SDD

System Design Document

LUPUS IN CAMPUS

Riferimento	NC12_SDD
Versione	0.12
Data	17/11/2024
Destinatario	Prof Carmine Gravino
Presentato da	NC12 Team
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
27/11/2024	0.1	Prima stesura	F.G S.G C.I
29/11/2024	0.2	Revisione	A.A
2/12/2024	0.3	Aggiunta diagramma architetturale	A.A F.G
9/12/2024	0.4	Aggiunta servizi di sottosistemi	Tutto il team
12/12/2024	0.5	Modifiche dei servizi dei sottosistemi	Tutto il team
15/12/2024	0.6	Modifiche dei sottosistemi, i loro servizi e diagrammi associati	S.G
20/12/2024	0.7	Aggiunte	A.A F.G
22/12/2024	0.8	Aggiunto matrice Controllo degli accessi e sicurezza + Modifiche al class diagram + Prima stesura delle boundary condition	S.G
11/02/205	1.8	Revisione dopo aver iniziato l'implementazione	S.G
17/02/2025	1.8	Revisione	F.G

Team members

Nome	Acronimo	Informazioni di contatto
Angelo Ascione	A.A	a.ascione19@studenti.unisa.it
Federica Graziuso	F.G	f.graziuso1@studenti.unisa.it
Stefano Gagliarde	S.G	s.gagliarde@studenti.unisa.it
Christian Izzo	C.I	c.izzo43@studenti.unisa.it

Sommario

1 Introduzione

1.1 Scopo del sistema

Il sistema **Lupus in Campus** è un'app basato sul famoso gioco da tavolo di deduzione "Lupus in Tabula". L'obiettivo principale è offrire la coinvolgente esperienza trascendendo i limiti dovuti alla posizione geografica dei giocatori.

Il sistema simula l'ambiente del gioco fisico, permettendo ai giocatori di assumere ruoli segreti (lupi mannari, veggenti, abitanti del villaggio, ecc.) e interagire tra loro tramite chat testuale o vocale per svelare i ruoli nascosti e vincere la partita. Il gioco si svolge in turni ciclici (giorno e notte) e le azioni segrete dei giocatori vengono automatizzate tramite il sistema. La narrazione, che solitamente è affidata ad un giocatore fisico, viene automatizzata, gestendo fasi come l'assegnazione dei ruoli, l'ordine dei turni e le votazioni. Inoltre, il sistema punta a valorizzare la comunità studentesca e la cultura locale, integrando riferimenti all'ambiente accademico e territoriale dell'Università di Salerno.

1.2 Obiettivi di design (Design Goals)

Design Goals

Rank	ID Design Goal	Descrizione	Categoria	RNF di origine
1	DG_1 Facilità d'uso	Il sistema deve risultare facilmente comprensibile con l'uso delle "8 regole d'oro di Shneiderman" per il design delle interfacce grafiche	End user	RNF_U_1 RNF_U _2 RNF_U_3 RNF_U_4
6	DG_2 Tempi di risposta	Il sistema deve fornire tempi di risposta entro massimo 5 secondi	Performance	RNF_P_2
4	DG_3 Navigazione	Il sistema deve poter gestire al massimo 100 utenti	Performance	RNF_P_1

	Concorrente	connessi.		
8	DG_4 Estensibilità	Il sistema deve permettere di aggiungere e modificare funzionalità e classi seguendo lo standard ISO/IEC/IEEE 14764:2022	Maintenance	RNF_S_1 RNF_S_2 RNF_I_1 RNF_I_2
5	DG_5 Informazioni sensibili	Il sistema deve proteggere le informazioni sensibili da possibili attacchi malevoli seguendo lo standard ISO/IEC/ 27001	Dependability	RNF_A_1
2	DG_6 Fallimento del sistema	Il sistema rileva errori e notifica l'utente per aiutarlo a correggere l'errore rilevato	Dependability	RNF_A_2
3	DG_7 Riavvio di sistema	Il sistema deve garantire la consistenza dello stato del sistema in caso di riavvio improvviso	Dependability	RNF_A_3
7	DG_8 Univocità codice	Il un dato istante di tempo un codice, può rappresentare una ed una sola lobby	Dependability	RNF_A_4

