

Applicazione Cloud per il Monitoraggio dei BigData nei Social Network

Introduzione

In un mondo pieno di messaggi social, proviamo a fare un po' di ordine e rendere l'esperienza dell'accesso ai contenuti più fruibile.

L'obiettivo è creare da zero una infrastruttura che permetta di interrogare big data dai social Facebook, Twitter e Instagram. L'applicazione sarà composta di una prima parte che offre la consultazione e l'interrogazione con interfaccia web per utente, e una seconda parte che offra dei servizi REST interrogabili. Per quanto riguarda i dati da interrogare lasciamo piena libertà agli studenti: ad esempio può essere interessante indicizzare messaggi contenenti un determinato hashtag su Facebook, Twitter e Instagram.

Sarà quindi necessario progettare una infrastruttura web scalabile. Proponiamo di utilizzare lo stack tecnologico della [Google Cloud Platform](#). Linguaggi di programmazione disponibili: Java, Php, Python. Piena libertà per l'implementazione dell'interfaccia (consigliato: HTML5, CSS3, jQuery, meglio se utilizzando un framework responsive come [Twitter Bootstrap](#)) e dei servizi Web (consigliato: [Google Endpoints](#)).

Zing S.r.l. si occuperà di illustrare l'utilizzo degli strumenti sopra elencati per progettare e realizzare una piattaforma web che sfrutti al meglio le potenzialità del cloud.

Tecnologie

Google Cloud Platform

La Google Cloud Platform (<https://cloud.google.com/>) è uno stack tecnologico composto da una serie di prodotti pensati per il supporto allo sviluppo nel cloud. Tali prodotti hanno tutti una versione di utilizzo gratuita e includono una interfaccia web (<https://console.developers.google.com>), sdk per l'uso da riga di comando e [REST API](#).

Tra i prodotti utili al progetto troviamo:

- [Google App Engine](#)
 - una [Platform as a Service](#) ideale per applicazioni web scalabili. App Engine riesce a scalare automaticamente al crescere delle risorse richieste e gestisce automaticamente il carico sui server.
- [Google Compute Engine](#)
 - una [Infrastructure as a Service](#) che abilita l'utente al lancio di macchine virtuali (VMS) on demand.
- [Google Cloud Storage](#)
 - servizio scalabile per lo storage di file online.
- [Google Cloud Datastore](#)
 - database NoSQL ad alte prestazioni
- [Google Cloud SQL](#):
 - database MySQL
- [Google BigQuery](#)
 - tool per l'analisi dei dati che utilizza query SQL-like per processare big data in pochi secondi
- [Google Cloud Endpoints](#)
 - strumento per creare web services in App Engine che possono essere utilizzati con iOS, Android e client JavaScript.

Main

Dashboard

[Instances](#)

[Logs](#)

[Versions](#)

[Cron Jobs](#)

[Task Queues](#)

[Quota Details](#)

Data

[Datastore Indexes](#)

[Datastore Viewer](#)

[Datastore Statistics](#)

[Blob Viewer](#)

[Prospective Search](#)

[Text Search](#)

[Datastore Admin](#)

[Memcache Viewer](#)

Administration

[Application Settings](#)

[Permissions](#)

[Blacklist](#)

[Admin Logs](#)

Billing

[Billing Status](#)

[Usage History](#)

Resources

[Documentation](#)

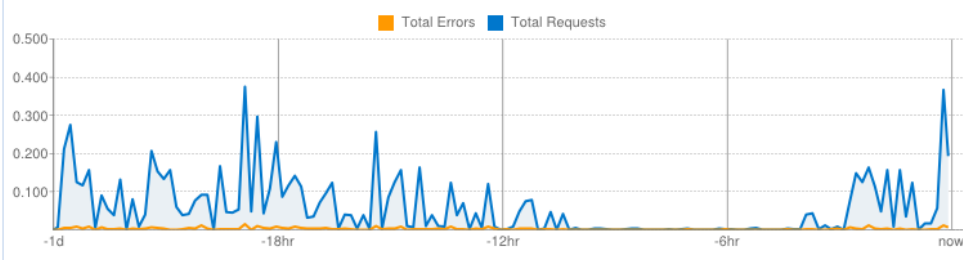
[FAQ](#)

