Università degli Studi di Salerno

Corso di Ingegneria del Software

UStrike!

SDD

Versione 1.0

Data: 25/11/2025

Informazioni sul Progetto

Progetto: UStrike!
Documento: **SDD**
Versione: 1.0
Data: 25/11/2025

Coordinatore del progetto:
Nome Matricola

Partecipanti:

Nome	Matricola
Russo Serena	0512119098
Stefanile Andrea	0512119557
Valito Marcello	0512119014

Scritto da:
Russo Serena, Stefanile Andrea, Valito Marcello

Revision History

Data	Versione	Descrizione	Autore
24/11/2025	0.1	Stesura del documento SDD	Russo Serena, Stefanile Andrea, Valito Marcello
25/11/2025	1.0	Rifinimento del SDD e consegna	Russo Serena, Stefanile Andrea, Valito Marcello

Sommario

1.0-Introduzione

1.1-Scopo del sistema

Nel settore dell'intrattenimento, strutture come sale da bowling e arcade si affidano ancora a metodi di gestione tradizionali, quali prenotazioni telefoniche e agende cartacee. Questo approccio manuale presenta diverse criticità: è soggetto a errori umani (prenotazioni sovrapposte o registrate in maniera errata), risulta inefficiente durante le ore di punta e offre un'esperienza al cliente poco moderna e frammentata.
Manca una piattaforma digitale unificata che possa semplificare e automatizzare la gestione delle prenotazioni per i vari servizi offerti (bowling, biliardo, go-kart). L'obiettivo del progetto UStrike! è colmare questa lacuna, creando un sistema centralizzato che migliori l'efficienza operativa dello staff e l'esperienza complessiva del cliente, riducendo gli errori e ottimizzando l'interazione tra l'arcade e il suo pubblico.

1.2-Obiettivi del sistema

Il sistema "UStrike!" è stato progettato considerando i seguenti obiettivi di design:

1.2.1-Criteri di performance

- Velocità delle azioni principali: Quando un cliente apre la disponibilità dei servizi (Bowling, Biliardo, Go-Kart) o decide di confermare o annullare una prenotazione, la pagina deve rispondere entro 1,5 secondi. Questo supporta l'esperienza utente in orari di punta.

- Carico elevato di utenti: Il sistema deve essere in grado di gestire fino a 200 persone contemporaneamente, senza mostrare messaggi di errore e rispettando il tempo di risposta di 1,5 secondi del punto precedente.

1.2.2-Criteri di usabilità

- Autonomia dell'utente: L'interfaccia web deve essere progettata per essere intuitiva, permettendo a un visitatore che la utilizza per la prima volta di prenotare senza bisogno di aiuto. L'interfaccia deve accompagnare l'utente passo dopo passo, con testi chiari, pulsanti ben visibili e una struttura che renda tutto immediatamente comprensibile.

- Semplicità del flusso di prenotazione: Il processo di prenotazione deve essere semplice e diretto, richiedendo all'utente un massimo di 6 passaggi in non più di 120 secondi. Ogni schermata deve avere un obiettivo chiaro (selezione del servizio, scelta data-ora, rivedere e confermare), con solo i campi essenziali.

1.2.3-Criteri di affidabilità

- Robustezza: Il sistema è progettato per continuare a funzionare anche in caso di problemi con un componente o durante picchi moderati di traffico. Se un servizio esterno (es. e-mail) dovesse guastarsi, le funzionalità principali rimangono operative e le richieste verso quel servizio vengono ripetute fino a due volte. In caso di blocco totale, il servizio si riattiva automaticamente, assicurando che le operazioni non rimangono a metà: ogni prenotazione sarà o confermata o annullata dopo un processo di riconciliazione che si occupa di sistemare le transazioni interrotte.