Trade-off

Tradeoff	Descrizione
User Experience vs Risorse di Sistema	Se le risorse di sistema sono limitate, è possibile ridurre la complessità della User Experience, ad esempio

	eliminando alcune animazioni, per garantire prestazioni più fluide.
Memoria vs Tempo di risposta	Se il sistema non soddisfa i requisiti di tempo di risposta o di velocità, è possibile utilizzare più memoria per accelerare il software
Tempo di rilascio vs Funzionalità	Se i tempi di rilascio sono stringenti, possono essere rilasciate meno funzionalità di quelle richieste, ma nei tempi giusti.
Tempo di rilascio vs Qualità	Se il sistema in fase di rilascio presenta bug critici, è preferibile posticipare la data di consegna per garantire la risoluzione dei problemi.

1.3 Definizioni, acronimi e abbreviazioni

Di seguito una lista di definizioni, acronimi e definizioni

- **Design Goal**: Obiettivi di design per il sistema proposto
- **Dati Persistenti**: Dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software**: Studio della connessione tra parti fisiche e logiche di cui si compongono il sistema.
- **Trade-off**: Scelte e compromessi tra design goals
- **SDD**: System Design Document
- **RNF**:Requisito non funzionale
- **DG**: Design Goals

1.4 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:

- Statement Of Work
- Requirements Analysis Document
- Design Patterns;
- Test Plan;
- Test Case Specification;

1.5 Panoramica

Questo documento di System Design è formato da cinque sezioni:

- **Introduzione**: Viene descritto in generale lo scopo del sistema, gli obiettivi di design che il sistema propone di raggiungere.
- **Architettura software corrente**: Viene descritto lo stato attuale dell'architettura del software già presente.
- **Architettura software proposta**: Viene descritto come il sistema sarà definito e partizionato in sottosistemi, il loro mapping Hardware/Software, la gestione dei dati persistenti. Verranno poi presentate la struttura dei singoli sottosistemi e le boundary conditions riguardanti l'intero sistema.
- **Servizi dei sottosistemi**: Vengono descritti i servizi offerti da ciascun sottosistema, specificando i compiti, le funzionalità principali e le interazioni con altri sottosistemi.
- **Glossario**: Contiene la lista dei termini usati nel documento con annessa spiegazione

2 Architettura del sistema corrente

Il sistema attuale è un prodotto fisico, ovvero un gioco da tavolo, e non un software. Di conseguenza, non è possibile individuare in esso alcuno schema di architettura software.

3 Architettura del sistema proposto

3.1 Panoramica

L'architettura scelta per il sistema proposto è **MVC**. Questa decisione deriva dalla semplicità di implementazione di ogni sottosistema che, in base alla responsabilità, può essere assegnato ad una delle tre categorie che il modello MVC identifica. Questa architettura garantisce inoltre l'aggiornamento di informazioni in tempo reale lato client, grazie al design pattern Observer-Observable, il quale fa sì che ogni vista sia sottoscritta ad eventi di notifica pubblicati dal model ad ogni suo cambiamento.

L'MCV, inoltre, garantisce una maggiore sicurezza sui dati, in quanto, la separazione di responsabilità, fa sì che il client non possa accedere direttamente al database. Infine, il modello MVC, permette di separare la logica di business dal client, facilitando la modifica, l'aggiunta o la rimozione di funzionalità, senza intaccare l'esperienza utente.

3.2 Scomposizione in sottosistemi

I sottosistemi identificati sono:

- **Gestione area utente:** si occupa di visualizzare l'area utente e della modifica dati dell'account
- **Autenticazione:** è responsabile delle funzionalità di Login, Logout, Registrazione
- **Gestione sistema di comunicazione:** si occupa della gestione sulla visualizzazione e nell'usare il sistema di comunicazione vocale e testuale
- **Gestione lobby:** si occupa di gestire la creazione, modifica e eliminazione di una lobby, di unirsi e uscire ad una lobby e di invitare amici nella propria lobby
- **Gestione partita:** si occupa di gestire il corretto svolgimento dei turni in una partita
- **Gestione notifiche:** si occupa di gestire l'invio di ogni tipo di notifica: pop-up, nel sistema, di errore, di invito e di amicizia
- **Gestione amici:** si occupa di gestire la logica degli amici
- **Persistenza:** si occupa di gestire la persistenza dei dati con un database, usando Spring JPA

