascio | 0512103219 |
| Simone Pirozzi | 0512104723 |
| Antonio Vitale | 0512104819 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Cambiamenti** | **Autori** |
| 27/11/2018 | 0.1 | Stesura del template del documento. | Antonio Junior Citarella |
| 28/11/2018 | 0.2 | Introduzione al documento, descrizione degli obiettivi del sistema. | Daniele Cerardi |
| 29/11/2018 | 0.3 | Introduzione agli obiettivi di design e all’architettura del sistema corrente. | Daniele Cerardi  Antonio Junior Citarella Antonio Vitale |
| 03/12/2018 | 0.4 | Introduzione al sistema proposto e all’insieme di strategie scelte. | Antonio Citarella  Antonio Vitale |
| 04/12/2018 | 0.5 | Revisione complessiva; aggiornato il glossario. | Daniele Cerardi  Antonio Junior Citarella |
| 08/12/2018 | 1.1 | Aggiornamento dello stile architetturale e del diagramma della divisione in sottosistemi. | Antonio Junior Citarella |
| 11/12/2018 | 1.2 | Aggiornamento del punto 3.4 | Antonio Junior Citarella |
| 16/12/2018 | 2.0 | Revisione complessiva. | Daniele Cerardi |

Sommario

1. [1. Introduzione 3](#_Toc436218379)

[1.1 Obiettivi del sistema](#_Toc436218380)

[1.2 Design Goals](#_Toc436218381)

[1.3 Definizioni, acronimi e abbreviazioni](#_Toc436218382)

[1.4 Riferimenti](#_Toc436218383)

[1.5 Panoramica](#_Toc436218384)

1. [2. Architettura del Sistema corrente 6](#_Toc436218385)
2. [3. Architettura del Sistema proposto 6](#_Toc436218386)

[3.1 Panoramica](#_Toc436218387)

[3.2 Decomposizione sottosistemi](#_Toc436218388)

[3.3 Mapping hardware/software](#_Toc436218389)

[3.4 Gestione dati persistenti](#_Toc436218390)

[3.5 Controllo degli accessi e sicurezza](#_Toc436218391)

[3.6 Controllo globale del software](#_Toc436218392)

[3.7 Condizione limite](#_Toc436218393)

1. [4. Servizi dei Sottosistemi 14](#_Toc436218394)
2. [Glossario 15](#_Toc436218395)

# 1. Introduzione

## 1.1 Obiettivi del sistema

Il sistema da realizzare ha come scopo la creazione di una piattaforma di comunicazione tra studenti che permetta loro di organizzare attività extra-universitarie e non.

La struttura del sistema dal punto di vista funzionale simula una bacheca in cui è possibile inserire annunci; essendo ogni annuncio pubblicato da uno studente, ne rappresenta le necessità in relazione al contesto didattico dell’informatica.

Il nostro obiettivo è facilitare l’interazione tra studenti del dipartimento di informatica, permettendo loro di valutare il proprio lavoro in relazione a quello degli altri.

Il sistema rappresenta uno strumento di supporto, veicolato da persone scelte che controllano il frequente inserimento di annunci e i corrispettivi contenuti.

Un utente registrato come studente sarà in grado di interagire con una serie di funzionalità che coinvolgono la pubblicazione, l’interazione e il mantenimento di annunci; ognuno di essi avrà a disposizione un profilo caratterizzato dalle proprie informazioni e da una serie di preferenze per la scelta degli argomenti per gli annunci.

Un amministratore godrà degli stessi diritti di uno studente e in più avrà a disposizione un’interfaccia per la gestione e il controllo della sicurezza sulla bacheca.

In sintesi, il sistema può essere distribuito in tre medio-grandi porzioni di funzionalità: gestione degli annunci, del profilo e dell’amministrazione.

Per la gestione degli annunci, sia per gli studenti che per gli amministratori, saranno messe a disposizione le funzionalità più ovvie per la loro manipolazione: partecipazione, inserimento, eliminazione, archiviazione, modifica.

Per la gestione del profilo, invece, gli stessi attori potranno modificare il loro profilo in relazione alle proprie credenziali e preferenze; in più, sarà presente un’interfaccia per visualizzare tutti gli annunci a cui si è iscritta la propria partecipazione o che si ha organizzato.

Per la gestione dell’amministrazione, infine, ogni attore di competenza potrà eliminare degli annunci non appropriati al contesto della piattaforma, inserire o rimuovere nel sistema studenti con i suoi stessi diritti, sospendere utenti o inserire nuove macro-preferenze, le quali potrebbero condizionare le scelte di pubblicazione di uno studente.