- Sicurezza: Le password degli utenti non vengono mai memorizzate in chiaro. Il sistema utilizza un algoritmo di hashing specifico per le password e salva solo l'hash risultante. Dopo il login, l'utente riceve token di sessione a breve termine che possono essere revocati in qualsiasi momento (ad esempio in caso di logout). Questi token sono trasmessi in cookie contrassegnati come HttpOnly e Secure.

1.2.4-Criteri di manutenzione:

- Modificabilità e Manutenibilità: Il codice deve seguire pattern standardizzati (MVC, Repository, Catalog Objects) e convezioni di naming chiare per facilitare la manutenzione futura da parte di nuovi sviluppatori.

1.3-Definizione, acronimi e abbreviazioni

- RAD: Requirements Analysis Document
- SDD: System Document Design
- UC: Use Case
- Utente: Termine generico per indicare un qualsiasi individuo che interagisce con la piattaforma.
- Cliente: Utente Registrato a UStrike!
- Staff: Membro del team di UStrike!
- Manager: Capo dello staff.
- Servizio: Uno dei tre servizi principali di UStrike! (Bowling, Biliardo, Go-Kart)

1.4-Riferimenti

Riferimento al Requirement Analysis Document di UStrike! e Bern Bruegge, Allen H. Dutoit, Object-Oriented Software Engineering - Using UML, Patterns, and JAVA.

1.5-Panoramica

Il documento è composto da 5 sezioni.
Nella prima sezione troviamo un'introduzione composta da scopo del sistema, definizione degli obiettivi del sistema con i vari criteri sui quali ci si basa, varie definizioni di acronimi e abbreviazioni usate nel seguente documento o termini relativi all'ambiente e per finire i riferimenti.
Nella seconda sezione si parla dell'architettura attuale.
Nella terza sezione si parla dell'architettura proposta per la risoluzione del problema. Si parte con una panoramica generale della soluzione per poi passare alla decomposizione in sottosistemi.
Una volta finita la definizione dei sottosistemi si passa alla mappatura Hardware e Software che ci dà una visione sui dispositivi hardware e le piattaforme software che verranno utilizzate nel progetto.
Dopodiché si passa alla gestione dei dati persistenti da conservare nel database.
Capiti i dati persistenti arriviamo alla fase di controllo nella quale definiamo quali utenti potranno accedere ai relativi servizi offerti dal sistema.
Sempre in questa fase definiamo anche il controllo globale del software e le condizioni di boundary da seguire.
Nella quarta sezione invece definiamo per ogni sottosistema quale funzionalità svolge.
Infine, nella quinta sezione è presente un glossario di termini tecnici utilizzati all'interno del documento.

2.0-Architettura Software Attuale

Essendo che UStrike! si propone di creare un'architetture software per l'arcade fisico non è presente alcuna architettura software precedente.

3.0-Architettura Software Proposta

3.1-Panoramica

Il sistema UStrike! adotta un'architettura MVC, tipica dei sistemi web. I sottosistemi principali previsti dall'architettura MVC sono i seguenti:

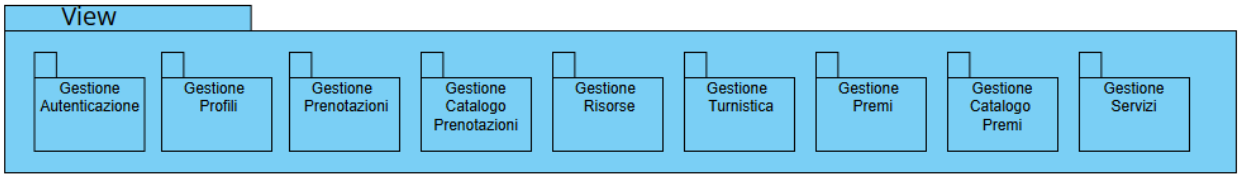
Model: rappresenta il sistema di gestione dei dati e della logica di business.

View: rappresenta il sistema di interazione diretta con l'utente, è composta da tutte le interfacce che l'utente è in grado di vedere e con le quali è in grado di interagire.

