WSIL (What Should I Learn)

Corso di Laurea Magistrale in Informatica
Integrazione Dati su Web

Professore	Studenti
Prof. Gennaro Costagliola	Mattia Tomeo
	Dario Di Pasquale
	Fabio Napoli

Anno Accademico Accedemico 2016-2017

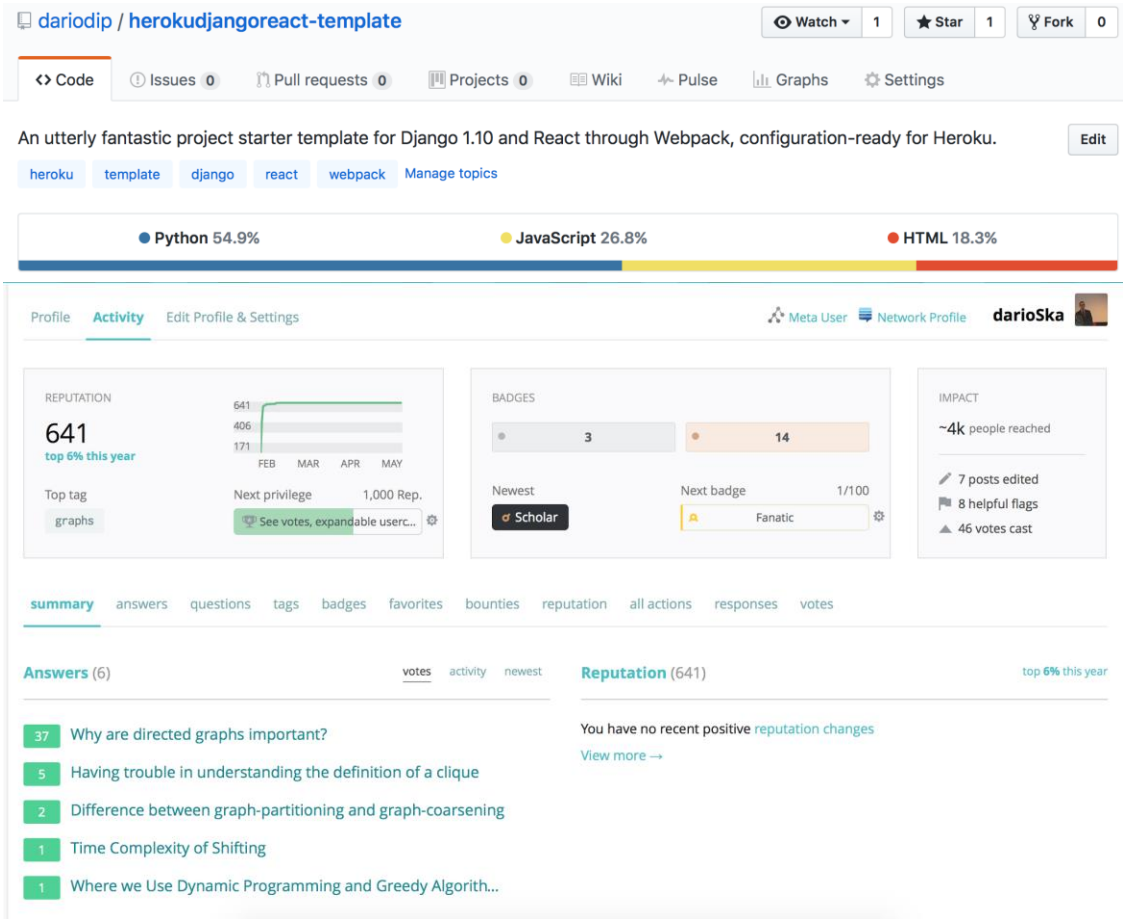
1. Sommario

2. Introduzione

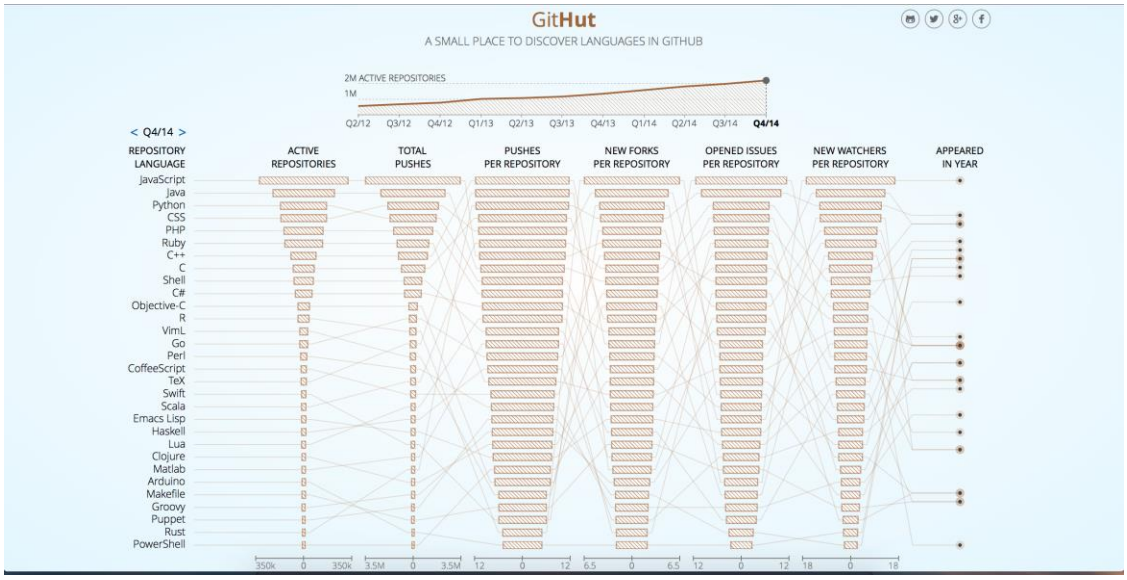
Con l'avanzare del progresso tecnologico, il lavoro dello sviluppatore diventa, paradossalmente, sempre più difficile.
Rispetto al passato in cui esistevano pochi linguaggi, anche se molto meno astratti e con una curva di apprendimento molto più alta, al giorno d'oggi esistono diverse centinaia di linguaggi di programmazione con diverse migliaia di librerie e framework basati su di essi.
Un odierno sviluppatore rischia di perdersi in un dedalo di nuovi linguaggi di programmazione e nuove tecnologie emergenti che scalano le vette del mercato e diventano sempre più richieste, per periodi di tempo sempre più variabili, mentre talvolta tecnologie vecchie, sebbene consolidate, tramontano.
Affacciarsi al mondo del lavoro con le giuste competenze, al momento giusto, risulta un requisito di *vitale* importanza per qualsiasi professionista. Tuttavia, fare una previsione su qual è e quale sarà la tecnologia più richiesta risulta talvolta impossibile.

3. Descrizione del problema

In rete, vista l'enorme mole di dati ivi presenti, è possibile trovare numerosissime piattaforme che offrono la possibilità di ricavare informazioni e statistiche circa l'utilizzo di svariati linguaggi di programmazione e delle relative tecnologie.
Al tempo stesso, al fine di rispondere alla sempre crescente necessità di aggiornamento da parte degli sviluppatori, sempre più siti offrono la possibilità di seguire online corsi riguardanti diversi argomenti relativi allo sviluppo software e web.
Anche molte piattaforme, il cui principale scopo è quello di offrire servizi per gli sviluppatori, offrono la possibilità di recuperare importanti statistiche circa i diversi linguaggi utilizzati. Tra queste, in primis, troviamo GitHub e StackOverflow.
Come si può vedere dalle immagini sottostanti, GitHub permette di ricavare, per ogni repository, il linguaggio o i linguaggi di programmazione utilizzati; mentre StackOverflow associa alle varie domande una serie di tag che possono essere utilizzati per ricavare il numero di domande e risposte relative ad una determinata tecnologia, identificata da uno o più tag.

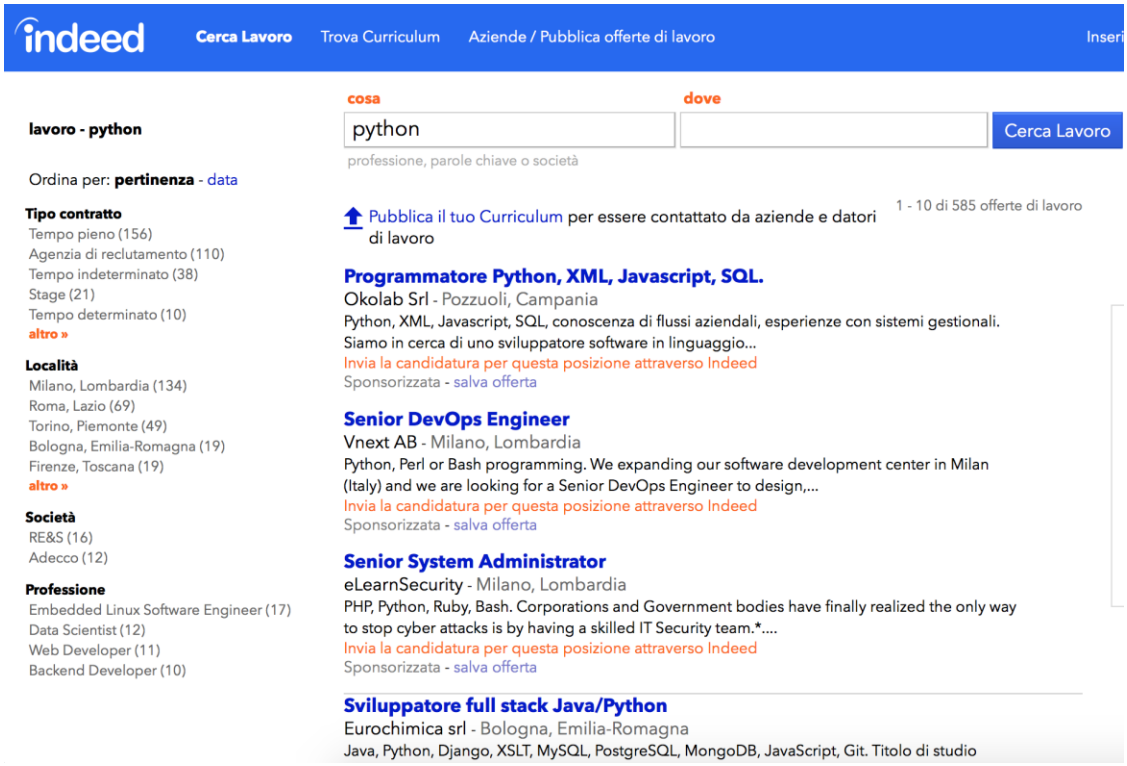


Il sito GitHub Archive(si veda l'immagine seguente) memorizza la timeline di GitHub e la rende disponibile per analisi future:

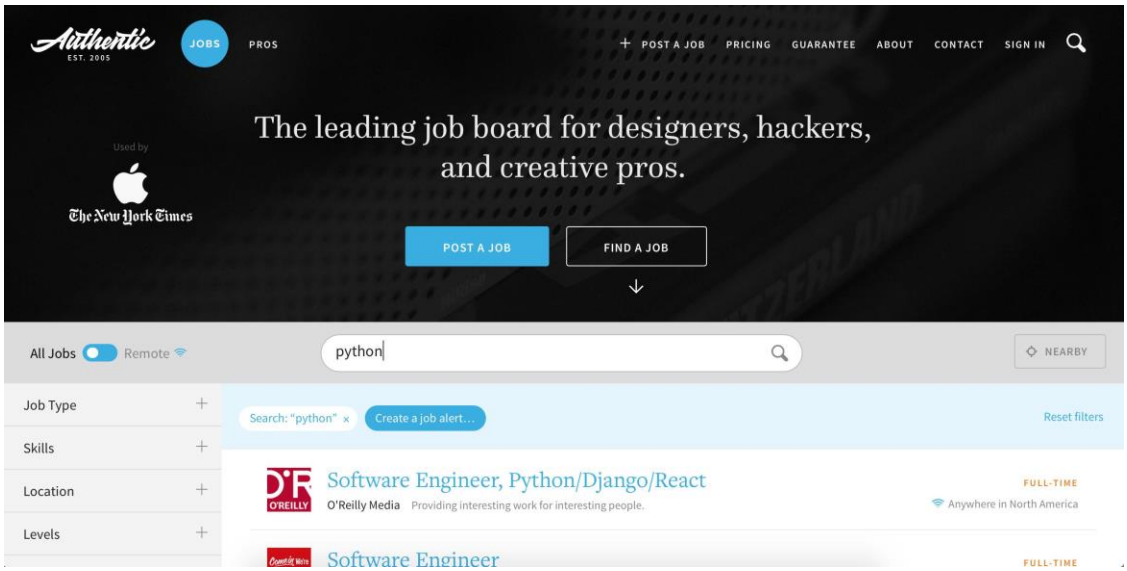


GitHut

Anche le piattaforme di recruitment e ricerca di lavoro nel settore dello sviluppo software sono in costante crescita e visitarle può favorire l'utente nel ricavare informazioni utili riguardo i linguaggi e i framework più ricercati nel mondo del lavoro.
Le piattaforme più utilizzate per la ricerca del lavoro sono Indeed e Authentic Jobs (si vedano le immagini sottostanti).