Version: **1 (Default)**

Charts

Summary

30 mins 3 hrs 6 hrs 12 hrs **24 hrs** 2 days 4 days 7 days 14 days 30 days



Instances

App Engine Release	Total number of Instances	Average QPS*	Average Latency*	Average Memory
1.9.14	1 total	0.000	Unknown ms	56.2 MBytes

Billing Status: Enabled (Daily budget: \$50.00) - [Settings](#)

Quotas reset every 24 hours. Next reset: 21 hrs

Resource	Usage	Billable	Price	Cost
Frontend Instance Hours	11.21 Instance Hours	0.00	\$0.05/ Hour	\$0.00
Backend Instance Hours	0.00 Instance Hours	0.00	\$0.05/ Hour	\$0.00
Datastore Stored Data	0.00 GBytes	0.00	\$0.006/ GByte-day	\$0.00
Logs Stored Data	0.06 GBytes	0.00	\$0.0009/ GByte-day	\$0.00
Task Queue Stored Task Bytes	0.00 GBytes	0.00	\$0.0009/ GByte-day	\$0.00

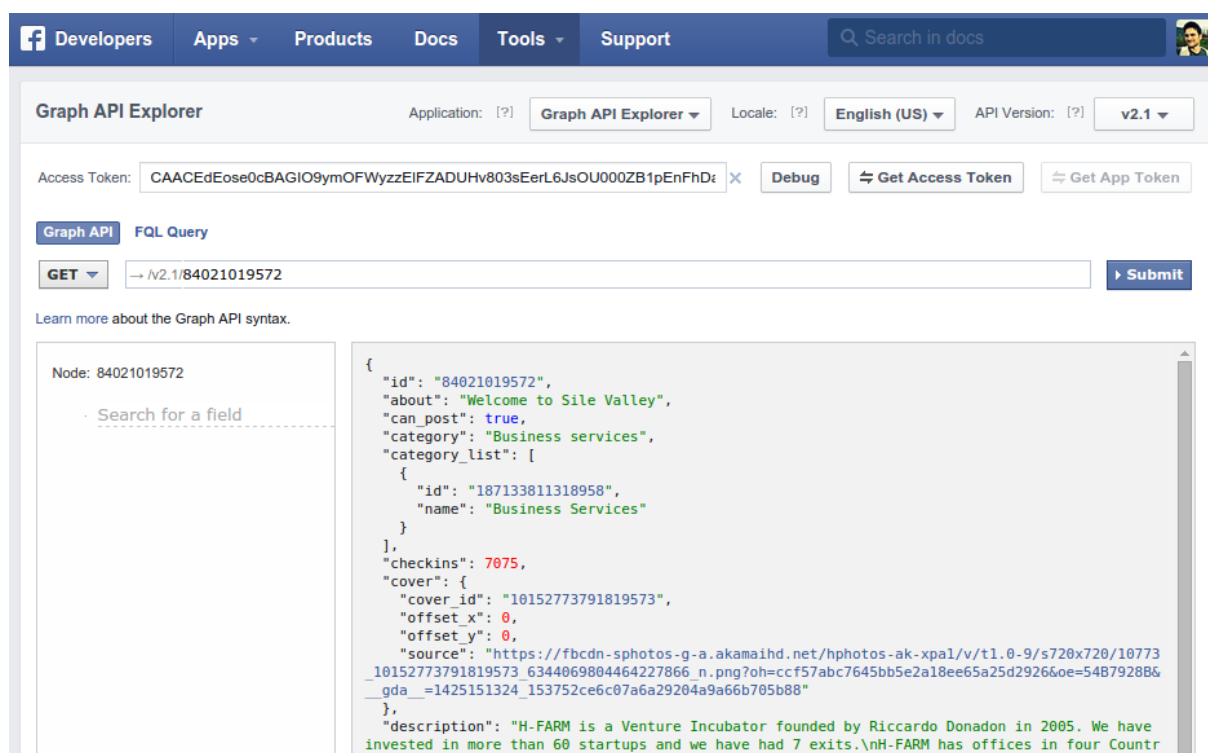
Api Facebook