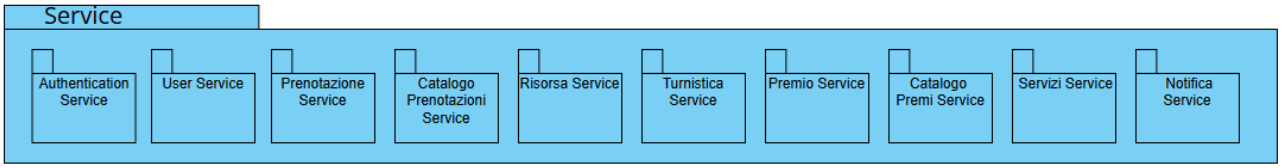
Controller: in questo sottosistema sono presenti tutte le componentistiche che elaborano i dati, si occupa anche delle interazioni tra View e Model.

3.2-Decomposizione in sottosistemi

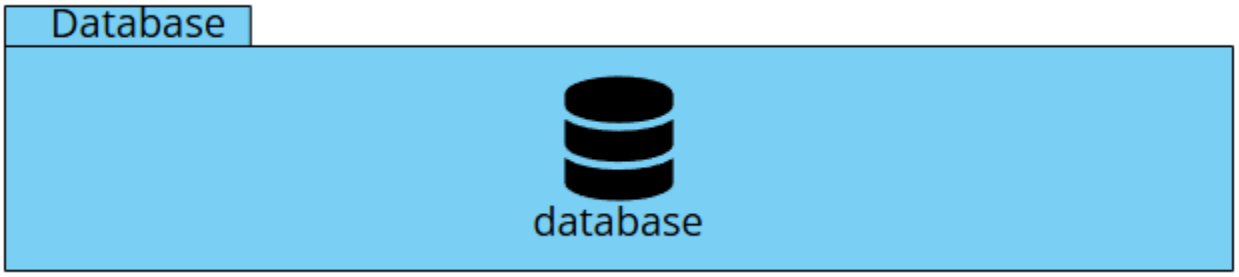
PresentationLayer



DataManagementLayer



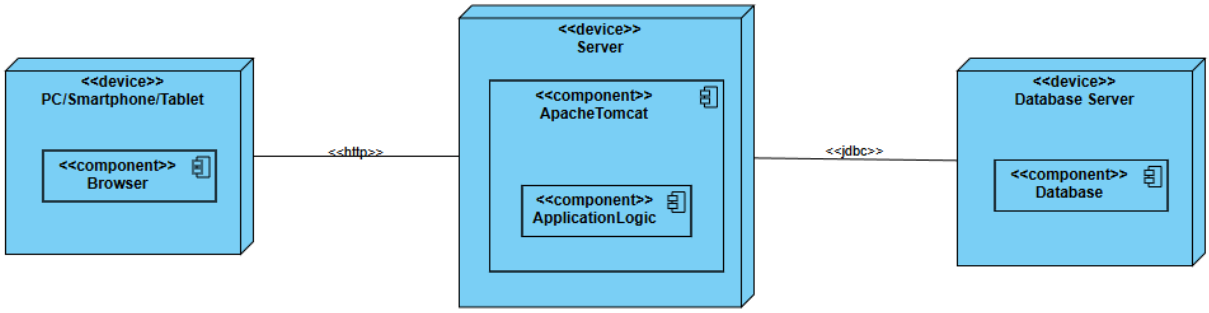
DataStorageLayer



Gestione Autenticazione	Comprende tutte le funzionalità relative al Login, Autenticazione e Password dimenticata
Gestione Profili	Comprende tutte le funzionalità relative alle informazioni personali e alle loro modifiche
Gestione Prenotazioni	Comprende tutte le funzionalità relative alla aggiunta, all'approvazione e rifiuto delle singole prenotazioni
Gestione Catalogo Prenotazioni	Comprende tutte le funzionalità relative al catalogo delle prenotazioni
Gestione Risorse	Comprende tutte le funzionalità relative all'aggiunta, rimozione e modifica allo stato delle risorse
Gestione turnistica	Comprende tutte le funzionalità relative all'aggiunta, modifica e rimozione dei turni
Gestione Premi	Comprende tutte le funzionalità relative all'aggiunta, rimozione, modifica dei singoli premi
Gestione Catalogo Premi	Comprende tutte le funzionalità relative al catalogo dei premi
Gestione Servizi	Comprende tutte le funzionalità relative alla visualizzazione dei servizi

