# Exif Viewer: lettore di informazione exif per immagini *.jpeg*

Francesco Gradi
Matricola: 7017190
francesco.gradi@stud.unifi.it

## Abstract

*In questo progetto è stata progettata un'interfaccia grafica mediante il framework PyQt per la lettura di immagini .jpeg e la visualizzazione delle relative informazioni exif. In particolare è stata riservata attenzione nell'utilizzo delle buone pratiche di programmazione di gui, cercando di separare le diverse responsabilità degli elementi tramite il paradigma di Model View Controller. Viene inoltre descritto brevemente un tipico caso d'uso che si verifica all'interno dell'app che fa uso proprio del paradigma. È stato incluso come esempio di utilizzo di informazioni exif, quello della lettura delle informazioni GPS e la conseguente apertura di un browser con Google Maps.*

## 1. Introduzione

Le immagini in formato *.jpeg* sono ormai pervasive, anche grazie alla rapida diffusione degli smartphones, che sono soliti salvare le immagini in questo formato. Tali files portano con sé informazioni aggiuntive, relative all'ora dello scatto, all'apertura focale, alle specifiche della lente, alle coordinate GPS e altre informazioni ancora [1].

Tali info sono salvate in formato *exif* ed è possibile leggerle per aumentare il grado di interattività di una certa applicazione con una certa foto.

Per il progettino del corso di *Human Computer Interaction* è stato quindi sviluppato un lettore di immagini con affiancate le informazioni Exif. L'esempio di utilizzo di tale informazioni implementato è stato l'utilizzo delle informazioni GPS per aprire un apposito url di Google Maps.

È stato scelto il linguaggio Python mediante il framework PyQt nella sua versione 5, poiché è abbastanza semplice estrarre le informazioni di *exif* (tramite la libreria *Pillow*), a differenza di altri linguaggi che rendono l'operazione più complessa (in particolare è stata realizzata in precedenza una bozza di applicazione in Angular).

Il codice è reperibile su GitHub [2].

## 2. Model View Controller

Quando si progetta un'interfaccia grafica è buona norma separare logicamente all'interno del codice le tre componenti logiche essenziali, secondo il paradigma Model View Controller [3].

- La **view** comprende tutte le componenti grafiche, all'interno del codice ci sono principalmente dei bottoni, una label per contenere l'immagine e la tabella per visualizzare le informazioni di *exif*. È stata generata a partire dal software *QtDesigner*, i file così forniti sono stati prima convertiti in python e poi inclusi nella classe principale, *MainWindow*.
- Il **modello** incapsula i dati fondamentali per l'applicazione. È stato previsto un modello per la tabella, contenente proprio le informazioni *exif*, chiamato *ExifTableModel*; esso eredita dal modello astratto della tabella direttamente dalla classe di QtCore e ne implementa le funzioni fondamentali, tra cui *data* per inserire i campi nella tabella. Non è stato implementato un modello anche per l'immagine perché era un po' esagerato per le informazioni che doveva gestire (una url) e per il fatto che nelle specifiche era richiesta soltanto un'immagine alla volta.
- Il **controller** è delegato all'applicazione principale, la quale implementa anche gli eventi collegati con i bottoni dell'interfaccia e gestisce il cambio di immagine e l'aggiornamento modello della tabella.

Uno scenario tipico di interazione potrebbe essere quindi:

1. L'utente vuole cambiare immagine e clicca sul relativo bottone all'interno della GUI.
2. Il bottone, che fa parte della *view*, è collegato a una funzione del *controller* apposita.

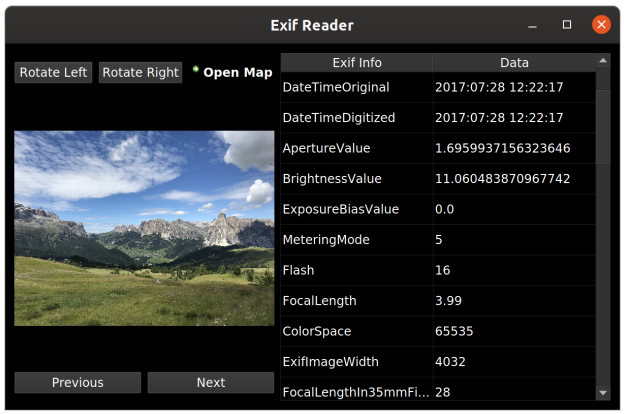


Figure 1. La vista dell'applicazione.

3. Tale funzione edita di conseguenza il *model*, l'aggiornamento di questo fa sì che si modifichi anche la vista corrente (vengono letti i nuovi dati). Si noti che il modello non sa nulla della vista.

## 3. Funzionalità

L'applicazione permette di visualizzare tutte le immagini con estensione *.jpeg* o similari all'interno della directory nella quale è eseguito il file *.python*. Viene visualizzata l'immagine sulla sinistra e la tabella delle informazioni *exif* a destra. Si può ridimensionare la finestra ottenendo un relativo cambio di proporzione da parte dell'immagine fino a che non raggiunge la grandezza massima (come da specifiche, 512 pixel).

È possibile cambiare l'immagine attuale semplicemente cliccando sui relativi bottoni, la foto e la tabella vengono opportunamente aggiornate in modo automatiche grazie al Model View Controller.

Funzioni aggiuntive sono la rotazione dell'immagine di 90 gradi a destra o a sinistra e un pulsante che legge le informazioni GPS dell'immagine (sempre *exif*) e apre il browser predefinito con la pagina di Google Maps contente un segnaposto nel luogo in cui è stata scattata la foto.

Questo è stato possibile mediante delle direttive specifiche di Google Maps su come è possibile effettuare una query a partire da delle coordinate GPS semplicemente tramite un apposito URL [4].

Il codice è stato testato sia su Ubuntu 19.10 sia su Mac OS 10.15.3. Su quest'ultimo setup sembra non funzionare ottimamente l'aggiornamento automatico, probabilmente per delle incompatibilità con PyQt (che spesso dà problemi con Mac).

## 4. Conclusioni

In conclusione, in questo progetto è stata analizzata l'importanza di utilizzare dei paradigmi tecnici specifici, quali il Model View Controller, per separare opportunamente le responsabilità degli elementi nella scrittura di codice di interfacce grafiche.

In particolare è stata progettata una GUI che intendesse valorizzare le informazioni di *exif* contenute nelle immagini *.jpeg* e consentire all'utente la visualizzazione della posizione dello scatto della fotografia tramite l'accesso a Google Maps.

## References

[1] *Exif Data*
`https://www.photographymad.com/pages/view/exif-data-explained`

[2] *Repository GitHub del progetto*
`https://github.com/FrancescoGradi/ExifViewer`

[3] *Model View Controller*
`https://www.codecademy.com/articles/mvc`

[4] *API di Google Maps per la visualizzazione di un luogo tramite una query con coordinate GPS*
`https://developers.google.com/maps/documentation/urls/guide`