Università degli Studi di Salerno
Corso di Ingegneria del Software

BeVoyager
ODD
Versione 1.2

Data: 10/01/2017

Partecipanti:

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

Revision History

Data	Versione	Descrizione	Autore
09/01/2017	1.0	Introduzione, object design trade-offs.	
10/01/2017	1.1	Prima stesura packages.	
10/01/2017	1.2	Linee guida per la documentazione dell'interfaccia.	

Sommario

1. Introduzione

L'Object Design Document (ODD) definisce l'object level design del sistema che si sta sviluppando. Attraverso questo documento viene definita l'architettura modulare della piattaforma, la suddivisione del suo contenuto in packages differenti. Esso sfrutta le conoscenze acquisite tramite la stesura dei precedenti documenti.

1.1 Object design trade-offs

- ***Tempo di risposta vs Spazio di memoria:***

Basandoci sui design goals descritti in precedenza (SDD sez. 1), il tempo risposta deve essere minimo. Le operazioni che usano più tempo all'interno del sistema sono gli accessi al database, in quanto, oltre all'accesso al disco bisogna effettuare operazioni di unione e controllo sulle tabelle generate dalle query. Dato che le operazioni effettuate dal sistema spesso risultano nella creazione delle stesse tabelle con gli stessi dati al loro interno, abbiamo rilevato che generare le tabelle necessarie in precedenza, in modo tale che quando viene effettuata un'operazione che ha bisogno di tali dati, non si ha la necessità di generare la tabella a runtime, ma semplicemente si effettua una ricerca in una già generata. Questo porta ad un utilizzo di memoria maggiore all'interno del database ma velocizza di molto le operazioni effettuate.

- ***Comprensibilità vs Costi***

All'interno del team di sviluppo utilizzeremo uno stile di programmazione ben definito tra tutti i componenti del team. In questo modo, nel caso di aggiunta di nuovi membri allo sviluppo, il tempo di training dei nuovi elementi sarà minimizzato, facendo in modo che possano subito iniziare a svolgere task di sviluppo, minimizzando i costi relativi al training, e velocizzando il tempo di sviluppo.

- ***Costi vs Mantenimento***

Il sistema verrà implementato con particolare attenzione alla mantenibilità dello stesso. Gli interventi di manutenzione devono essere resi semplici e veloci per minimizzare il downtime del sistema, e di conseguenza la perdita di guadagno relativa alla mancata disponibilità del servizio. Questo vuol dire che bisognerà prestare più attenzione all'implementazione del codice, e mantenere una documentazione adeguata alla complessità del sistema, il che prevede costi maggiori in quanto ci sarà bisogno di un team dedicato alla documentazione stessa.

- ***Interfaccia vs Easy-use***

L'interfaccia del sistema sarà costruita in modo tale da risultare semplice all'utilizzo dell'utente. Aumentare la facilità di utilizzo significa allargare le funzionalità della piattaforma ad un maggior numero di utenti. Per questo il team di sviluppo, nel momento di implementare l'interfaccia grafica, non solo cercherà di dare alla pagina uno style piacevole alla vista ma fornire una struttura di interazione tale da rendere semplice all'utente la gestione delle sue operazioni.

1.2 Linee guida per la documentazione dell'interfaccia

Tipo	Regole sui nomi	Esempi
Packages	Il prefisso di un nome di un pacchetto comincia sempre con una lettera maiuscola. Se il nome del package comprende due o più parole, allora il nome completo prevederà che la lettera iniziale della parola nel mezzo sarà maiuscola.	package AccessController;
Classes	I nomi delle classi iniziano sempre con una lettera minuscola. Esse devono racchiudere in breve il campo in cui operano. Se il nome è composto da più parole, allora le parole nel mezzo avranno la prima lettera maiuscola e le parole risulteranno divise da dei trattini.	class creaLuogo; class search-route-result;
Interfaces	I nomi dell'interfaccia sono scelti in modo tale da racchiudere il significato di ciò che la stessa mostra all'utente e cosa permette di fare.	interface search; interface feedback;
Methods	I nostri metodi sono verbi, nel caso in cui mescolato con la prima lettera minuscola, con la prima lettera di ogni parola in maiuscolo interna.	insert(); createReport();
Variables	I nomi associati alle variabili sono scritti totalmente con caratteri minuscoli. Essi sono scelti in modo tale da rendere semplice il loro significato ed esplicare significativamente il loro utilizzo.	int i; char c; float latitude;

2. Packages

Di seguito si elencano i *packages* presenti all'interno del sistema:

1. ***AccessController:*** package la cui classe opera controlli sugli accessi.
2. ***DatabaseConnection:*** package le cui classi si occupano delle connessioni al database.
3. ***Feedback:*** package che contiene le classi che creano ed operano su feedback.
4. ***Location:*** package che contiene le classi che creano ed operano su luoghi.
5. ***Log:*** package che contiene le classi per login e logout.
6. ***Notification:*** package che contiene le classi che creano ed operano sulle notifiche.
7. ***Poll:*** package che contiene le classi che creano sondaggi ed operano su essi.
8. ***Route:*** package che contiene le classi che creano ed operano sugli itinerari.
9. ***Travel:*** package che contiene le classi che creano ed operano sui viaggi.
10. ***User:*** package che contiene le classi che creano ed operano sugli utenti.