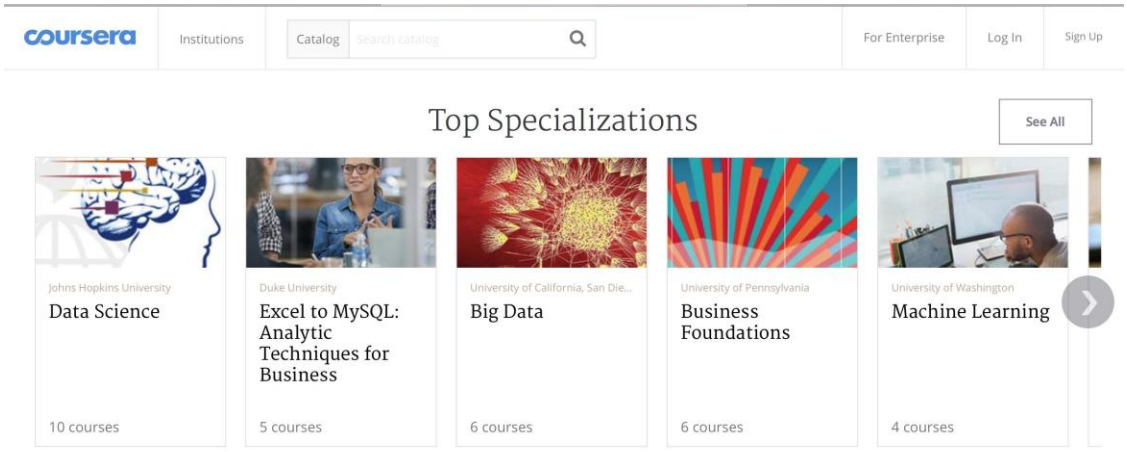


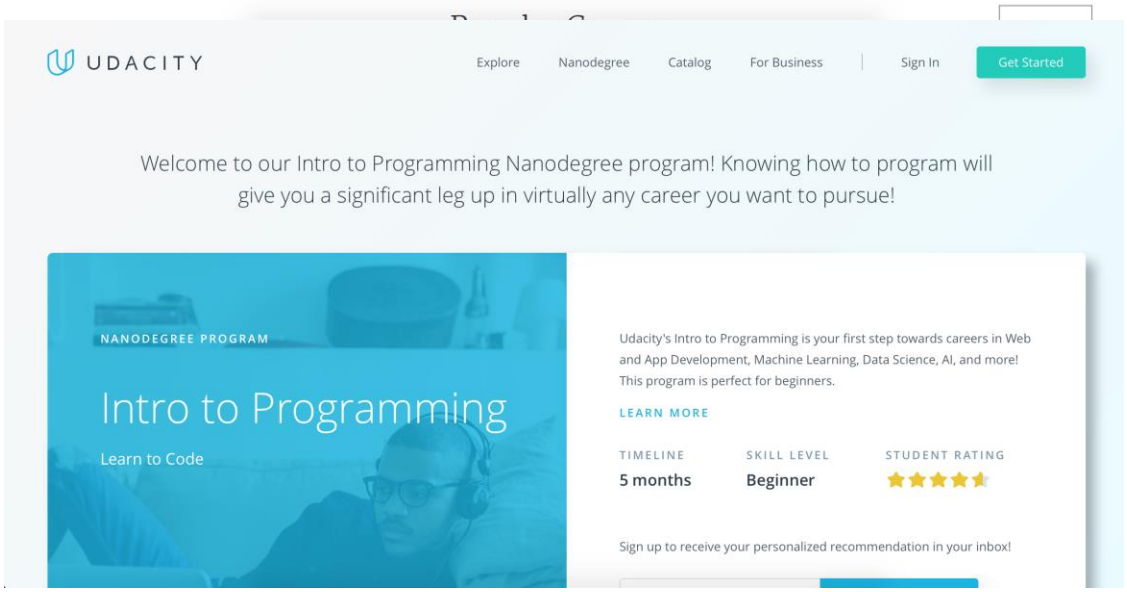
Indeed



Authentic Jobs

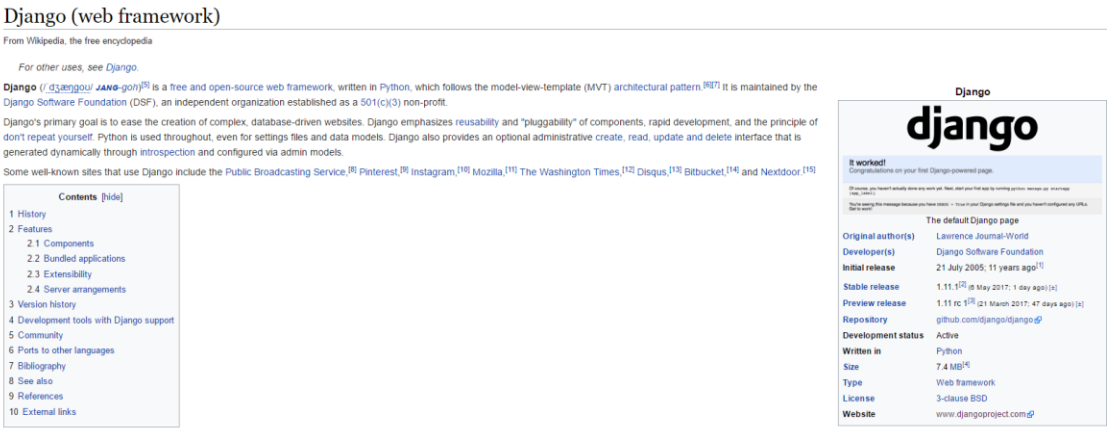
Le piattaforme di apprendimento online più gettonate sono Coursera e Udacity (si vedano le immagini seguenti). Queste ultime offrono numerosi corsi che permettono agli sviluppatori di migliorare le proprie competenze.





Rimane tuttavia compito del programmatore navigare tra i vari siti, ricavare statistiche circa l'utilizzo e la richiesta attuale dei diversi linguaggi di programmazione al fine di prevederne il trend.

Anche Wikipedia, un'enciclopedia aperta gestita da editori volontari che ha come aspirazioni fondamentali contenuto libero ed articoli oggettivi, fra i vari contenuti, contiene pagine dedicate a linguaggi, framework e librerie.



Django (web framework)

From Wikipedia, the free encyclopedia

For other uses, see Django.

Django (/ˈdʒæŋgoʊ/ JANG-goh)[5] is a free and open-source web framework, written in Python, which follows the model-view-template (MVT) architectural pattern.[6][7] It is maintained by the Django Software Foundation (DSF), an independent organization established as a 501(c)(3) non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that use Django include the Public Broadcasting Service,[8] Pinterest,[9] Instagram,[10] Mozilla,[11] The Washington Times,[12] Disqus,[13] Bitbucket,[14] and Nextdoor.[15]

Contents [hide]

1 History
2 Features
  2.1 Components
  2.2 Bundled applications
  2.3 Extensibility
  2.4 Server arrangements
3 Version history
4 Development tools with Django support
5 Community
6 Ports to other languages
7 Bibliography
8 See also
9 References
10 External links

Django

django

It worked!
Congratulations on your first Django-powered page.

The default Django page

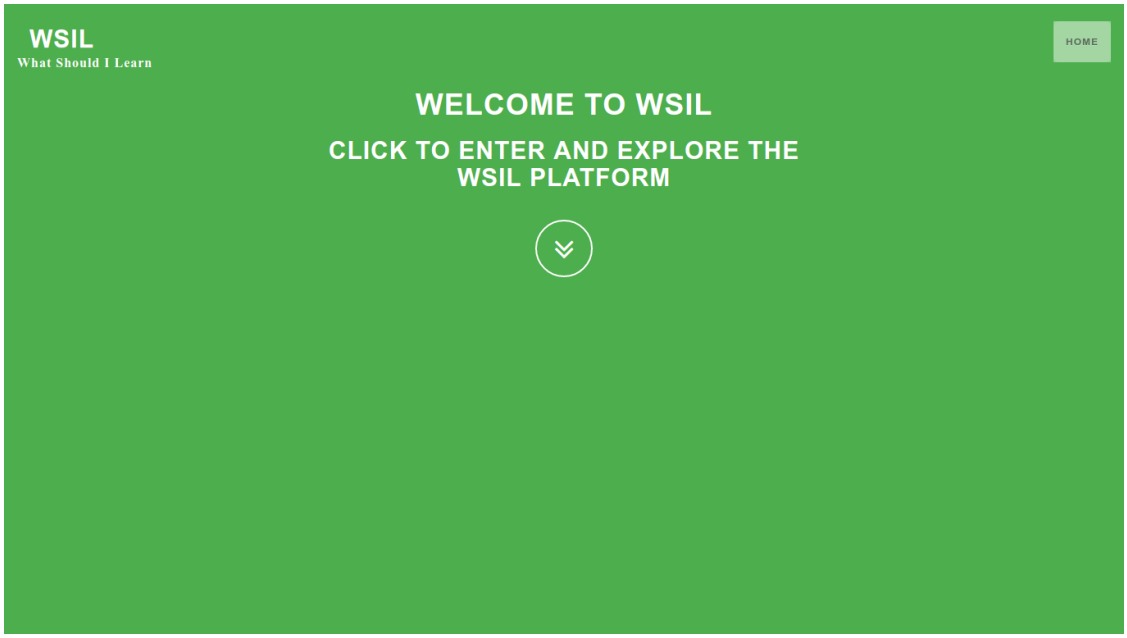
Original author(s)	Lawrence Journal-World
Developer(s)	Django Software Foundation
Initial release	21 July 2005; 11 years ago[1]
Stable release	1.11.1[2] (8 May 2017; 1 day ago) [±]
Preview release	1.11 rc 1[3] (21 March 2017; 47 days ago) [±]
Repository	github.com/django/django
Development status	Active
Written in	Python
Size	7.4 MB[4]
Type	Web framework
License	3-clause BSD
Website	www.djangoproject.com

Immagine della pagina di Django su Wikipedia

Da questa immagine possiamo notare come siano presenti una serie di informazioni riguardanti ad esempio versione, linguaggio e tipologia del framework Django, così come una descrizione sintetica sulle caratteristiche principali di esso.

