Università degli Studi di Salerno
Corso di Ingegneria del Software

BeVoyager
SDD
Versione 1.4

Data: 07/01/2017

Partecipanti:

Nome	Matricola
Donato Tiano	0512102916
Alessandro Longobardi	0512102910
Paolo Zirpoli	0512102862
Salvatore Ruggiero	0512103002

Revision History

Data	Versione	Descrizione	Autore
07/01/2017	1.0	Introduzione, purpose of the system.	
07/01/2017	1.1	Design goals.	
07/01/2017	1.2	Definitions, acronyms and abbreviations.	
07/01/2017	1.3	Current Software Architecture.	
07/01/2017	1.4	Proposal Software Architecture: introduction.	

Sommario

1. Introduction

Il progetto che si vuole proporre è una piattaforma di viaggi, chiamata BeVoyager, la quale nasce dall'idea di un sito all-in-one, cioè una piattaforma dove sia possibile ritrovare tutto quello che all'utente serve per poter organizzare un viaggio in maniera più efficiente. Sul sito saranno unite tutte le comuni ricerche che servono all'organizzazione del viaggio: dalla formazione di un gruppo di persone alla creazione di un luogo, dalla ricerca di itinerari ai feedback su utenti, luoghi e itinerari.

1.1 Purpose of the system

Lo sviluppo di tale progetto è stato pensato al fine di fornire all'utente la possibilità di organizzare viaggi e trovare informazioni su di essi. BeVoyager permetterà all'utente di:

- creare viaggi pubblici, dove ogni iscritto alla piattaforma potrà partecipare e formare un gruppo di persone interessate allo stesso itinerario;
- creare viaggi privati, dove sarà lo stesso utente ad isrivere le persone a tale viaggio in modo da creare un gruppo chiuso di persone conosciute e scegliere insieme l'itinerario di cui il viaggio sarà composto.
- diventare un Tour Operator, ovvero utilizzare la piattaforma per la creazione di viaggi non modificabili da rendere disponibili agli utenti. Dalla vendita di tali viaggi, il Tour Operator ne guadagnerà la percentuale pervenuta.
- lasciare feedback, su persone, itinerari e luoghi, in modo tale da facilitare le ricerche da parte di altre persone sull'utilizzo di un itinerario, sulla scelta di un luogo e sull'aggiunta o meno di un altro utente all'organizzazione di un viaggio.

1.2 Design goals

1.2.1 Criteri di sviluppo e mantenimento

Estensibilità:

- Il sistema verrà progettato in modo tale da facilitare l'introduzione di nuove funzionalità per l'utente. Essendo una web application, verranno usati i linguaggi di scripting come HTML, Javascript, ed il back-end sarà implementato in Java.

Numero minimo di errori:

- Il sistema verrà implementato dando particolare attenzione a minimizzare l'occorrenza di errori durante l'utilizzo.

Interfacciamento con oggetti già esistenti:

- Il sistema deve trattare componenti off-the-shelf come l'utilizzo di librerie Java già esistenti.

Velocità di implementazione:

- Il sistema verrà progettato in maniera tale da massimizzare la velocità di implementazione dello stesso, cercando di garantire uno sviluppo che proceda senza difficoltà

Mantenibilità:

- Il codice sorgente deve essere scritto e documentato con attenzione, permettendo a nuovi programmatori che verranno incaricati di mantenerlo di poter familiarizzare con lo stesso in un periodo di tempo rapido.

Portabilità:

- La piattaforma deve essere aperta all'aggiunta di funzionalità appartenenti ad altre piattaforme. In pratica si deve fare in modo che non venga modificata in maniera lenta nel caso in cui un'altra piattaforma ne richiede l'uso.
- Il sistema deve utilizzare funzionalità esterne in modo tale da non modificare quelle già esistenti. In pratica deve interfacciarsi a sistemi diversi senza subire modifiche a basso livello (codice).

1.2.2 Criteri di affidabilità

Robustezza:

- Il sistema garantisce un funzionamento adeguato anche se vengono inseriti input non validi (fault-tolerance).
- Il sistema deve saper gestire errori che possono verificarsi durante l'uso quotidiano da parte dell'utente.

Affidabilità:

- Il sistema deve garantire che le operazioni effettuate dagli utenti vadano a buon fine rispettando le aspettative degli stessi e soprattutto deve essere coerente con le scelte fatte da essi, in quanto le scelte dell'utenza non devono essere altera da un funzionamento

inadeguato.

