Università degli Studi di Modena e Reggio Emilia

Sistemi e Applicazioni di Rete

PROGETTAZIONE E REALIZZAZIONE DI UNA APPLICAZIONE CLOUD-BASED: NEWS ANALYZER

A cura di Luca Gallinari

Anno Accademico 2015/16

1. Introduzione 2

2. Analisi del problema 3

3. Descrizione dei servizi e del workload 4

3.1 Servizi per tutti i tipi di utente 4
3.2 Servizi per utenti autenticati 5
3.3 Workload atteso 5

4. Soluzioni architetturali 8

4.1 Hosting 8
4.2 Database 9

5. Componenti software 10

5.1 Linguaggio e piattaforma 10
5.2 Framework e librerie 11
5.3 User interface: framework e librerie 13
5.4 API esterne utilizzate 13
5.5 API esportate 15
5.6 Comunicazione asincrona 16
5.7 CRON 17

6. Stima dei costi 18

6.1 Riassunto dei costi 18
6.2 Stime ad intensità crescente 19

7. Sviluppi futuri 23

7.1 Miglioramenti 23
7.2 Aggiunte 24

8. Conclusioni 25

1. Introduzione

Nel presente elaborato verranno descritte e motivate tutte le scelte che sono state prese nella progettazione e realizzazione dell'applicazione, inoltre, verranno analizzati i vari componenti software che la costituiscono, verrà descritto il workload e fornita una possibile stima dei costi.

News Analyzer è un'applicazione *cloud-based* che permette agli utenti di rimanere aggiornati sulle ultime notizie dal mondo e di poter analizzare una notizia in modo da estrapolarne le entità principali. Quest'ultima funzionalità consente agli utenti di farsi un'idea generale del contenuto della notizia e di poter analizzare nel dettaglio ogni entità da essa estratta.

Le funzionalità di visualizzazione delle notizie di tendenza e di analisi di una particolare notizia sono usufruibili anche da utenti non autenticati poiché non fanno uso di informazioni personali. Invece, gli utenti che vorranno personalizzare la propria esperienza potranno autenticarsi con il proprio account Google+ ottenendo l'accesso a qualche funzionalità aggiuntiva. Potranno, ad esempio, creare dei filtri, cioè un insieme di parole chiave, che permettono all'utente sia di poter visualizzare nella *homepage* solo le notizie relative ad un particolare filtro, sia di poter ricevere tramite email ad un dato orario le notizie di tendenza rispetto ad un filtro.

Per motivi di Copyright, il contenuto delle notizie mostrate agli utenti è solo un estratto del reale contenuto della notizia (immagine, titolo e breve descrizione), inoltre, è sempre presente un link che rimanda alla notizia originale. Invece, l'analisi delle notizie non è soggetta a problemi di Copyright poiché, la sola visualizzazione delle entità caratteristiche del testo, non è paragonabile alla reale lettura della notizia.
Gli utenti autenticati potranno revocare in qualsiasi momento i permessi di accesso, da parte dell'applicazione, alle informazioni personali del proprio account Google+.
L'applicazione esporta a sua volta delle Web API, utilizzabili da un qualsiasi sistema esterno.

2. Analisi del problema

L'obiettivo principale di *News Analyzer* è quello di riuscire a mettere in una sola pagina, a portata di click, il maggior numero di informazioni riguardanti le principali entità che caratterizzano una notizia. Infatti, molto spesso capita, leggendo un articolo, di affrontare uno o più argomenti nuovi e/o dei quali non si hanno particolari conoscenze. L'operazione di ricerca di questi argomenti richiede però parecchio tempo, specie quando gli argomenti sono tanti e/o quando si sta utilizzando un cellulare (la ricerca multipla da cellulare è più difficoltosa che da pc).
News Analyzer permette di risolvere in parte questo problema fornendo informazioni specifiche in base al tipo di entità che si vuole approfondire.
Ad esempio, nel caso di un entità di tipo *Place* verrà visualizzata una mappa centrata sulle coordinate di quel posto e alcune foto che lo caratterizzano, mentre nel caso di un *Film* verranno mostrate delle immagini e qualche video preso da Youtube.

Inoltre, come supporto alla macro-funzionalità di analisi appena descritta, viene fornito all'utente un sistema di ricerca e gestione delle notizie di tendenza. In questo modo, un utente potrà rimanere aggiornato sulle notizie di giorno in giorno e analizzarle nel caso voglia approfondire un certo argomento. Questo sistema è assolutamente indipendente da quello di analisi, ma si presta molto bene come incentivo all'analisi delle notizie, poiché per ogni notizia mostrata all'utente verrà sempre visualizzato un bottone per poterla analizzare.

Come abbiamo già accennato, si vuole rendere l'applicazione facilmente utilizzabile anche da utenti mobile, per far questo sono stati adottati particolari accorgimenti sia nella realizzazione della User Interface, per migliorare l'usabilità da mobile, sia nello scambio di dati tra client (utente) e server (*News Analyzer*), per ridurre il consumo di dati e non sovraccaricare il client. Questo aspetto verrà trattato più nel dettaglio nel capitolo 5 delle *Componenti Software*.

3. Descrizione dei servizi e del workload

In questo capitolo verranno brevemente descritte tutte le funzionalità offerte dall'applicazione ad utenti anonimi o autenticati (figura 1) e verrà fornita una descrizione del possibile workload atteso su base giornaliera, settimanale, mensile e annuale.