4. Descrizione funzionale della soluzione proposta

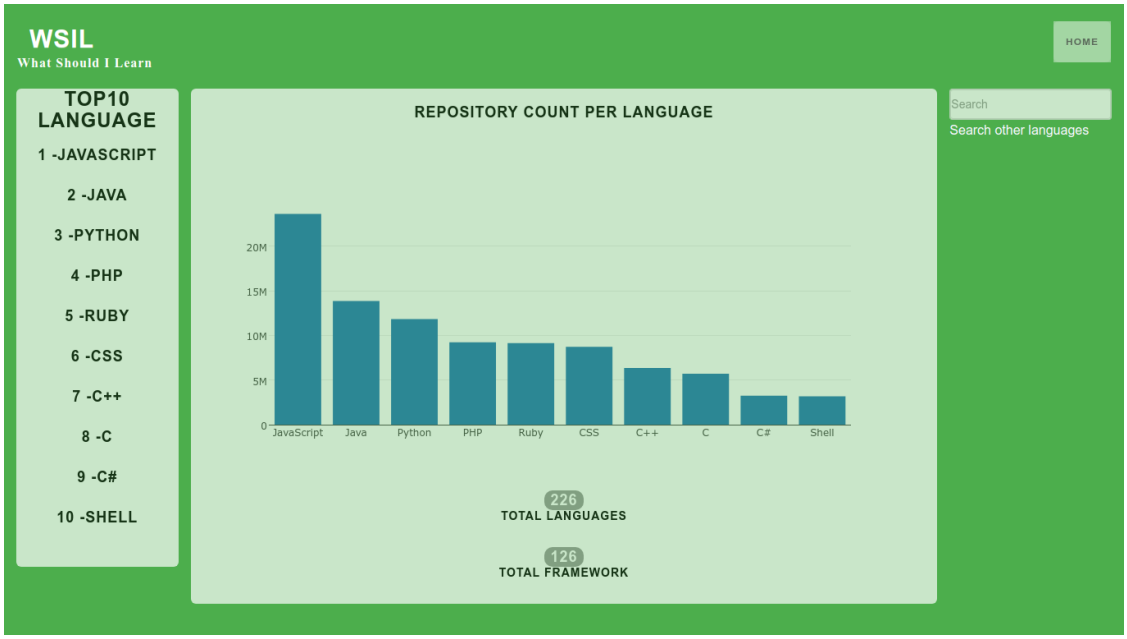
Nel seguente capitolo descriveremo l'applicazione WSIL dal punto di vista funzionale.
La web application si presenta con una schermata di **benvenuto** per l'utente. Da qui, tramite un click sul bottone centrale, potrà accedere alla piattaforma senza sistemi di autenticazione.



Welcome Page

Quindi, cliccando sul button centrale, l'utente verrà reindirizzato alla ***Homepage*** della Web application: tale pagina sarà formata da tre sezioni:

- **la parte sinistra** della homepage conterrà una classifica che elencherà i 10 linguaggi di programmazione più utilizzati del momento; l'utente, cliccando su uno dei linguaggi nella classifica, verrà reindirizzato ad una pagina dalla quale potrà visionarne le statistiche dettagliate, i framework, i lavori e i corsi relativi a quel linguaggio(ne discuteremo poi in seguito);
- **la parte centrale** della homepage presenterà un grafico che mostrerà il numero di repository su github per ciascun linguaggio presente nella top 10; inoltre illustrerà il numero totale di framework e linguaggi disponibili sul mercato;
- **la parte destra** della homepage consentirà invece all'utente di effettuare delle ricerche personali riguardo specifici linguaggi; all'immissione di ciascun carattere, verranno mostrati dei suggerimenti cliccabili. Una volta selezionato un linguaggio, l'utente verrà reindirizzato alla pagina di dettaglio per quel linguaggio.



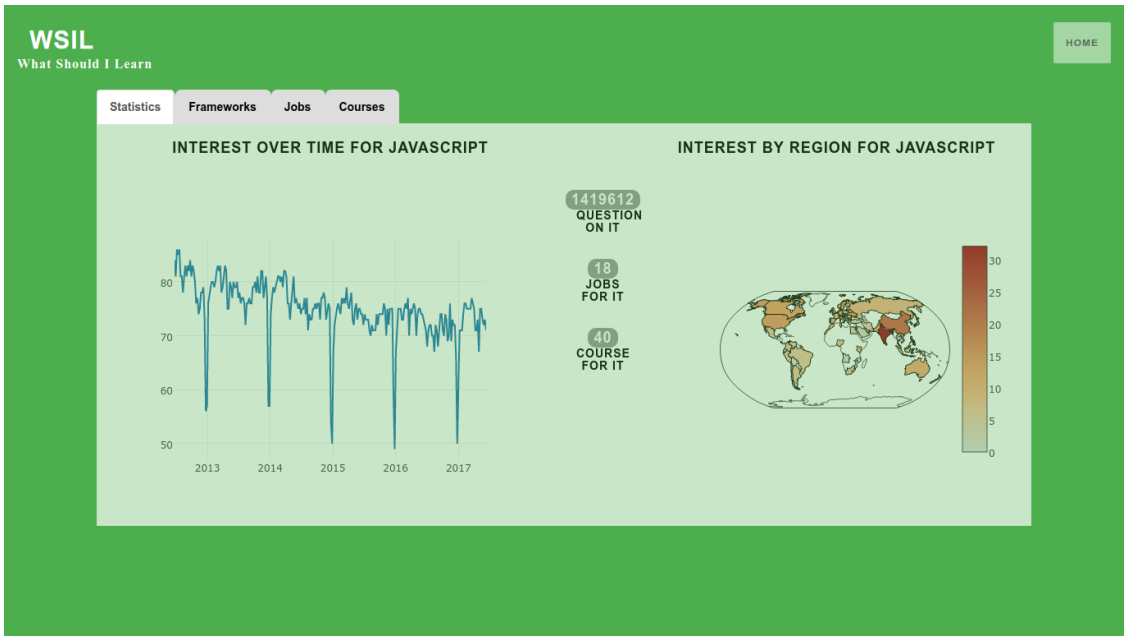
Homepage

Selezionato un linguaggio, cliccando una entry nella top 10 oppure utilizzando la search bar, l'utente verrà reindirizzato ad una pagina "**Details**" di dettaglio del linguaggio contenente 4 tab:

- Statistics:
In questo tab vengono mostrati due grafici:

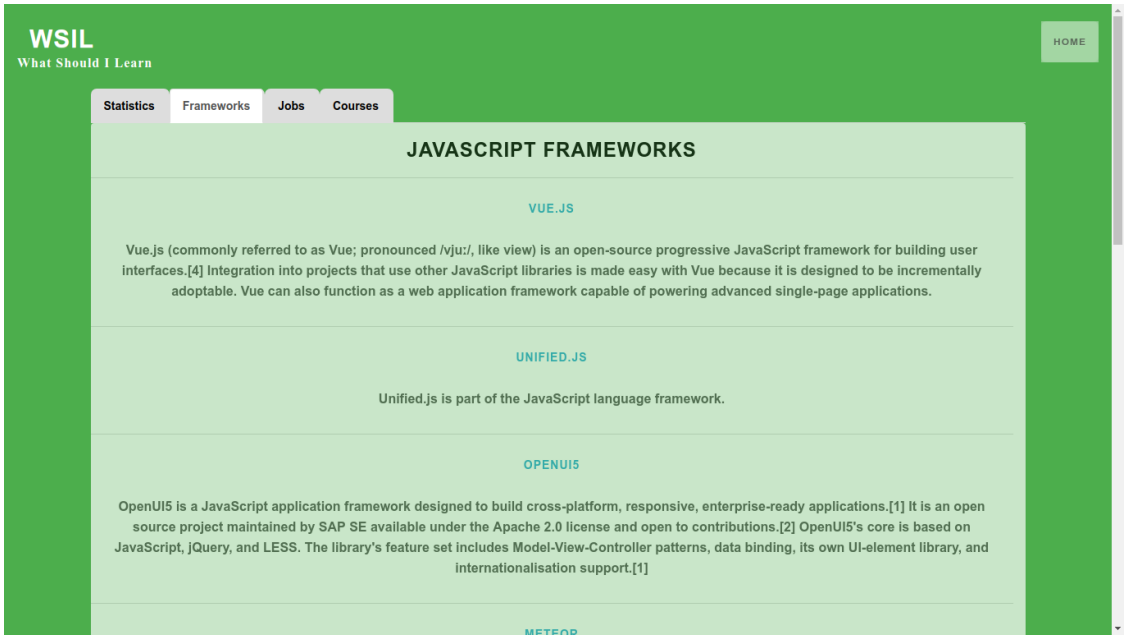
- il primo va a descrivere l'interesse nel tempo negli ultimi 5 anni rispetto a quel determinato linguaggio;
- il secondo grafico mostra l'interesse per quel linguaggio rispetto alla regione geografica.

Infine l'utente ha la possibilità di visualizzare delle informazioni generali riguardo il numero di domande presenti nella piattaforma StackOverflow, il numero di offerte di lavoro e il numero dei corsi inerenti al linguaggio selezionato.



Statistics Tab

- Frameworks:
In questo tab l'utente potrà visualizzare i framework esistenti per il linguaggio selezionato assieme ad una breve descrizione.



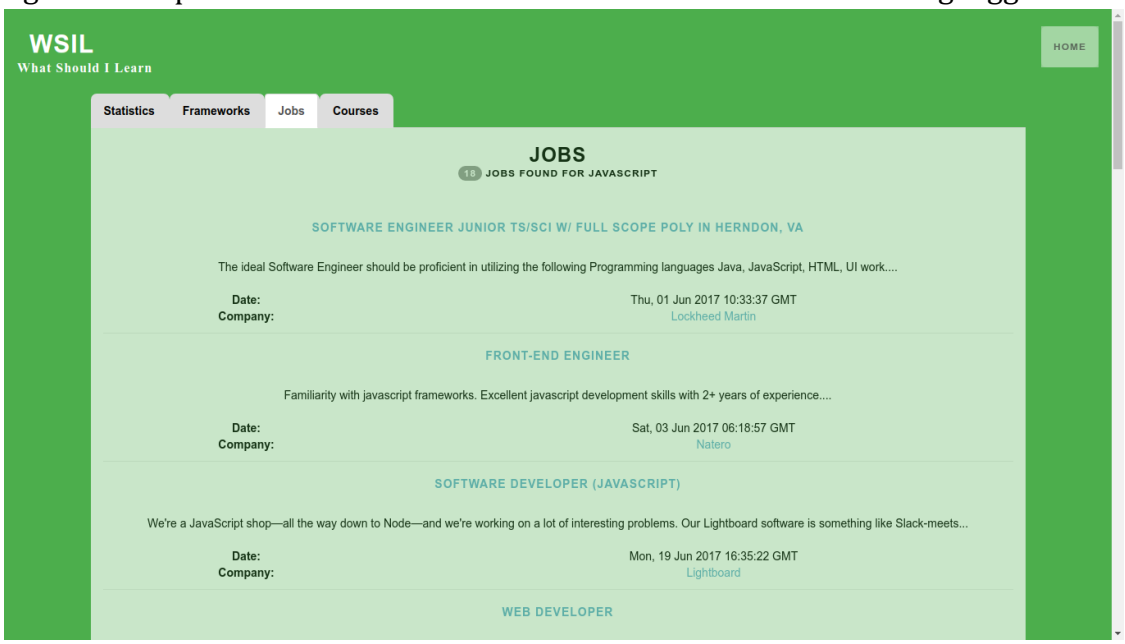
Frameworks Tab

Ciascuna entry dell'elenco è inoltre cliccabile. Quando l'utente selezionerà un elemento della lista, verrà reindirizzato alla pagina "**DetailFw**" simile alla pagina **Detail** dei linguaggi: essa avrà un tab Statistics, un tab Jobs e un tab Courses con lo stesso tipo di contenuto degli omonimi tab

presenti nella pagina **Detail**, ad eccezione del tab Framework che viene sostituito dal tab **Details** che presenterà le feature del framework preso in considerazione.

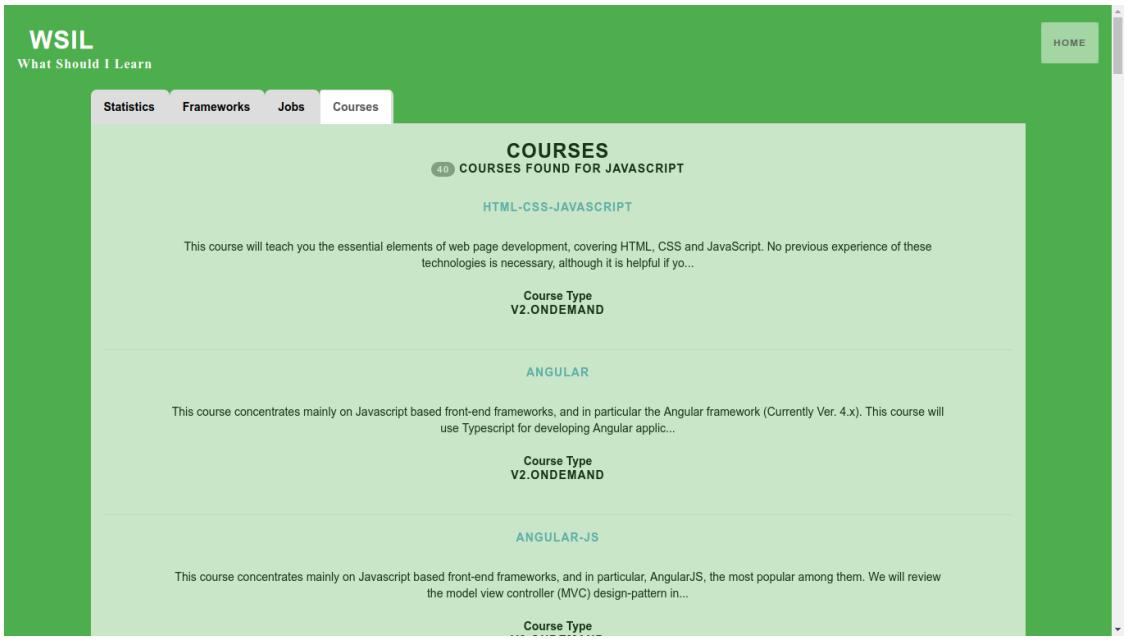
Details tab

- ***Jobs***:
  la seguente tab presenterà una lista di tutti i lavori trovati inerenti al linguaggio selezionato.