3.3-Mappatura Hardware/Software

Il sistema utilizza un'architettura Client/Server. Il Web Server è rappresentato da Apache Tomcat 11 ed è situato su una singola macchina, la logica del sistema è composta da Java Servlet mentre l'interfaccia utente è realizzata tramite l'utilizzo di pagine JSP. Il client è rappresentato dal Web Browser utilizzato dall'utente. La comunicazione tra i nodi viene effettuata tramite richieste e risposte http tra client e server mentre tra server e database verranno effettuate delle query JDBC.



3.4-Gestione dei dati persistenti

I dati saranno gestiti tramite un DBMS MySQL. La scelta dei dati persistenti è stata effettuata nel RAD che include le seguenti entità principali.

Premio

Nome	Tipo	Vincoli	Chiave
IDPremio	Int	NOT NULL	PRIMARY KEY
Nome	Varchar(52)	NOT NULL	
Descrizione	Varchar(52)	NULL	
ValoreTicket	Int	NOT NULL	

Staff

Nome	Tipo	Vincoli	Chiave
IDStaff	Int	NOT NULL	PRIMARY KEY
Nome	Varchar(52)	NOT NULL	
Cognome	Varchar(52)	NOT NULL	
Email	Varchar(52)	NOT NULL	
Password	Varchar(52)	NOT NULL	
Ruolo	Enum	NOT NULL	

Cliente

Nome	Tipo	Vincoli	Chiave
IDCliente	Int	NOT NULL	PRIMARY KEY
Nome	Varchar(52)	NOT NULL	
Cognome	Varchar(52)	NOT NULL	
Email	Varchar(52)	NOT NULL	
Password	Varchar(52)	NOT NULL	
PuntiTicket	Int	NOT NULL	

Manager

Nome	Tipo	Vincoli	Chiave
IDManager	Int	NOT NULL	PRIMARY KEY
Nome	Varchar(52)	NOT NULL	
Cognome	Varchar(52)	NOT NULL	
Email	Varchar(52)	NOT NULL	
Password	Varchar(52)	NOT NULL	

Turno

Nome	Tipo	Vincoli	Chiave
IDTurno	Int	NOT NULL	PRIMARY KEY
Data	TimeStamp	NOT NULL	
FasciaOraria	Enum	NOT NULL	
IDStaff	Int	NOT NULL	FOREIGN KEY
IDManager	Int	NOT NULL	FOREIGN KEY

Risorsa

Nome	Tipo	Vincoli	Chiave
IDRisorsa	Int	NOT NULL	PRIMARY KEY
Stato	TinyInt	NOT NULL	
Capacità	Int	NOT NULL	
IDServizio	Int	NOT NULL	FOREIGN KEY

Servizio

Nome	Tipo	Vincoli	Chiave
IDServizio	Int	NOT NULL	PRIMARY KEY
Nome	Varchar(52)	NOT NULL	
Stato	TinyInt	NOT NULL	

Prenotazione

Nome	Tipo	Vincoli	Chiave
IDPrenotazione	Int	NOT NULL	PRIMARY KEY
Data	TimeStamp	NOT NULL	
Orario	TimeStamp	NOT NULL	
Stato	Enum	NOT NULL	
Partecipanti	Varchar(52)	NULL	
IDServizio	Int	NOT NULL	FOREIGN KEY
IDRisorsa	Int	NOT NULL	FOREIGN KEY
IDCliente	Int	NOT NULL	FOREIGN KEY
IDStaff	Int	NULL	FOREIGN KEY

