SDD

**1.1 Scopo del sistema**

La piattaforma LetsMeet di pone l’obiettivo di poter dare l’opportunità, a qualsiasi tipologia di utente interessato alla creazione e monitoraggio di eventi, di poter creare eventi in real-time sul territorio attraverso una mappa interattiva.

Usualmente, chi interagisce con sistemi che offrono servizi di creazione e condivisione degli stessi eventi, mettono in secondo luogo l’interazione con la mappa, la quale funge solo come eventuale scorciatoia per accedere ad altre applicazioni, come Google Maps, per la visualizzazione del luogo d’incontro.

LetsMeet vuole concentrare l’attenzione proprio sull’utilizzo di una mappa dando anche la possibilità agli utenti di visionare in tempo reale anche il numero dei partecipanti effettivi agli eventi, cosa che effettivamente manca ad altre tipo di piattaforme in circolazione che si prefiggono l’obiettivo di creare eventi.

**1.2 Obiettivi di Design**

La piattaforma LetsMeet deve poter essere il più efficiente ed intuitiva possibile attraverso agevoli interfacce. Tale efficienza sarà costruita attraverso rapidi tempi di risposta ad ogni genere di input ma anche differenti politiche di tolleranza all’errore.

Per ottenere gli obiettivi finali vanno seguiti dei criteri di progettazione tenendo presente:  
Performance, Affidabilità, Costi, Manutenzione e Utente Finale.

**1.2.1 Criteri di Performance**

**Tempo di Risposta:**

LetsMeet deve essere reattivo per tutte le operazioni più immediate come la creazione di un evento o la verifica della partecipazione ad un evento.

Per operazioni massive, come il caricamento di più eventi sulla mappa, il sistema deve garantire tempi di risposta ragionevoli nell’ordine delle decine di secondi, ma, essendo un sistema web, molto dipenderà dalla qualità della connessione e dalla congestione della rete sul sistema online.

**Throughput:**

I picchi di carico, fino a circa 500 (cinquecento) utenti simultaneamente collegati, devono essere gestiti dal sistema senza rallentamenti, garantendo fluidità e una latenza molto bassa.

Il sistema deve garantire consistenza in tutte le operazioni che includono chiamate al database, specialmente nei momenti di maggiore carico.

**Memoria:**

Il sistema utilizza un database relazionale per memorizzare tutti i dati e la mole dei dati non rappresenterà un problema di performance del sistema.

**1.2.2 Criteri di affidabilità**

**Robustezza**

I componenti devono essere affidabili ed essere in grado di poter  
mantenere i propri dati anche in caso di guasti come problemi  
elettrici, guasti dell'hardware, attacchi informatici, problemi legati al  
browser.

**Disponibilità**

LetsMeet deve essere disponibile all’uso, 24 ore su 24, da parte degli  
utenti, grazie ad un server sempre attivo.

**Tolleranza all’errore**

**(vede vittorio)**

**Sicurezza**

Le tecniche utilizzate, per garantire la sicurezza, sono basate prevalentemente su una ‘login’, la quale permette il riconoscimento dell’utente.  
Tutti gli accessi al sistema avvengono tramite un’operazione di routing che controlla se l’utente che sta operando sul sistema ha i permessi adeguati per usufruire di determinate funzionalità.

C’è, quindi, un controllo che verifica i permessi degli utenti che verranno adeguatamente distribuiti sulla base del grado di importanza e responsabilità degli Utenti.

L’Utente Moderatore potrà godere di funzionalità aggiunte sopra gli altri Utenti, potendo sospenderli o meno dalla piattaforma.

La figura del super-admin è sviluppata per poter indicare un Utente della piattaforma e renderlo Moderatore.

🡪(controllare se è fattibile) Inoltre, le password sono codificate in MD5 senza essere rese in chiaro in nessun campo di nessuna tabella che tiene traccia di  
queste informazioni all’interno del database.

**1.2.3 Criteri di costo**

**Sviluppo**

I costi di sviluppo previsti riguardano l’affitto del server per la fase di sviluppo. È nostra intenzione utilizzare come web Server Altervista(vedere vittorio). Per quanto riguarda il DBMS utilizzeremo MySQL che è open source.