Disponibilità:

- Essendo il sistema pensato per supportare gli utenti anche durante il viaggio che hanno programmato tramite la piattaforma, si presenta la necessità di rendere il sistema disponibile in maniera continuativa. Bisogna quindi minimizzare i tempi di downtime in caso di manutenzione.

Sicurezza:

- Il sistema deve garantire la sicurezza dei dati inviati dall'utente alla piattaforma. Gli utenti della piattaforma forniranno i loro dati sensibili, che devono essere trattati in maniera congrua, per non permettere possibili furti di dati.

Accesso:

- Il sistema deve gestire l'accesso dell'utenza tramite l'utilizzo di credenziali che serviranno al riconoscimento del singolo utente ed al reindirizzamento al profile personale.
- Il sistema non deve consentire operazioni fatte da utenti a cui non è consentito l'accesso. Deve fornire i permessi solo a coloro che in generale possono effettuare un'operazione che può modificare lo stato di una singola entità.

1.2.3 Criteri utente finale

Tracciabilità:

- Il sistema deve essere in grado di soddisfare tutti i requisiti definiti in fase precedente In pratica le funzionalità del sistema devono essere compatibili con quelle desiderate inizialmente.

Usabilità:

- Il sistema aiuterà l'utente in ogni sua scelta, guidandolo al raggiungimento del proprio obiettivo.
- L'interfaccia utente sarà di facile comprensione, per poter garantire un utilizzo semplificato.

Facilità di riadozione:

- Il design della piattaforma deve permettere all'utente di riprendere da dove aveva interrotto le sue operazioni anche dopo un periodo di assenza dalla piattaforma.

1.2.4 Criteri di performance

Efficienza:

- Il sistema viene progettato in maniera tale da poter fornire risultati rapidamente all'utente. Tempi di risposta accettabili sono nell'ordine di pochi secondi, tenendo conto che la velocità della connessione di ogni utente giocherà un ruolo molto importante nella visualizzazione dei risultati.
- Il sistema dovrà effettuare in tempi accettabili le varie operazioni a runtime nascoste all'occhio dell'utente, ovvero quelle fornite dal server, in modo da ridurre ulteriormente il tempo di risposta prestabilito.

Utilizzo di memoria:

- Il sistema utilizza un database relazionale basato su MySQL. Tutti i dati persistenti verranno salvati in tale database. C'è ovviamente la necessità di garantire uno spazio di archiviazione sufficiente per gestire un numero elevato di accessi e dati.
- L'operazione di lettura dal database deve avere tempi bassi in modo tale da poter gestire i dati in esso velocemente.

1.3 Definitions, acronyms and abbreviations

Acronimo	Definizione
SQL	**Structured Query Language, linguaggio usato per interagire col database**
MySQL	DBMS scelto per l'implementazione di BeVoyager
DB	Il database sul quale vengono memorizzati i dati
Java	Linguaggio di programmazione OOP in cui è scritto BeVoyager
Javascript	Linguaggio di scripting per web
HTML	Linguaggio di markup per la strutturazione di pagine web
RAD	Requirements Analisys Document
SDD	Software Design Document
ODD	Object Design Document

2. Current Software Architecture

Dopo aver effettuato varie ricerche sul web, appare evidente che non esistono siti la cui funzione sia simile a quella di BeVoyager. In genere, un sito offre la viasualizzazione di informazioni su luoghi ed eventi di interesse, ma non permette di creare un itinerario personalizzabile, e sopratutto non è stata mai rilevata la funzione di poter creare "viaggi", aggiungendo amici e partecipanti in modo tale da coordinare l'organizzazione e la scelta delle mete.
Quindi, in pratica, a parte qualche funzionalità basilare, non ci sono sistemi già realizzati su cui BeVoyager possa risultare basato.

3. Proposed Software Architecture

3.1 Introduction

Nella fase di analisi del sistema, è stata individuata una prima divisione logica del sistema proposto, ed alcune delle relazioni tra i sottosistemi trovati.
Per la struttura del sistema, essendo una web application, si è ritenuto opportuno utilizzare lo stile architetturale Client-Server, per poter semplificare le operazioni logiche del sistema.
Il client, eseguito in locale sulla macchina personale dell'utente, comunica esclusivamente con il server, del quale utilizza i servizi offerti per portare a termine le operazioni scelte.
Il server si occupa di gestire la logica dell'applicazione e di memorizzare i dati sul database.
Per la memorizzazione dei dati è ovviamente necessario l'utilizzo di un sistema per la gestione dei

dati persistenti a lungo termine. Si è deciso di optare, quindi, per un database relazionale, che permette di gestire le varie operazioni sui dati in maniera corretta ed efficace.
I servizi offerti dal database sono accessibili solamente al server.