3.5-Controllo Accesso e Sicurezza

3.5.1 User Model

- 4 attori (Ospite, Cliente, Staff, Manager)
- 5 Classi (User, Prenotazione, Turnistica, Premio, Servizio)
- 38 Use Cases del RAD

3.5.2 Access Matrix Implementation

- Tipo: Access Control List (ACL)
- Struttura: (actor, operation) pairs per class
- Default Policy: DENY (se non trovata in ACL)

3.5.3 Authentication Scheme

- Metodo: E-mail + Password
- Storage of password: hash in User table
- Trasporto: HTTP

3.5.4 Struttura Tabella

- Righe: 4 Actors (Guest, Cliente, Staff, Manager)
- Colonne: 5 Classes (User, Prenotazione, Turnistica, Premio, Servizio)
- Celle: ✅ ALLOWED o ❌ DENIED
- Check: (actor, operation) ∈ Class.ACL?

Legenda
✅ = ALLOWED - (actor, operation) ∈ Class.ACL
❌ = DENIED - (actor, operation) ∉ Class.ACL
C=Create, R=Read, U=Update, D=Delete, A*=Accept, R*=Reject

USER CLASS

UC	Use Case	Operation	Guest	Cliente	Staff	Manager	(actor,op) Check
UC1	Registrazione	CREATE	✅	❌	❌	❌	(Guest,CREATE)∈ACL? ✅
UC2	Login	READ	✅	✅	✅	✅	(Guest,READ)∈ACL? ✅
UC3	Logout	DELETE_OWN	❌	✅	✅	✅	(Guest,DELETE_OWN)∈ACL? ❌
UC4	Visualizza Profilo	READ_OWN	❌	✅	✅	✅	(Guest,READ_OWN)∈ACL? ❌
UC5	Modifica Profilo	UPDATE_OWN	❌	✅	✅	✅	(Guest,UPDATE_OWN)∈ACL? ❌
UC6	Recupero Password	RESET_TOKEN	✅	✅	✅	✅	(Guest,RESET_TOKEN)∈ACL? ✅
UC7	Cambio Password	UPDATE_OWN	❌	✅	✅	✅	(Guest,UPDATE_OWN)∈ACL? ❌

PRENOTAZIONE CLASS

UC	Use Case	Operation	Guest	Cliente	Staff	Manager	(actor,op) Check
UC8	Prenotazione Bowling	CREATE	❌	✅	❌	❌	(Cliente,CREATE)∈ACL? ✅
UC9	Prenotazione Go-Kart	CREATE	❌	✅	❌	❌	(Cliente,CREATE)∈ACL? ✅
UC10	Prenotazione Biliardo	CREATE	❌	✅	❌	❌	(Cliente,CREATE)∈ACL? ✅
UC11	Modifica Prenotazione	UPDATE_OWN	❌	✅	❌	❌	(Cliente,UPDATE_OWN)∈ACL? ✅
UC11b	Cancella Prenotazione	DELETE_OWN	❌	✅	❌	❌	(Cliente,DELETE_OWN)∈ACL? ✅
UC12	Rifiuta Bowling	REJECT	❌	❌	✅	❌	(Staff,REJECT)∈ACL? ✅
UC13	Rifiuta Go-Kart	REJECT	❌	❌	✅	❌	(Staff,REJECT)∈ACL? ✅
UC14	Rifiuta Biliardo	REJECT	❌	❌	✅	❌	(Staff,REJECT)∈ACL? ✅
UC15	Accetta Bowling	ACCEPT	❌	❌	✅	❌	(Staff,ACCEPT)∈ACL? ✅