The primary way for apps to read and write to the Facebook social graph. The Graph API has multiple versions available, read about [what has changed](#) and how to [upgrade from older versions](#).

Graph API Reference <https://developers.facebook.com/docs/graph-api/reference/v2.1>

How to use Graph API <https://developers.facebook.com/docs/graph-api/using-graph-api/v2.1>

Graph API tools <https://developers.facebook.com/tools/explorer/>



Graph API Explorer

Application: [?] Graph API Explorer Locale: [?] English (US) API Version: [?] v2.1

Access Token: CAACEdEose0cBAGIO9ymOFWyzzEIFZADUHV803sEerL6JsOU000ZB1pEnFhDz Debug Get Access Token Get App Token

Graph API FQL Query

GET → /v2.1/84021019572 Submit

Learn more about the Graph API syntax.

Node: 84021019572

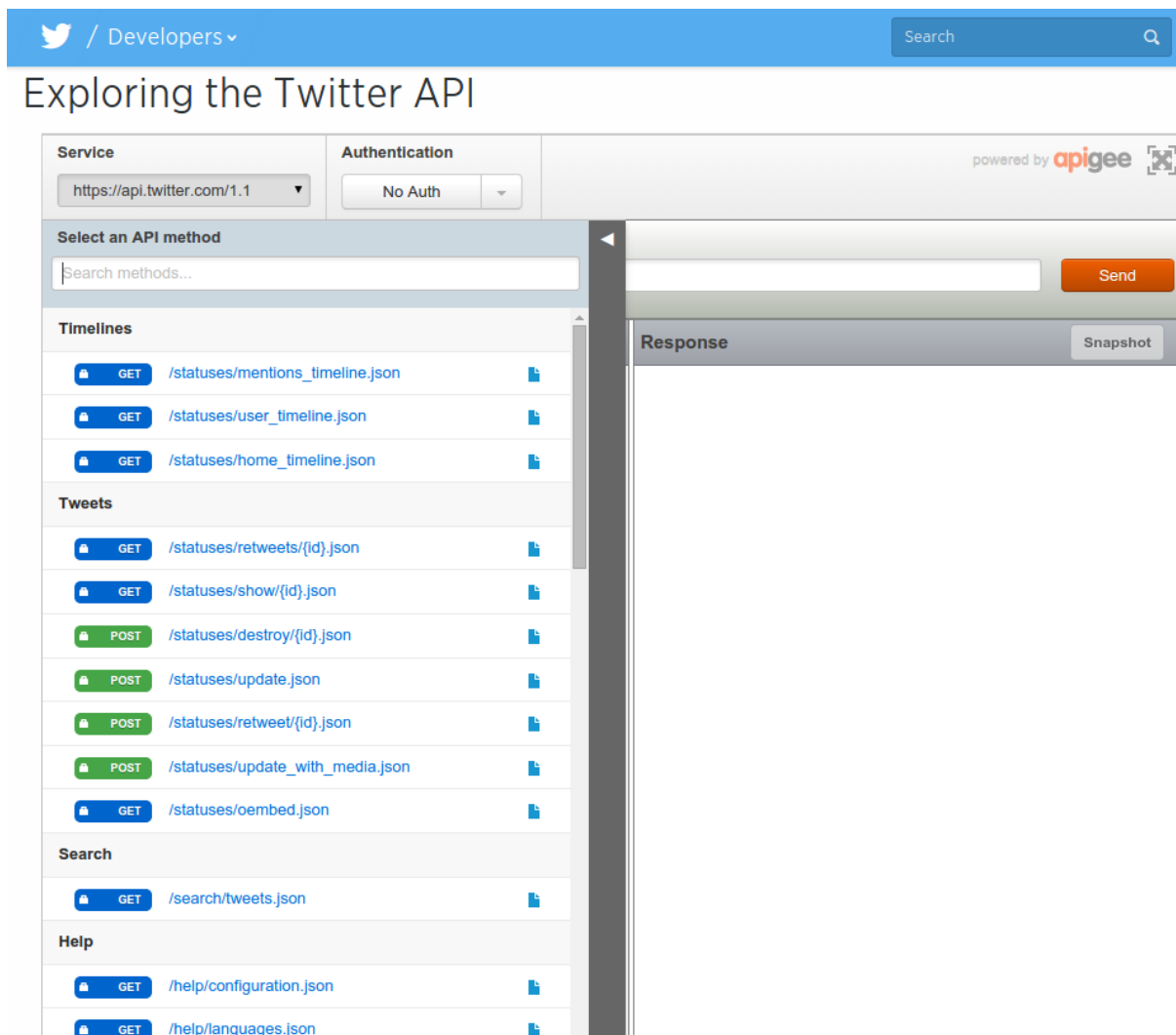
Search for a field

```
{
  "id": "84021019572",
  "about": "Welcome to Sile Valley",
  "can_post": true,
  "category": "Business services",
  "category_list": [
    {
      "id": "187133811318958",
      "name": "Business Services"
    }
  ],
  "checkins": 7075,
  "cover": {
    "cover_id": "10152773791819573",
    "offset_x": 0,
    "offset_y": 0,
    "source": "https://fbcdn-sphotos-g-a.akamaihd.net/hphotos-ak-xpal/v/t1.0-9/s720x720/10773_10152773791819573_6344069804464227866_n.png?oh=ccf57abc7645bb5e2a18ee65a25d29266oe=54B7928B&_gda__=1425151324_153752ce6c07a6a29204a9a66b705b88"
  },
  "description": "H-FARM is a Venture Incubator founded by Riccardo Donadon in 2005. We have invested in more than 60 startups and we have had 7 exits.\nH-FARM has offices in four Countries: Italy, Germany, France, UK, Spain, USA, Canada, UK, and Mexico. (Total 10 offices)"
}
```

