Scelte Implementative

Ingegneria del Software

Stefano Ravetta, Simone Renzo, Matteo Scarpone, Mattia Tollari

29 Febbraio, 2016

1 Introduzione

1.1 Informazioni su questo ducumento

In questo documento verranno trattate e discusse le scelte implementative che sono state prese per la realizzazione del progetto "Team Diary Management" e le motivazioni delle stesse .

1.2 Contenuti

Si porrà l'attenzione in modo particolare su

- Software per la creazione dei diagrammi
- Software per la stesura della documentazione
- Linguaggio di programmazione
- Formato e struttura della base di dati
- Librerie per la creazione dell'interfaccia utente
- Struttura dei sistemi di sicurezza adottati

2 Linguaggio di Programmazione

2.1 Introduzione

É stato scelto il linguaggio di programmazione Python[1] per alcune caratteristiche piuttosto interessanti che lo possono far risaltare nell'ecosistema dei linguaggi di programmazione odierno.

[1] https://www.python.org/

2.2 Alto livello

Python consente di programmare ad un livello molto alto: con poche righe riesce a gestire in modo semplice e diretto operazioni complesse senza che il codice risulti troppo complesso da leggere e comprendere.

Questo ha permesso al team di sviluppo di concentrarsi maggiormente sulle parti più astratte e progettuali (come i design pattern e la rimozione di vari code smell) rispetto a questioni più vicine alla comprensione del funzionamento del linguaggio.

2.3 Filosofia

Una parte di un famoso "easter egg"[2] recita:

> If the implementation is hard to explain, it's a bad idea.
>
> If the implementation is easy to explain, it may be a good idea.

Queste righe sembrano denotare con toni sarcastici la filosofia del linguaggio. Uno degli obiettivi del linguaggio è appunto far sì che il codice scritto risulti il miglior compromesso tra stringatezza e comprensibilità. Si intende, quindi, che codice estremamente compatto non è sinonimo di efficienza a lungo termine in quanto provocherà quasi sicuramente un dilatamento delle tempistiche sia in fasi di sviluppo successive, sia in fase di testing ed ottimizzazione. Un esempio che può far notare il modo in cui questo tipo di filosofia inserito nella struttura del linguaggio possa effettivamente influire sul codice è quello dell'indentazione: per forzare il programmatore ad indentare sempre il codice (e a farlo in modo corretto) questo linguaggio non usa caratteri di begin ed end nella definzione di classi, espressioni condizionali, espressioni iterative, funzioni e procedure (come fa ad esempio Java con le parentesi graffe) ma usa solo l'indentazione. In tal modo, da una parte i programmatori che di norma non indentano il loro codice risulteranno "costretti" a farlo, e dall'altra i programmatori che gia' di norma indentano il proprio codice non dovranno inserire altri simboli per esprimere ciò che concettualmente può essere già espresso dall'indentazione.

Secondo il nostro team di sviluppo, questa caratteristica risulta molto interessante per la gestione di un progetto in cui è richiesta particolare attenzione alla qualità del codice prodotto. Inoltre, a livello pratico, tutto ciò è risultato addirittura comodo perche' si è mostrata la necessità di lavorare con uno stile di stesura del codice unico (ogni membro del gruppo ne presentava uno diverso).

[2] Questo easter egg, "The Zen of Python" è stato inserito nell'interprete Python e che può apparire appena si prova ad importare la libreria "this".

2.4 Portabilità

Python è un linguaggio precompilato e il suo interprete è disponibile per numerose[3] piattaforme. La presenza di questa caratteristica non è stata sicuramente ignorata dato che quasi tutti i membri del team di sviluppo usano piattaforme ed ambienti diversi. Anche la distribuzione dell'applicazione e di pacchetti contenenti le varie dipendenze risulta estremamente semplificata grazie alla portabilità di Python.

2.5 Dizionari

Una delle funzionalità che sono state più apprezzate per la stesura del codice relativa al progetto è senza dubbio la gestione dei dizionari che Python offre e la loro semplice e diretta interazione con classi e anche dati Json.

3 Librerie per la creazione dell'interfaccia utente

3.1 Introduzione

Sono state scelte le librerie Qt[4] che fanno parte di uno storico framework open source.

3.2 Potenzialità

Le librerie Qt sono in grado di gestire interfacce grafiche, basi di dati, attività multimediali, flussi di rete, gestione di file e notifiche di sistema. Una caratteristica piuttosto interessante di queste librerie è che possono essere usate attraverso molti[5] linguaggi di programmazione e non sono legate a nessuna piattaforma specifica. Permettono anche lo sviluppo di applicazioni mobile[6]

3.3 Ambienti di sviluppo

Per lo sviluppo dell'interfaccia grafica è stato comodo usare lo strumento QtDesigner[7] (che segue il paradigma WYSIWYG) che permette di disegnare GUI, modificare le proprietà dei vari oggetti e generare codice XML dell'interfaccia realizzata.

