Università degli Studi di Salerno
Corso di Ingegneria del Software

Esbet START
System Design Document
Versione 1.0

Data: 25/11/2024

Progetto: EsbetSTART	Versione: 2.0
Documento: System Design Document	Data: 25/11/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Aversana Marco	0512118978
Petrillo Francesco	0512116329
Polise Michele	0512116854
Zaccaro Valeria	0512117733

Scritto da:	

Revision History

Data	Versione	Descrizione	Autore
25/11/2024	1.0	Prima stesura SDD	

1. Introduzione

In questo documento descriveremo come si intende procedere con l'architettura del sistema del sito EsbetSTART e quali sono gli obbiettivi di design che intendiamo raggiungere.

1.1. Scopo del sistema

EsbetSTART si propone di presentarsi come un sito di scommesse dedicato agli e-sports. Come in ogni altro *bookmaker* si metterà a disposizione dell'utente registrato la possibilità di piazzare scommesse, gestire il proprio conto e il proprio profilo e partecipare a offerte proposte.
Il sito verrà gestito da utenti con privilegi, che potranno gestire gli eventi e le offerte disponibili.

1.2. Obbiettivi di design

Gli utenti principali di EsbetSTART sono gli scommettitori, a cui è necessario fornire un'esperienza di utilizzo semplice e intuitiva. Per evitare di perdere utenti il sistema dovrà essere solido e poter gestire picchi di carico aspettati in prossimità di competizioni importanti, oltre che permettere l'inserimento di eventi sempre nuovi per stare al passo con i tempi.
Trattando inoltre dati sensibili e denaro, uno sguardo importante va anche alla sicurezza e alla corretta gestione dei malfunzionamenti.
In base a queste osservazioni, si identificano i seguenti obbiettivi di design:

- *Disponibilità*: vista la globalità delle competizioni presenti sulla scena, il sito dovrà essere disponibile h24, in modo da dare agli utenti massima libertà nelle scommesse ed evitare che si rivolgano alla competizione.

- *Scalabilità*: Il sistema deve permettere la scalabilità sia in termini di persone connesse, in modo da gestire picchi di carico in prossimità di eventi particolarmente importanti, sia in termini di numero di eventi supportati; è prevedibile, infatti, che con il passare del tempo la scena degli e-sports si espanda a macchia d'olio, e EsbetSTART deve essere pronto ad espandersi con essa.

- *Sicurezza*: Per rispettare le legislazioni presenti sul gioco d'azzardo il sito si troverà a dover gestire dati estremamente sensibili degli utenti (le immagini dei loro documenti d'identità ad esempio), per questo il sito dovrà garantire la massima sicurezza nel trattamento degli stessi.

- *Gestione dei malfunzionamenti*: Dovendo gestire il denaro degli utenti, i malfunzionamenti dovranno essere gestiti in una maniera che garantisca l'integrità e la correttezza dei dati, nelle operazioni di piazzamento delle scommesse ma soprattutto in quelle di prelievo e deposito dal conto di gioco. Uno sguardo particolarmente attento dovrà essere dato al rispetto delle proprietà *ACID* per le transazioni su memoria persistente.

- *Facilità di aggiornamento degli eventi*: Il sistema per inserire nuovi eventi, competizioni o giochi dovrà essere intuitivo e veloce. Un aggiornamento lento e problematico infatti

potrebbe portare alla mancata disponibilità di eventi in voga, spingendo gli utenti a spostarsi su altre piattaforme.

1.3. Referenze

Requirement Analysis Document EsbetSTART

2. Sistema corrente

Non è presente un sistema corrente per l'applicazione da sviluppare.

3. Architettura del sistema proposto

3.1. Panoramica

Nelle prossime sezioni andremo a definire l'architettura del sistema in termini di:

- *Decomposizione in sottosistemi*: si propone un'architettura a tre livelli con suddivisione tra interfacce, logica applicativa e logica di persistenza.