Nei costi di sviluppo rientrerebbero anche le licenze di Google per poter usufruire del servizio Maps, costi mensili per il numero di interazioni con la mappa.

**Deployment**

**Vedere vittorio**

**1.2.4 Criteri di mantenimento**

**Estensibilità**

La progettazione del sistema sarà condotta in modo da agevolare la facile introduzione di nuove funzionalità utilizzando il linguaggio di markup HTML5, i fogli di stile CSS3, JQuery e   
Java.

**Modificabilità**

Deve essere possibile intervenire sul codice esistente per correggere eventuali bugs o implementare nuove funzionalità. Bisogna garantire che il codice sia leggibile per rendere agevole la modifica.

**Leggibilità**

Il codice sarà ben strutturato per semplificare eventuali interventi su di esso.

**Tracciabilità dei requisiti**

Grazie alla tracciabilità̀ dei requisiti, sarà̀ possibile effettuare le modifiche necessarie al corretto funzionamento del sistema, valutando correttamente i costi e i rischi che le modifiche porteranno.

**1.2.5 Criteri per l’utente finale**

**Usabilità**

LetsMeet deve essere facilmente apprendibile (l’utente deve essere in grado di interagire con il sistema e padroneggiare le funzionalità in modo rapido), deve essere flessibile e robusto (l’utente deve essere in grado di capire quando ha successo nel perseguire i suoi obiettivi nel sistema o quando sta sbagliando qualcosa per poter intervenire).

**1.3 Definizione ed acronimi**

* **LetsMeet:** La piattaforma di creazione eventi in questione.
* **Utente:** Utente generico registrato alla piattaforma che può creare, partecipare ad eventi ed utilizzare tutte le funzioni a disposizione della piattaforma.
* **Moderatore:** Utente generico registrato alla piattaforma che può visualizzare le segnalazioni fatte dagli Utenti e può sospenderli eventualmente.
* **Super-Admin:** Utente esterno che può indicare un Utente come Moderatore.
* **Login:** attività di accesso all’account;
* **Logout:** attività di uscita dell'account connesso;
* **DBMS:** Database Management System;
* **SQL:** Structured Query Language, ed è linguaggio di interrogazione (dei database)  
  strutturato.
* **RAD:** Requirements Analysis Document

**1.4 Riferimenti**

L’insieme del materiale di riferimento utilizzato per la realizzazione del progetto e per la  
stesura di questo stesso documento comprende:

* Libro di testo: B.Bruegge, A.H. Dutoit, Object Oriented Software Engineering - Using  
  UML, Patterns and Java, Prentice Hall, 3rd edition, 2009;
* Slide del docente, reperibili sulla piattaforma;
* Documento RAD del progetto LetsMeet;

**1.5 Panaromica**

Il documento si compone di una prima parte in cui vengono introdotti gli obiettivi di design.  
Verrà poi brevemente spiegata l’architettura corrente del sistema prima e dopo il reengineering ma prima di addentrarci nel discutere l’architettura, è doveroso fare un excursus alle attività di system design che compongono le basi per l’architettura software del sistema.

* **Decomposizione del sistema**: il sistema viene decomposto in sottosistemi allo scopo di  
  poter assegnare ad un team parti di software semplici da sviluppare. Un sottosistema è  
  una collezione di classi, associazioni, operazioni, eventi e vincoli che sono in relazione tra  
  di loro. Un insieme di operazioni correlate forma un servizio. Ogni sottosistema quindi sarà  
  caratterizzato dai servizi che offre ad altri sottosistemi; l’insieme dei servizi che un  
  sottosistema espone sarà denominato Interfaccia(API).
* **Mapping Hardware/Software**: descrive come i sottosistemi vengono assegnati  
  all’hardware e alle componenti “off-the-shelf”. Elenca anche le problematiche introdotte da  
  nodi multipli e dal riuso del software.
* **Gestione dati persistenti**: descrive i dati persistenti memorizzati dal sistema e  
  l’infrastruttura di gestione richiesta per essi.
* **Politiche di accesso e sicurezza**: descrive il modello utente del sistema in termini di una  
  matrice degli accessi, stabilendo in modo più preciso le operazioni e le informazioni  
  effettuabili da ogni singolo attore e come questi si autenticano al sistema.
* **Flusso di controllo globale**: descrive quali operazioni eseguire ed in che ordine, per  
  garantire il corretto flusso di controllo del sistema.
* **Condizioni Limite**: descrive lo start-up, lo shutdown e i comportamenti errati del sistema.