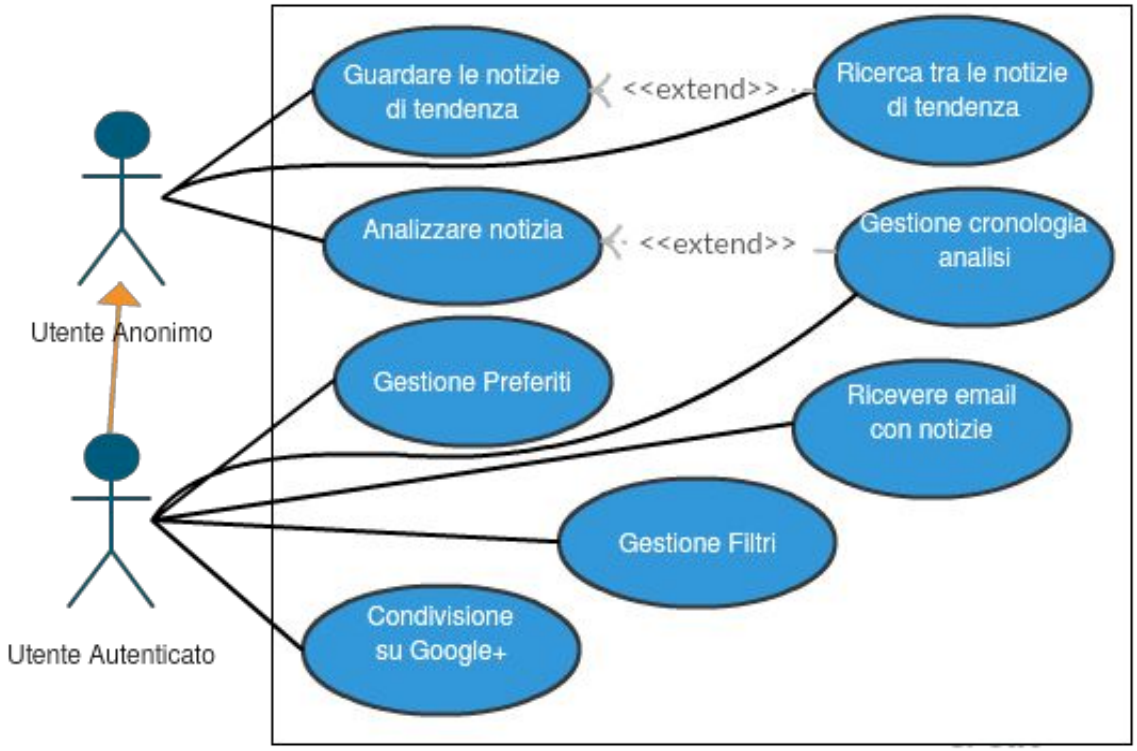

Figura 1: Use Case Diagram che mostra i due tipi di utente e le funzionalità messe a loro disposizione

3.1 Servizi per tutti i tipi di utente

I servizi di seguito descritti sono disponibili a tutte le tipologie di utente:

- Visualizzare l'immagine principale, il titolo, un breve estratto del contenuto e qualche informazione aggiuntiva, delle notizie di tendenza (in lingua inglese) prese da svariati siti web;
- Ricercare notizie di tendenza in base a delle parole chiave;
- Analizzare una notizia, trovata sul sito o fornendo un URL, in modo da estrapolarne le entità principali;
- Ogni entità estratta dalla procedura di analisi può essere consultata, ottenendo un insieme di informazioni (immagini, video, testi ecc) specifiche in base alla tipologia dell'entità (film, persone, posti ecc).

3.2 Servizi per utenti autenticati

I servizi elencati di seguito sono invece dedicati ai soli utenti che hanno effettuato l'accesso con il proprio account Google+ e hanno autorizzato l'applicazione all'accesso alle informazioni personali.

- Aggiungere le notizie di tendenza ai propri preferiti, in modo da poterle consultare o analizzare in un secondo momento;
- Creare, eliminare e modificare dei filtri costituiti da un nome (con cui verranno identificati) e da un insieme di parole chiave (in base alle quali verranno effettuate le ricerche);
- Visualizzare nella *homepage* solo le notizie di tendenza relative alle parole chiave di un particolare filtro;
- Rimanere sempre aggiornati facendosi inviare quotidianamente per email, ad una data ora, una lista delle ultime notizie selezionate in base alle parole chiave di un particolare filtro (questo aspetto richiede accorgimenti architetturali non banali poiché deve essere possibile, a livello di piattaforma, eseguire delle azioni programmate);
- Condividere le notizie su Google+ attraverso il proprio account;
- Consultare ed eliminare la cronologia delle proprie analisi.

3.3 Workload atteso

Il modello del workload atteso è quello tipico delle applicazioni che offrono un servizio web collocabile nella tipologia svago e tempo libero. Questo tipo di servizio presenta una utilizzazione annuale e mensile abbastanza uniforme, senza particolari periodi di picco. La distribuzione del carico settimanale è fortemente concentrato nei giorni infra-settimanali, mentre nel weekend si ha un'utilizzazione minore ma più uniforme nel corso della giornata. Anche per quanto riguarda il workload giornaliero bisogna fare distinzione tra giorni infra settimanali e il weekend. Nel primo caso, si avranno dei picchi di utilizzo in concomitanza con i pasti, quindi colazione (pre-lavoro), pranzo e cena (periodo serale), intervallati da lassi di tempo di inutilizzo. Invece, il weekend, presenterà dei picchi di utilizzo molto meno elevati e un'utilizzazione lievemente maggiore tra i pasti (figura 2).