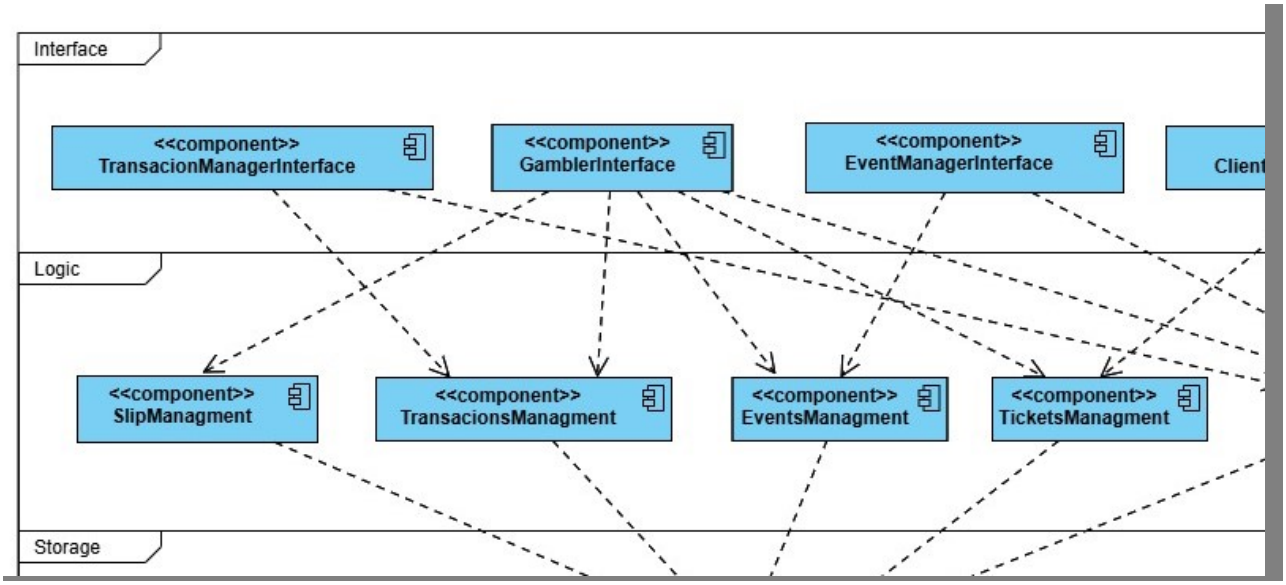
- *Hardware/software mapping*:
- *Gestione dei dati persistenti*:
- *Controllo di accessi e sicurezza*:
- *Controllo globale del software*:
- *Boundary conditions*:

3.2. Decomposizione in sottosistemi

Per la divisione in sottosistemi abbiamo osservato i requisiti funzionali e non funzionali. Notando che l'applicazione risulta molto strutturata e non necessita di aggiornamenti in tempo reale a bassa latenza (come potrebbe un videogioco) abbiamo optato per un'architettura a tre livelli in cui riconosciamo

1. Le interfacce degli utenti (divisi per ruoli)
2. La logica applicativa
3. La logica di persistenza

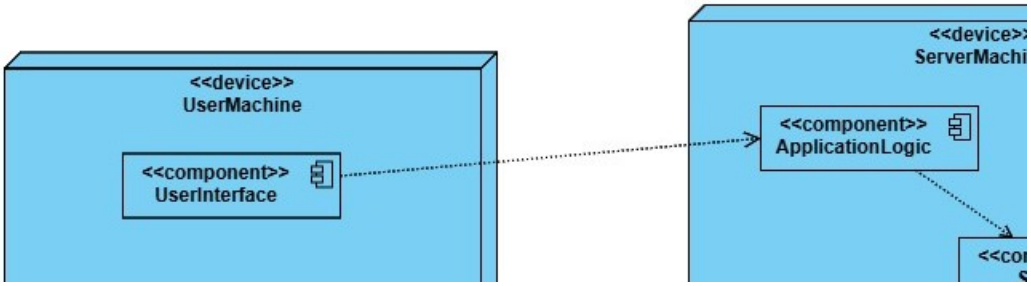
Le interfacce utente comunicano direttamente con i sottosistemi del livello inferiore, che si occupano di offrire servizi sia per gli utenti scommettitori che per i gestori. I permessi di accesso sono descritti nella tabella degli accessi (sez. 3.5).
I sottosistemi della logica applicativa comunicano poi con *Storage* per salvare i dati in modo persistente.