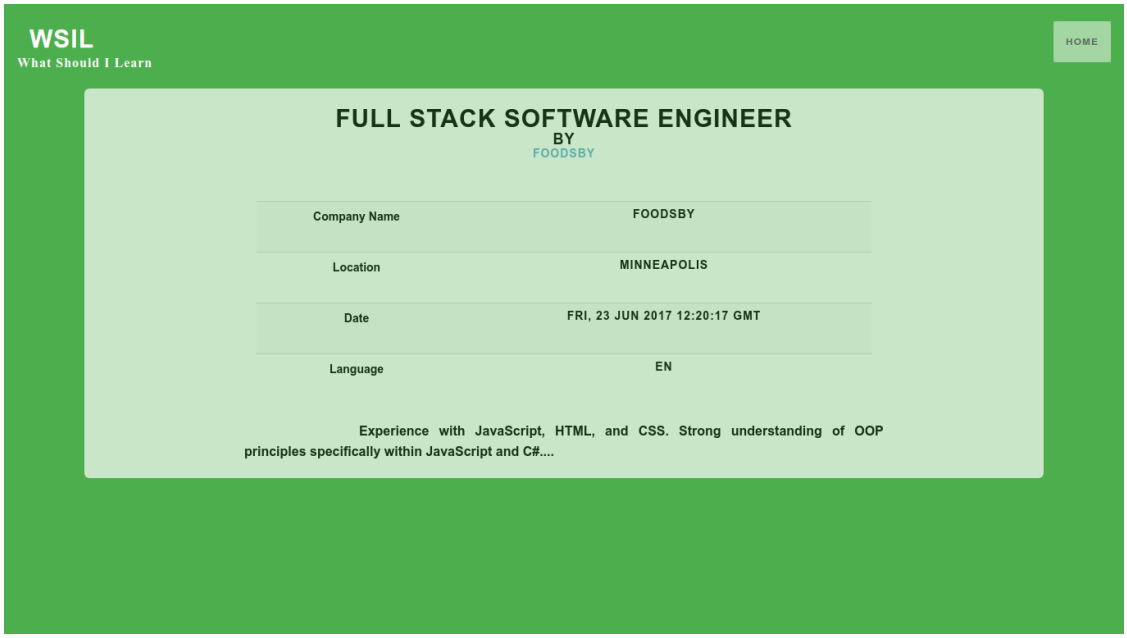
Jobs Tab

- ***Courses***
  la seguente tab presenterà una lista di tutti i corsi trovati inerenti al linguaggio selezionato.



Courses Tab

Dai tab jobs e courses è possibile cliccare su uno dei job o corsi trovati per visualizzarne i dettagli; l'utente sarà quindi reindirizzato ad una pagina "***Job***" dalla quale si potrà poi essere reindirizzati all'annuncio di lavoro:



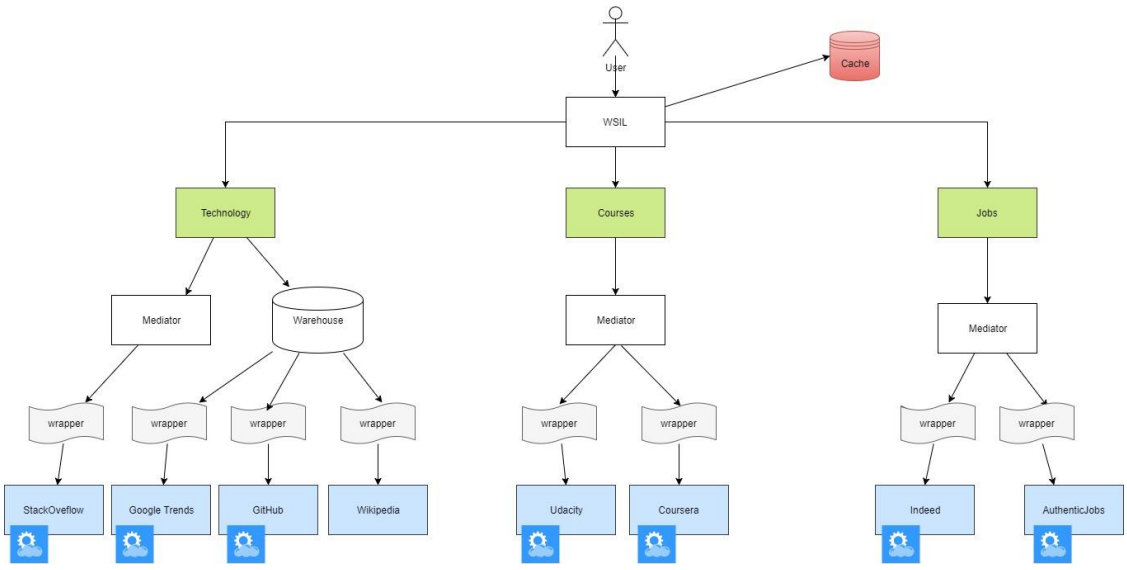
Job Page

oppure ad una pagina **Course** dalla quale si potrà poi essere reindirizzati al corso:

Course Page

5. Architettura del sistema proposto

Il sistema proposto presenta un'architettura ibrida, in quanto l'applicazione sfrutta sia sistemi mediator, sia sistemi datawarehouse:



Il sistema proposto si compone di tre funzionalità principali:

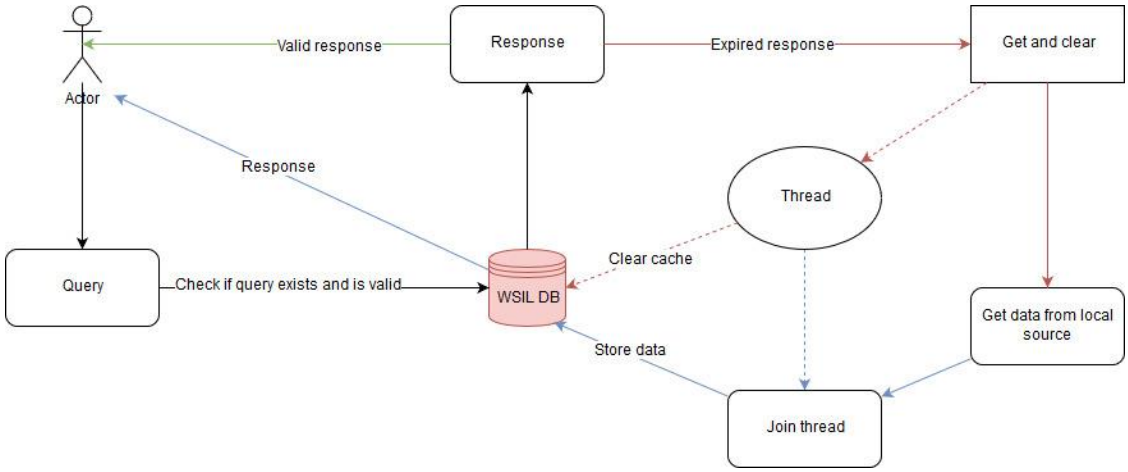
- ***Technology*** con la quale si ricavano informazioni inerenti a linguaggi e framework presenti sul mercato della programmazione, basato su un approccio ibrido;

- ***Courses*** per informazioni sui corsi a disposizione dei programmatori basato su un approccio data warehouse;
- ***Jobs*** per informazioni sui lavori a disposizione dei programmatori basato su approccio mediator.

Tale architettura permette di prelevare dati altamente volatili in real time, come si fa ad esempio per le informazioni inerenti a possibilità di lavoro, domande postate sulla piattaforma StackOverflow, e allo stesso tempo ci dà l'opportunità di memorizzare in un data warehouse informazioni statiche o poco volatili, come le informazioni prese da Google Trends degli ultimi 5 anni, oppure dati da Wikipedia.
Nei capitoli successivi illustreremo le query attraverso le quali è possibile ricavare le informazioni e i vari wrapper utilizzati per l'intera architettura proposta.

5.1. La cache

Nel caso di dati molto volatili, mantenerli nella base di dati per più di un giorno rischia di comprometterne la validità: si potrebbero mostrare informazioni ormai scadute e quindi non ancora valide. A tal fine abbiamo realizzato un sistema di cache che si basa sulla tecnica della Lazy List utilizzata nella programmazione concorrente: i dati, anche se scaduti, restano in cache finché non viene lanciata una query il cui risultato contiene dati scaduti. A quel punto viene lanciato un thread che si occupa di pulire la cache da *tutti* i dati scaduti, mentre si recuperano e trasformano i dati dalle fonti locali. Al termine di questa operazione si effettua l'operazione di join sul thread e si inseriscono i nuovi dati in cache.



6. Le Fonti

In questo capitolo descriveremo le fonti utilizzate per l'applicazione; WSIL si basa essenzialmente su 8 fonti principali:

- GitHub;
- StackOverflow;
- Wikipedia;

- Google Trends;
- Udacity;
- Coursera;
- Indeed;
- Authentic Jobs.

Attribuiremo ad ogni fonte un grado di volatilità e una tipologia:

	GitHub	**Stack Overflow**	**Wikipedia**	**Google Trends**
Tipologia	Database	API	Web Page	API
Volatilità	Media	Alta	Bassa	Media

	Udacity	**Coursera**	**Authentic Jobs**	**Indeed**
Tipologia	API	API	API	API
Volatilità	Bassa	Bassa	Alta	Alta

Il lavoro di analisi delle fonti è stato diviso tra i membri del gruppo nel seguente modo:

- GitHub, StackOverflow (Fabio Napoli);
- Wikipedia, Coursera, Udacity (Mattia Tomeo);
- Authentic Jobs, Indeed, Google Trends (Dario Di Pasquale);

6.1. Descrizione delle fonti

Ogni fonte trattata corrisponderà ad una delle fonti S1, …, S8 come indicato in tabella:

Fonte	Nome
GitHub	S1
StackOverflow	S2
Wikipedia	S3
Google Trends	S4
Udacity	S5
Coursera	S6
Authentic Jobs	S7
Indeed	S8

6.1.1. GitHub

GitHub è un servizio di hosting ed un repository di controllo versione per progetti software. Il nome deriva da Git, un sistema di controllo di versione utilizzato per tracciare i cambiamenti su una serie di file e per coordinare il lavoro di un team di persone che lavorano su tali file. GitHub fornisce un'interfaccia web con cui espone le funzionalità di Git, assieme ad una serie di feature

proprie della piattaforma stessa.
Il sito è attivamente utilizzato dagli sviluppatori di software open source che sfruttano questa piattaforma per coordinare il lavoro dei loro team, così come per fornire agli utenti finali l'accesso diretto ai propri prodotti. Questi ultimi possono interagire con lo sviluppatore tramite un sistema di issue tracking e commenti che permettono agli sviluppatori di avere un contatto diretto con i propri utenti. Inoltre, grazie al sistema di pull request, gli utenti possono contribuire direttamente allo sviluppo del prodotto software proponendo modifiche al codice stesso.
Github produce pagine dettagliate che riassumono il grado di attività degli sviluppatori nei repository. Le statistiche relative alla piattaforma GitHub sono accessibili tramite il sito gitArchive;
GitHub Archive è un progetto sviluppato per memorizzare la timeline pubblica di GitHub, archiviarla e renderla facilmente accessibile per analisi future.

