# Извличане на информация

## Софийски Университет

## Факултет по математика и информатика

## 20.02.2016

## Git Insights

## Мартин Боянов, ФН 25187, Извличане на информация и откриване на знания

# Съдържание



# Декларация за плагиатство

Тази курсова работа е моя работа, като всички изречения, илюстрации и програми от други хора са изрично цитирани. Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция. Разбирам, че ако се установи плагиатство в работата ми ще получа оценка “Слаб”.
Мартин Костадинов Боянов:

# I. Мотивация

С развитието на софтуерното инженерство, проектите в практиката стават все по-сложни и все по-големи. В някои случаи става дума за проекти с милиони редове програмен код, по които работят стотици хора. Често се получава хората, които подържат кода да са съвсем различни от хората, които са го написали. Git е технология, която позволява на разработчиците да работят по различни части от проекта едновременно. Освен че следи промените, които разработчиците правят, Git позволява да се добави и описание на направените промени.
Целта на настоящата работа е да се създаде извличаща система, която да позволява свободно търсене в историята на разработката на приложението. С помощта на тази система ще може да се намерят всички компоненти, засягащи дадена функционалност или да се открие човека, компетентен да отговори за развитието на функционалността.

# II. Кратък обзор

Git като приложение за контролиране на програмния код има много вградени функционалности за търсене. Подържат се следните функционалности:

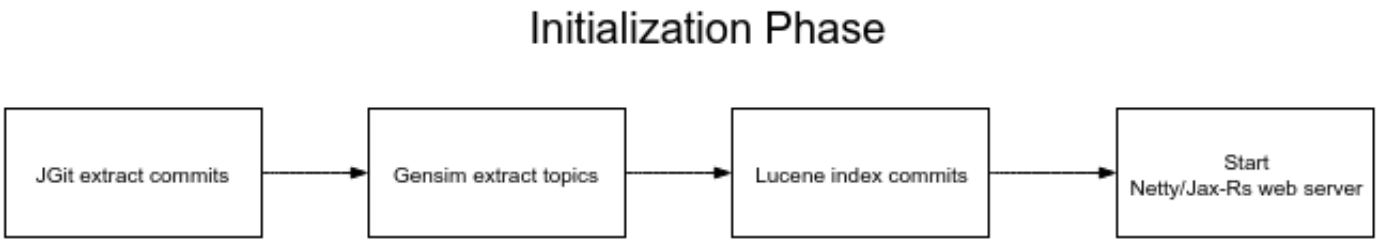
- Търсене по автор
- Търсене по id
- Търсене по файл
- Търсене чрез grep
- и др.

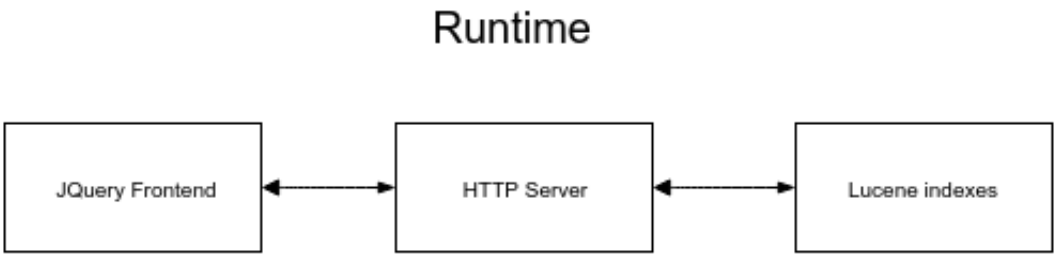
Проблемът с въпросните функционалности е, че те са достъпни чрез специализиран синтаксис, които е предназначен за напреднали потребители. Друг недостатък е, че е необходим физически достъп до сорс кода на приложението. Още един проблем е, че въпросните функционалности следват булевия модел на извличане на информация и не позволяват ранкиране.
Целта на настоящата разработката е да надстрои над тези базови функционалности като добави възможност за пълнотекстово търсене с ранкиране, достъпно чрез уеб приложение. Това приложение ще позволи на по-голям набор от хора да взаимодействат с историята на проекта, както и ще им даде достъп до различни статистики за проекта.

Трябва да се отбележи , че съществуват и много приложения за статичен анализ на сорс код. Тези приложения позволяват намирането на бъгове и на потенциални проблеми със сигурността, както и идентифицирането на компоненти, които са критични за ефикасността и производителността на програмата. Интеграция с такива приложения би била много ползотворна, но е извън обхвата на сегашната разработка.