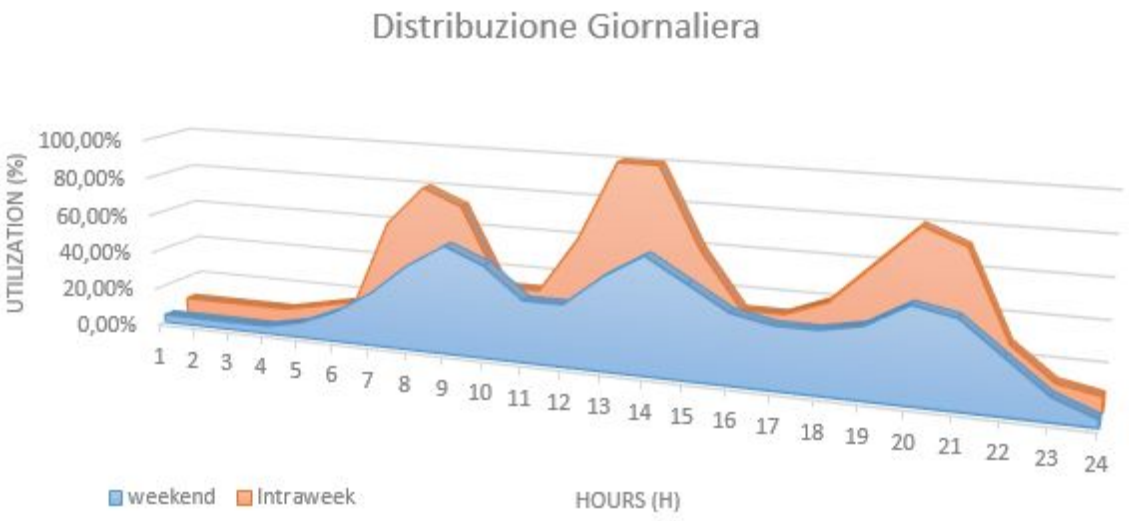

Figura 2: Si possono notare le differenze nell'utilizzazione giornaliera tra weekend e giorni infra-settimanali.

Le possibili distribuzioni viste finora, sono da considerarsi veritiere solo nel caso di utilizzo dell'applicazione da parte di utenti dello stesso continente della locazione del server di *News Analyzer*. Questo perché l'applicazione potrebbe, col tempo, iniziare ad essere utilizzata da utenti provenienti da continenti diversi (visto che è interamente realizzata in lingua inglese). In questo caso, il workload globale giornaliero varierebbe notevolmente, portandosi verso una distribuzione molto più uniforme nelle varie ore della giornata. La distribuzione settimanale, invece, non varierebbe tantissimo.

Ci spingiamo più nel dettaglio, andando ad elencare i possibili valori del workload:

- 20 accessi/giorno da parte di utenti autenticati o non;
- 5 accessi/giorno da parte di utenti autenticati;
- 200 ricerche/giorno di notizie (considerando i 20 accessi/giorno e il fatto che la *homepage* visualizzi sempre almeno 10 notizie);
- 100 notizie/giorno da analizzare (una media di 5 notizie/utente considerando i 20 accessi/giorno);
- 2 notizie/giorno inserite nei preferiti da ogni utente autenticato;
- Una media di 3 filtri/utente inseriti al momento dell'iscrizione e l'aggiunta di 2 filtri/mese nel periodo successivo;

- 20 email al giorno inviate automaticamente (considerando 20 utenti registrati e la media precedente di 3 filtri/utente, di cui magari solo un 1 filtro/utente viene abilitato all'invio tramite email);
- 15 accessi/giorno alle Web API esportate dall'applicazione.

Questi valori del workload sono stati stimati considerando un basso utilizzo dell'applicazione. Ovviamente, se questa dovesse avere successo, i valori andrebbero totalmente riconsiderati e, come accennato precedentemente, si dovrebbe rivedere il modello del workload (per via del possibile utilizzo su scala mondiale).

4. Soluzioni architetturali

Nel precedente capitolo abbiamo trattato la prima fase del ciclo di sviluppo del software chiamata *analisi*, in questo andremo ad affrontare la fase successiva ovvero quella di *progettazione*.

Elenchiamo di seguito i vari aspetti architetturali che presentano varie possibili alternative e rispetto alle quali andranno prese delle decisioni:

- Una infrastruttura o una piattaforma nella quale *hostare* l'applicazione;
- Un database nel quale salvare preferiti, filtri, utenti ecc.;

4.1 Hosting

La prima scelta da fare è rispetto all'alternativa "make or buy", ovvero, bisogna decidere se creare e gestire un proprio server web oppure affidarsi a servizi di terze parti. Si è stimato che il costo derivante dalla prima alternativa sarebbe molto elevato, questo perché non si possiede già un proprio server web e, inoltre, la quantità di tempo richiesta per l'installazione, configurazione e successiva gestione del sistema sarebbe elevatissima. Considerando tra l'altro che, in caso di successo (probabilità bassissima ma esistente), si dovrebbero gestire grossi problemi di scalabilità del sistema. Questo aspetto è, in genere, gestito nativamente dai servizi di hosting cloud a fronte, ovviamente, di un costo maggiore in proporzione alle "dimensioni" del sistema (*pay per use*).

Si è quindi deciso di appoggiarsi ad un servizio cloud per l'hosting dell'applicazione. Tra i 3 possibili paradigmi di servizio (SaaS, PaaS e IaaS) caratteristici del cloud, solo 2 sono effettivamente adatti al nostro scopo: Platform as a Service (PaaS) e Infrastructure as a Service (IaaS). La tipologia Software as a Service (SaaS) è da escludere a priori poiché, per definizione, non è adatta alle esigenze della nostra applicazione.

Il modello IaaS, in breve, mette a disposizione un'intera infrastruttura virtualizzata, scalabile e fortemente personalizzabile. Il livello di dettaglio fornito da questa soluzione è però eccessivo rispetto alle esigenze di *News Analyzer*, ci si dovrebbe occupare di aspetti non rilevanti e dei quali non ci si vuole preoccupare. Per queste ragioni, è stata considerata migliore la soluzione PaaS (anche se introduce notevoli limitazioni) che fornisce una piattaforma di sviluppo, implementazione e gestione di una applicazione.

Esistono tanti fornitori di PaaS (Google, Amazon, Microsoft ecc..), ognuno con le proprie caratteristiche. Questo aspetto verrà trattato più a fondo nel capitolo delle *Componenti Software* poiché la scelta è legata al linguaggio di programmazione che andremo ad utilizzare, anche se ultimamente la diversità di linguaggi supportati dai vari PaaS si è ridotta sempre più.