Lo schema locale per S1 è il seguente:

Linguaggio

Permette di recuperare le informazioni riguardo un linguaggio

```
S1.language(?language_name, ?numberRepo);
```

6.1.2. StackOverflow

StackOverflow è un sito, facente parte del network Stack Exchange, che permette di porre domande e dare risposte circa argomenti relativi alla programmazione.
StackOverflow mette a disposizione una serie di API, presenti all'indirizzo https://api.stackexchange.com/docs, che seguono le convenzioni REST.
Lo schema locale per S2 è il seguente:

6.1.3. Tag

Permette di recuperare le informazioni inerenti ad uno specifico tag, come ad esempio Java, Javascript ecc…

```
S2.tag(?tag_name, count);
```

6.1.4. Wikipedia

Wikipedia è un'enciclopedia online a contenuto libero, collaborativa, multilingue e gratuita, nata nel 2001, sostenuta e ospitata dalla Wikimedia Foundation, un'organizzazione non a scopo di lucro statunitense. Con più di 35 milioni di voci in oltre 280 lingue, è l'enciclopedia più grande mai scritta ed è tra i dieci siti web più visitati al mondo. Rappresenta la maggiore e più consultata opera di riferimento generalista di Internet. Nel nostro progetto verrà utilizzata per ottenere informazioni relative a framework e librerie.
Lo schema locale per S3 è il seguente:

Framework o librerie

Permette di recuperare le informazioni di base di un particolare framework o libreria

- `S3.library_framework(?name, ?type, ?initial_release, ?stable_release, ?repository, ?development_status, ?language_used, ?license, ?website, ?description);`

Feature supportate

Permette di recuperare una serie di feature supportate dai vari framework o librerie

- `S3.feature(?library_framework_name, ?ajax, ?mvc_framework, ?mvc_push_pull, ?localization, ?orm, ?testing_framework, ?db_migration_framework, ?security_framework, ?template_framework, ?caching_framework, ?form_validation_frameworks);`

6.1.5. Google Trends

Google Trends è uno strumento, basato su Google, che permette di conoscere la frequenza di ricerca sul web di una determinata parola o frase. La ricerca e la visualizzazione sono impostabili per nazione e per lingua. I risultati (cioè i trends, ovvero le "tendenze" correnti) sono mostrati accompagnando l'occorrenza con un grafico che sintetizza, nel tempo, l'andamento della sua popolarità (ricerca o visualizzazione).
È possibile accedere alle API di Google Trends utilizzando la libreria *pytrends* per Python, disponibile all'indirizzo https://github.com/GeneralMills/pytrends.
Lo schema è il seguente:

- `S4.interest_over_time(?keyword, ?date, ?interest_rate);`
- `S4.interest_by_region(?keyword, ?region, ?interest_rate);`

6.1.6. Udacity

Udacity è una piattaforma educativa che offre corsi online aperti, dando la possibilità agli utenti di acquisire certificati al superamento dei corsi stessi, raggruppati in una serie di percorsi denominati "tracks". Molti corsi riguardano proprio la Computer Science, e permettono di studiare una determinata tecnologia sfruttando un determinato linguaggio. Udacity fornisce una API pubblica che permette di accedere a tutti i corsi, tutte le tracks e a tutte le certificazioni. L'indirizzo utilizzato per accedere a tale API è https://www.udacity.com/public-api/v1/courses.
Lo schema offerto per S5 è il seguente

Courses

Permette di recuperare tutti i corsi, assieme alla loro descrizione completa:

- `S5.course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url);`
- `S5.affiliates(?course_key, ?name, ?image);`

6.1.7. Coursera

Coursera è un'azienda statunitense che opera nel campo delle tecnologie didattiche, fondata da docenti d'informatica dell'Università di Stanford. Come Udacity, fornisce numerosi corsi e percorsi online grazie a partnership con altre aziende e università. Questi corsi spaziano in vari

campi, fra cui quello dell'informatica.
Cousera fornisce 2 endpoint per l'accesso alle informazioni su corsi e partner:

- https://api.coursera.org/api/courses.v1 per ottenere l'elenco completo dei corsi;
- https://api.coursera.org/api/partners.v1 per ottenere l'elenco completo dei partners;

Lo schema per S6 è il seguente:

Course

I seguenti schemi contengono le informazioni complete su tutti i corsi offerti da Coursera, offerti in diverse lingue assieme a numerosi sottotitoli. A causa del cambiamento della loro piattaforma, questo schema contiene attributi i cui valori differenziano i corsi appartenenti alla vecchia e alla nuova piattaforma.

- `S6.course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url);`

Partners

Permette di estrarre l'elenco dei partner di Coursera, assieme ai corsi offerti.

- `s6.partner(?partner_id, ?partner_name).`
- `S6.partnership(?course_id, ?partner_id);`

6.1.8. Authentic Jobs

The leading job board for designers, hackers, and creative pros.

Authentic Jobs è una web application che permette di offrire e trovare lavoro a sviluppatori e designer, sia in sede sia remoti. Grazie allo svariato numero di lavori presenti su di essa, la piattaforma è tra le più utilizzate.
Authentic Jobs mette a disposizione una serie di API, disponibili al seguente indirizzo: https://authenticjobs.com/api/docs.
Lo schema offerto dalla fonte è il seguente:

Job

Permette di recuperare liste di lavori, più diverse informazioni utili, tra cui *categoria* e *tipo* di un lavoro, informazioni circa una *compagnia* e una *location*, secondo determinate query:

- `S7.job(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lng, ?keywords);`

6.1.9. Indeed

Indeed è un motore di ricerca per trovare lavoro online. Il sito raccoglie annunci di lavoro da migliaia di siti web, incluse bacheche di lavoro, quotidiani, associazioni, e pagine d'impiego di compagnie private.
Indeed mette a disposizione una serie di API che permettono di postare e cercare annunci di

lavoro tramite interfaccia REST.
Lo schema è il seguente:

- `S8.result(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query).`

7. Definizione dello schema globale

In questo capitolo definiremo i vari schemi globali utilizzando il linguaggio Datalog. La descrizione degli schemi globali verrà divisa in paragrafi. Tutte i paragrafi meno l'ultimo descriveranno gli schemi globali attraverso i mapping GAV, mentre il paragrafo finale mostrerà infine il mapping LAV delle fonti locali.

7.1. Informazioni tecnologia

I seguenti schemi permettono di recuperare varie informazioni dalle tecnologie ricercate:

- `Language(?name) :- S1.language(?name, _).`

- `LibraryOrFramework(?name, ?type, ?initial_release, ?stable_release, ?repository, ?development_status, ?language, ?license, ?website, ?description) :- S3.library_framework(?name, ?type, ?initial_release, ?stable_release, ?repository, ?development_status, ?language, ?license, ?website, ?description).`

- `Features(?library_framework_name, ?ajax, ?mvc_framework, ?mvc_push_pull, ?localization, ?orm, ?testing_framework, ?db_migration_framework, ?security_framework, ?template_framework, ?caching_framework, ?form_validation_frameworks) :- S3.feature(?library_framework_name, ?ajax, ?mvc_framework, ?mvc_push_pull, ?localization, ?orm, ?testing_framework, ?db_migration_framework, ?security_framework, ?template_framework, ?caching_framework, ?form_validation_frameworks).`

7.2. Popolarità tecnologia

I seguenti schemi permettono di estrarre informazioni circa la popolarità di una determinata tecnologia (linguaggio di programmazione o framework):

- `RepositoriesUsingIt(?language_name, ?repo_count) :- S1.language(?language_name, ?repo_count).`

- `QuestionOnIt(?tag, ?count) :- S2.tag(?tag, ?count).;`

- `InterestOverTimeLanguage(?language_name, ?date, ?interest_rate) :- S4.interest_over_time(?language_name, ?date, ?interest_rate), S1.language(?language_name, _).`

- `InterestOverTimeFrameworkLibrary(?fw_or_lib, ?date, ?interest_rate) :- S4.interest_over_time(?fw_or_lib, ?date, ?interest_rate), S3.library_framework(?fw_or_lib, _, _, _, _, _, _, _, _, _).`

- `InterestByRegionLanguage(?language, ?region, ?interest_rate):- S4.interest_by_region(?language, ?region, ?interest_rate), S1.language(?language, _).;`

- `InterestByRegionFrameworkLibrary(?fw_or_lib, ?region, ?interest_rate):- S4.interest_by_region(?fw_or_lib, ?region, ?interest_rate), S3.library_framework(?fw_or_lib, _, _, _, _, _, _, _, _, _).;`

7.3. Corsi

In questa sezione sarà mostrato lo schema globale relativo ai corsi:

- `Course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url) :- S5.course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url).`

- `Course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url) :- S6.course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url).`

- `CoursePartner(?course_id, ?partner_id, ?partner_name) :- S5.affiliates(?course_id, ?partner_name), S6.partner(?partner_id, ?partner_name).`

- `CoursePartner(?course_id, ?partner_id, ?partner_name) :- S6.partnership(?course_id, ?partner_id), S6.partner(?partner_id, ?partner_name).`

7.4. Lavoro

In questa sezione sarà mostrato lo schema globale relativo ai lavori:

- `Job(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query) :- S7.job(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query);`

- `Job(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query) :- S8.result(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query).`

7.5. Mapping LAV

Dopo aver mostrato tutte le tabelle dello schema globale, assieme ai relativi mapping GAV, passiamo al descrivere i corrispondenti mapping LAV.

7.5.1. GitHub

- `S1.language(?language_name, ?numberRepo) :- Language(?language_name), RepositoriesUsingIt(?language_name, ?numberRepo), InterestOverTimeLanguage(?language_name, _, _), InterestByRegionLanguage(?language_name, _, _).`

7.5.2. StackOverflow

- S2.tag(?tag, ?count) :- QuestionOnIt(?tag, ?count).;

7.5.3. Wikipedia

- S3.library_framework(?name, ?type, ?initial_release, ?stable_release, ?repository, ?development_status, ?language, ?license, ?website, ?description) :- LibraryOrFramework(?name, ?type, ?initial_release, ?stable_release, ?repository, ?development_status, ?language, ?license, ?website, ?description), InterestOverTimeFrameworkLibrary(?name, _, _), InterestByRegionFrameworkLibrary(?name, _, _).

- S3.feature(?library_framework_name, ?ajax, ?mvc_framework, ?mvc_push_pull, ?localization, ?orm, ?testing_framework, ?db_migration_framework, ?security_framework, ?template_framework, ?caching_framework, ?form_validation_frameworks) :- Features(?library_framework_name, ?ajax, ?mvc_framework, ?mvc_push_pull, ?localization, ?orm, ?testing_framework, ?db_migration_framework, ?security_framework, ?template_framework, ?caching_framework, ?form_validation_frameworks).

7.5.4. Google Trends

- S4.interest_over_time(?keyword, ?date, ?interest_rate) :- InterestOverTimeLanguage(?keyword, ?date, ?interest_rate).

- S4.interest_over_time(?keywork, ?date, ?interest_rate) :- InterestOverTimeFrameworkLibrary(?keyword, ?date, ?interest_rate).

- S4.interest_by_region(?keyword, ?region, ?interest_rate) :- InterestByRegionLanguage(?keyword, ?region, ?interest_rate).;

- S4.interest_by_region(?keyword, ?region, ?interest_rate) :- InterestByRegionFrameworkLibrary(?keyword, ?region, ?interest_rate).;

- S4.suggestion(?keyword, ?suggested_keyword) :- Suggestion(?keyword, ?suggested_keyword);

7.5.5. Udacity

- S5.course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url) :- Course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url).

- S5.affiliates(?course_id, ?partner_name) :- CoursePartner(?course_id, ?partner_name).

7.5.6. Coursera

- S6.course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url) :- Course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description, ?workload, ?url).

- S6.partnership(?course_id, ?partner_id) :- CoursePartner(?course_id, ?partner_id, _, _).

- `S6.partner(?partner_id, ?partner_name) :- CoursePartner(_, ?partner_id, ?partner_name).`

7.5.7. AuthenticJob

- `S7.job(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query) :- Job(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query);`

7.5.8. Indeed

- `S8.result(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query) :- Job(?job_title, ?description, ?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon, ?query).`

8. Wrapper

Nel nostro progetto utilizzeremo esclusivamente **wrapper mauali**; quindi i dati verranno presi delle sorgenti **S**, che hanno schemi **Ts** definiti nei capitoli precedenti con l'utilizzo di API; dopodicé vengono definiti i wrapper **W** che avranno un target schema definito manualmente **Tw** che comprenderà i dati che occorrono alla data integration.