[3] L'elenco delle piattaforme supportate è reperibile a https://www.python.org/download/other/
[4] http://www.qt.io/
[5] È possibile visionare tutti i linguaggi supportati a questa pagina: http://wiki.qt.io/Category:LanguageBindings
[6] il progetto SailfishOs può dare un'idea delle potenzialità delle librerie Qt: http://sailfishos.org/develop/sdk-overview/
[7] http://doc.qt.io/qt-4.8/designer-manual.html

3.4 interazione con Python

Per far interagire il file XML generato da QtDesigner con Python è stato usato lo strumento Pyuic4[8] che svolge il lavoro meccanico di conversione da file che descrive un'interfaccia ad uno script Python che mette a disposizione classi che rappresentano gli oggetti usati ed i metodi per accedervi.

4 Base di dati

4.1 Introduzione

Dato che la base di dati usata ha una struttura molto semplice è stato scelto di non usare uno strumento complesso come un database relazionale. È stata creata una base di dati in formato Json che è risultata semplice da strutturare e da gestire. È risultato comodo interagire assieme sulla base di dati tramite Git. L'unico problema riscontrato è stato dover gestire i vari file dal codice dell'applicazione. Questo problema è stato praticamente risolto con l'introduzione di una struttura di metodi e costanti. Con un database relazionale sarebbe stato necessario usare i file ottenuti dal dump del database ed importarli in locale ad ogni modifica. L'ideazione e la gestione del database relazionale avrebbero occupato piú tempo della soluzione adottata.

4.2 Struttura

La struttura della base di dati è così definita:

```
database
├── user.json
├── location.json
├── activity.json
├── group.json
└── project.json
```

4.2.1 User.json

Contiene i dati relativi all'utente: id, username, password, nome, cognome, gruppi (una lista che contiene l'id del gruppo e il ruolo ("partecipante" o "team leader"), vacanze (una listas che contiene un id, un nome, data di inizio e data di fine).

4.2.2 Location.json

Contiene i dati relativi ai luoghi: un id ed un nome.

[8] http://pyqt.sourceforge.net/Docs/PyQt4/designer.html

4.2.3 Activity.json

Contiene i dati relativi alle attività: id, nome, id del progetto di riferimento, id dell'utente che ha creato l'attività, tipo di attività (se di gruppo, di progetto o singola), l'id del luogo, id del grupppo, lista di id dei partecipanti.

4.2.4 Group.json

Contiene i dati relativi ai grupi: id, nome, father (False se il gruppo non è un sottogruppo, id del sottogruppo altrimenti), lista degli id dei sottogruppi.

4.3 Interazione con Python

Per importare e gestire la base di dati dal programma sono stati usati i dizionari costruiti, come Json, con [chiave: valore]. Questo tipo di dato ha reso molto agile l'accesso e la generazione di codice Json.

5 Struttura dei sistemi di sicurezza adottati

5.1 Codifica delle password

Per evitare di salvare le password degli utenti in chiaro è stato fatto ricorso ad un algoritmo di hashing one-way: esiste una funzione semplice da calcolare che associa ad ogni stringa un hash ma non esiste una funzione semplice da calcolare che associa ad ogni hash una stringa tale che, se applicata la funzione iniziale, l'hash ottenuto sia quello di partenza. È stato scelto SHA512 in quanto oggi risulta quello più robusto a livello di problemi di collisione e preimmagine. Per l'implementazione in Python è stata usata la libreria Hashlib.

5.2 Token, autenticazione e gestione delle sessioni

Per gestire autenticazione e sessioni è stato implementato un sistema di token. Un token è un hash SHA512 di utente, password e timestamp. Con un controllo su questi tre campi si impedisce ad un utente malintenzionato (che non dispone dei una password valida) di poter accedere ad una sessione già attiva. Il timestamp serve per imporre scadenze al token (limite attualmente posto uguale ad un giorno). Così facendo risulterà difficile anche provare un attacco che mira a scoprire il token dato che in poco tempo i tentativi precedenti risulteranno inutili.

5.3 Password dimenticate

Nel caso venisse smarrita la password di un utente verrà visualizzato un messaggio che indicherà di contattare gli amministratori di sistema. È stata presa questa scelta perchè

non erano presenti le risorse necessarie per implementare e testare a fondo un sistema di recupero password che usi email o SMS.