## 1.2 Design Goals

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Priorità | ID Design Goal | Descrizione Design Goal | Categoria | Origine | Trade-off |
| 1 | DG\_1 | **Tempo di risposta:**   * Il tempo di risposta per la visualizzazione di un annuncio, del profilo e della bacheca è tra 0.5 e 1 secondo/i. * il tempo di risposta per la modifica, eliminazione e archiviazione di un annuncio è tra 1 e 3 secondo/i. * Il tempo di risposta per la specifica o l’eliminazione di una partecipazione ad un annuncio è tra 0.5 e 2 secondi. | Performance | Requisito Non Funzionale 3.3.3 | Velocità – Memoria: Il tempo di risposta sarà relativo alla dimensione dei dati immagazzinati sulla piattaforma. |
| 2 | DG\_2 | **Memoria:** La dimensione complessiva del sistema dipende dalla memoria utilizzata per il mantenimento della base di dati. | Performance | Requisito  Non Funzionale 3.3.3 |  |
| 3 | DG\_3 | **Security:** L’accesso al sistema è garantito da un meccanismo di autenticazione attraverso un nome utente ed una password.  La sicurezza dei dati persistenti è garantita dalla generazione di file di ripristino, utilizzati qualora dovessero verificarsi guasti alla base di dati o al sistema. | Dependability | Requisito Non Funzionale 3.3.2 |  |
| 4 | DG\_4 | **Usabilità:** Questo sistema sarà molto semplice all’uso senza la necessità di consultare la documentazione associata. | End User | Requisito Non Funzionale 3.3.1 | Usabilità – Tempo di consegna: Il tempo dedicato alla cura del sistema per facilitarne l’uso andrà ad impattare sul tempo di consegna dello stesso. |
| 5 | DG\_5 | **Tolleranza agli errori:** Il sistema dovrà fornire una gestione accurata degli errori e dovrà prevedere il maggior numero possibile di casi di errore e di warning che la piattaforma potrebbe generare. | Dependability | Requisito Non Funzionale 3.3.2 / Criteri di Accettazione | Tolleranza agli errori – Tempo di consegna: Il tempo impiegato al controllo degli errori incrementa il tempo necessario per la consegna del sistema. |
| 6 | DG\_6 | **Costo di manutenzione:** Il sistema dovrà prevedere una buona manutenibilità; dovranno,dunque, essere previsti i costi eventuali da affrontare. | Cost | Criteri di Accettazione |  |
| 7 | DG\_7 | **Robustezza:** Dovrà essere stata fatta un’attenta fase di testing andando ad effettuare un branch coverage di almeno il 75% dei casi di test.  Eventuali input non validi immessi dall’utente saranno opportunamente segnalati attraverso messaggi di errore. | Dependability | Criteri di Accettazione |  |
| 8 | DG\_8 | **Costo di sviluppo:**  Il costo dello sviluppo della piattaforma è valutato in termini di tempo dedicato alla produzione dell’intero prodotto software e non dovrebbe superare le 300 ore. | Cost | Dominio di Applicazione | Tempo di consegna – Gestione del personale: Il tempo dedicato allo sviluppo dal team e il numero di persone che lo compongono andranno ad impattare sul tempo di consegna del sistema. |
| 9 | DG\_9 | **Costo di installazione:** L’utente, al fine di utilizzare la piattaforma al suo pieno potenziale, dovrà provvedere ad installare la versione più aggiornata del suo browser. | Cost | Requisito Non Funzionale  3.3.7 |  |

|  |
| --- |
| Daniele Cerardi Antonio Junior Citarella |

## 1.3 Definizioni, acronimi e abbreviazioni

* DG = Design Goal, obiettivo di design.
* SDD = System Design Document.
* Event Driven = Programmazione ad eventi.
* GUI = Grafical User Interface.
* Flat File =Vengono chiamati “Flat file” tutti quei file di dati contenenti tracciati record privi di relazioni strutturali.
* MVC= In ingegneria del software e nell'ambito dei sistemi informatici, l'espressione architettura MVC : Model-View-Controller, talvolta tradotto in [italiano](https://it.wikipedia.org/wiki/Lingua_italiana) con la dicitura modello-vista-controllo, è un [pattern architetturale](https://it.wikipedia.org/wiki/Design_pattern) molto diffuso nello sviluppo di sistemi [software](https://it.wikipedia.org/wiki/Software), in particolare nell'ambito della [programmazione orientata agli oggetti](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_oggetti), in grado di separare la [logica di presentazione](https://it.wikipedia.org/wiki/Livello_di_presentazione#Altro) dei dati dalla [logica di business](https://it.wikipedia.org/wiki/Logica_di_business).
* DBMS = DataBase Management System.
* MySQL = MySQL o Oracle MySQL è un Relational Database Management System (RDBMS) composto da un client a riga di comando e un server.
* Greenfield Engineering = Tipo di Requirements Elicitation; non esiste una base da cui partire per lo sviluppo del prodotto software.