**2. Architettura del software corrente**

Attualmente un sistema come LetsMeet non è in circolazione, ma esistono piattaforme che si prefiggono l’obiettivo di condividere eventi ed utilizzano una mappa interattiva unicamente per poter visionare il percorso oppure il luogo dell’evento interessato.

La maggior parte delle piattaforme offrono all’utente unicamente una pagina con le informazioni dell’evento ed eventualmente un numero di partecipanti, senza dare un’effettiva conferma del numero dei partecipanti effettivi ad un evento.

Google in parte offre un servizio servizi di visualizzazione di punti d’interesse su di una mappa, ma non ha le funzionalità del sistema LetsMeet proposto.

**3. Architettura del software proposto**

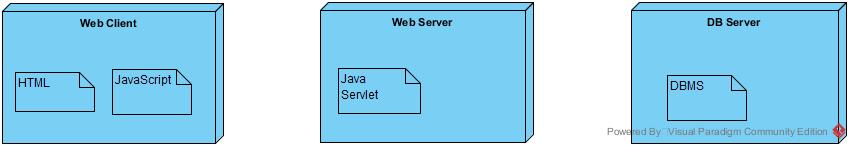
Il sistema proposto è una fusione di servizi che offrono Google ed altri sistemi di condivisione di eventi.

**3.1 Panoramica**

Il sistema proposto è un software web-based per la gestione dei propri eventi online. Le tipologie di utenti si dividono in: utente, moderatore e super-admin. L’utente usufruirà della quasi totalità dei servizi offerti dalla piattaforma; il moderatore accederà alle stesse funzionalità di un utente e in più potrà accedere alle funzionalità che si occupano di gestire gli utenti, dando la possibilità di poter visionare le eventuali segnalazione di commenti o eventi e poter sospendere dalla piattaforma gli utenti segnalati; infine il super-admin potrà gestire solamente l’eventuale trasformazione di un utente in moderatore, poi potrà accedere come utente alla piattaforma.

L’architettura del sistema si presenta come una architettura Client/Server per i seguenti motivi:

* **Portabilità:** il sistema (essendo web-based) potrà essere utilizzato su una varietà di  
  macchine e sistemi operativi.
* **Trasparenza:** il sistema nonostante sia distribuito è in grado di fornire i propri servizi al  
  singolo utente senza interferire con gli altri utenti del sistema.
* **Performance:** il sistema garantirà reattività per i task degli utenti collegati ma essendo  
  web-based molto dipenderà dalla qualità della connessione online.
* **Scalabilità:** il sistema sarà in grado di gestire un grosso numero di utenti connessi  
  contemporaneamente che effettuano le operazioni in contemporanea.
* **Flessibilità:** Il sistema fornirà un’interfaccia grafica intuitiva e con funzionalità  
  specifiche per il ruolo dell’utente che effettuerà l’accesso.
* **Affidabilità:** Entrambi i componenti client e server devono essere affidabili ed essere  
  in grado di mantenere i propri dati anche in seguito a guasti, quindi deve essere  
  possibile effettuare dei backup periodici al database.



**3.2 Decomposizione del sistema**

Per realizzare la piattaforma LetsMeet è stato utilizzato lo stile architetturale three-tier in versione  
Client/server. L’architettura three-tier ("a tre strati") indica una particolare architettura software  
di tipo multi-tier per l’esecuzione di un'applicazione web che prevede la suddivisione  
dell'applicazione in tre strati dedicati rispettivamente alla interfaccia utente, alla logica  
funzionale e alla gestione dei dati persistenti.