UC16	Accetta Go-Kart	ACCEPT	❌	❌	✅	❌	(Staff,ACCEPT)∈ACL? ✅
UC17	Accetta Biliardo	ACCEPT	❌	❌	✅	❌	(Staff,ACCEPT)∈ACL? ✅
UC18	Assegna Pista Bowling	ASSIGN_RESOURCE	❌	❌	✅	❌	(Staff,ASSIGN)∈ACL? ✅
UC19	Assegna Pista Go-Kart	ASSIGN_RESOURCE	❌	❌	✅	❌	(Staff,ASSIGN)∈ACL? ✅
UC20	Assegna Tavolo Biliardo	ASSIGN_RESOURCE	❌	❌	✅	❌	(Staff,ASSIGN)∈ACL? ✅

TURNISTICA CLASS

UC	Use Case	Operation	Guest	Cliente	Staff	Manager	(actor,op) Check
UC21	Crea Turnistica Bowling	CREATE	❌	❌	❌	✅	(Manager,CREATE)∈ACL? ✅
UC22	Crea Turnistica Go-Kart	CREATE	❌	❌	❌	✅	(Manager,CREATE)∈ACL? ✅
UC23	Crea Turnistica Biliardo	CREATE	❌	❌	❌	✅	(Manager,CREATE)∈ACL? ✅
UC24	Modifica Turnistica Bowling	UPDATE	❌	❌	❌	✅	(Manager,UPDATE)∈ACL? ✅
UC25	Modifica Turnistica Go-Kart	UPDATE	❌	❌	❌	✅	(Manager,UPDATE)∈ACL? ✅
UC26	Modifica Turnistica Biliardo	UPDATE	❌	❌	❌	✅	(Manager,UPDATE)∈ACL? ✅
UC27	Visualizza Turnistica Bowling	READ_OWN	❌	❌	✅	✅	(Staff,READ_OWN)∈ACL? ✅
UC28	Visualizza Turnistica Go-Kart	READ_OWN	❌	❌	✅	✅	(Staff,READ_OWN)∈ACL? ✅
UC29	Visualizza Turnistica Biliardo	READ_OWN	❌	❌	✅	✅	(Staff,READ_OWN)∈ACL? ✅

PREMIO CLASS

UC	Use Case	Operation	Guest	Cliente	Staff	Manager	(actor,op) Check
UC30	Aggiungi Premio	CREATE	❌	❌	✅	❌	(Staff,CREATE)∈ACL? ✅
UC31	Rimuovi Premio	DELETE	❌	❌	✅	❌	(Staff,DELETE)∈ACL? ✅
UC32	Modifica Premio	UPDATE	❌	❌	✅	❌	(Staff,UPDATE)∈ACL? ✅
UC33	Catalogo Premi	READ	✅	✅	✅	✅	(Guest,READ)∈ACL? ✅

SERVIZIO CLASS

UC	Use Case	Operation	Guest	Cliente	Staff	Manager	(actor,op) Check
UC34	Servizio Bowling	READ	✅	✅	✅	✅	(Guest,READ)∈ACL? ✅
UC35	Servizio Go-Kart	READ	✅	✅	✅	✅	(Guest,READ)∈ACL? ✅
UC36	Servizio Biliardo	READ	✅	✅	✅	✅	(Guest,READ)∈ACL? ✅
UC37	Abilita Servizio	UPDATE_STATUS	❌	❌	✅	❌	(Staff,UPDATE)∈ACL? ✅
UC38	Disabilita Servizio	UPDATE_STATUS	❌	❌	✅	❌	(Staff,UPDATE)∈ACL? ✅

3.6-Controllo software globale

Poiché UStrike! è un'applicazione web, il Web Server ha il compito di gestire le richieste provenienti dai client. Quando arriva una richiesta, il server la inoltra alla Java Servlet appropriata, che si occupa di elaborarla, interagendo se necessario con il modello applicativo. Una volta ottenuti i dati necessari, il server provvede a generare la pagina JSP, la quale verrà trasformata in HTML e infine mostrata all'utente.