Sono mostrate di seguito le dipendenze tra i sottosistemi attraverso un component diagram UML:

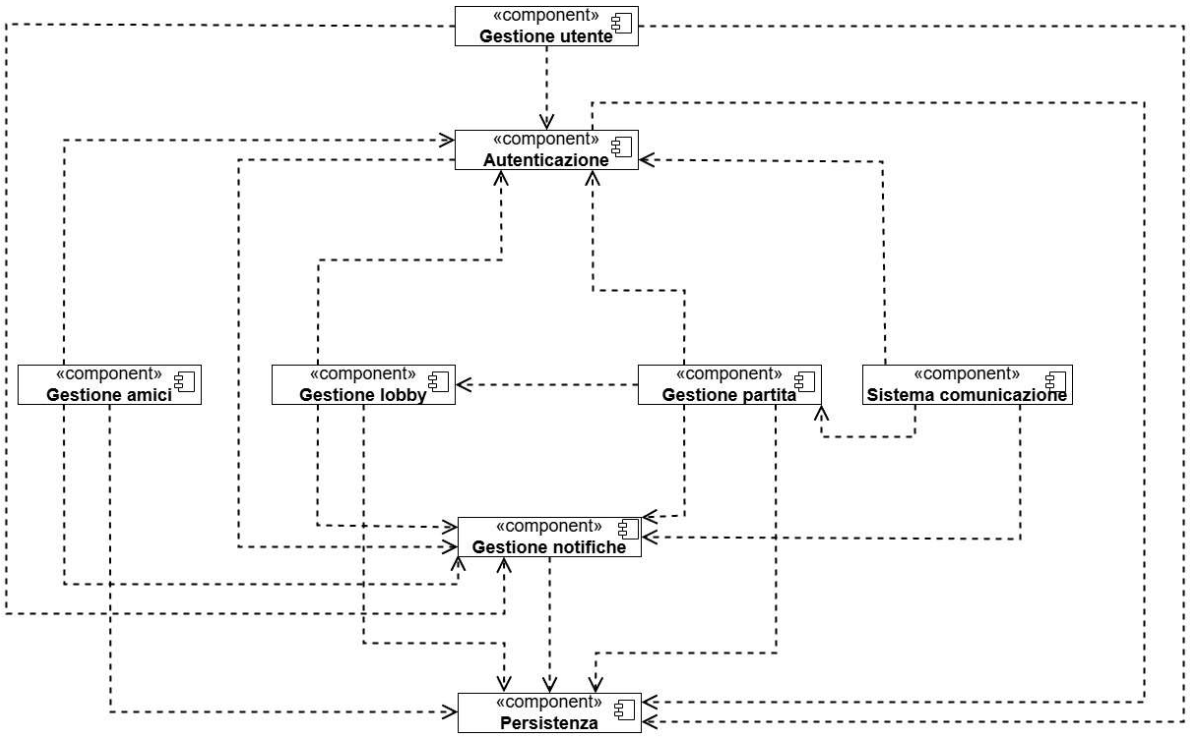
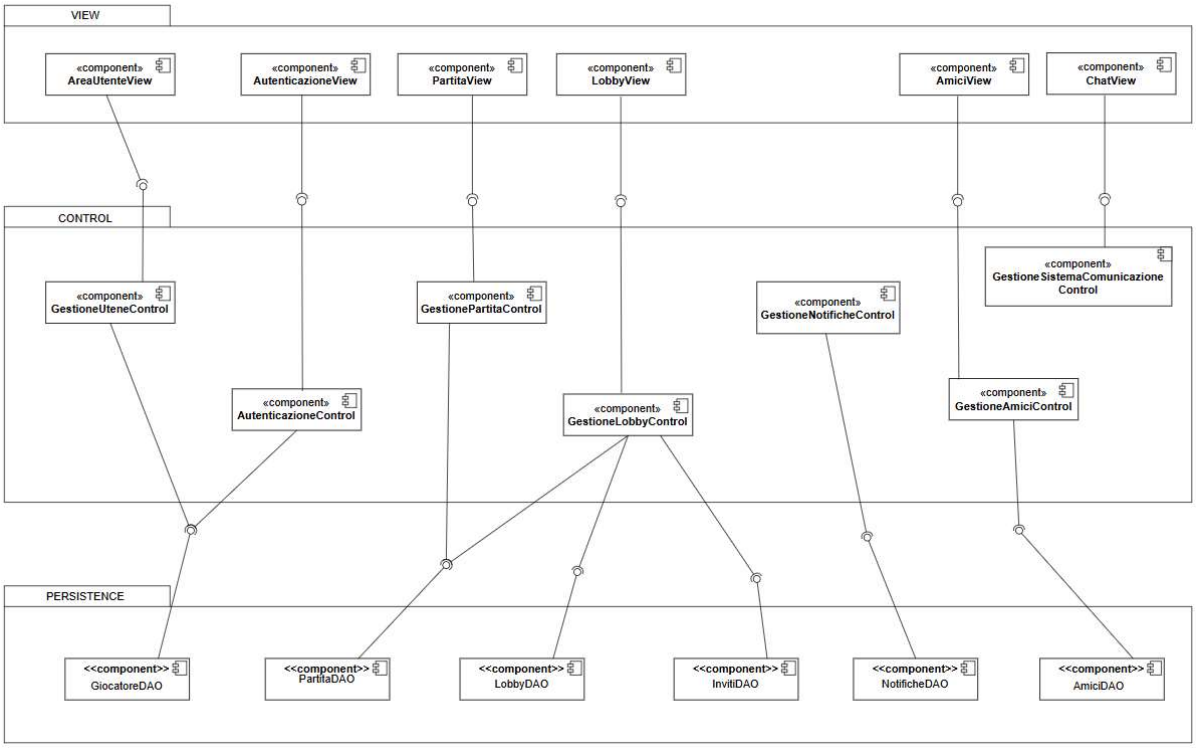
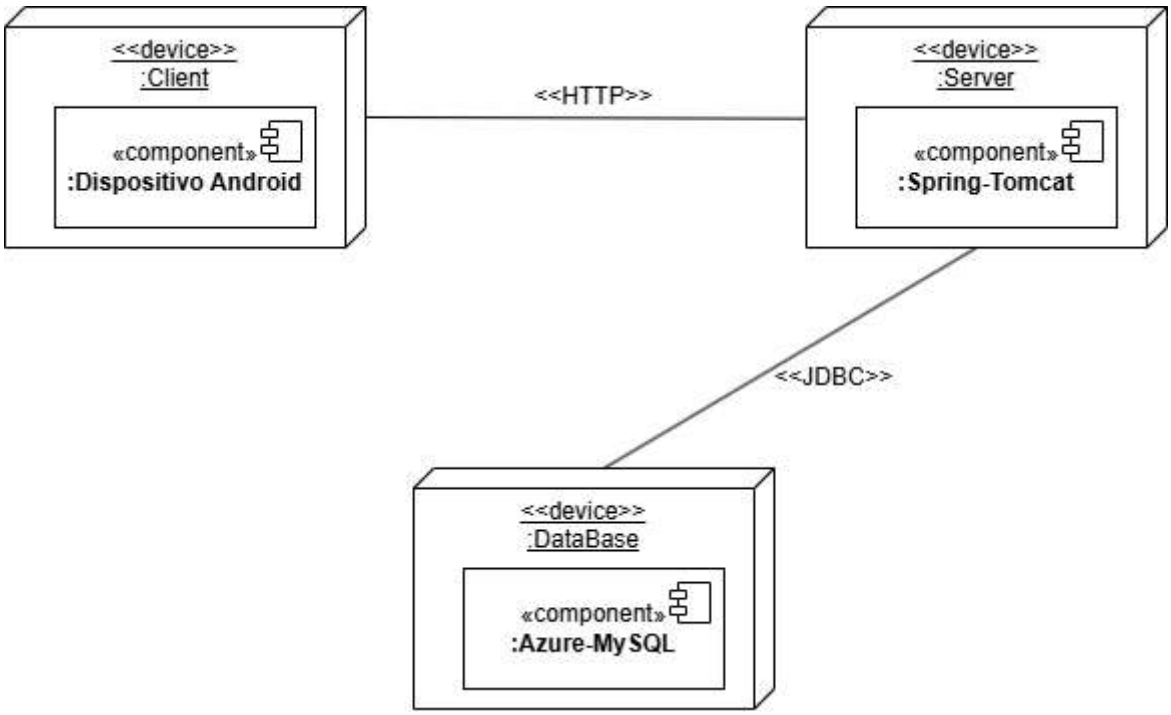