4.2 Database

Come è già stato evidenziato in fase di analisi, l'applicazione necessita di un database per il salvataggio e il recupero di informazioni. Dalle specifiche (ma anche dal modello di workload) si può facilmente notare che *News Analyzer* non richieda database con particolari prestazioni: filtri, preferiti e utenti sono rappresentati da informazioni molto contenute e potenzialmente poco numerose. Gli unici gruppi di informazioni che richiederebbero maggiori accorgimenti (poiché numerosi e voluminosi) sono le notizie di tendenza (da mostrare agli utenti) e le informazioni relative alle entità ottenute dall'analisi di una notizia (la sola entità non costituirebbe un problema per il database). Si è però deciso di non conservarli e ricrearli di volta in volta. Se il numero di utenti dovesse aumentare notevolmente, si potrebbero iniziare a salvare quelle informazioni in modo da ridurre sensibilmente il traffico di rete verso i servizi di terze parti.
In realtà, l'URL delle notizie analizzate dagli utenti autenticati e il nome delle entità in esse presenti, verranno salvate nel database per realizzare alcune delle API (capitolo 5.5). Questi dati però non rappresentano un problema per il datastore, in quanto alcuni di essi vengono eliminati quotidianamente.

Inoltre, bisogna considerare che, come visto nel precedente capitolo, è stato deciso di affidarsi ad un PaaS, che presenta dei limiti rispetto alle tipologie di database supportate e, in genere, ne presenta già uno integrato direttamente nel servizio.

5. Componenti software

Entriamo ora nello specifico andando a delineare i dettagli implementativi, ovvero, verranno descritti i vari componenti software utilizzati e le scelte che sono state prese rispetto al linguaggio di programmazione, alla piattaforma, alle librerie e alle API esterne.
Non verrà trattato nessun aspetto riguardante il codice scritto, poiché tutto il codice è ben commentato e facilmente consultabile online al seguente indirizzo: https://github.com/LucaGallinari/NewsAnalyzer.

5.1 Linguaggio e piattaforma

Come già accennato nel capitolo delle *Soluzioni Architetturali*, i linguaggi di programmazione messi a disposizione dai vari PaaS sono molteplici: PHP, Java, Python, Go, Ruby, .NET e altri. Ognuno di questi presenta uno o più framework creati ad hoc per lo sviluppo di applicazioni web e, di conseguenza, i vantaggi che si avrebbero dall'utilizzo di un linguaggio piuttosto che un altro sarebbero davvero pochi. Personalmente, non si hanno particolari interessi per .NET e Ruby, mentre PHP e Java sono già stati trattati a fondo in altre occasioni e, in questo progetto, ci si voleva cimentare su un linguaggio di programmazione non ancora utilizzato. La sola piattaforma che offre supporto per il Go è Google App Engine mentre Python è utilizzabile sulla maggior parte dei PaaS. Si è quindi deciso di utilizzare quest'ultimo per non scartare fin da subito tutte le altre piattaforme.
Tra tutti i possibili fornitori di PaaS, sono di nostro interesse quelli che:

- Offrono supporto al Python;
- Presentano un free-tier (il servizio è gratuito se si rimane entro certi limiti);
- Sono *pay per use* (al di fuori del free-tier);
- Scalano automaticamente;
- Sono localizzati in Europa.

Visti i numerosi criteri di ricerca è stato utilizzato il sito web *paasify.it*[1] per ottenere il seguente elenco di possibili scelte:

- Google App Engine (GAE), soluzione molto commerciale che però soddisfa a pieno le caratteristiche elencate qui sopra e fornisce un SDK (Software Development Kit)

[1] http://www.paasify.it/filter, permette di confrontare le diverse soluzioni PaaS

per tutti i linguaggi di programmazione supportati. E' gratuito finché si rimane entro le 28h di istanze, 1GB di storage e 1GB di traffico in ingresso ed in uscita al giorno;

- IBM Bluemix, basata su Cloud Foundry (PaaS open source), presenta un ampio catalogo e delle funzioni interessati (Servizi Watson, Internet Of Things, ecc..), ma essendo un sistema molto recente (2014) presenta poca documentazione;
- Amazon Elastic Beanstalk (AWS EB), viene fornito un sistema molto versatile e flessibile (per questo è forse un può più complicato degli altri da configurare) sul quale caricare e gestire le proprie applicazioni (non presenta un SDK come GAE) con la possibilità di utilizzare numerose funzioni aggiuntive di Amazon (Load Balancing, Autoscale, ecc..). Il prezzo dipende dai costi delle istanze di EC2 utilizzate e da numerosi altri fattori. E' possibile usufruire di istanze EC2 gratuitamente per 12 mesi con un massimo di 750h/mese di istanze;
- Microsoft Azure, PaaS molto completa e la quantità di prodotti e servizi a disposizione dei clienti è davvero notevole. Fornisce un SDK che in combinazione con Visual Studio creano una piattaforma di sviluppo molto potente. Non presenta un vero e proprio free-tier ma attualmente offre 170€ di buono da utilizzare sui propri servizi.

Tutte queste piattaforme sono fornite da società "Blue Chip" (società ad alta capitalizzazione azionaria), ciò vuol dire che non ci dobbiamo preoccupare di problemi di supporto al servizio (non spariscono da un giorno all'altro). Ma tra queste soluzioni, quella che meglio si adatta alle esigenze dell'applicazione, è sicuramente Google App Engine. A differenza delle altre presenta una documentazione vastissima e molto dettagliata, senza considerare la facilità di integrazione con i servizi di Google che, considerando le specifiche di *News Analyzer*, risulterebbero di notevole utilità. Inoltre, fornisce diretto supporto per il Google Datastore, un database schema-less transazionale, le cui funzionalità sono facilmente accessibili dalla piattaforma.

5.2 Framework e librerie

In questo capitolo vengono brevemente descritte le principali librerie e i framework utilizzati in *News Analyzer*. Alcune sono già presenti in GAE mentre altre sono state importate manualmente inserendo l'apposita cartella nella *root* dell'applicazione.

Si è deciso di seguire il pattern architetturale MVC (Model View Controller) in modo da separare la logica di presentazione dei dati da quella di business (figura).

5.2.1 WebApp2

Web framework molto leggero direttamente supportato da Google App Engine. In parole povere, si fa carico di gestire le richieste web dei client "eseguendo" la classe Python associata all'URL richiesto. Queste associazioni vengono fatte direttamente nel codice attraverso il componente *WSGIApplication*.