In particolare, i tre strati si occupano di differenti funzionalità del sistema di seguito descritte:

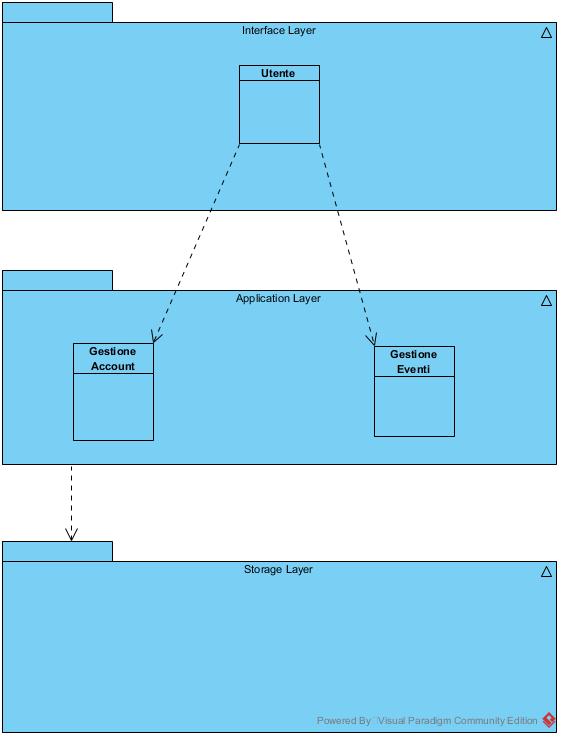
* **Interface Layer:** Include tutte le interfacce grafiche e in generale i boundaryobjects, come le form con cui interagisce l'utente. L’ interfaccia verso l’utente è rappresentata da un Web server e da eventuali contenuti statici (es. pagine HTML).
* **Application Layer** include tutti gli oggetti relativi al controllo e all’elaborazione dei  
  dati. Questo avviene interrogando il database tramite lo storage layer per generare  
  contenuti dinamici e accedere a dati persistenti
* **Storage Layer** effettua la memorizzazione, il recupero e l'interrogazione degli  
  oggetti persistenti. I dati, i quali possono essere acceduti dall’application layer, sono  
  depositati in maniera persistente su un database tramite DBMS.

Riportiamo una breve descrizione delle gestioni del sistema:

* **Gestione account:** Questa funzionalità raccoglie tutte le informazioni per gestire l’autenticazione degli utenti su LetsMeet, la trasformazione di un profilo utente in moderatore e la ricerca/visualizzazione dei profili utenti.
* **Gestione Eventi:** Questa funzionalità raccoglie tutte le informazioni per la gestione degli eventi sulla piattaforma compresa la loro ricerca e l’invio dei messaggi inerenti a quegli eventi.
* **Gestione Segnalazione:** Questa funzionalità raccoglie tutte le informazioni per la gestione delle segnalazioni degli utenti.

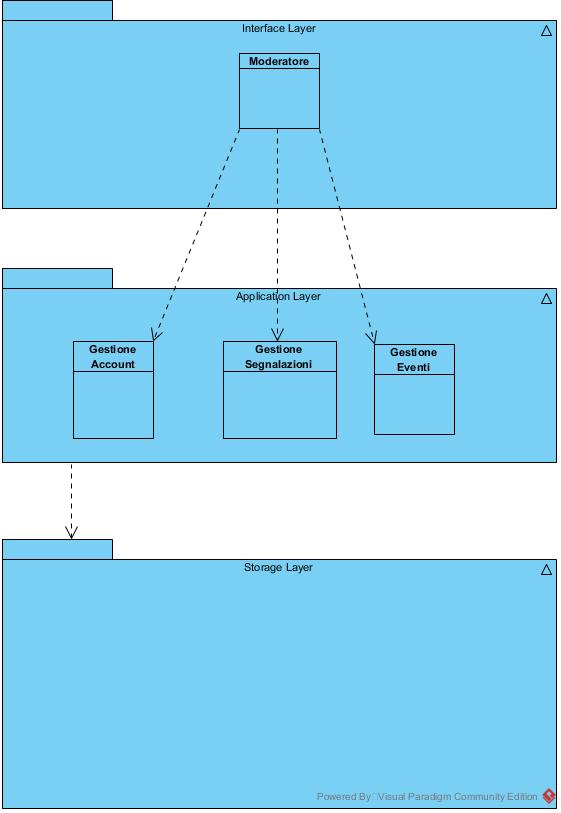
In seguito, verranno mostrati ogni singolo layer.