Diagramma architetturale



3.3 Mapping Hardware/Software

L' applicazione che verrà sviluppata si basa su una piattaforma hardware costituita da un server che risponde alle richieste effettuate dai clienti dai loro dispositivi mobile. Essendo che il nostro sistema è un videogioco, esso è diviso in una parte server e in una parte applicativa che gli utenti utilizzano per poter giocare.
Di seguito un UML deployment diagram che descrive il mapping hardware/software.



3.4 Gestione dei dati persistenti

Per gestire il salvataggio dei dati persistenti del sistema, abbiamo deciso di utilizzare un database relazionale, che permette di gestire facilmente l'accesso concorrente ai dati e di garantire la loro consistenza tramite l'uso di un DBMS.

La scelta di un DBMS è stata fatta per rispettare gli obiettivi di progettazione, offrendo:

- **Integrità dei dati**: Il DBMS applica vincoli per garantire l'integrità dei dati e li verifica durante gli aggiornamenti del database.
- **Privacy dei dati**: Diversi utenti possono accedere solo a specifiche parti del database e con permessi differenziati.
- **Affidabilità dei dati**: Il DBMS consente di effettuare backup e offre metodi per ripristinare il database in caso di guasti software o hardware.

- **Atomicità delle operazioni**: Le transazioni vengono eseguite interamente o non vengono eseguite affatto, mantenendo la coerenza del database con il modello reale.

Abbiamo deciso di utilizzare un database in cloud tramite Azure. Questa soluzione ci consente di lavorare su un'unica istanza condivisa, riducendo i problemi legati all'hosting locale del database e, di conseguenza, il tempo speso per risolvere eventuali errori.

3.5 Controllo degli accessi e sicurezza

Di seguito viene mostrata la matrice degli accessi per poter tenere traccia di quali attori possono accedere a quali servizi offerti dal sistema.

	Giocatore
Gestione utente	VisualizzaAreaUtente, ModificaDatiUtente, CancellazioneAccount, StoricoPartiteConGiocatori
Autenticazione	Registrazione, Login, Logout, RecuperaPassword
Sistema comunicazione	VisualizzaSchermataVocale, AttivaMicrofono, DisattivaMicrofono, VisualizzaSchermataTestuale, InviaMessaggio
Gestione lobby	CreazioneLobby, ModifcaLobby, EliminaLobby, UnioneLobby, VisualizzaLobby, VisualizzaInviti, InvitaAmici
Gestione partita	TurnoVotazione SvolgimentoTurno
Gestione notifiche	InviaNotifica

Gestione amici	AggiungiAmico, RimuoviAmico, CercaUtente, AccettaRichiestaAmicizia, RifiutaRichiestaAmicizia, InviaRichiestaAmicizia

3.6 Controllo globale del software

Il sistema di gioco è un'applicazione mobile interattiva in cui ogni funzionalità viene avviata a seguito di una richiesta dell'utente, inviata tramite l'interfaccia grafica dell'app. Quando l'utente desidera accedere a una determinata funzionalità, come effettuare il login o visualizzare una finestra specifica, invia un comando che viene interpretato dal sistema.

La richiesta viene gestita da un componente dedicato che analizza l'input dell'utente, esegue le operazioni necessarie sul server tramite il backend Spring e produce una risposta che, a sua volta, viene gestita dal codice lato client. Il server è responsabile dell'indirizzamento del flusso verso i moduli applicativi appropriati, come la logica di controllo e la logica applicativa, per garantire una corretta elaborazione e un'esperienza fluida per l'utente.