5.2.2 Jinja2

Linguaggio di templating che permette di realizzare parte del pattern MVC dividendo logica di presentazione e di business. E' stato anche utilizzato per *renderizzare* il template delle email.

5.2.3 OAuth2Client[2]

E' una libreria Python che permette l'accesso a risorse tramite OAuth2. Nel nostro caso viene utilizzato per interfacciarsi con le API di Google+ (aspetto descritto meglio nel capitolo 5.4.2).

OAuth2 è uno standard per l'**autorizzazione** che permette ad un servizio di terze parti di ottenere l'accesso alle risorse (anche informazioni) di un utente gestite da un *resource owner*. L'utente dovrà acconsentire all'utilizzo delle risorse e, in caso affermativo, verrà generato un token di una certa durata che il servizio esterno dovrà comunicare al *resource owner* di volta in volta.

5.2.4 Google ApiClient

Permette di interfacciarsi con le API di Google. I servizi che si vogliono utilizzare vanno abilitati dalla "developer console" del progetto.

5.2.5 GAESession

Libreria Python di terze parti (non di proprietà di Google) che permette di gestire delle sessioni. In *News Analyzer* viene utilizzato per controllare se un utente è autenticato e per salvare le informazioni dell'account Google+, senza così doverle richiedere ogni volta ai server di Google.

[2] https://github.com/google/oauth2client

5.2.6 PyTZ

Viene utilizzata, insieme alla libreria di Python *datetime*, per calcolare la data e l'orario attuale rispetto al fuso orario "Europe/Rome".

5.3 User interface: linguaggi e librerie

Abbiamo appena descritto alcune delle librerie con cui è stata realizzata la parte di back-end, andiamo adesso ad introdurre le componenti utilizzate per creare la sua controparte: il front-end.

5.3.1 jQuery

Famosa libreria JavaScript che permette di semplificare notevolmente tante operazioni che risulterebbero complicate da realizzare con le sole funzioni JavaScript. Nell'applicazione viene utilizzata per gestire le chiamate AJAX (che verranno discusse nel capitolo 5.6) e migliorare la *user experience*.

5.3.2 MaterializeCSS

E' un front-end framework bassato su Material Design interamente realizzato in CSS e JavaScript. Ci ha permesso di realizzare, con poco sforzo, una *User Interface* elegante e leggera.

5.3.3 jQueryWaterfall

Semplice plugin JavaScript per JQuery utilizzato, nel nostro caso, per mostrare le notizie con uno stile cascata molto simile a quello utilizzato da Pinterest[3].

5.3.4 jQcloud

Permette di generare una nuvola di nomi di grandezze diverse a seconda del peso ad essi associato. Nel nostro caso viene utilizzato per mostrare le entità estratte da una notizia, con una grandezza diversa a seconda del loro grado di accuratezza.

5.4 API esterne utilizzate

Di seguito vengono descritte le varie API utilizzate per la realizzazione della maggior parte delle funzionalità di *News Analyzer*.

[3] https://it.pinterest.com/

5.4.1 Google Users API

Uno dei grossi vantaggi nell'utilizzo di Google App Engine è che risulta veramente facile effettuare l'**autenticazione** degli utenti e accedere quindi a email e nickname. L'email è di fondamentale importanza poiché, essendo univoca, può essere utilizzata come ID per l'interazione (salvataggio e caricamento di informazioni) col datastore.

5.4.2 Google OAuth2 e Google+

In *News Analyzer* viene utilizzato OAuth2 per consentire l'accesso ad alcune informazioni dell'account Google+. Vengono utilizzati l'email[4], il "display name" e l'immagine di profilo in modo da mostrare in ogni pagina una "navigation bar" molto simile a quelle dei servizi di Google (FIGURA).

5.4.3 FAROO

Risulta essere il solo servizio gratuito e allo stesso tempo performante che esporta delle Web API per la ricerca di notizie di tendenza. I risultati ritornati sono completi di molte informazioni. L'unico aspetto negativo è il limite di 1 richiesta/secondo (1.000.000 di richieste al mese) che limita fortemente l'utilizzo "concorrente" dell'applicazione da parte di più utenti. Per questo motivo, in caso di successo dell'applicazione, bisognerà salvare periodicamente nel datastore le notizie di tendenza, in questo modo ogni utente richiederà le notizie al datastore e non direttamente a FAROO, oppure spostarsi su un servizio commerciale a pagamento (Yahoo Boss, Bing Web Search API, ecc..).

5.4.4 Dandelion

Startup italiana specializzata in Semantics Analysis e Big Data. Mette a disposizione svariate API, tra le quali quella per l'estrazione delle entità (da un URL o da un testo) necessaria per realizzare una delle funzionalità *core* di News Analyzer. L'offerta base è gratis per un limite di 1000 richieste/giorno[5] di analisi.

Le entità possono essere estratte in base ad un grado minimo di accuratezza (l'entità viene scartata se non lo soddisfa), inoltre, ad ognuna delle entità vengono allegate informazioni aggiuntive prese da Wikipedia e DBpedia: una breve descrizione, un'immagine e le categorie di appartenenza. Quest'ultime sono fondamentali perché ci permettono di trattare in modo

[4] In realtà l'email è già disponibile tramite la Google Users API introdotta nel capitolo 5.4.1.
[5] Sarebbero 1000unità/giorno, ma l'estrazione delle entità da un testo viene conteggiato come 1unità per richiesta.

diverso le varie entità e mostrare informazioni di diverso tipo a seconda delle categoria di appartenenza.

5.4.5 Flickr

Di proprietà di Yahoo!, è una piattaforma online pubblica per la condivisione di fotografie personali. Le API sono molto semplici da utilizzare, mettono a disposizione tantissimi parametri e, nota molto importante, non presentano particolari restrizioni. In *News Analyzer* vengono caricate 5 foto alla volta in modo da non sovraccaricare il client ed è presente un bottone per caricare altre immagini se necessario.