Architettura del Controllo Software

Il controllo del software in UStrike! è implementato mediante un'architettura web three-tier basata su:
• Web Server (Tomcat)
• Application Logic Layer (Servlet Java)
• Database Layer (MySQL)

3.7-Condizioni di boundary

Le condizioni di boundary riguardanti accensione e spegnimento del sistema sia da lato server vengono qui di seguito riportate:

- Avvio del sistema: Con l'accensione del sistema vengono avviati anche web server e database già popolato.
- Spegnimento del sistema: Quando il software viene chiuso si ha la terminazione del sistema con il logout dell'utente. Per quanto riguarda la terminazione del server essa viene delegata al manager. Una volta terminato il server nessun client potrà più accedere e il server dovrà garantire la terminazione di tutte le transazioni in corso (in caso di terminazione imprevista).

4.0-Servizi dei sottosistemi

Gestore Autenticazione	
Login	Permette all'utente di accedere al sistema e svolgere tutte le operazioni a lui consentite a patto che acceda con le giuste credenziali.
Registrazione	Consente all'utente di registrare un account nel sistema.
Password dimenticata	Consente all'utente di accedere all'account con una password provvisoria fornita dal sistema.

Gestione Profili	
Visualizza informazioni personali	Permette all'utente di visualizzare le proprie informazioni personali.
Modifica informazioni personali	Permette all'utente di modificare le proprie informazioni personali quali nome, cognome e e-mail.
Logout	Permette all'utente di uscire dall'account presente in quel momento sul sistema lato client.
Modifica password	Permette all'utente di modificare la propria password inserita nel sistema.

Gestione Prenotazioni	
Effettua prenotazione	Permette all'utente di effettuare una prenotazione per un dato servizio ad un dato giorno ed orario.
Accetta prenotazione	Permette all'utente dello staff di accettare una prenotazione effettuata da un utente.
Rifiuta prenotazione	Permette all'utente dello staff di rifiutare una prenotazione effettuata da un utente.
Visualizza prenotazione	Permette all'utente di visualizzare una prenotazione specifica in maniera dettagliata.

Gestione Catalogo Prenotazioni	
Visualizza catalogo prenotazioni	Permette all'utente di vedere lo storico delle prenotazioni effettuate e in quale stato sono.
Ordina catalogo prenotazioni	Permette all'utente staff di filtrare lo storico delle prenotazioni

Gestione Risorse	
Visualizza risorse	Permette all'utente staff di visualizzare le risorse e in quale stato si trovano.
Disabilita risorsa	Permette all'utente staff di rendere lo stato di una risorsa "disabilitato".
Abilita risorsa	Permette all'utente staff di rendere lo stato di una risorsa "abilitato".

Gestione Turnistica	
Visualizza tabella turni settimanale	Permette di vedere la tabella turni settimanale.
Assegna turno	Permette allo user manager di assegnare un turno a uno user staff.

Gestione Premi	
Aggiungi premi	Permette di aggiungere un premio nel sistema.
Rimuovi premio	Permette di rimuovere un premio dal sistema.
Visualizza premio	Permette di vedere un singolo premio nel dettaglio.
Modifica premio	Permette di modificare le informazioni relative ad un determinato premio.

Gestione Catalogo Premi	
Visualizza catalogo premi	Permette all'utente di visualizzare il catalogo dei premi con i loro prezzi in ticket.

Gestione Servizi	
Visualizza servizio	Permette all'utente di visualizzare un servizio.
Disabilita servizio	Permette all'utente staff di rendere lo stato di un servizio "disabilitato".
Abilita servizio	Permette all'utente staff di rendere lo stato di un servizio "abilitato".

5.0-Glossario

MySQL: database service utilizzato da UStrike! per la gestione dei dati.
DBMS: è un software che permette di creare, gestire, e interrogare database in modo efficiente e sicuro.
Client: componente che accede ai servizi del server.
Server: componente che fornisce servizi ai client.
JSP: tecnologia che permette di creare pagine web dinamiche.
Servlet: programma java eseguito su un web server per elaborare richieste da parte dei client.