Api Twitter

Twitter API Overview <https://dev.twitter.com/overview/api>

Twitter API Explorer <https://dev.twitter.com/rest/tools/console>

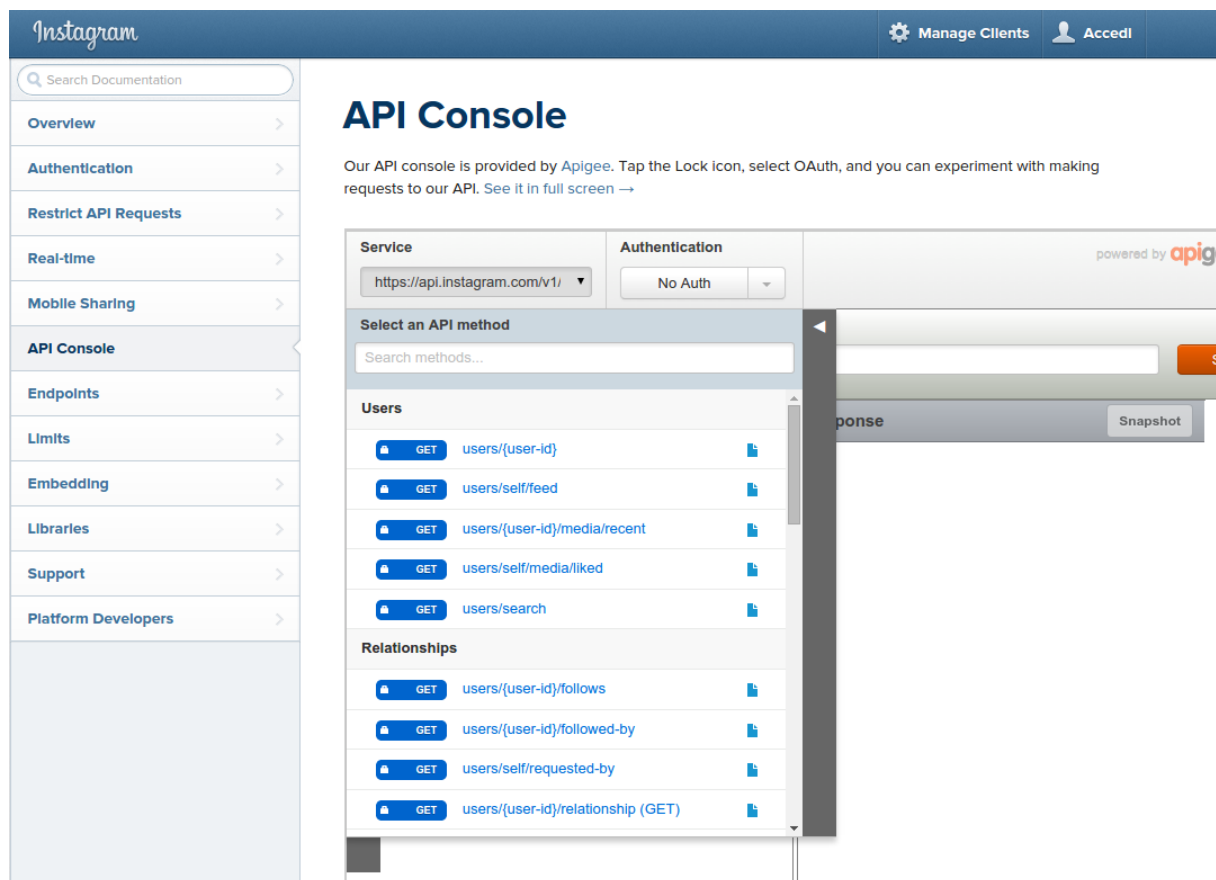


The screenshot shows the Twitter API Explorer interface. At the top, there's a blue header with the Twitter logo, a dropdown menu set to 'Developers', and a search bar. Below the header, the title 'Exploring the Twitter API' is displayed. The main interface is divided into two panels. The left panel, titled 'Service', shows the API endpoint 'https://api.twitter.com/1.1' and an 'Authentication' dropdown set to 'No Auth'. Below this is a 'Select an API method' section with a search bar. The methods are categorized into 'Timelines', 'Tweets', 'Search', and 'Help'. The 'Timelines' section lists three GET methods: '/statuses/mentions_timeline.json', '/statuses/user_timeline.json', and '/statuses/home_timeline.json'. The 'Tweets' section lists seven methods: three GET methods ('/statuses/retweets/{id}.json', '/statuses/show/{id}.json', and '/statuses/oembed.json') and four POST methods ('/statuses/destroy/{id}.json', '/statuses/update.json', '/statuses/retweet/{id}.json', and '/statuses/update_with_media.json'). The 'Search' section lists one GET method: '/search/tweets.json'. The 'Help' section lists two GET methods: '/help/configuration.json' and '/help/languages.json'. The right panel, titled 'Response', shows a 'Send' button and a 'Snapshot' button. The interface is powered by Apigee.