5.4.6 Youtube

Famosa piattaforma per la condivisione e streaming video. Essendo un servizio Google, è facilmente utilizzabile all'interno di Google App Engine attraverso il pacchetto *"apiclient.discovery"*. Poiché i video sono oggetti molto pesati (in termini computazionali e di memoria), verrà visualizzato all'utente solo la thumbnail[6] e non verranno caricati finché l'utente non ci clicca sopra[7].

5.4.7 Google Maps Embed

Viene utilizzata la versione embedded di Google Maps per mostrare una mappa nel caso di entità di tipo "Posto/Luogo".

5.5 API esportate

Di seguito vengono elencate le Web API che sono state rese accessibili pubblicamente.

5.5.1 List users by entity

Descrizione: data un'entità è possibile ottenere, in formato JSON, la lista di utenti (loggati) che l'hanno estratta oggi.

Rotta: "/api/analyze/list_users_by_entity"

Argomenti: entity=[nome], nome dell'entità in codifica UTF-8.

Risposta: [(email1),(email2),(email3),..]

[6] L'immagine di anteprima del video.

[7] Per il "load after click" si è preso spunto dal seguente script: http://www.labnol.org/internet/light-youtube-embeds/27941/

5.5.2 List users by URL

Descrizione: dato l'URL di una notizia è possibile ottenere, in formato JSON, la lista di utenti (loggati) che l'hanno analizzata (possono comparire anche più di una volta) e il *timestamp* dell'analisi. Ammesso che l'utente non abbia rimosso l'analisi dalla cronologia. La lista può essere ordinata dal timestamp più recente a quello più vecchio con il corrispondente parametro aggiuntivo.

Rotta: "/api/analyze/list_users_by_url"

Argomenti: url=[url], stringa dell'url in codifica UTF-8.
order=[t/u], t=ordina per timestamp decrescenti, u=ordina per email,
defaut=no order

Risposta: [{email:(email),timestamp:(timestamp), ..}]

5.5.3 List top ten entities

Descrizione: ottenere la lista delle 10 entità più estratte oggi abbinate al numero di volte che sono state estratte.

Rotta: "/api/analyze/list_top_ten_entities"

Argomenti: nessuno.

Risposta: [[entity1, count1],[entity2, count2],..,[entity10, count10]]

5.6 Comunicazione asincrona

Il discreto numero di Web API delle quali la pagina di *analisi* usufruisce, in aggiunta al possibile grande numero di entità estrapolate da una notizia[8], potrebbe generare un elevato traffico di rete e la richiesta potrebbe richiedere molto tempo per essere soddisfatta. Per ovviare a questo problema, le informazioni specifiche di un'entità vengono ricavate solo quando esplicitamente richiesto dall'utente attraverso un *click* su di essa. A questo punto, il sistema provvederà ad effettuare delle richieste in background attraverso AJAX ad alcune delle Web API (sempre in base al tipo di entità), aggiornando la pagina con le risposte ricevute.

[8] Mediamente una notizia presenta tra le 15 e le 20 entità (poiché filtrate con il grado di accuratezza) ma non è raro trovare notizie con anche più di 30 entità.

In questo modo, oltre a ridurre il tempo di elaborazione di una richiesta da parte del server, viene fortemente ridotta la quantità di dati scambiati tra client e server (caratteristica fondamentale nel caso di dispositivi mobile) all'apertura della pagina di analisi.

5.7 CRON[9]

Come avevamo già anticipato precedentemente, dobbiamo programmare delle azioni automatiche nel tempo, sia per poter inviare le email con le news ai vari utenti, sia per eliminare quotidianamente le entità estratte in quel giorno.
In Google App Engine si può usufruire di questa funzionalità attraverso il file *cron.yaml*, nel quale vanno specificati (per ogni CRON):

- Una descrizione;
- Un URL;
- Una frequenza con la quale avviare il CRON;
- La timezone, ovvero rispetto a quale fuso orario.

Nel nostro caso avremo:

```
cron:
- description: submit filters emails
  url: /cron/submit_emails
  schedule: every 1 hours synchronized
  timezone: Europe/Rome
- description: clear today extracted entities
  url: /cron/clear_today_entities
  schedule: every 24 hours synchronized
  timezone: Europe/Rome
```

[9] I CRON sono, in parole povere, delle azioni programmate nel tempo.

6. Stima dei costi

Come abbiamo già accennato, il livello free-tier di Google App Engine presenta dei limiti rispetto alle risorse messe a disposizione e, di conseguenza, una volta sforati questi limiti, si passa nella "fascia" del *pay per use*. Dobbiamo inoltre considerare che anche le API utilizzate sono gratuite fino ad un certo numero di richieste e, in genere, a differenza di Google App Engine, presentano delle fasce di prezzi e non sono *pay per use*.

6.1 Riassunto dei costi

Prima di procedere alla stima dei costi, riassumiamo i limiti delle fasce gratuite e i prezzi dei vari servizi utilizzati.

6.1.1 Google App Engine

La fascia gratuita mette a disposizione (al giorno):

- 28 h-istanza;
- Datastore, 1GB di storage, 50.000 letture, scritture e "small operation";
- Network, 1GB di traffico in uscita;
- Mail, 100 chiamate alla Mail API (nel nostro caso una chiamata corrispondente all'invio di un email) al giorno ma può essere esteso fino a 1.700.000 se richiesto.

Mentre i costi aggiuntivi sono i seguenti:

- 0.05$ per ogni "ora-istanza" aggiuntiva[10];
- 0.18$ per ogni GB di storage aggiuntivo al mese;
- 0.06$ per 100.000 operazioni di lettura e scrittura aggiuntive;
- 0.12$ per ogni GB di traffico in uscita aggiuntivo al giorno;
- Le email non prevedono costi aggiuntivi.

6.1.2 Google APIs

Google Maps Javascript API presenta un limite di 25.000 mappe al giorno per 90 giorni consecutivi e 0.50$ per ogni 1000 mappe aggiuntive al giorno.

[10] In realtà esistono tante classi di istanze con prezzi differenti a seconda delle prestazioni. Noi per semplicità consideriamo il modulo base F1 con 256MB di memoria e una CPU da 600Mhz.