8.1. Wrapper1: Udacity (API)

La fonte udacity.com offre il seguente endpoint :
`https://www.udacity.com/public-api/v1/courses`

che fornisce il seguente schema

```
Ts=(key, title, homepage, subtitle, level, starter, image, banner_image,
teaser_video, summary, short_summary, required_knowledge, expected_learning,
featured, syllabus, faq, full_course_available, expected_duration,
expected_duration_unit, new_release, tracks, affiliates, instructors,
affiliatesName, affiliatesImage)
```

da questo schema si crea il wrapper avente target schema

```
Tw=course(?course_id, ?slug, ?course_type, ?logo, ?photo_url, ?description,
?workload, ?url), affiliates(?course_key, ?name, ?image)
```

8.2. Wrapper2: AuthenticJobs(API)

La fonte authenticjobs.com offre il seguente endpoint :

`https://authenticjobs.com/api/`

che esposto al seguente url:

```
https://authenticjobs.com/api?api_key=<API_KEY>&method=aj.jobs.search&keywords=<SE
ARCH_STRING>
```

dove `api_key` rappresenta la `key` necessaria per le chiamate, `method` il metodo da chiamare (nel nostro caso l'elenco delle posizioni correnti) e `keyword` per la parola chiave da ricercare nel titolo o nella descrizione del lavoro, fornisce il seguente schema

```
Ts=(id, title, description, perks, howToApply, postDate, relocation_assistance,
telecommuting, keywords, applyurl, url, categoryID,
category,Name,categoryType,categoryLogo,categoryTagline,category, locationID,
locationName, locationCity, locationCountry, locationLatitude, locationLongitute,
locationState)
```

da questo schema si crea il wrapper avente target schema

```
Tw=job(?job_title, ?description, ?post_date, ?company_name, ?company_url,
?location_name, ?lat, ?lon, ?query)
```

8.3. Wrapper3: Indeed(API)

La fonte indeed.com offre il seguente endpoint : http://api.indeed.com/ads/ che esposto al seguente url:
`http://api.indeed.com/ads/apisearch?publisher=<PUBBLISHER_ID>&q=<SEARCH_STRING>&v=<VERSION_NUMBER_API>&latlong=1`
dove `publisher` rappresenta l'id di un publisher iscritto al sito, `q` la query di ricerca, `v` la versione delle api da utilizzare e `latlong` che, settato a 1, ritorna le coordinate di ciascun lavoro, restituisce il seguente schema

```
Ts=(query,jobTitle, company, city, state, country, language, formattedLocation,
source, date, snippet, url, jobKey, sponsored, station, expired,
indeedApply,latitude, longitude)
```

da questo schema si crea il wrapper con target Schema:

```
Tw=job(?job_title, ?description, ?post_date, ?company_name, ?company_url,
?location_name, ?lat, ?lon, ?query)
```

8.4. Wrapper4: Coursera(API)

La fonte coursera.com offre l'endpoint https://api.coursera.org/api/courses.v1 che esposto all'url:

```
https://api.coursera.org/api/courses.v1?q=search&query=<SEARCH_STRING>
```

dove `q` rappresenta il metodo di ricerca e `query` le parole chiave da cercare, separate dal simbolo `+`, restituisce il seguente schema

```
Ts=(id, slug, name, primaryLanguages, subtitleLanguages, partnerLogo, instructors,
partnerIds, photoUrl, certifcates, description, startDate, workload, previewLink,
specializations)
```

da questo schema si costruisce il wrapper con target Schema

```
Tw=course(course_id, slug, course_type, logo, photo_url, description, workload,
url)
```

Coursera.com offre un altro endpoint:
https://api.coursera.org/api/partners.v1:course_id/includes=partnerIds

dove `course_id` rappresenta l'id del corso e `includes` i parametri aggiuntivi da inserire, che ci consente di recuperare i dati sui partner e restituisce il seguente schema:

```
Ts=(id, name, shortName, description, banner, courselds, instructorIds, links,
location)
```

da questo schema si costruisce il wrapper con target Schema:

```
Tw=partner(partner_id, partner_name), partnership(partner_id, course_id)
```

8.5. Wrapper5: Stackoverflow

La fonte Stack Exchange mette a disposizione una serie di API al seguente endpoint: https://api.stackexchange.com/docs .

Al fine di poter calcolare delle statistiche sul numero di domande/risposte relative ad un linguaggio o ad una tecnologia, ci siamo serviti dei tag: degli identificativi che vengono posti ad ogni domanda per specificarne l'area di interesse.
All'endpoint esposto al seguente url:

```
https://api.stackexchange.com/2.2/tags?order=desc&sort=popular&inname=<keyword>&si
te=stackoverflow
```

dove `order` specifica l'ordinamento dei parametri di ricerca, `sort` il valore di ordinamento, `site` il sito della piattaforma stackexchange da cui estrarre i dati e `inname` la query di ricerca, è possibile recuperare l'elenco di tutti i tag, avente il seguente schema:

```
Ts=(has_synonyms, is_moderator_only, is_required, count, name)
```

dal quale costruiamo lo schema:

```
Tw=tag(name, count)
```

8.6. Wrapper6: Google Trends

Per accede ai dati di Google Trends ci siamo serviti della libreria Python pytrends. La suddetta permette di accedere ai dati relativi alle ricerche all'interno dell'ambiente di Python.

Per poter utilizzare pytrends, occorre avere un account. Una volta creato tale account, è possibile connettersi a google trends grazie al seguente metodo:

```
TrendReq(GOOGLE_USR, GOOGLE_PWD)
```

Connessi a google, è possibile impostare la e le keyword da cercare grazie agli endpoint con la seguente funzione

```
build_payload(kw_list)
```

Gli endpoint utilizzati sono:

Interest over time

Il seguente metodo:

```
pytrends.interest_over_time()
```

restituisce il numero di ricerche (in migliaia) di una determinata keyword. Lo schema di questa chiamata è:

```
Tw=(keyword, interval, interest_rate)
```

in questo caso, `Ts = Tw`.

Interest by region

`Il seguente metodo:`

```
pytrends.interest_by_region()
```

restituisce il numero di ricerche (in migliaia) di una determinata keyword per una particolare area geografica. Lo schema di questa chiamata è:

```
Tw=(keyword, region, interest_rate)
```

in questo caso, `Ts = Tw`.

8.7. Wrapper7: GitHub

L'estrazione dei dati da GitHub è stata effettuata utilizzando il sito GitHub Archive, dal quale è possibile estratte utili statistiche relativamente all'utilizzo dei diversi linguaggi nei vari repository.
I dati messi a disposizione da GitHub Archive sono accessibili scaricando il file http://data.githubarchive.org/YYYY-MM-DD-HH.json.gz, dove YYYY è l'anno, MM è il mese, DD è il giorno e HH è l'ora della giornata.
Il file succitato contiene una serie di *eventi*, ossia attività che vengono effettuate su GitHub. Questi eventi possono essere di vario tipo (e.g. Push, Push Request, Fork, etc). Nel nostro caso, al fine di estrarre informazioni circa l'utilizzo dei vari linguaggi di programmazione, ci serviremo solo degli eventi aventi tra gli attributi, l'attributo *language*. Tale attributo indica il linguaggio di programmazione dell'evento (i.e. quando si esegue un'operazione di push, indica i linguaggi di programmazione utilizzati nel codice modificato). Appare chiaro che gli eventi associati all'apertura, la chiusura, la risoluzione, il commento o la risposta ad un'issue non contengono informazioni rilevanti all'analisi dei linguaggi.
Tramite uno script basato su BaseCommand di Django, abbiamo implementato una funzione che, partendo dalla data *2014-01-01-0*, fino alla data, *2016-12-31-23* scarica tutti i pacchetti *.json.gz* associati alle statistiche di GitHub tramite una chiamata `urlretrieve`: un metodo simile allo script *wget* di UNIX.
Scaricato e decompresso il pacchetto d'interesse, itera su ogni evento, e, qualora l'evento dovesse contenere un valore per la chiave *language*, incrementa il valore di un dizionario contenente, per

ogni linguaggio, il numero di volte in cui è stato trovato nei vari eventi trovati fino a quel momento.
Ad intervalli regolari salva queste informazioni nel warehouse, sugli schemi

```
RepositoryUsingIt(language_name, repo_count)
Language(language_name)
```

Quindi:

```
Tw = language(language_name, repo_count).
```

8.8. Wrapper8: Wikipedia

Per l'estrazione dei dati da Wikipedia, è stato necessario sviluppare tecniche di scraping, utilizzando Scrapy.
Alla pagina https://en.wikipedia.org/wiki/Comparison_of_web_frameworks è possibile trovare un elenco dei web framework più diffusi. Da questa pagina è possibile ricavare due schemi:

Library Framework

Per ogni framework è possibile estrarre informazioni di vario tipo (e.g. data di rilascio, tipo di licenza, etc). Queste informazioni sono parzialmente disponibili effettuando scraping dell'HTML tramite la seguente espressione XPath:

```
//*[@id="mw-content-text"]/div/table[position()<18 and
position()>2]/tr[position()>1]
```

Ogni tabella di framework ha come titolo il nome del linguaggio utilizzato per sfruttare tali framework. Per estrarre il nome di tale linguaggio, è stata utilizzata la seguente espressione xpath

```
./../preceding-sibling::h3[1]/span/text()
```

utilizzando come nodo contesto un nodo appartenente al nodeset estratto dalla espressione xpath precedente.
Tuttavia, fra tutte le tabelle contenenti i nomi dei framework, vi è una con titolo "Others". In tale tabella, i linguaggi richiesti sono memorizzati in una colonna della tabella. In questo caso, i linguaggi vengono prelevati a partire dalla prima espressione xpath sopra mostrata.

Dalle precedenti espressioni ricaviamo lo schema:

```
Ts1 = (name, stable_release, release_date, license, language_used)
```

Seguendo il link che porta alla pagina dedicata ai singoli framework, estraibile tramite l'espressione:

```
//*[@id="mw-content-text"]/div/table[position() < 18]/tr/th/a/@href
```

è possibile ottenere altre informazioni relative al framework, estraibili tramite l'espressione:

```
//*[@id="mw-content-text"]/div/table[position()<=3]/tr
```

dalla quale ricaviamo:

`Ts2 = (initial_release, repository, development_status, type, license, website)`

infine, tramite l'espressione

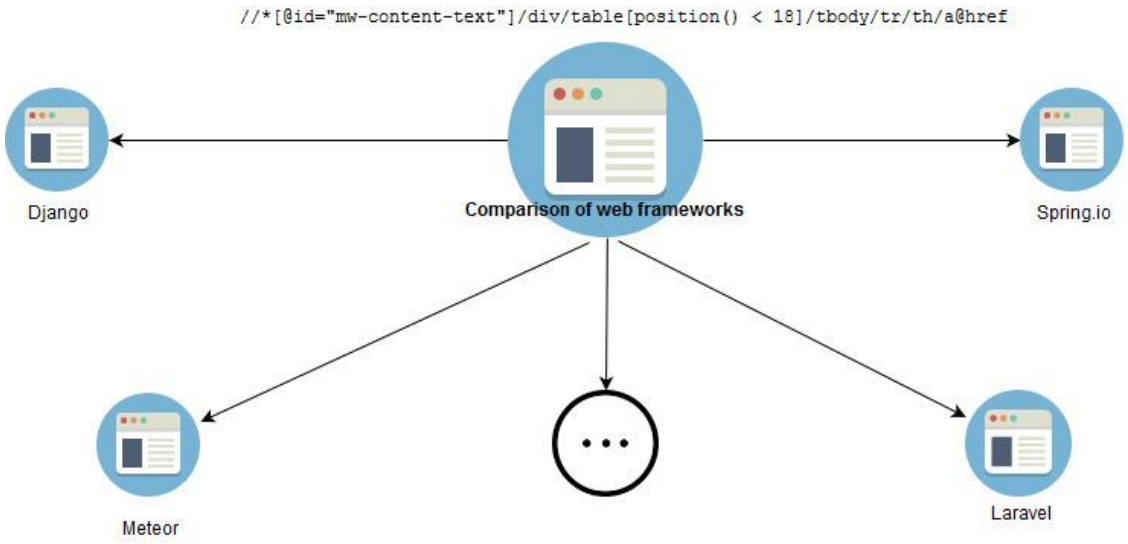
`//*[@id="mw-content-text"]/div/p[1] | //*[@id="mw-content-text"]/p[1]`

estraiamo

`Ts3 = (description).`

Con le informazioni ricavate sopra, possiamo estrarre lo schema locale:

```
Tw = library_framework(name, type, initial_release, stable_release, repository,
development_status, language_used, license, website, description)
```