# III. Архитектура и алгоритми

Архитектурата на приложението може да се обобщи със следната графика:




# JGit извличане на информация

Първата стъпка в процеса е да се подаде адреса .git базата, в която се пази цялата история на проекта. JGit е Java библиотека, която може да борави със специалните .git формати. С помощта на тази библиотека приложението извлича информацията за всички промени, които са правени по проекта от създаването му и ги десериализира в удобен за употреба обект.

Обектът следва структурата на промяната и съдържа:

- Автор
- уникален хаш
- списък от променени файлове
- кратко описание
- пълно описание

# Gensim извличане на ключови/релевантни думи

След като се десериализират, промените се сериализират отново в .json формат и се подават на специален python скрипт, който извлича ключовите думи от тях и ги връща обратно на приложението.
Ключовите думи се извличат чрез известния алгоритъм за намаляване измеренията *латентно семантично индексиране* (LSI). Това преобразувание се изпълнява от библиотеката GenSim. За целта документите трябва да минат през следните трансформации:

1. Токенизация - тъй като алгоритъмът е ненадзирован, резултатите от него силно се влияят от качеството на токенизатора
2. Трансформация към bag-of-words
3. Обучение/Пресмятане на ключовите вектори
4. Реиндексация чрез получените вектори

LSI получава като единствен параметър бройката ключови вектори. В хода на проекта бе избрана стойността 200 ключови вектора, като тя подлежи на промяна.
След реиндексацията, документите вече са част от векторното пространство на ключовите вектори, които са премерена сума (weighted sum) на оригиналните термове. За ключови думи се избират термовете с положително тегло, които са в 3-те най-тежки ключови вектори.

# Lucene индексиране

След като се извлекат ключовите думи, коментарите се предават на персонализиран Lucene индекс. Задават се специфични правила за токенизация, които да олесняват последващото търсене. Освен това се задават и правила за увеличаване на тежестта за търсенето по полетата *кратко описание*, *автор*, *ключови думи* като целта е тези тежести да могат лесно да се променят.
Докато тече индексацията се изчисляват и различни статистики за авторите:

- Обща бройка промени
- Обща бройка променени файлове

Също така за всеки файл се пазят авторите, които са го променяли.

# Стартиране на web server

Последната стъпка в инициализацията на проекта е да се стартира уеб сървър, към който да се

пращат заявките. Самият уеб сървър е netty, като комуникацията се извършва чрез Jax-Rs Resteasy ресурси и Jackson JSON parser.

# Процес на работа

След инициализацията, комуникацията с приложението се извършва чрез HTTP и JSON. Съществуват 3 ендпойнта:

1. `http://localhost:1337/query?query={query}` - поема пълнотекстовата заявка и връща първите 20 резултата.
2. `http://localhost:1337/users/:email` - поема емейл на user и връща статистики за него
3. `http://localhost:1337/files/:path` - поема път към файл и връща емейлите на авторите, които са го променяли.

Написан е тънък слой от HTML и JQuery, който да комуникира със сървъра и да визуализира резултите. Този слой е имплементиран като отделен проект като идеята е да се изолира презентацията на данните от структурата на приложението.
Презентационният слой има минималистичен дизайн като предоставя единствено една форма за въвеждане на пълнотекстовата заявка, а навигирането към различните ресурси се осъществява чрез хипервръзки:

# Заключение

Софтуерната индустрия се развива с огромни темпове и това води до все по-големи и по-сложни проекти. Програмният код подлежи на ентропия и с времето става все по-труден за подържане и модифициране. Затова е важно да може да се навигира не само текущата структура на проекта, но и историческото му развитие. Полезно е да може да се извлича и семантична информация за функционалността. Сорс кода казва на машините как да функционират, но историята на проекта разказва и **какво** и **защо** го правят. Има скрита семантика и в имената на променливите и методите, както и в структурата на взаимовръзките между компонентите.
Git Insights е един инструмент, който позволява да се намери част от тази скрита семантика. Той е в начална фаза на своето развитие като може да се подобри по следните(и други) начини:

- Индексиране на самия сорс код
- Анализ на връзките между файловете
- Интеграция с продукти за следене на развитието ( Github Issues, Jira и др.)
- Интеграция с продукти за статичен анализ на код
- Подобрение на токенизаторите
- Подобрение на извличане на ключовите думи
- Техники за query expansion - плурализация, синоними, преобразуване във векторното пространство на ключовите думи
- по-детайлни статистики

Въпреки липсата на тези функционалности, Git Insights дава инструменти за търсене на базовия потребител по лесен и достъпен начин.

# Литература и библиотеки


1. Lucene https://lucene.apache.org/core/
2. Resteasy HTTP Server http://resteasy.jboss.org/
3. GenSim topic modelling https://radimrehurek.com/gensim/
4. JGit Git Integration https://eclipse.org/jgit/