Api Instagram

Instagram API overview <http://instagram.com/developer/>

Instagram API console <http://instagram.com/developer/api-console/>



The screenshot shows the Instagram API Console interface. The top navigation bar includes the 'Instagram' logo, a 'Manage Clients' gear icon, and an 'Acced' user icon. A search bar for documentation is located on the left. A sidebar menu on the left lists various API sections: Overview, Authentication, Restrict API Requests, Real-time, Mobile Sharing, API Console (highlighted), Endpoints, Limits, Embedding, Libraries, Support, and Platform Developers. The main content area is titled 'API Console' and contains a message: 'Our API console is provided by Apigee. Tap the Lock icon, select OAuth, and you can experiment with making requests to our API. See it in full screen →'. Below this, there are two tabs: 'Service' and 'Authentication'. The 'Service' tab is active, showing a dropdown menu for the service URL (https://api.instagram.com/v1/). The 'Authentication' tab is set to 'No Auth'. A 'Select an API method' section is visible, with a search bar and a list of methods categorized under 'Users' and 'Relationships'. The 'Users' category includes methods like users/{user-id}, users/self/feed, users/{user-id}/media/recent, users/self/media/liked, and users/search. The 'Relationships' category includes methods like users/{user-id}/follows, users/{user-id}/followed-by, users/self/requested-by, and users/{user-id}/relationship (GET). A 'powered by apigee' logo is visible in the top right corner of the console area.

Sviluppo del sistema

Contesto del prodotto

Fissiamo come obiettivo la realizzazione di un'applicazione web che permetta all'utente che dispone delle autorizzazioni necessarie di interrogare una vastissima mole di dati recuperata utilizzando le API delle più diffuse piattaforme social. La natura dei dati raccolti è a discrezione degli studenti, purché vengano scelti secondo un criterio specifico di utilità (eg. dati necessari a monitorare la popolarità di brand sui social, dati utili a verificare la distribuzione di attività in una certa area, frequenza di hashtag, etc.)

Funzionalità dell'applicazione web

Possono essere individuate quattro principali parti del progetto:

Data mining

Il recupero dei dati tramite l'interrogazione via API dei social network deve essere puntuale, continua e per quanto possibile automatizzata.

Elaborazione e persistenza

I dati raccolti dovranno essere elaborati e salvati nella base dati. Viene concessa massima libertà nella fase di analisi, progettazione e sviluppo della parte applicativa, a patto che queste scelte vengano successivamente giustificate dal tipo di servizio che l'applicazione offrirà.

Aggiornamento periodico e continuo dei dati raccolti

Un processo dedicato dell'applicazione dovrà occuparsi dell'aggiornamento periodico dei dati raccolti nel caso in cui essi non siano persistenti ma variabili, in modo da poter esporre tramite i servizi dell'applicazione dati il più possibile aggiornati. A tal fine App Engine offre funzionalità di schedulazione dei task (cron) nativamente integrato.

Esporre dati tramite API e interfaccia web

Sarà necessario creare servizi web ad hoc per esporre e interrogare l'applicazione ai fini di aggiungere, modificare, visualizzare e cancellare i dati raccolti. Per rendere disponibili alla fruizione i dati raccolti all'utente finale sarà necessario inoltre realizzare un pannello di gestione web, con accesso privato alle sezioni che lo richiedono.

Vincoli generali

Consentiamo piena libertà nella selezione dei dati da recuperare dai social network e nella scelta di come sviluppare l'architettura a patto che le scelte strutturali siano coerenti con il tipo di servizio offerto dall'applicazione. Ad esempio, è opportuno giustificare le scelte fatte riguardo alla persistenza, specificando i motivi per cui si è scelta una base dati di tipo non relazionale o relazionale. È inoltre opportuno illustrare la frequenza delle interrogazioni alle API dei social network, fatte lato server, in fase di costruzioni della base di dati: perché aggiornare i dati in database con cadenza giornaliera o settimanale? Perché scegliere una specifica soglia per limitare le richieste?

