Requirement Engineering - CMMTickets

Caprinali Michele 1087210
Mazzoleni Gabriel 1086530
Mazzoleni Raphael 1086531

Indice:

1- Introduzione

Il seguente documento ha lo scopo di definire quali sono i requisiti del progetto CMMTickets, dei modi in cui sono stati elicitati, definendo per importanza quali sono necessari da implementare, utilizzando la tecnica del MoSCoW. Il documento permetterà di avere una visione chiara e lucida del progetto che si vuole sviluppare e delle funzionalità implementate.

1.1 - Obiettivi e scopo

CMMTickets ha come scopo lo sviluppo di un'applicazione desktop per la gestione di acquisto e vendita di biglietti per eventi o partite. L'obiettivo è offrire un sistema efficace e intuitivo per utenti finali e amministratori, garantendo funzionalità come registrazione utenti, selezione di eventi, acquisto e consultazione di biglietti, e creazione/gestione di eventi da parte degli amministratori del sistema.
Gli obiettivi del sistema sono:

- fornire un sistema per la vendita e acquisto di biglietti
- garantire strumenti di gestione per gli amministratori, con creazione, modifica e configurazione di eventi
- consentire agli utenti di visualizzare informazioni dettagliate sui biglietti acquistati.

1.2 - Definizione, acronimi e termini

Termini e definizioni:

- Utente registrato: persona che utilizza il sistema per acquistare biglietti
- Amministratore: utente con privilegi avanzati per la gestione del sistema e degli eventi
- Evento: attività come concerti, partite sportive o gare
- Evento a posto numerato: attività con un posto numerato disponibile per ogni settore
- Evento a posto unico: attività senza la necessità di avere un posto numerato
- Settore: spazio all'interno di un luogo con un totale di posti disponibili per ognuno di esso

Acronimi:

- GUI: Graphical User Interface
- DBMS: DataBase Management System

1.3 - Riferimenti

- documento del piano di progetto (project plan) fornito dal team di sviluppo
- standard di sviluppo del software: Pascal Case, Camel Case, Snake Case

—----------

2- Descrizione generale

2.1 - Prospettiva del prodotto

Il sistema sarà sviluppato come un'applicazione desktop incrementale, con iterazioni che consentono verifiche continue. Gli utenti potranno selezionare eventi, acquistare biglietti e accedere ai dettagli, mentre gli amministratori avranno strumenti per configurare eventi e gestire i biglietti

2.2 - Funzionalità principali

1. Per gli utenti:
    - registrazione/login
    - visualizzazione eventi disponibili ed eventi in arrivo in futuro secondo la data imposta
    - acquisto di biglietti (rispettando i limiti per evento)
    - accesso ai dettagli dei biglietti acquistati

2. Per gli amministratori:
    - creazione/modifica di eventi
    - creazione/modifica dei luoghi disponibili
    - configurazione dei settori
    - gestione prezzi dei biglietti

2.3 - Caratteristiche degli utenti

- Utenti finali: accedono al sistema per acquistare i biglietti
- Amministratori: configurano e gestiscono gli eventi

2.4 - Vincoli

- Tecnologici: utilizzo di SQLite come database e Eclipse IDE per sviluppo
- Temporali: consegna del progetto completa entro il 24 gennaio 2025
- Sicurezza: accesso amministratore protetto da credenziali.

—----------

3- Requisiti specifici

3.1 - Elicitazione dei requisiti

In questa sezione del documento, ci occuperemo di definire i modi in cui i requisiti del progetto vengono elicitati in relazione al problema da risolvere.
I metodi con cui i requisiti vengono estratti sono:

- Analisi basata sullo scenario:
    - Utente → si suppone che un utente che voglia acquistare dei biglietti relativi ad un evento, debba accedere al sito. Se non è registrato deve effettuare la registrazione, se lo è basta accedere con le proprie credenziali. Una volta effettuato l'accesso, l'utente può navigare nell'applicazione scorrendo la lista di eventi disponibili, e successivamente quando ne trova uno di suo gradimento può procedere all'acquisto selezionando il settore/posto e il numero di biglietti che vuole acquistare. Infine può accedere alla sezione del carrello in cui trova tutti i biglietti a cui è interessato ed acquistarli procedendo con il pagamento.
    - Admin → gli admin dell'applicazione possono accedere al sistema, tramite le proprie credenziali, e gestire gli eventi, inserendone di nuovi, modificare il luogo o il numero di biglietti per evento e talvolta cancellarli.
- Derivazione da un sistema esistente: si è preso spunto dai maggiori siti di vendita di biglietti di eventi, inclusa la gestione dei settori e l'acquisto da parte degli utenti.
- Prototipazione: la prototipazione nello sviluppo del nostro sistema permette di avere ogni volta una versione sempre aggiornata e corretta, con nuove funzionalità implementate, questo perchè non è possibile avere una versione corretta e finita sin dal primo sviluppo. Tramite questo metodo avremo inoltre la possibilità di avere un prodotto eseguibile, al quale potremo poi effettuare delle migliorie