## Riferimenti

* Bernd Bruegge & Allen H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, (3rd edition), Prentice-Hall, 2003.
* https://it.wikipedia.org/wiki

## Panoramica

Segue nel documento la seconda sezione dedicata alla descrizione dell’architettura dei sistemi esistenti che offrono la stessa tipologia di servizi. Successivamente sarà identificata la descrizione dell’architettura del sistema proposto, descrivendo i sottosistemi individuati, il tipo di architettura scelto e i livelli individuati. Indicheremo, inoltre, in che modo tratteremo i dati persistenti della piattaforma, come gestiremo il flusso globale e la sicurezza sugli accessi del sistema.

# 2. Architettura del Sistema corrente

Non esiste attualmente un sistema software nel dipartimento d’informatica che aiuti gli studenti ad organizzarsi per attività didattiche extracurriculari, tra cui semplici gruppi di studio o per formare progetti. Eventuali attività vengono segnalate mediante avvisi nelle bacheche situate nei corridoi del dipartimento.

Si tratta, dunque, di un sistema che rientra nel campo della Greenfield Engineering, dove lo sviluppo comincia da zero, non essendoci alcun sistema da sostituire o migliorare. I requisiti sono ottenuti dall’utente finale e dal cliente, facendo così che la piattaforma nasca per soddisfare i loro bisogni.

# 3. Architettura del Sistema proposto

## 3.1 Panoramica

Il sistema proposto sarà un’applicazione web. L’obiettivo che ci poniamo è di fornire uno strumento per la pubblicazione di annunci contestualizzati al dipartimento di informatica dell’università di Salerno. Questi ultimi saranno divisi in due macrocategorie: quelli per la formazione di gruppi di studio o di un progetto e quelli per la realizzazione di corsi di insegnamento relativi al mondo dell’informatica.

Il sistema sarà diviso sostanzialmente in tre sottosistemi principali: la gestione degli annunci, la gestione dell’utenza e la gestione amministrativa della piattaforma.

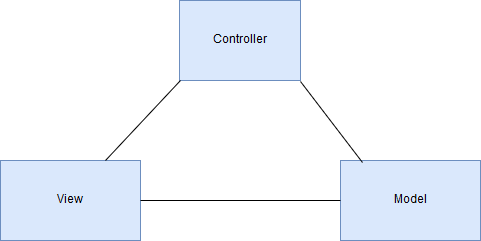
Non abbiamo la possibilità di automatizzare le operazioni di amministrazione e quelle che riguardano il mantenimento di una piattaforma pulita da spam e contenuti inopportuni, sia per motivi di sicurezza, sia per motivi di budget; sarà la figura dell’amministratore ad avere accesso ad un’area della piattaforma appropriata per effettuare operazioni sui dati persistenti desiderati. Gli studenti avranno la possibilità di inserire dati all’interno del sistema e di visualizzarli, effettuando operazioni di inserimento e interrogazione al database, mentre gli utenti ospite avranno solo la possibilità di effettuare operazioni di interrogazione. Il sistema si modellerà, come vedremo dettagliatamente in seguito, su un’architettura di tipo Model-View-Controller (MVC).

## 3.2 Decomposizione in sottosistemi

**3.2.1 Stile di Architettura**

L’architettura utilizzata dal nostro sistema è di tipo MVC(Model-View-Controller) aperta, in cui ogni livello potrà inviare e ricevere informazioni dal livello adiacente:

* La View è il livello con cui l’utente si interfaccia e in cui vengono presentate tutte le funzionalità proposte. La View mostra all’utente i dati presenti nel livello Model.
* Il Controller è il livello che gestisce la logica applicativa e di business. All’interno di esso avverrà l’implementazione effettiva delle operazioni che l’utente potrà effettuare.
* Il Model è il livello che si occupa di del mantenimento dei dati persistenti del sistema e dello scambio di quest’ultimi tra i vari sottosistemi.