3.3. Hardware/software mapping

EsbetSTART nasce come sistema distribuito, in cui più client da tutto il mondo si possono collegare a un server centrale che gestisce le richieste. È compito del server capire quale utente può eseguire quali comandi (a seconda dei permessi).

Il sistema sarà sviluppato utilizzando il framework Spring Boot, che facilita la creazione di applicazioni web riducendo al minimo la necessità di configurazioni manuali. Spring Boot è ideale per implementare architetture a microservizi, offrendo server integrati e una gestione efficiente delle dipendenze. Per garantire un controllo sicuro degli accessi e la gestione dei ruoli, verrà utilizzato il modulo Spring Security. Sul lato client verrà impiegato Angular, un framework front-end che permette di sviluppare interfacce utente interattive, modulari e facilmente scalabili, semplificando l'integrazione con i servizi back-end tramite il supporto nativo alle API REST.



Userinterface e ApllicationLogic racchiudono tutti i sottosistemi relativi

3.4. Gestione dei dati persistenti

Il sistema dovrà gestire il salvataggio in modo persistente di:

- I dati degli utenti
- Gli eventi a disposizione con le relative quote
- Le promozioni disponibili
- I ticket compresi di relativi messaggi
- Le transazioni effettuate da tutti gli utenti

Per farlo verrà utilizzato un database relazionale, in modo da garantire accesso veloce ai dati e il rispetto delle proprietà ACID nelle transazioni. Il database verrà implementato tramite il DBMS *PostgressSQL* e la comunicazione con quest'ultimo verrà fatta tramite JPA.

3.5. Controllo di accessi e sicurezza

La **Global Access Table** definisce i permessi associati ai vari ruoli all'interno del sito, descrivendo in modo chiaro e organizzato le azioni che ciascun ruolo è autorizzato a compiere sulle entità del sistema. La tabella è strutturata in modo tale da rappresentare gli attori nelle righe e le entità del sistema a cui possono accedere nelle colonne, specificando le operazioni consentite per ciascuna combinazione.

Actors / Objects	Slip	Bet Placed	Event	Ticket	Offer	Gambler
Gambler	addOdd removeOdd placeBet	viewHistory	search view	open sendMessage	activate	withdraw deposit updateInfo
Transaction Manager					create update remove	viewAllTransactions
Event Manager			create remove update end			
Client Service Manager				viewRequests open replyToMessage close		

Il controllo degli accessi sarà affidato al componente **User Management**, che, in base al ruolo associato al profilo, concede o nega il permesso di accedere ad una funzionalità.

3.5.1. Controllo globale del software

In EsbetSTART, il **control flow** sarà implementato utilizzando un approccio **event-driven**, supportato dai meccanismi nativi di Spring Boot. Gli eventi generati dagli attori, come richieste HTTP degli utenti o notifiche di aggiornamento dai servizi interni, saranno catturati dai controller e gestiti da componenti dedicati. Il **DispatcherServlet** di Spring fungerà da main loop centrale, occupandosi di inoltrare ogni richiesta al controller appropriato. All'interno dei controller, i servizi necessari verranno invocati per elaborare le azioni richieste, centralizzando l'elaborazione degli input e mantenendo una struttura ben organizzata.

Gestione dei Processi Concorrenti

Per i processi concorrenti o asincroni, EsbetSTART sfrutterà thread gestiti automaticamente tramite il supporto dell'annotazione `@Async` e del thread pool configurato nel contesto dell'applicazione. Questo approccio consente di combinare la semplicità e la leggibilità di un modello event-driven con le capacità di elaborazione parallela quando richiesto, garantendo un'esecuzione fluida e scalabile anche in scenari più complessi.