Con Youtube abbiamo a disposizione 50.000.000 di unità al giorno ed effettuiamo delle operazioni di ricerca prelevando il solo ID dei video per un costo totale di 101 unità.

6.1.3 Altre API

Flickr è gratuito e non presenta restrizioni per le applicazioni senza scopo di lucro (non commerciali) e non verrà quindi considerato nelle stime. Stessa questione per FAROO che è anch'esso gratuito (con delle restrizioni descritte nel capitolo 5.4.3).

Dandelion offre 1000 unità al giorno di base gratuite mentre le fasce a pagamento prevedono 2000, 10.000, 15.000 e 75.000 unità al giorno per, rispettivamente, 49$, 99$, 239$ e 749$ al mese. Dobbiamo considerare che ogni operazione di estrazione delle entità costa in media 1,5 unità[11].

6.2 Stime ad intensità crescente

Andiamo quindi ad effettuare una stima dei costi per mantenere attivo il servizio, considerando 3 diversi scenari ad intensità crescente del numero di utenti giornalieri: 10, 100 e 1000. Assumeremo, per semplicità, che tutti gli utenti siano utenti autenticati e che la cronologia delle analisi e i preferiti non vengano mai cancellati.

Basandosi sui valori stimati nel modello di workload descritto nel capitolo 3, andiamo a stimare gli utilizzi di un singolo utente "intermedio" autenticato:

- 20 minuti-istanza (0.33 h-istanza) al giorno[12];
- 3,75 KB al giorno per il salvataggio delle entità estratte (capitolo 5.5 sulle API esportate), ottenuti da 5 notizie analizzate al giorno che comportano un salvataggio di 0,75 KB, considerando che ognuna di queste è mediamente costituita da 15 entità da 50 byte[13];
- 1 KB al giorno per mantenere la cronologia delle analisi, considerando 5 analisi al giorno e 200 byte per ogni *entry* della cronologia;
- 1 KB al giorno per il salvataggio di preferiti, dato da 500 byte per ogni preferito moltiplicato per 2 preferiti al giorno;

[11] L'estrazione costa 1 unità ogni 4.000 caratteri presenti nel testo analizzato e abbiamo stimato una media di 6.000 caratteri per notizia, quindi 1,5 unità per notizia.

[12] In realtà, sarebbe molto difficile stimare le h-istanza per utente poiché c'è un costo per avvio dell'istanza pari a 15 minuti e inoltre mentre un'istanza è attiva può servire più richieste contemporaneamente.

[13] Non consideriamo il fatto che più utenti potrebbero analizzare la stessa notizia e questo non comporterebbe una nuova scrittura sul database dopo la prima.

- 2 KB di filtri, dato da una media di 10 filtri da 200 byte;
- 400 scritture e 200 letture sul datastore[14];
- 2 MB di traffico in uscita al giorno verso ogni utente, considerando 200 KB per ogni pagina di analisi, 50 KB ogni 10 notizie visualizzate nella homepage, 50 KB per la pagina dei preferiti e 10 KB per la pagina dei filtri;
- 2 email al giorno.

Infine, facciamo notare che le unità fornite da Youtube e Google Maps sono sufficienti anche in caso di 1000 utenti[15]. Di conseguenza, la sola API che andremo a considerare, nei vari scenari, è quella di Dandelion.

6.2.1 10 utenti al giorno

Con un totale di 10 utenti al giorno avremmo:

- 3h20m-istanza di utilizzazione;
- 57,5KB di spazio occupato sul datastore, di cui 20KB di filtri e 37,5KB per le entità[16];
- 20 KB di scritture sul datastore, di cui 10 KB per i preferiti e 10 KB per la cronologia delle analisi;
- 4.000 scritture e 2.000 letture sul datastore;
- 20 MB di traffico in uscita;
- 20 email al giorno.

Tutti questi valori sono ampiamente dentro i limiti del free-tier di Google App Engine e, anche senza mostrare calcoli, si può notare che tutte le API rimarrebbero entro i vincoli della fascia gratuita. I 20 KB scritti quotidianamente sul datastore porterebbero ad esaurire il GB di spazio gratuito in 50.000 giorni, circa 136 anni.
Il costo totale per 10 utenti attivi al giorno sarebbe pari a 0$.

6.2.2 100 utenti al giorno

Con un totale di 100 utenti al giorno avremmo:

- 33h20m-istanza di utilizzazione;

[14] Le scritture sono molto alte poiché sono presenti 2 indici in ciascuna delle 2 entità (lista entità e cronologia analisi) create ad hoc per servire i CRON dell'applicazione.
[15] Youtube: (50.000.000unità / 1000utenti) / 101unità/richiesta = 495 richieste/utente
Google Maps: 25.000 / 1000utenti = 25 mappe/utente
[16] Questo valore è fisso e non si accumula per natura dei dati considerati (ad esempio, le entità vengono rimosse quotidianamente)

- 575 KB di spazio occupato sul datastore, di cui 200 KB di filtri e 375 KB per le entità;
- 200 KB di scritture sul datastore, di cui 100 KB per i preferiti e 100 KB per la cronologia delle analisi;
- 40.000 scritture e 20.000 letture sul datastore;
- 200 MB di traffico in uscita;
- 200 email al giorno.

Il solo valore oltre i limiti risulta essere quello delle h-istanza che sfora di 5h20m oltre le 28h di base per un costo totale di 0,27$ al giorno. Con 200 KB al giorno, finiremmo il GB di spazio gratuito del datastore in 5.000 giorni, circa 14 anni. Andrebbe inoltre richiesto un aumento del numero di email che è possibile inviare al giorno.
Per quanto riguarda le API di Dandelion, su un totale di 1000 unità, ogni utente avrebbe a disposizione 10 unità e, ricordando che ogni analisi richiede in media 1,5 unità, potrebbe effettuare un massimo di 6 analisi. In questo caso, potremmo sforare facilmente il limite con un picco di richieste di analisi. Questo comporterebbe un disservizio fino all'inizio del giorno successivo (si resetta alle 00:00), quindi, per evitare inconvenienti, converrebbe fare l'upgrade a 2000 unità al giorno per 49$ al mese (1,63$ al giorno).
Il costo totale con 100 utenti attivi quotidianamente sarebbe di circa 57$ al mese (1,90$ al giorno).