***Utente***



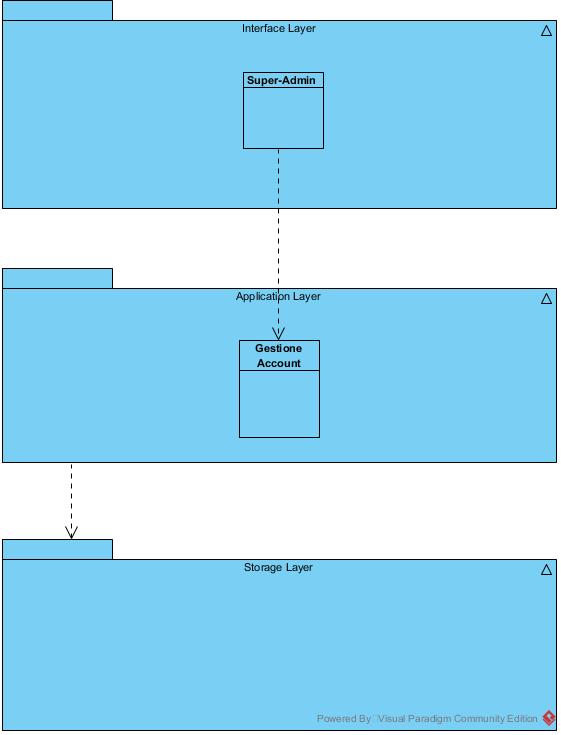
* **Gestione Account:** L’Utente potrà godere delle funzioni riguardanti la propria autentificazione all’interno della piattaforma, la visualizzazione del proprio profilo personale e quello degli altri utenti con eventuale ricerca.
* **Gestione Eventi:** L’Utente potrà godere delle funzionalità di creazione, partecipazione, verifica partecipazione di un evento e la possibilità di ricerca degli eventi con visualizzazione delle eventuali pagine di informazioni.

***Moderatore***



* **Gestione Account:** Il Moderatore potrà godere delle funzioni riguardanti la propria autentificazione all’interno della piattaforma, la visualizzazione del proprio profilo personale e quello degli altri utenti con eventuale ricerca.
* **Gestione Eventi:** Il Moderatore potrà godere delle funzionalità di creazione, partecipazione, verifica partecipazione di un evento e la possibilità di ricerca degli eventi con visualizzazione delle eventuali pagine di informazioni.
* **Gestione Segnalazioni:** Il Moderatore potrà godere delle funzioni di visualizzazione ed accettazione, compreso scarto, di segnalazioni effettuate dagli altri profili della piattaforma, sospendendo gli eventuali utenti segnalati.

