Università degli Studi di Salerno
Corso di Ingegneria del Software

UStrike!

Problem Statement

Versione 1.1
Data: 13/10/2025

Informazioni sul Progetto

Progetto: UStrike!
Documento: Problem Statement
Versione: 1.1
Data: 13/10/2025

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
12/10/2025	1.0	Problem Statement, scenario, requisiti funzionali	Russo Serena, Stefanile Andrea, Valito Marcello
13/10/2025	1.1	Problem Statement, requisiti non funzionali, target environment e deadlines	Russo Serena, Stefanile Andrea, Valito Marcello

Indice

1 Problem Domain

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 modo errato), risulta inefficiente durante le ore di punta e offre un'esperienza 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.

2 Scenari

Di seguito sono riportati alcuni scenari d'uso che illustrano le interazioni chiave con il sistema.

2.1 Scenario: `prenotazioneSerataGruppo`

- **Istanze di attori partecipanti:** `giulia:Cliente`, `marco:Staff`
- **Flusso degli eventi:**
  1. Giulia vuole organizzare un'uscita di gruppo e accede al sistema UStrike! per prenotare.
  2. Dopo aver creato un account ed effettuato il login, visualizza i servizi disponibili e sceglie due piste da bowling per sabato sera.
  3. Completa la prenotazione e riceve una notifica con lo stato "in attesa".
  4. Marco, membro dello staff, accede alla sua dashboard e visualizza la nuova richiesta.
  5. Verifica la disponibilità e conferma la prenotazione.
  6. Giulia riceve la notifica di conferma e il suo stato di prenotazione viene aggiornato a "confermata".
  7. All'arrivo, Marco recupera facilmente la prenotazione dal sistema, garantendo un check-in rapido.

2.2 Scenario: `gestioneTurniSettimanali`

- **Istanze di attori partecipanti:** `elena:Manager`, `marco:Staff`
- **Flusso degli eventi:**
  1. Elena, la manager, deve pianificare i turni di lavoro della settimana.
  2. Accede al sistema con il suo account ed entra nella sezione di gestione delle turnistiche.
  3. Crea i turni, specificando orari e ruoli, e li assegna ai membri dello staff, tra cui Marco.
  4. Una volta pubblicato l'orario, Marco accede al suo profilo.
  5. Visualizza i suoi turni di lavoro assegnati per la settimana tramite l'apposita funzione.

2.3 Scenario: `eliminazionePrenotazione`

- **Istanze di attori partecipanti:** `matteo:Cliente`, `sara:Staff`
- **Flusso degli eventi:**
  1. Matteo ha una prenotazione per una pista da bowling ma deve cancellarla a causa di un imprevisto.
  2. Accede al suo account UStrike! e visualizza le sue prenotazioni attive.

3. Seleziona la prenotazione in questione e sceglie l'opzione per annullarla.
4. Sara, membro dello staff, riceve una notifica nel pannello di gestione delle prenotazioni riguardo alla cancellazione.
5. Il sistema aggiorna automaticamente lo stato della prenotazione a "annullata" e libera lo slot, rendendolo nuovamente disponibile per altri clienti.
6. Matteo riceve una conferma dell'avvenuto annullamento.

2.4 Scenario: `aggiornamentoPremiCatalogo`

- **Istanze di attori partecipanti:** `marco:Staff`
- **Flusso degli eventi:**
  1. A Marco viene chiesto di aggiornare il catalogo dei premi riscattabili con i ticket vinti nell'area arcade.
  2. Accede al sistema con il suo profilo staff.
  3. Entra nella sezione "Gestione pagina premi con ticket".
  4. Rimuove un premio esaurito e aggiunge un nuovo articolo, caricando una descrizione e impostando il valore in ticket necessario per il riscatto.
  5. Salva le modifiche, che sono immediatamente visibili ai clienti.

3 Requisiti Funzionali

Sulla base del documento di progetto, i requisiti funzionali sono suddivisi per tipologia di utente.

3.1 Funzionalità Comuni

- Il sistema deve permettere a tutti gli utenti di creare un account ed effettuare il login.
- Il sistema deve consentire a tutti gli utenti di visualizzare i servizi offerti dall'arcade e le relative informazioni.

3.2 Funzionalità per i Clienti

- Il sistema deve permettere ai clienti di prenotare una pista da bowling, un tavolo da biliardo o una pista go-kart, scegliendo data e orario.
- Il sistema deve consentire ai clienti di consultare lo stato (confermata, in attesa, annullata) e i dettagli delle proprie prenotazioni.

3.3 Funzionalità per lo Staff

- Il sistema deve permettere allo staff di visualizzare, confermare, modificare o annullare le prenotazioni dei clienti.
- Il sistema deve consentire allo staff di aggiornare e gestire il catalogo dei premi riscattabili con i ticket.
- Il sistema deve permettere ai membri dello staff di visualizzare i propri turni di lavoro.