3.2 - Requisiti

In questa sezione del documento ci occuperemo di definire i requisiti che permettono al progetto di risolvere il problema impostato. I requisiti definiti nel sistema per la sua implementazione sono:

1. Implementazione della fase di accesso/registrazione da parte di utenti/admin del sistema
2. Fase di login
    a. per gli utenti registrati, accesso tramite nome utente e password scelti durante la fase di registrazione
    b. per gli admin accesso tramite nome utente e password pre assegnati in fase di configurazione del sistema, che permetterà di entrare nell'area riservata a loro
3. Fase di registrazione (per utenti non ancora registrati)
    a. Inserimento di username, password, codice fiscale, telefono, email
    b. Controllo del sistema che il codice fiscale sia di 16 caratteri, il telefono di 10 caratteri e che l'email sia nel formato corretto
    c. I dati forniti sono inseriti in un embedded database
    d. Il database deve mantenere i dati integri e consistenti
4. Accesso admin all'area riservata
    a. Possibilità di aggiungere un evento inserendo il nome, la data, l'ora, numero max di biglietti a persona, data di inizio vendita, il luogo dell'evento e gestione dei settori.
    b. Aggiunta dei luoghi disponibili in cui può essere effettuato un evento sportivo o concerto, indicando il nome, città, indirizzo e l'immagine.
    c. Modifica ed eliminazione di un evento, selezionandolo dalla tabella
    d. Modifica ed eliminazione di un luogo, selezionandolo dalla tabella
    e. La modifica della gestione dei settori in modifica evento
    f. Riepilogo degli utenti che hanno acquistato i biglietti
5. Accesso utente al sistema
    a. Visualizzazione di tutti gli eventi, disponibili e non, con la possibilità di filtrare tramite nome o luogo, e seleziona il numero biglietto di un evento
    b. Visualizzazione del carrello con tutti i biglietti di eventi selezionati in precedenza
    c. Visualizzazione di tutti i gli ordini effettuati da un utente
    d. Quando viene acquistato uno o più biglietti non c'è possibilità di rimborso

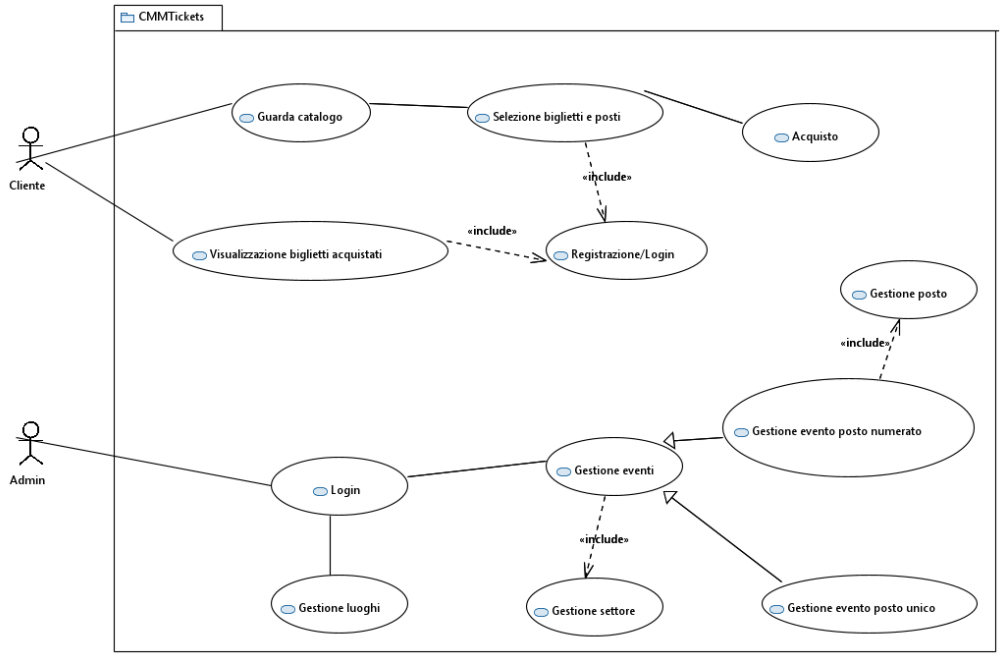
3.3 - MoSCoW

I requisiti elencati nella sezione precedente sono suddivisi in base alla loro priorità di implementazione:

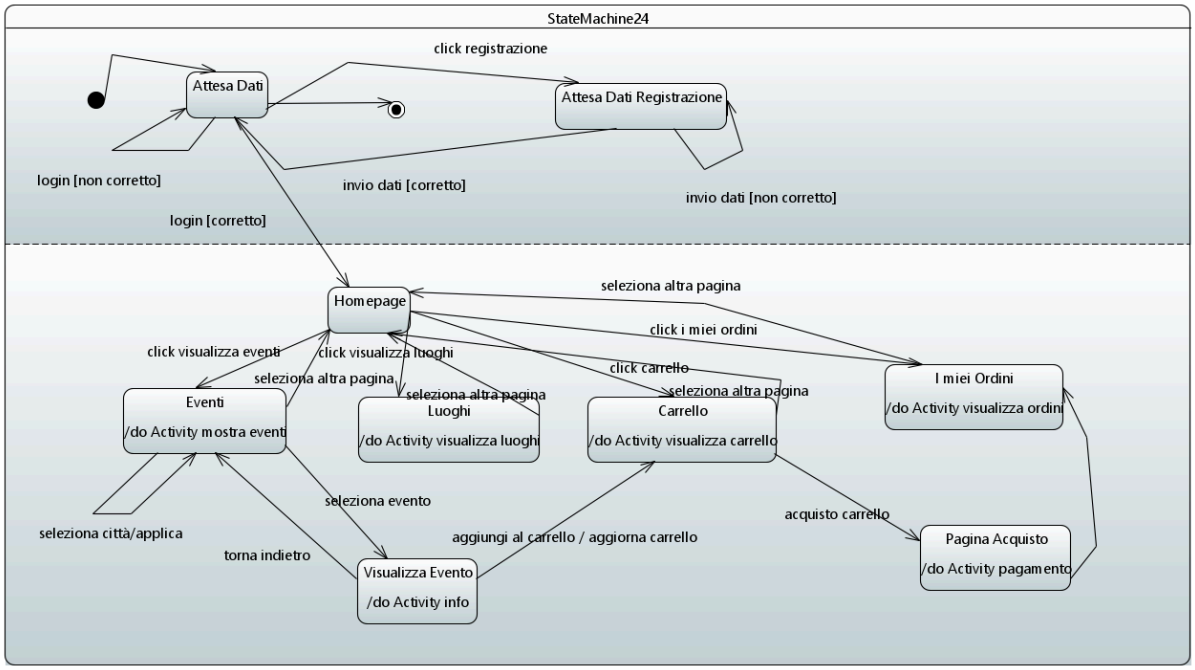
MUST HAVE	SHOULD HAVE	COULD HAVE	WON'T HAVE
1			
2.a, 2.b			
3.a, 3.d	3.b, 3.c		
4.a	4.b, 4.c, 4.d	4.e	4.f
5.a, 5.b	5.c		5.d

3.4 - UML

Use Case Diagram:



State Machine Diagram:



4- Software Quality

Per quanto riguarda i fattori di qualità del sistema si è deciso di seguire con impegno alcuni criteri e fattori di qualità stabiliti nel modello ISO 9126. Le caratteristiche qualitative di questo modello in cui si è posta maggiore attenzione sono i seguenti:

- Affidabilità: coerenza del sistema nel soddisfare, tramite un insieme di funzioni, attività e obiettivi dell'utente.
- Interoperabilità: capacità del software di interagire con uno o più sistemi specificati.
- Usabilità: capacità del software di essere comprensibile ed utilizzato in maniera semplice ed efficace da parte dell'utente.
- Manutenibilità: possibilità di garantire che il software possa essere modificato dopo la consegna, in caso di rilevamento di errori o guasti. Inoltre, si vuole che il software sia di facile lettura e comprensibile per i programmatori.
- Efficienza: si vuole che il software prodotto possa fornire prestazioni adeguate, rispetto alla quantità di risorse utilizzate