Wikipedia Scraping

Freature supportate

Per ogni framework sono elencate le sue feature principali. Lo schema in questione è:

```
Ts = (library_framework_name, ajax, mvc_framework, mvc_push_pull, localization,
orm, testing_framework, db_migration_framework, security_framework,
template_framework, caching_framework, form_validation_frameworks).
```

In tal caso, `Tw = Ts`.
L'espressione XPath utilizzata per estrarre tali informazioni è:

`//*[@id="mw-content-text"]/div/table[position() >= 18]/tr`

9. Query

Di seguito saranno elencate tutte le query che è possibile effettuare sul sistema:

9.1. Query Datalog

9.1.1. Nomi di linguaggi e framework

1. **Elenco dei linguaggi**:
```
q(?name) :- Language(?name).
```
2. **Elenco dei nomi delle librerie e dei framework disponibili**:
```
q(?name) :- LibraryOrFramework(?name, ?type, ?initial_release,
?stable_release, ?repository, ?development_status, ?language, ?license,
?website, ?description).
```
3. **Elenco dei nomi delle librerie e dei framework disponibili con il relativo linguaggio (solo se presenti informazioni sul linguaggio)**:
```
q(?name, ?language_name) :- LibraryOrFramework(?name, ?type, ?initial_release,
?stable_release, ?repository, ?development_status, ?language, ?license,
?website, ?description), Language(?language_name).
```
4. **Elenco di nomi delle librerie e dei framework di un dato linguaggio**:
```
q(?fw_name, 'LINGUAGGIO') :- LibraryOrFramework(?fw_name, ?type,
?initial_release, ?stable_release, ?repository, ?development_status,
'LINGUAGGIO', ?license, ?website, ?description), Language('LINGUAGGIO').
```

9.1.2. Caratteristiche di framework

1. **Elenco di tutti i framework e di tutte le librerie, con le caratteristiche principali**:
```
q(?name, ?ajax, ?mvc_framework, ?mvc_push_pull, ?localization, ?orm,
?testing_framework, ?db_migration_framework, ?security_framework,
?template_framework, ?caching_framework, ?form_validation_frameworks) :-
LibraryOrFramework(?name, _, _, _, _, _, _, _, _, _), Features(?name, ?ajax,
?mvc_framework, ?mvc_push_pull, ?localization, ?orm, ?testing_framework,
?db_migration_framework, ?security_framework, ?template_framework,
?caching_framework, ?form_validation_frameworks).
```

9.1.3. Popolarità

1. **Mostra per ogni linguaggio l'utilizzo su GitHub**:
```
q(?language_name, ?repo_count) :- RepositoriesUsingIt(?language_name,
?repo_count), Language(?language_name).
```

2. **Mostra per ogni linguaggio il numero di domande postate su stackoverflow**:
```
q(?language_name, ?question_count) :- QuestionOnIt(?language_name,
?question_count), Language(?language_name).
```

3. **Mostra per ogni linguaggio, il grado di interesse nel tempo misurato come il numero di ricerche che gli utenti hanno lanciato su Google**:
```
q(?language_name, ?date, ?interest_rate) :-
InterestOverTimeLanguage(?language_name, ?date, ?interest_rate),
Language(?language_name).
```

4. **Mostra per ogni libreria o framework, il grado di interesse nel tempo misurato come il numero di ricerche che gli utenti hanno lanciato su Google al variare del tempo**:
```
q(?fw_name, ?date, ?interest_rate) :-
InterestOverTimeFrameworkLibrary(?fw_name, ?date, ?interest_rate),
LibraryOrFramework(?fw_name, _, _, _, _, _, _, _, _, _).
```

5. **Mostra per ogni linguaggio, il grado di interesse per regione misurato come il numero di ricerche che gli utenti hanno lanciato su Google**:
```
q(?language_name, ?region, ?interest_rate) :-
InterestByRegionLanguage(?language_name, ?region, ?interest_rate),
Language(?language_name).
```

6. **Mostra per ogni libreria o framework, il grado di interesse per regione misurato come il numero di ricerche che gli utenti hanno lanciato su Google al variare del tempo**:
```
q(?fw_or_lib, ?region, ?interest_rate) :-
InterestByRegionFrameworkLibrary(?fw_or_lib, ?region, ?interest_rate),
LibraryOrFramework(?fw_or_lib, _, _, _, _, _, _, _, _, _).
```

9.1.4. Corsi

1. **Mostra tutti i corsi per un determinato linguaggio**:
```
q(?slug, ?course_type, ?language_name, ?logo, ?photo_url, ?description,
?workload, ?url) :- Course(_, ?slug, ?course_type, ?language_name, ?logo,
?photo_url, ?description, ?workload, ?url), Language(?language_name).
2q(?slug, ?course_type, ?logo, ?photo_url, ?language_name, ?workload, ?url) :-
Course(_, ?slug, ?course_type, ?logo, ?photo_url, ?language_name, ?workload,
?url), Language(?language_name).
```

2. ** Mostra tutti i corsi per un determinato framework o libreria**:
```
q(?slug, ?course_type, ?fw_name, ?logo, ?photo_url, ?description, ?workload,
?url) :- Course(_, ?slug, ?course_type, ?fw_name, ?logo, ?photo_url,
?description, ?workload, ?url), LibraryOrFramework(?fw_name, _, _,_, _, _, _,
_ _, _).
q(?slug, ?course_type, ?logo, ?photo_url, ?fw_name, ?workload, ?url) :-
Course(_, ?slug, ?course_type, ?logo, ?photo_url, ?fw_name, ?workload, ?url),
LibraryOrFramework(?fw_name, _, _,_, _, _, _, _ _, _).
```

3. **Partner per ogni corso**:
```
q(?partner_id, ?partner_name) :- Course(_?course_id, _, _, _, _, _, _, _, _),
CoursePartner(?course_id, ?partner_name).
```

9.1.5. Lavori

1. **Tutti i lavori per un linguaggio**:
```
q(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name,
?lat, ?lon, ?language_name) :- Job(?job_title, ?description, ?post_date,
?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name),
Language(?language_name).
```
2. **Tutti i lavori per un framework**:
```
q(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name,
?lat, ?lon, ?fw_name) :- Job(?job_title, ?description, ?post_date,
?company_name, ?url, ?location_name, ?lat, ?lon, ?fw_name),
LibraryOrFramework(?fw_name, _, _,_, _, _, _, _ _, _)
```

9.2. Unfolding

Query scelta:
```
q(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat,
```

```
?lon, ?language_name) :- Job(?job_title, ?description, ?post_date, ?company_name,
?url ?location_name, ?lat, ?lon, ?language_name), Language(?language_name).
```

9.2.1. Unfolding GAV

Partendo dalla query:

```
q(?job_title, ?description, ?post_date, ?company_name, ?company_url,
?location_name, ?lat, ?lon, ?language) :- Job(?job_title, ?description,
?post_date, ?company_name, ?company_url, ?location_name, ?lat, ?lon,
?language_name), Language(?language_name).
```

Sostituiamo gli schemi mediati con le rispettive fonti locali, ottenendo:

```
q(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat,
?lon, ?language_name) :- S7.job(?job_title, ?description, ?post_date,
?company_name, ?company_url, ?location_name, ?lat, ?lon, ?language_name),
S1.language(?language_name, _).
```

e

```
q(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat,
?lon, ?language_name) :- S8.result(?job_title, ?description, ?post_date,
?company_name, ?company_url, ?location_name, ?lat, ?lon, ?language_name),
S1.language(?language_name, _).
```

9.2.2. Unfolding LAV

L'attività di unfolding della query tramite mapping LAV verrà svolta utilizzando il Bucket Algorithm. L' algoritmo si dividerà in tre fasi.

Creazione dei Bucket

Dati i due atomi contenuti nel corpo della query:

- `Job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name);`
- `Language(?language_name).`

otteniamo i corrispondenti bucket

bucket(g1)	bucket(g2)
`S8.result(?job_title, ?company_name, ?location_name, ?post_date, ?description, ?url, ?lat, ?lon, ?language_name)`	`S1.language(?language_name, ?v1)`
`S7.job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name)`	

Creazione delle riscritture candidate

In questa fase creeremo le possibili riscritture della query, considerando tutte le possibili combinazioni di atomi prelevati dai bucket. A partire dai due bucket ottenuti nella fase precedente, otteniamo i seguenti candidati:

- `q1(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name) :- S7.job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name), S1.language(?language_name, ?v1);`
- `q2(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name) :- S8.result(?job_title, ?company_name, ?location_name, ?post_date, ?description, ?url, ?lat, ?lon, ?language_name), S1.language(?language_name, ?v1);`

Verifica della correttezza dei candidati

In questa fase occorre verificare, fra tutti i candidati ottenuti nella fase precedente, quali siamo validi e quali no. Per dimostrare questo occorre dimostrare che le query candidate, espanse tramite i mapping LAV, siano contenute all'interno della query di partenza.
Detto questo dobbiamo verificare se le query

- `exp_q1(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name) :- Job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name), Language(?language_name), RepositoriesUsingIt(?language_name, ?v1), InterestOverTimeLanguage(?language_name, ?v2, ?v3), InterestByRegionLanguage(?language_name, ?v4, ?v5);`
- `exp_q2(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name) :- Job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name), Language(?language_name), RepositoriesUsingIt(?language_name, ?v1), InterestOverTimeLanguage(?language_name, ?v2, ?v3), InterestByRegionLanguage(?language_name, ?v4, ?v5);`

ottenuta espandendo `q1` e `q2`, siano contenute in `q`. Tuttavia notiamo immediatamente che entrambe le query sono identiche. Quindi tutto questo si riduce al verificare se la query:

- `exp_q1(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name) :- Job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name), Language(?language_name), RepositoriesUsingIt(?language_name, ?v1), InterestOverTimeLanguage(?language_name, ?v2, ?v3), InterestByRegionLanguage(?language_name, ?v4, ?v5);`

Per fare questo costruiamo un database canonico per `q1`. Per ogni variabile `v`, sia "v" la corrispondente costante. Il database canonico ottenuto da `q1` è il seguente:

Job	Language	RepositoriesUsingIt
("job_title", "description, "post_date", "company_name", "url", "location_name", "lat", "lon", "language_name")	("language_name")	("language_name", "v1")
InterestOverTimeLanguage	**InterestByRegionLanguage**	
("language_name", "v2", "v3")	("language_name", "v4", "v5")	

da cui otteniamo la seguente testa congelata:

- `q2("job_title", "description", "post_date", "company_name", "url", "location_name", "lat", "lon", "language_name");`

Ora eseguiamo la query iniziale sul seguente database canonico:

- le variabili contenute in`Job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name)` possono essere unificate con i valori contenuti nella tupla di Job del database canonico, ottenendo il fatto `Job("job_title", "description, "post_date", "company_name", "url", "location_name", "lat", "lon", "language_name");`
- dall'unificazione della fase precedente otteniamo il fatto `Language("language_name")` , che è contenuto nel database canonico. Da questo otteniamo il risultato finale della query `q("job_title", "description", "post_date", "company_name", "url", "location_name", "lat", "lon", "language_name")` .

Essendo il risultato finale ottenuto di fatto la testa congelata di `q1`, segue necessariamente che `q1` è contenuto in `q` e che quindi la riscrittura ottenuta applicando il Bucket Algorithm è valida. Da questo segue che il risultato dell'unfolding LAV della query in esame è il seguente:

- `q1(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name) :- S7.job(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name), S1.language(?language_name, ?v1);`
- `q2(?job_title, ?description, ?post_date, ?company_name, ?url, ?location_name, ?lat, ?lon, ?language_name) :- S8.result(?job_title, ?company_name, ?location_name, ?post_date, ?description, ?url, ?lat, ?lon, ?language_name), S1.language(?language_name, ?v1);`

ovvero l'OR delle query `q1` e `q2`.