3.4 Funzionalità per il Manager Staff

- Il sistema deve permettere al manager di creare, assegnare e modificare i turni di lavoro per tutti i membri dello staff.

4 Requisiti Non Funzionali

La versione 1.0 del progetto non specificava requisiti non funzionali. Di seguito, vengono definiti i requisiti di usabilità, affidabilità, performance, supportabilità e legalità.

4.1 Usabilità

- Autonomia dell'utente: La piattaforma 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.
- Accessibilità: Tutte le pagine pubbliche e di prenotazione devono essere facilmente raggiungibili. I contenuti devono essere accessibili tramite tastiera; i contrasti devono assicurare una lettura agevole dei testi e degli elementi interattivi; le dimensioni dei target di tocco devono essere adeguate per l'uso su dispositivi mobili.

4.2 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, come email, SMS o pagamenti, dovesse guastarsi, le funzionalità principali rimangono operative e le richieste verso quel servizio vengono ripetute fino a due volte. In caso di un blocco totale, il servizio si riattiva automaticamente, assicurando che le operazioni non rimangano "a metà": ogni prenotazione sarà o confermata o annullata dopo un processo di riconciliazione che si occupa di sistemare le transazioni interrotte.
- Sicurezza: Il sistema è progettato per prevenire situazioni problematiche, come l'overbooking, evitando di avere due prenotazioni sullo stesso servizio nella stessa fascia oraria. Se la disponibilità è incerta, la prenotazione viene rifiutata piuttosto che confermata in modo ambiguo.
- Protezione: 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. Il traffico tra browser, applicazione e API è forzato su HTTPS e utilizza protocolli TLS per cifrare i dati sensibili in transito. Inoltre, le interazioni con il database utilizzano prepared statements, evitando la concatenazione di stringhe con input dell'utente per prevenire SQL injection. L'applicazione adotta un modello a ruoli per il controllo degli accessi, consentendo ogni funzionalità solo se esplicitamente autorizzata per il ruolo dell'utente (cliente, staff, manager).

4.3 Performance

- Velocità delle azioni principali: Quando un cliente apre la disponibilità dei servizi (come bowling, biliardo o go-kart) o decide di confermare o annullare una prenotazione, la pagina deve rispondere entro 1,5 secondi.
- Carico elevato di utenti: Il sistema deve essere in grado di gestire fino a 200 persone contemporaneamente, senza mostrare messaggi di errore e rispettando i tempi indicati nel punto precedente.
- Prestazioni garantite nei momenti di punta: Se si supera la capacità massima, il sistema deve rifiutare immediatamente nuove richieste, evitando di rallentare il servizio per tutti gli utenti già attivi.

4.4 Supportabilità

- Compatibilità Display: L'interfaccia è stata progettata per essere utilizzata su schermi che vanno da 360 px a 1920 px, coprendo così tutti i dispositivi senza compromettere le funzionalità della piattaforma.
- Compatibilità Browser: Il sistema funziona senza intoppi su browser supportati - come Chrome, Firefox, Edge e Safari - assicurando che le operazioni principali siano sempre accessibili e coerenti.

4.5 Legalità

- Diritti d'autore del progetto: Il progetto UStrike! è destinato unicamente alla valutazione del corso. Il codice e la documentazione rimangono di proprietà degli autori e non possono essere riutilizzati o ridistribuiti al di fuori dell'esame.
- Proprietà dei dati: I dati dimostrativi, come prenotazioni, clienti e incassi di prova, sono di proprietà del titolare dell'istanza e possono essere esportati in formati comuni (CSV/JSON).

5 Target Environment

Tutti i membri del team lavorano su Windows 11 utilizzando IntelliJ IDEA con JDK 21; l'applicazione viene avviata direttamente dall'IDE tramite una normale configurazione di run. Il backend si appoggia a un'istanza locale di MySQL Server 8.0, amministrata con MySQL Workbench. Lo schema del database, le migrazioni SQL e i dati di demo sono tracciati nel repository e applicati mediante script dedicati, così gli ambienti restano sempre allineati. La configurazione è esterna e accentrata in un solo file `.env`, garantendo che password e altre credenziali riservate non siano mai parte integrante del codice.

6 Deadlines

Di seguito è riportata la pianificazione delle scadenze di progetto:

Data	**Documento / Attività**
7 ottobre	Start-up del progetto, creazione repository GitHub, kick-off meeting
14 ottobre	Problem Statement, scenari, requisiti funzionali e non funzionali, target environment
28 ottobre	Requisiti e casi d'uso
11 novembre	Requirements Analysis Document
25 novembre	System Design Document
16 dicembre	Piano di test e specifica interfacce dei moduli del sistema
31 dicembre	Esecuzione dei test
15 gennaio	Object design e implementazione
20 gennaio	Consegna finale del progetto