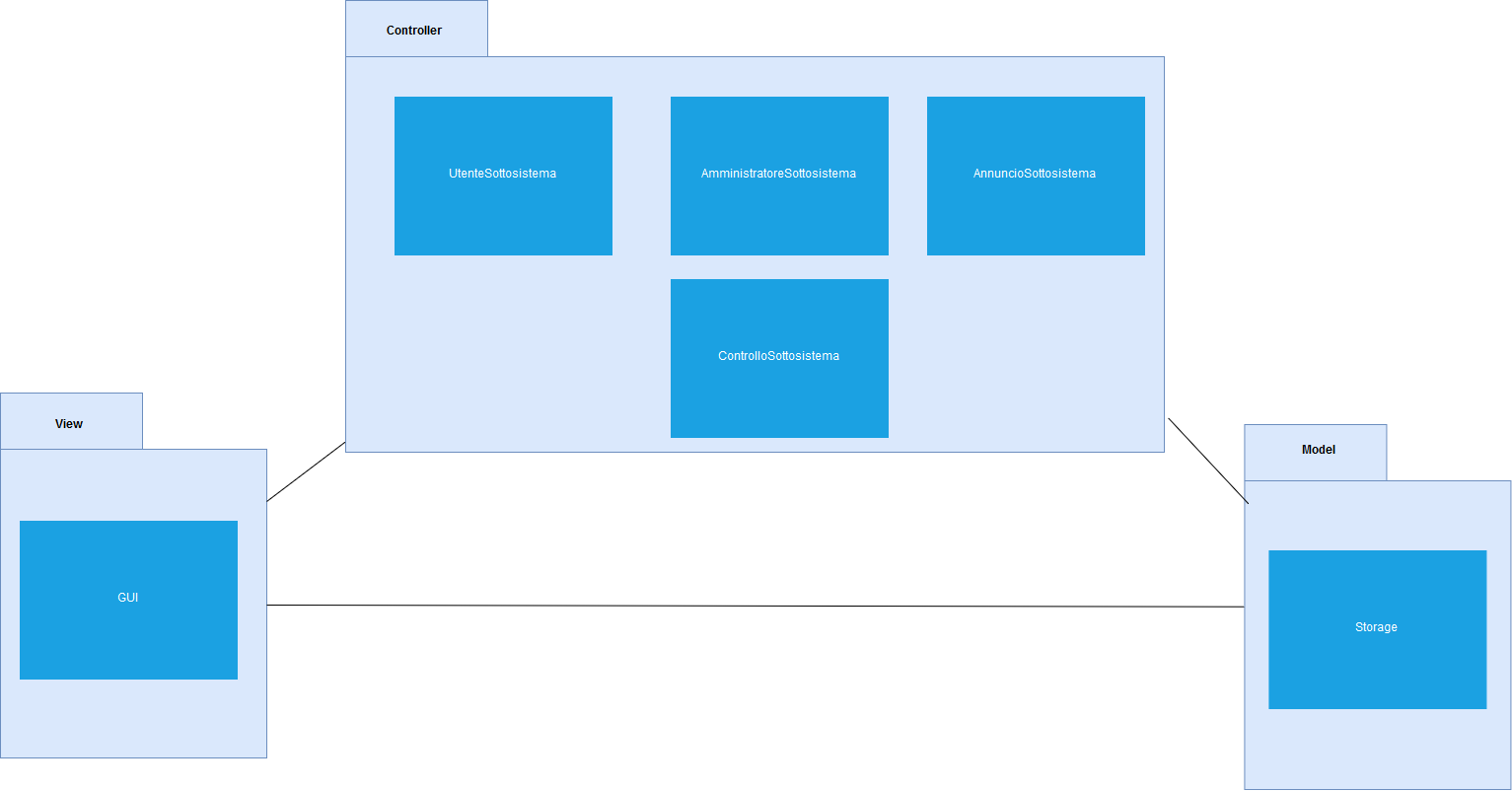


|  |
| --- |
| Antonio Junior Citarella |

**3.2.2 Decomposizione in sottosistemi**

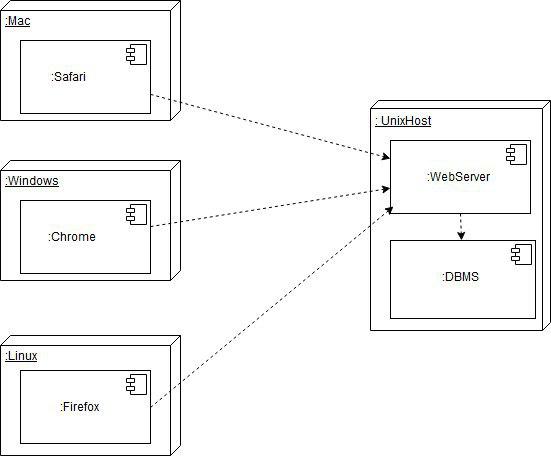
Al fine di ottenere un accoppiamento basso e un’elevata coesione, abbiamo deciso di dividere i sottosistemi per categoria di funzionalità:

* GUI: Sottosistema presente nel livello di presentazione che si occupa di fornire un’interfaccia grafica all’utente.
* UtenteSottosistema: Sottosistema dedicato alle operazioni che riguardano la gestione del profilo di un utente.
* AnnuncioSottosistema: Sottosistema dedicato alle operazioni effettuabili su un annuncio della piattaforma.
* AmministratoreSottosistema: Sottosistema dedicato alle operazioni effettuabili dagli amministratori per la gestione della piattaforma.
* ControlloSottosistema: Sottosistema che effettua operazioni di comunicazione tra il sottosistema di amministrazione e i sottosistemi dell’utente e dell’annuncio al fine di non coinvolgere direttamente il livello dei dati persistenti per ogni operazione che ne richiede la presenza.
* Storage: Sottosistema d’interfacciamento con il DBMS per la gestione delle operazioni sui dati persistenti.



|  |
| --- |
| Antonio Junior Citarella |

## 3.2.3 Diagramma di Deployment



|  |
| --- |
| Antonio Junior Citarella |

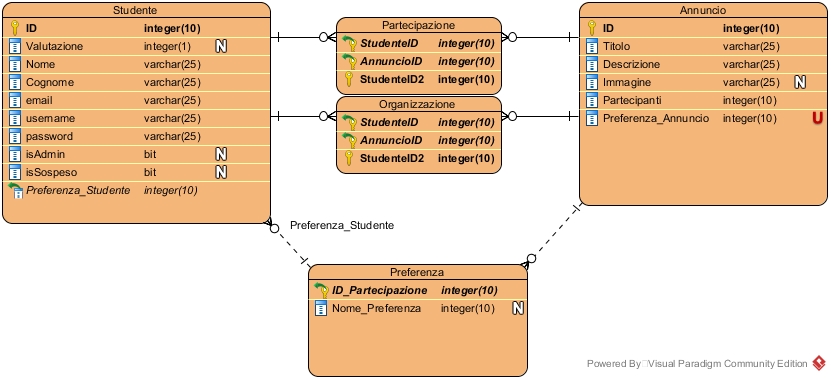
## 3.3 Mapping hardware/software

## Il sistema verrà installato su un Web Server che utilizzerà il DBMS MySQL per gestire il database. In questa prima versione del sistema, il tutto verrà gestito su una sola macchina che effettuerà sia le operazioni di interfacciamento con il client che le operazioni dal lato del server. In una versione successiva potrebbe essere effettuata una divisione tra queste operazioni, inserendole appropriatamente su macchine differenti. Per quanto riguarda la gestione del Web Server utilizzeremo Apache Tomcat che ci permetterà di creare un server in locale e di testare l’applicazione che inseriremo su di esso.

## 3.4 Gestione dati persistenti

La nostra strategia di gestione dei dati persistenti prevede l’utilizzo di due diverse opzioni.   
Per quanto riguarda gli attori e gli annunci verrà utilizzato un database relazionale, che fornisce un’astrazione dei dati ad un livello più alto. Sono state, dunque, create tre tabelle: una per gli annunci, una per le preferenze ed una per l’utente “studente”.  
In tutte e tre è stato inserito un ID per specificare l’unicità del dato. Inoltre, in quella degli studenti, sono state inserite due variabili booleane per i controlli a tempo d’esecuzione per un utente sospeso o amministratore. Per i file di log e di errore è stato previsto l’utilizzo dei flat file, che hanno un livello di astrazione relativamente basso e consentono all'applicazione di essere ottimizzata in termini di dimensioni e velocità.  
Dal punto di vista delle priorità, il database relazionale ha una rilevanza maggiore rispetto ai flat file.

**Schema E-R**

****

|  |
| --- |
| Antonio Junior Citarella |

**Studente**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Tipo** | **Null** | **Key** |
| Nome | Varchar(15) | Not Null |  |
| Cognome | Varchar(15) | Not Null |  |
| Email | Varchar(30) | Not Null |  |
| Username | Varchar(15) | Not Null |  |
| Password | Varchar(30) | Not Null |  |
| ID | Integer(5) | Not Null | Primary Key |
| Sospeso | Boolean | Nullable |  |
| Admin | Boolean | Nullable |  |
| Valutazione | Integer(1) | Nullable |  |
| Preferenza\_Studente | Integer(5) | Not Null | Foreign Key(Preferenza) |

**Annuncio**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Tipo** | **Null** | **Key** |
| Titolo | Varchar(35) | Not Null |  |
| Descrizione | Varchar(500) | Not Null |  |
| Immagine | Varchar(35) | Nullable |  |
| Preferenza\_Annuncio | Varchar(15) | Not Null | Foreign Key(Preferenza) |
| Partecipanti | Integer(2) | Not Null |  |
| ID | Integer(5) | Not Null | Primary Key |
| ID Studente | Integer(5) | Not Null | Foreign Key  (Studente) |