6.2.3 1000 utenti al giorno

Con un totale di 1000 utenti al giorno avremmo:

- 333h20m-istanza di utilizzazione;
- 5,75 MB di spazio occupato sul datastore, di cui 2 MB di filtri e 3,75 MB per le entità;
- 2 MB di scritture sul datastore, di cui 1 MB per i preferiti e 1 MB per la cronologia delle analisi;
- 400.000 scritture e 200.000 letture sul datastore;
- 2 GB di traffico in uscita;
- 2000 email al giorno.

In questo caso, sforiamo di 305h20m-istanza per un totale di circa 15,25$ al giorno. A questi si aggiungono i 0,12$ per il GB aggiuntivo di traffico in uscita. Il GB di storage verrebbe

esaurito in 500 giorni con 2 MB di scritture al giorno. Considerando che un solo GB aggiuntivo al mese costa solo 0.18$, per i primi mesi possiamo trascurare il costo che genererebbe, magari dopo qualche anno potrebbe diventare consistente e andrebbe preso in considerazione. Infine, dobbiamo aggiungere le 350.000 scritture e 150.000 letture che corrispondono a 0,21$ e 0,09$ al giorno, circa 6$ e 3$ al mese.
Dandelion presenta una differenza di prezzo notevole tra le 10.000 e le 15.000 unità al giorno, che costano rispettivamente 99$/mese e 249$/mese. Andrebbe quindi valutata bene la situazione, magari anche con delle valutazioni statistiche di utilizzo della funzione di analisi. Secondo le stime effettuate, per garantire il servizio a 1000 utenti al giorno, si arriverebbe a pagare circa 718$/mese (460$ + 249$ + 9$).

7. Sviluppi futuri

In questa sezione, andremo prima a delineare qualche possibile miglioramento da apportare all'applicazione e, successivamente, indicheremo qualche possibile aggiunta.

7.1 Miglioramenti

Come si è potuto notare nel capitolo precedente, l'utilizzo della API di Dandelion costituisce la maggior parte del costo per mantenere l'applicazione attiva con 1000 utenti. Per far fronte a questo problema, si potrebbero salvare sul datastore tutte le informazioni estratte da una notizia (identificata con un URL), ovviamente, solo se questa non è già presente. Le richieste successive di analisi, rispetto alla stessa notizia (allo stesso URL), faranno riferimento ai dati sul datastore e non effettueranno richieste a Dandelion. Del resto, le notizie, per loro natura, hanno molte più possibilità di essere analizzate in un breve periodo successivo alla pubblicazione. Di conseguenza, potremmo diminuire notevolmente le richieste dirette alle API di Dandelion e, quindi, diminuire il costo totale. L'aumento delle scritture sul datastore non dovrebbe portare a nessun costo, poiché l'utilizzo totale era ampiamente dentro i limiti anche con 1000 utenti. Infine, le dimensioni del datastore potrebbero costituire un problema dopo un po' di tempo, quindi si potrebbero eliminare le notizie dopo una settimana dall'ultima analisi.

E' presente un altro grosso problema nell'applicazione dovuto all'utilizzo di AJAX e alla possibilità di utilizzare la piattaforma anche senza autenticazione. Infatti, per poter effettuare delle richieste asincrone in background, sono state create delle "API interne" (in realtà sono pubbliche, per natura dell'applicazione) le quali fanno da ponte verso le API utilizzate dall'applicazione. Ad esempio, la rotta "/api/flickr", se interrogata con i giusti parametri, andrà ad effettuare le opportune richieste alle API di Flickr e restituirà una risposta JSON. E' stata adottata questa soluzione, per evitare di scrivere in chiaro, sulla pagina web, le chiavi private utilizzate per accedere alle API. Quindi, chiunque potrebbe accedere alle mie API ed effettuare delle richieste "senza controllo".
Per evitare questo problema, si potrebbe, ad esempio, limitare l'accesso alle funzioni di *News Analyzer* ai soli utenti autenticati, oppure limitare le richieste che ogni utente non autenticato può effettuare al secondo e al giorno.

7.2 Aggiunte

Non c'è limite al numero di funzioni che potrebbero essere aggiunte all'applicazione, ma di seguito ne elenchiamo qualcuna che potrebbe portare ad un miglioramento della user experience generale e/o ad un ampliamento del target di utenti.

Si potrebbe aggiungere il supporto a molte più lingue (per ora c'è solo l'inglese), magari utilizzando i Feed RSS di Google o di altre API per le notizie e cercare qualche altro servizio per l'estrazione delle entità (Alchemy API).

Si potrebbero aggiungere altre categorie di entità (Arte, cibo, ecc..) e cercare altre API per fornire più informazioni specifiche nella pagina di analisi.

L'integrazione con altri social network (Twitter, Facebook, ecc..) porterebbe di sicuro ad un aumento del numero di utenti. Non tutti utilizzano Google o vogliono dare i permessi di accesso al profilo Google+.

8. Conclusioni

Molto probabilmente *News Analyzer* non diventerà mai un'applicazione famosa o frequentemente utilizzata, ma la sua creazione e la stesura di questo documento hanno contribuito a creare una bella esperienza. I temi che sono stati affrontati sono tanti e alcuni di essi sono di notevole importanza nel mondo di oggi: il cloud, le Web API, i servizi.

Il confronto delle alternative tra le piattaforme PaaS è stato abbastanza interessante. Ma l'argomento che più mi ha colpito è stato quello della stima dei costi, che non avevo mai affrontato prima e non ritenevo così complicato. Nel nostro caso, sono state fatte molte assunzioni e sono stati dati per scontati certi aspetti, cose che in progetti reali andrebbero gestite meglio con conseguente aumento della complessità.

Sono quindi felice del risultato ottenuto e dell'esperienza in generale.