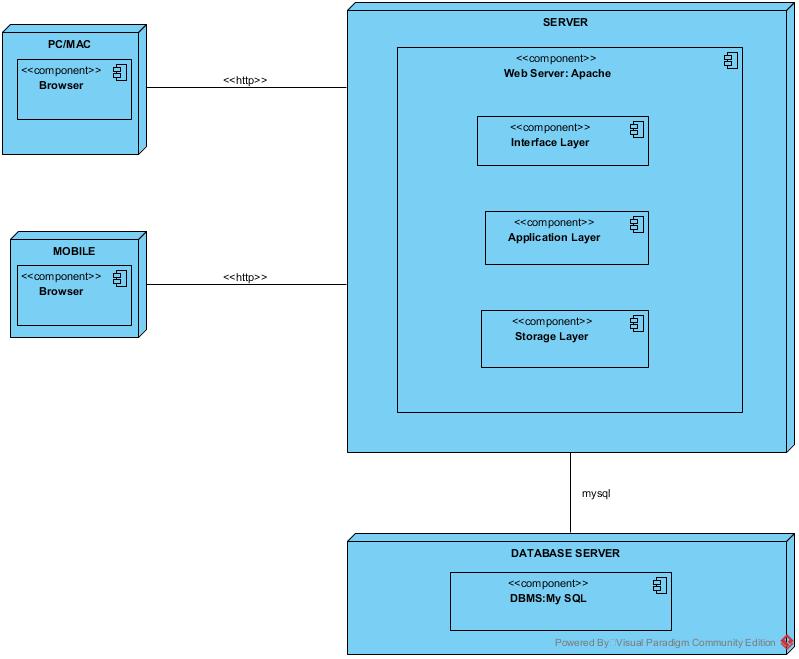
***Super-Admin***



* **Gestione Account:** Il Super-admin potrà godere della funzione di trasformazione degli utenti in moderatori.

**3.3 Mapping Hardware /Software**

La struttura hardware proposta è costituita da un server centrale e dai client cioè un qualsiasi  
computer al quale un utente può collegarsi per sfruttare il browser per accedere al sistema  
LetsMeet. Al server si collegano i client ed il database, il tipo di utente è determinato in fase di  
autenticazione controllando nel database il tipo di utente che corrisponde allo Username  
inserito in quel client. I client dovranno effettuare richieste al server per eseguire le operazioni  
legate alle loro rispettive funzionalità. II client ed il server saranno connesse tramite una rete  
che utilizzerà il protocollo TCP/IP.



**Interface Layer**

L’utente utilizza il sistema mediante un Browser installato all’interno del suo calcolatore (ad es.  
Opera, Firefox, Chrome).  
**Application layer**Il sistema, e quindi le funzionalità, sono implementate in linguaggio JAVA. Il codice in JAVA  
verrà tradotto in linguaggio HTML e il codice risultante viene inviato al browser del client.  
**Storage layer**Rappresenta il collegamento con il server da parte del sistema e si occupa di tutte le richieste  
di accesso e modifiche sui dati permanenti presenti nel database.  
**Database Server***Il DBMS usato* è MySQL il quale presenta molte API che permettono l’interazione tra sistema  
e database.

**//Forse Diagrammi a Design-Time:**

Okkei arrivati a questo punto arriva la parte più complicata.

Doobbiamo fare i Design-Time, se vedi l’esempio capisci, è una sorta di descrizione di passaggio da pagine html a control a model e ne facciamo una per i diversi sequence diagram , questo non significa che per ogni sequence dobbiamo fare sto disgramma, ma un diagramma ingloba più sequence per esempio, la partecipazione e la verofica alla fine si muovono uguali, in questo caso vedi esempio e conforntati con gli altri.

A questo punto si fanno i diagrammi Run-Time, sincer n aggia e capit, arrop vrim.

3.4 Gestione dati persistenti

Dobbiamo dire che utilizziamo un db Relazionale,(vedi esempio che dice merdatecome sicurezza ecc)

3.4.1 Gli mettiamo il nostro ER(che sta praticamente nel RAD)

3.4.2 Digramm Entity-Relation è l’ER nu poc chiu stitimat

3.4.3 Gli dobbiamo scrivere le tabelle del db

3.5 Controllo degli accessi e della sicurezza

Tutt o blocc è na tabella dove diciamo per ogni “attore” utente moderatore ecc che operazioni possono fare

3.6 Controllo del software globale

(Se manteniamo il client/server e le servlet)

Il controllo del flusso software viene gestito da classi java che interagendo con il client, il quale  
si interfaccia tramite un web browser, svolgono le varie operazioni. Il server smista ogni nuova  
richiesta alla classe java adeguata, inoltrando poi la risposta al client.

3.7 Condizioni Boundary

Vedere pure con vittorio ma grossolanamente dovrebbe essere un analisi delle situazioni di start-up, terminazione e failure che il sistema, al lato server, puo avere esi fanno dei piccoli casi d’uso per indicare i processi.

4.0 Servizi dei sottosistemi

Qui facciamo delle tabelle in funzione delle nostre gestioni ex Gestione autenficiazione ed indichiamo i servizi offeri(vedere esempio)