**Preferenza**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Tipo** | **Null** | **Key** |
| Nome Preferenza | Varchar(15) | Not Null |  |
| ID Preferenza | Integer(5) | Not Null | Primary key |

**Partecipazione**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Tipo** | **Null** | **Key** |
| ID\_Studente | integer (15) | Not Null | Primary Key |
| ID\_Annuncio | Integer(5) | Not Null | Primary key |

**Organizzazione**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Tipo** | **Null** | **Key** |
| ID\_Studente | integer (15) | Not Null | Primary Key / Foreign Key ( Studente) |
| ID\_Annuncio | Integer(5) | Not Null | Primary key / Foreign Key ( Studente ) |

**Studente**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nome** | **Cognome** | **Email** | **Username** | **Password** | **ID** | **Sospeso** | **Admin** | **Valuta**  **zione** | **Preferenza \_Studente** |
|  |  |  |  |  |  |  |  |  |  |

**Annuncio**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Titolo** | **Descrizione** | **Immagine** | **Partecipanti** | **ID** | **Preferenza\_Annuncio** |
|  |  |  |  |  |  |

**Preferenza**

|  |  |
| --- | --- |
| **Nome Preferenza** | **Id Preferenza** |
|  |  |

#### **Partecipazione**

|  |  |
| --- | --- |
| **ID\_Studente** | **ID\_Annuncio** |
|  |  |

|  |  |
| --- | --- |
| **ID\_Studente** | **ID\_Annuncio** |
|  |  |

#### **Organizzazione**

|  |
| --- |
| Antonio Junior Citarella Antonio Vitale |

## 3.5 Controllo degli accessi e sicurezza Il controllo degli accessi è gestito mediante l’utilizzo di un nome utente e una password, che verranno richieste ad ogni singolo accesso. La sicurezza sui dati sensibili degli utenti “studente” è garantita dall’accesso controllato, in quanto solo loro possono accedere ai dati riservati del proprio profilo. Le operazioni possibili sono:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sottosistema**  **Attore** | **Gestione** | | |
| **UtenteSottosistema** | **AnnuncioSottosistema** | **AmministratoreSottosistema** |
| **Utente Ospite** | •Registrazione | •Ricerca Annuncio |  |
| **Studente** | •Login •Modifica Profilo Utente •Valutazione Organizzatore •Segnalare Utente | •Ricerca Annuncio  •Inserire Annuncio •Modificare Annuncio •Eliminare Annuncio •Eliminare Partecipazione Corso •Partecipare Annuncio |  |
| **Amministratore** | •Login |  | •Gestione Preferenze  •Inserire Amministratore •Eliminare Amministratore •Eliminare Annuncio (Amministratore) •Sospendere Utente |

|  |
| --- |
| Antonio Vitale |

## 3.6 Controllo flusso globale del sistema

Il meccanismo per il controllo del flusso che abbiamo deciso di implementare è l’Event Driven, basato sulla gestione di eventi. L’utente interagirà con gli elementi del nostro sistema e genererà eventi che verranno controllati e che daranno luogo all’esecuzione delle corrispettive operazioni.

## 

## 3.7 Condizione limite

**3.7.1 Start-Up**

**Scenario**

|  |  |  |
| --- | --- | --- |
| Nome SC\_20 | Avvio del Sistema | |
| Istanza attori partecipanti | Antonio: Amministratore. | |
| Descrizione Scenario | Attore | Sistema |
|  | Antonio ha la necessità rendere la piattaforma accessibile all’utenza; effettua,dunque, le operazioni necessarie attraverso un terminale,affinchè il sistema diventi operativo. Procede, inoltre, ad inserire i comandi per avviare il DBMS. |  |
|  |  | Il sistema si avvia, rendendo visibile all’utenza la piattaforma e i dati persistenti contenuti all’interno del DBMS. |

|  |
| --- |
| Antonio Junior Citarella Revisione: Daniele Cerardi |

**Use Case**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Identificativo  UC\_19\_AvvioSistema | | | *Avvio del sistema* | *Data* | | *04/12/2018* |
| *Vers.* | *00.00.001* | |
| *Autore* | *Daniele Cerardi* | |
| Descrizione | | | *Lo Use Case fornisce la descrizione dell’avvio del Sistema.* | | | |
| Attore Principale | | | **Amministratore**  L’amministratore avvia il Sistema. | | | |
| Attori secondari | | | NA | | | |
| Entry Condition | | | NA | | | |
| Exit condition  On success | | | L’amministratore ha avviato il sistema con successo. | | | |
| Exit condition  On failure | | | L’amministratore non ha avviato il sistema. | | | |
| Rilevanza/User Priority | | | Alta | | | |
| Frequenza stimata | | | 0-1 usi/giorno | | | |
| Extension point | | | NA | | | |
| Generalization of | | | NA | | | |
| Flusso di Eventi Principale/Main Scenario | | | | | | |
| 1 | Amministratore: | Avvia il Sistema attraverso I comandi necessari da un terminale; avvia, inoltre, il DBMS. | | | | |
| 2 | Sistema: | Rende disponibile le operazioni e I dati contenuti sul DBMS. | | | | |