9.3. Query SQL

9.3.1. Nomi di linguaggi e framework

1. **Elenco dei linguaggi**:

```
select name from Language
```

2. **Elenco dei nomi delle librerie e dei framework disponibili**:

```
`select name
from LibraryOrFramework`
```

3. **Elenco dei nomi delle librerie e dei framework disponibili con il relativo linguaggio**

```
`select LibraryOrFramework.name, Language.name
from LibraryOrFramework join Language
on LibraryOrFramework.language_used = Language.name`
```

4. **Elenco di nomi delle librerie e dei framework di un dato linguaggio**:

```
`select LibraryOrFramework.name
from LibraryOrFramework join Language
on LibraryOrFramework.used_language = Language.name
where Language.name = 'LINGUAGGIO'`
```

9.3.2. Caratteristiche di framework

1. **Elenco di tutti i framework e di tutte le librerie, con le caratteristiche principali**

```
`select Features.*
from LibraryOrFramework join Features
on LibraryOrFramework.name = Features.library_framework_name`
```

9.3.3. Popolarità

1. **Mostra per ogni linguaggio il livello di utilizzo su GitHub**:

```
`select repository_count, Language.name
from RepositoriesUsingIt join Language
on RespositoriesUsingIt.language = Language.name`
```

2. **Mostra per ogni linguaggio il numero di domande postate su stackoverflow**

```
`select name, QuestionOnIt.count
from QuestionOnIt join Language
on QuestionOnIt.tag = Language.name`
```

3. **Mostra per ogni libreria o framework il numero di domande postate su StackOverflow**:

```
 `select LibraryOrFramework.name, QuestionOnIt.count
from QuestionOnIt join LibraryOrFramework
on QuestionOnIt.tag = LibraryOrFramework.name`
```

4. **Mostra per ogni linguaggio, il grado di interesse nel tempo misurato come il numero di ricerche che gli utenti hanno lanciato su Google**:

```
`select interest_rate, date, language_name
from InterestOverTimeLanguage join Language
on InterestOverTimeLanguage.language_name = Language.name`
```

5. **Mostra per ogni libreria o framework, il grado di interesse nel tempo misurato come il numero di ricerche che gli utenti hanno lanciato su Google al variare del tempo**:

```
 `select interest_rate, date, fw_or_lib
from InterestOverTimeFrameworkLibrary join LibraryOrFramework
on InterestOverTimeFrameworkLibrary.fw_or_lib = LibraryOrFramework.name`
```

6. **Mostra per ogni linguaggio, il grado di interesse per regione misurato come il numero di ricerche che gli utenti hanno lanciato su Google**:

```
 `select interest_rate, region, language
from InterestByRegionLanguage join Language
on InterestByRegionLanguage.language = Language.name`
```

7. **Mostra per ogni libreria o framework, il grado di interesse per regione misurato come il numero di ricerche che gli utenti hanno lanciato su Google al variare del tempo**:

```
 `select interest_rate, region, fw_or_lib
from InterestByRegionFrameworkLibrary join Language
on InterestByRegionFrameworkLibrary.fw_or_lib = LibraryOrFramework.name`
```

9.3.4. Corsi

1. **Mostra tutti i corsi per un determinato linguaggio**:

```
`select slug, course_type, Language.name, logo, photo_url, description, workload,
url
from Course join Language
where Course.description like %Language.name% `
```

2. **Mostra tutti i corsi per un determinato framework o libreria**:

```
`select slug, course_type, LibraryOrFramework.name, logo, photo_url, description,
workload, url
from Course join LibraryOrFramework
where Course.description like %LibraryOrFramework.name%`
```

3. **Partner per ogni corso**:

```
`select partner_name, slug
from Course join CoursePartner
on Course.course_id = CoursePartner.course_id`
```

9.3.5. Lavori

1. **Tutti i lavori per un linguaggio**:

```
`select *
from Job join Language
where Job.description like %Language.name%`
```

2. **Tutti i lavori per un framework**:

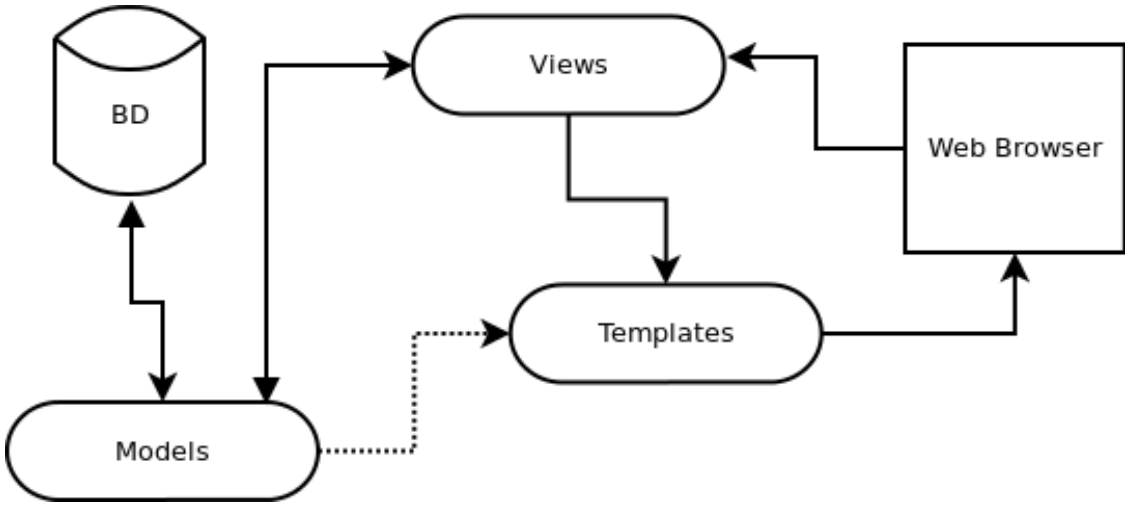
```
`select *
from Job join LibraryOrFramework
where Job.description like %LibraryOrFramework.name%`
```

9.4. Tecnologie utilizzate nella soluzione sviluppata

9.4.1. Back-End

Per lo sviluppo dei servizi REST abbiamo utilizzato **Django**: un web framework open source per lo sviluppo di applicazioni web, scritto in linguaggio Python, seguendo il paradigma Model-View-Template (MVT).
Perchè usare Django? Ecco i principali motivi:
• è sviluppato in Python;
• prevede tempi di sviluppo minori;
• segue il principio DRY (Don't Repaet Yourself);
• offre un'interfaccia admin generata automaticamente;
• corredato da un'ottima documentazione e comunità attiva.

Lo schema globale è mappato interamente nei *model* dell'applicazione. Questo permette di gestire in maniera semplice e sicura la cache.



9.4.2. Front-End

I linguaggi utilizzati sono HTML, CSS e Javascript. Lo sviluppo della webapp ha richiesto l'ausilio di diversi componenti e librerie esterne:
• **Bootstrap3**: celeberrimo framework front-end per il design di applicazioni.
• **ReactJS**: una libreria Javascript creata da Facebook e Instagram (https://facebook.github.io/react/) che permette di realizzare applicazioni single-page (Single Page Applications (SPA)) attraverso una struttura che la divide in componenti dinamici e riutilizzabili.
Le motivazioni dietro la scelta di React sono le seguenti:

Virtual Dom

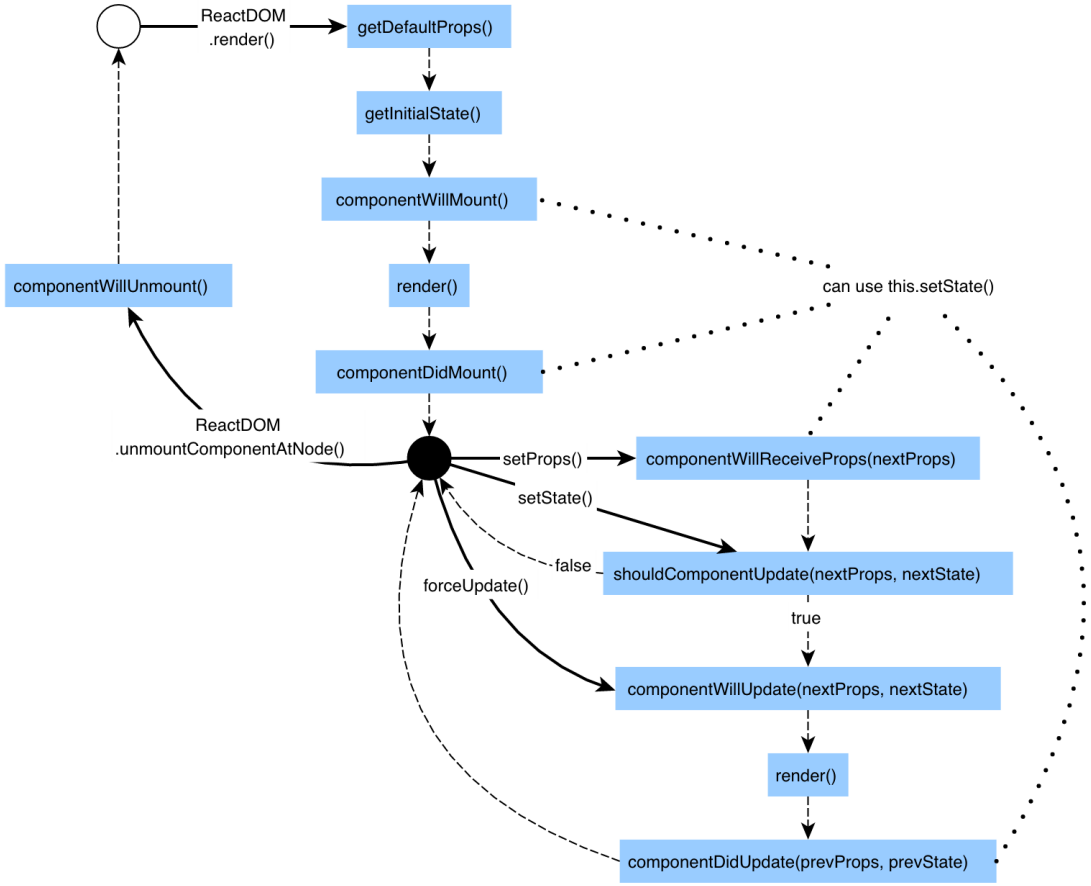
A causa della natura dei browser, che eseguono gli script JavaScript in un unico thread, dove eseguono il rendering della pagina, le performance delle applicazioni web possono ridursi drasticamente sui dispositivi che hanno a disposizione poca capacità di calcolo o poca memoria. Il problema non è dovuto alla lentezza di JavaScript, ma solitamente alla lentezza delle chiamate alla manipolazione degli elementi del DOM della pagina, le quali, oltre ad essere lente, risultano bloccanti nell'esecuzione del codice.
Tuttavia questi problemi di performance dovuti al DOM possono essere migliorati utilizzando ReactJS. ReactJS mantiene in memoria una copia virtuale del DOM, chiamata appunto Virtual DOM, e ogni qualvolta viene applicata una modifica alla pagina, invece di applicare direttamente le modifiche al DOM, ReactJS modifica il DOM virtuale e calcola il set minimo di modifiche da applicare sul DOM reale, riducendo quindi l'impatto sulle performance.

Declarative

L'approccio classico allo sviluppo web è solitamente legato al concetto di evento e callback. Tale logica risulta semplice per applicazioni di piccole dimensioni, ma man mano che l'applicazione cresce, il codice diventa sempre più incomprensibile e difficile da leggere.
L'approccio di ReactJS è totalmente diverso; quando si sviluppa una view in ReactJS si definisce a priori il comportamento della suddetta in ogni momento del suo lifecycle, in base al suo stato. Questo approccio risulta molto più comprensibile sul lungo termine.

Componenti Riutilizzabili

ReactJS è basato su dei componenti debolmente accoppiati tra di loro, che, messi insieme, contribuiscono allo sviluppo dell'intera applicazione.
Ogni componente è formato da una view, che ne determina l'aspetto, da una serie di funzioni ed uno stato.
Ad ogni variazione allo stato, viene chiamata la funzione `render()`. Tale funzione ha lo scopo di ricaricare il componente.
Nel dettaglio, ogni componente ReactJS ha una serie di hook che si attivano in base allo stato nel lifecycle nel quale si trova un componente.



React Lifecycle