Essendo un'applicazione Android che interagisce con un server, il sistema adotta un approccio basato su eventi (event-driven), in cui ogni azione dell'utente genera un evento che viene elaborato in tempo reale, a seconda delle risposte ricevute dal server.

3.7 Condizioni limite

Nel progetto non è prevista la gestione esplicita delle condizioni limite, poiché tale gestione è già automaticamente gestita da Spring con Tomcat e MySQL.

4 Servizi dei sottosistemi

In questa sezione vengono descritti i servizi di ogni sottosistema precedentemente elencati

Sottosistema Autenticazione

Servizio	Descrizione	Operazioni
Login	Questo servizio permette di effettuare l'accesso al sistema tramite le proprie credenziali	login
Logout	Questo servizio permette di disconnettersi dal sistema.	logout
Registrazione	Questo servizio permette di registrarsi sulla piattaforma.	registrazione
Cambio password	Questo servizio permette di cambiare la propria password se dimenticata	passwordDimenticata

Sottosistema Gestione utente

Servizio	Descrizione	Operazioni
Visualizza area giocatore	Permette di visualizzare i dati relativi alla propria area giocatore.	storicoPartite; listaAmici;
Modifica dati	Permette di modificare i dati del giocatore.	modificaNickname; resettaPassword;
Cancellazione account	Permette di cancellare il proprio account sulla piattaforma.	cancellaGiocatore;

Sottosistema Sistema comunicazione

Servizio	Descrizione	Operazioni
Visualizza sistema	Permette di visualizzare	vediChatTestuale;

	l'interfaccia dei vari sistemi di comunicazione (vocale e testuale)	vediChatVocale;
Strumenti per la comunicazione	Permette di utilizzare gli strumenti per effettuare la comunicazione	attivaMicrofono; disattivaMicrofono; inviaMessaggio;
Comunicazione	Permette il corretto funzionamento della comunicazione	inviaAudio; inviaMessaggio;

Sottosistema Gestione lobby

Servizio	Descrizione	Operazioni
Creazione lobby	Questo servizio permette di creare una nuova lobby specificando il tipo (Pubblica, Privata, Locale)	creaLobby
Modifica lobby	Questo servizio permette di modificare il tipo di una lobby	modificaLobby
Elimina lobby	Questo servizio permette di eliminare una lobby creata	eliminaLobby
Entrare e Uscire da lobby	Questo servizio permette ai giocatori di unirsi e uscire da una lobby	joinLobby leaveLobby
Invita amici	Questo servizio permette di inviare inviti a amici per unirsi alla lobby	invitaAmicoLobby

Sottosistema Gestione partita

Servizio	Descrizione	Operazioni
Assegnazione Ruolo	Questo servizio si occupa di gestire l'assegnazione dei ruoli allí'inizio della partita, farà in modo di creare una partita bilanciata tra numero di giocatori e ruoli assegnati	AssegnazioneRuoliService
Controllo turni	Questo servizio si occupa di controllare il corretto flusso della partita, fa si che ogni ruolo possa agire al momento giusto, gestisce anche i turni di discussione e di votazione	ControlloTurniService

Sottosistema Gestione notifiche

Servizio	Descrizione	Operazioni
Invia notifiche push	Questo servizio invia notifiche per eventi come inviti nelle lobby, richieste di amicizia e messaggi all'interno del sistema.	inviaNotificaPush
Notifiche di Partita	Gestisce notifiche relative allo stato della partita, come l'inizio di un turno o la fine di una fase di gioco.	inviaNotificaPartita
Notifiche di Errori o Avvisi	Gestisce le notifiche in caso di errori o problemi tecnici nel sistema.	InviaNotificaInfo

Sottosistema Gestione amici

Servizio	Descrizione	Operazioni
Invia richiesta di amicizia	Permette di inviare una richiesta di amicizia ad un giocatore	inviaRichiestaAmicizia
Risultato della richiesta di amicizia	Permette di aggiornare le liste amici di entrambi i giocatori quando il destinatario accetta l'invito, altrimenti elimina la richiesta di amicizia.	risultatoRichiestaAmicizia
Rimuovi amico	Permette di rimuovere un giocatore dalla propria lista amici.	rimuoviAmico
Cerca utente	Permette di cercare un giocatore inserendo il suo nickname	cerca