Per il primo avvio del sistema “DInfBoard” è necessario avviare il Web Server in cui il sistema è presente, inizializzando il database MySQL utilizzato per la gestione dei dati persistenti come le credenziali degli utenti iscritti alla piattaforma. Lo Studente sarà poi in grado al suo primo accesso di effettuare accesso o registrazione tramite credenziali memorizzate nel database. Il sistema presenterà allo Studente tutte le possibili operazioni che è possibile poter compiere sulla piattaforma.

**3.7.2 Start-Up (in seguito ad un fallimento)**

A seguito di un fallimento il sistema dovrà essere riavviato, riportando la piattaforma all’ultimo stato funzionante. Per fare ciò è necessario mantenere invariati i dati presenti sul database MySQL. Questo verrà garantito effettuando un dump periodico dei dati e andando a salvare su una struttura ausiliaria il contenuto del database e la struttura di quest’ultimo.

**3.7.3 Terminazione**

**Scenario**

|  |  |  |
| --- | --- | --- |
| Nome SC\_21 | Terminazione del Sistema | |
| Istanza attori partecipanti | Daniele: Amministratore. | |
| Descrizione Scenario | Attore | Sistema |
|  | Daniele deve rendere la piattaforma DInfBoard inaccessibile all’utenza per effettuare delle operazioni di manutenzione. Provvede, dunque, a digitare da un terminale i comandi per la terminazione del Web Server e del DBMS. |  |
|  |  | Il sistema termina la propria esecuzione, rendendo inaccessibile la piattaforma all’utenza. |

|  |
| --- |
| Antonio Junior Citarella Revisione: Daniele Cerardi |

**Use Case**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Identificativo  UC\_20 TerminazioneSistema | | | *Terminazione del sistema* | *Data* | | *04/12/2018* |
| *Vers.* | *0.00.001* | |
| *Autore* | *Daniele Cerardi* | |
| Descrizione | | | *Lo Use Case fornisce la descrizione della terminazione del sistema.* | | | |
| Attore Principale | | | **Amministratore**  L’amministratore spegne il sistema. | | | |
| Attori secondari | | | NA | | | |
| Entry Condition | | | NA | | | |
| Exit condition  On success | | | L’amministratore termina con successo l’esecuzione del sistema. | | | |
| Exit condition  On failure | | | L’amministratore non termina l’esecuzione del sistema. | | | |
| Rilevanza/User Priority | | | Media | | | |
| Frequenza stimata | | | 0-1 usi/giorno | | | |
| Extension point | | | NA | | | |
| Generalization of | | | NA | | | |
| Flusso di Eventi Principale/Main Scenario | | | | | | |
| 1 | Amministratore: | Termina il Web Server e il DBMS associato attraverso una serie di comandi da un terminale. | | | | |
| 2 | Sistema: | Termina la propria esecuzione e quella del DBMS associato. | | | | |

La terminazione dell’utilizzo della piattaforma avviene nel momento in cui un utente effettua il logout dalla piattaforma; il sistema, dunque, dovrà occuparsi di mantenere i dati consistenti per il prossimo accesso da parte dell’utente.

**3.7.4 Fallimento**

Le situazioni che possono portare ad una situazione di fallimento sono diverse:

* Per fallimenti del database dovuti a sovraccarichi o cambiamenti di utenti malevoli sarà previsto un salvataggio preventivo dei dati che permetteranno il ripristino della piattaforma ad uno stato funzionante.
* Un ulteriore tipo di fallimento potrebbe essere rappresentato da un crash del web server, causa di inaccessibilità ai servizi della piattaforma. Non sono previste misure di contromisura, escludendo l’effettuare un riavvio del sistema il prima possibile.
* Un ultimo tipo di fallimento potrebbe essere dovuto ad un guasto hardware, per il quale non sarà prevista alcuna misura di correzione.