Documentazione

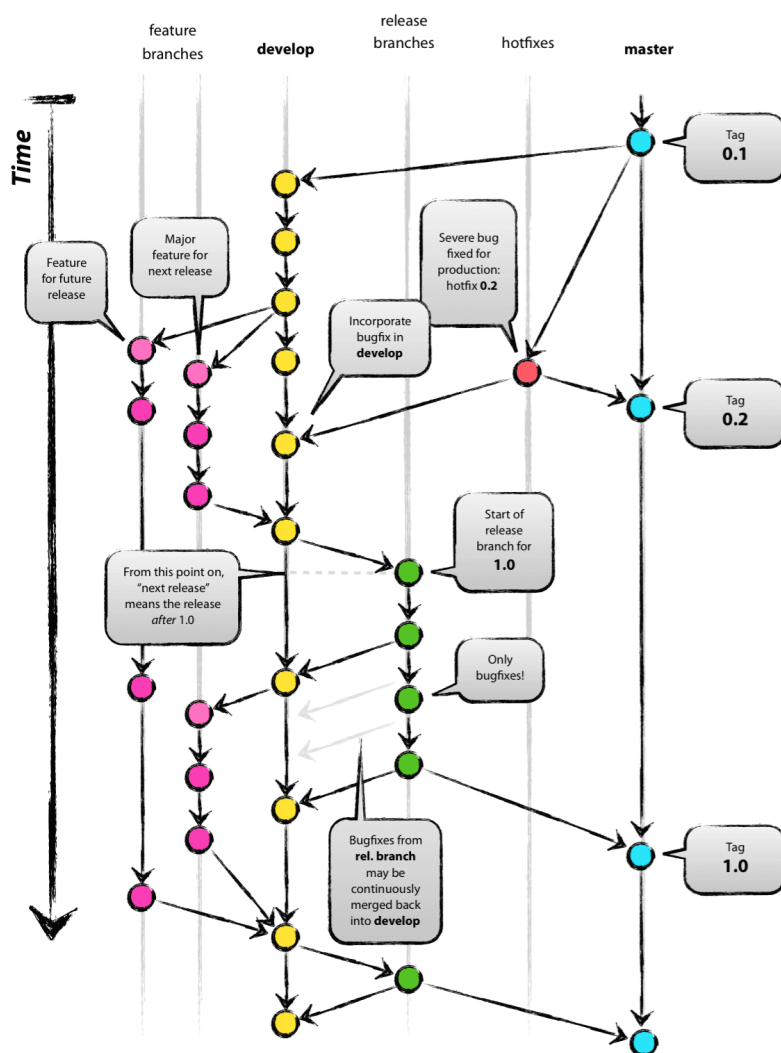
Oltre al codice prodotto in formato sorgente, sono richiesti i seguenti documenti secondo la tempistica e le scadenze da concordare con il committente.

Materiale da consegnare prima della messa in produzione del sistema

- Modello Logico e diagramma a oggetti, diagramma di sequenza delle principali operazioni che saranno svolte sul sistema, schema della base dati utilizzato dagli studenti per la realizzazione del progetto
- Documentazione dettagliata di tutte le API, che descrive la firma dei servizi che si andranno ad implementare sia via servizio web che tramite interfaccia utente.
- Piano dei test di unità che descrive le operazioni necessarie per verificare il corretto funzionamento del sistema.

Materiale da consegnare in fase di esercizio del sistema

- Bug reporting al fine di tenere traccia delle anomalie riscontrate e sul loro stato di risoluzione.
- Utilizzo di sistemi di versionamento del codice (git, mercurial) e branching (vedi immagine)
- utilizzo di repository online (github, bitbucket) per approcciare allo sviluppo in modo sicuro, condiviso e concorrente.



Un modello efficace di branching (source: <http://nvie.com/posts/a-successful-git-branching-model>)

I rilasci in produzione del software applicativo ed ogni modifica della configurazione del sistema nella sua versione in produzione deve essere effettuata nella collaborazione reciproca del fornitore e del committente.

Si intende che gli sviluppatori del prodotto oggetto di questo capitolato ne ritengono il copyright intellettuale e materiale.

Per consentire la diffusione, maturazione ed eventualmente l'uso del risultato, si richiede di farne comunicazione all'indirizzo di posta elettronica info@zing-store.com.