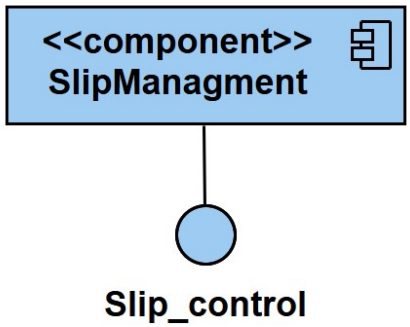
3.6. Boundary conditions

Le condizioni di confine in una web application si riferiscono a tutti i casi limite o alle situazioni estreme che potrebbero influenzare il comportamento del sistema, come carichi elevati, accessi simultanei o situazioni di errore. Il monitoraggio e la gestione delle condizioni di confine sono affidati a un amministratore di sistema che ha il compito di supervisionare l'infrastruttura, eseguire operazioni di manutenzione e intervenire in caso di malfunzionamenti. L'amministratore si occupa di gestire, ad esempio, eventuali errori del sistema, e garantire che la web application funzioni in modo affidabile anche nelle condizioni di carico massimo o durante eventuali guasti parziali.

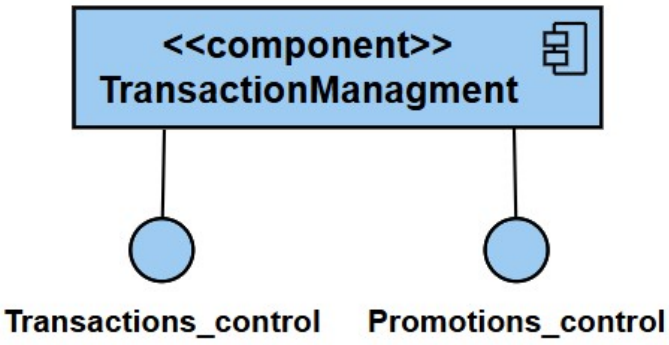
4. Servizi dei sottosistemi

Analizziamo ora per i sottositemi di logica i servizi che devono offrire.

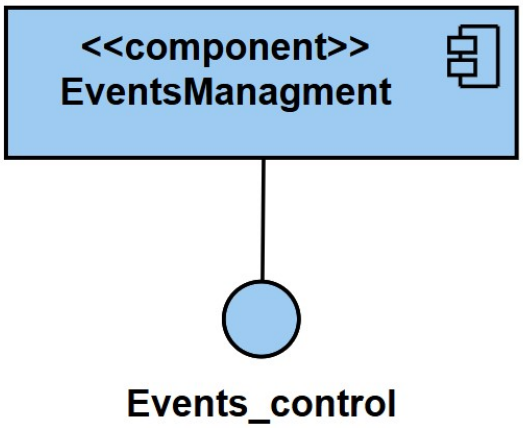
SlipManagment
È definito il servizio *SlipControl*, che racchiude tutte le funzionalità di modifica di una schedina.



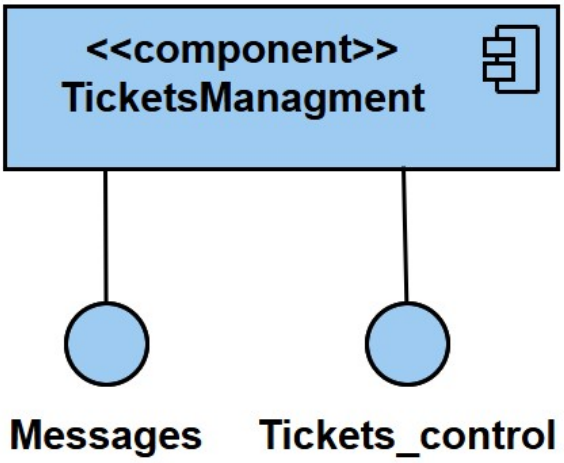
TransactionsManagment
Sono definiti i servizi *TransactionsControl* e *PromotionsControl*, che raccolgono le funzionalità di visualizzazione e modifica delle transazioni e delle promozioni.



EventsManagment
È definito il servizio EventsControl, che raccoglie le funzionalità di visualizzazione e modifica di eventi, competizioni e giochi.



TicketsManagment
Sono definiti i servizi TicketsManagment e Messages, che raccolgono le funzionalità di visualizzazione e modifica dei ticket e di invio dei messaggi.



UsersManagment
Sono definiti i servizi Authentication e Authorization, che raccolgono le funzionalità di gestione degli utenti registrati sul sito e di gestione dei ruoli per il controllo degli accessi alle funzionalità degli altri componenti.