# 4. Servizi dei Sottosistemi

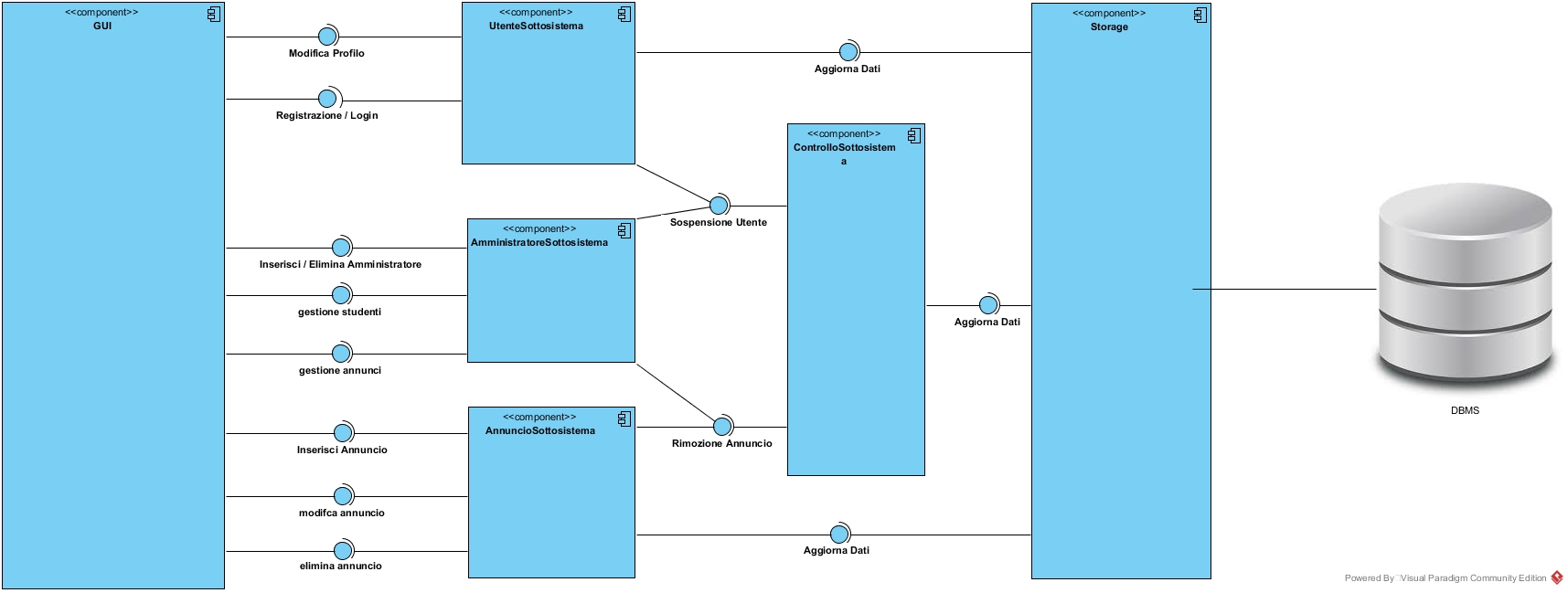


Diagramma delle component che esplicitano I servizi di ogni sottosistema e le interazioni tra gli stessi.

|  |  |
| --- | --- |
| Nome Operazione | Descrizione |
| Modifica Profilo | Operazioni di modifica delle credenziali presenti nel profilo. |
| Registrazione / Login | Operazione che permette all’utente di accedere alla piattaforma registrandosi, o se già registrato, effettuando il login. |
| Inserisci / Elimina Amministratore | Operazione per l’inserimento e l’eliminazione di un amministratore. |
| Gestione Annunci | Operazioni effettuabili dall’amministratore sugli annunci. |
| Gestione Studenti | Operazioni effettuabili dall’amministratore sugli studenti della piattaforma |
| Inserisci Annuncio | Operazione di inserimento dell’annuncio da parte di uno studente. |
| Modifica Annuncio | Operazione di modifica dell’annuncio da parte di uno studente. |
| Elimina Annuncio | Operazione di cancellazione dell’annuncio da parte di uno studente. |
| Sospensione Utente | Operazione di sospensione dell’utente da parte dell’amministratore. |
| Rimozione Annuncio | Operazione di rimozione di un annuncio da parte dell’amministratore. |
| Aggiorna Dati | L’Operazione che effettua aggiornamenti sui dati persistenti presenti nel DBMS |

|  |
| --- |
| Antonio Junior Citarella |

# Glossario

* **Warning**: Potenziali sorgenti di errori a run-time o di

comportamenti non voluti

* **Branch**: Un punto che rappresenta un potenziale cambiamento nel flusso di un programma,
* **Spam**: Messaggio non richiesto, inviato a un numero molto elevato di utenti di Internet.
* **Budget**: Potenziale economico a disposizione dal team di sviluppo.
* **Dump:** elemento di un database contenente un riepilogo della struttura delle [tabelle](https://it.wikipedia.org/wiki/Data_Definition_Language#Tabelle) del database medesimo e/o i relativi dati, ed è normalmente nella forma di una lista di dichiarazioni [SQL](https://it.wikipedia.org/wiki/SQL)