МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ИДЗ

по дисциплине «Разработка ПО ИС»

Тема: Поиск скрытых групп

Рымарь М.И.
Алешин С.В.
Студенты гр. 4383 Коптюг А.Д.

Преподаватель Заславский М.М.

Санкт-Петербург
2017

ЗАДАНИЕ
К ИДЗ

Студенты Рымарь М.И., Алешин С.В., Коптюг А.Д.

Группа 4383

Тема работы: поиск скрытых групп в социальной сети

Исходные данные:
Данные пользователей социальной сети «Вконтакте»

Содержание пояснительной записки:
«Содержание», «Введение», «Качественные требования к решению сценарии использования «, «Модель данных», «Разработанное приложение», «Выводы», «Заключение», «Список использованных источников».

Дата выдачи задания: 07.09.2017

Дата сдачи реферата: 27.12.2017

Дата защиты реферата: 28.12.2017

Рымарь М.И.

Алешин С.В.

Студенты Коптюг А.Д.

Преподаватель Заславский М.М.

АННОТАЦИЯ

Индивидуальное домашнее задание по данному курсу предусматривает освоение навыков по работе с нереляционными базами данных: запись в базу данных, обработка данных в базе, получение данных из базы. В работе была использована такая база, как «Neo4j», которая представляет собой графовую нереляционную базу данных. В работе анализировались данные из социальной сети «Вконтакте», были созданы критерии для скрытых групп, в которые могут попасть различные пользователи сети. Пользователи скрытых групп отображаются на странице приложения в виде графа.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность

Социальные сети на текущий момент занимают значимое место в жизни многих людей. Многие пользователи интернета заводят себе страницы в разных сетях для общения со своими друзьями, поиска новых знакомств либо ради получения какой-либо полезной информации из новостей или сообществ. Социальная сеть «Вконтакте» является одной из самых популярных сетей в России на текущий день: более 97 000 00 человек пользуется Вконтакте ежемесячно. Огромное количество людей оставляют самую разную информацию о себе. Вконтакте имеет встроенные фильтр по поиску людей, однако не предоставляет выбора всех опций, который могут быть интересны. Такую задачу стремится выполнить приложение «Поиск скрытых групп Вконтакте», которое позволяет найти группы людей, которые было бы не так просто найти через Вконтакте обычным поиском.

Постановка задачи

Из вышеизложенного ясно, что поставленной задачей является поиск выбор пользователей из какой-либо большой группы по определенным признакам для записи из в более узконаправленную группу. Признаки могут быть самыми различными, этим руководствуются разработчики приложения. Таким образом необходимо получать данные о пользователях из социальной сети ВК, загружать данные в базу данных, затем осуществлять поиск данных в базе по определённым признакам и выгружать полученные данные с дальнейшей демонстрацией их пользователю.

Предлагаемое решение

В данной работе предлагается использовать следующее решение. У ВК существует API, который позволяет получать данные о пользователях, группах и т.д. Запросы к API будут осуществляться из приложения, написанного на php. Полученные данные будут приходить в формате JSON. Из данного формата приложение будет разбирать необходимые данные для записи, подготавливать данные в необходимом формате для записи из в базу данных Neo4j, затем записывать. Далее к данным можно совершать запросы из приложения для получения необходимого результата.

КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Текущие

Решение должно быть реализовано в виде сайта (одностраничного), где присутствуют формы для ввода пользователем информации, через которую он сможет донести, какого конкретно результат он ожидает (какую конкретную скрытую группы пользователь ищет). Для анализа могут быть использованы как различные сообщества социально сети, так и пользователи. В первом случае должны анализироваться участники сообществ, во втором – друзья указанного пользователя ВК. Результаты анализа полученных данных должны выводится пользователю на той же странице приложения, с которой он осуществлял ввод информации в формы. Результат должен быть предоставлен пользователю в виде графа, с которым пользователь может интерактивно взаимодействовать: менять положение узлов на экране, получить дополнительную информацию об узлах (ФИО, ссылку на страницу в социальной сети).

Перспективные

При росте популярности приложения необходимо проработать более эргономичный дизайн интерфейса сайта, с которым взаимодействует, пользователь для привлечения еще большей аудитории. Также необходимо будет добавить встроенную возможность «Поделиться с друзьями», которая бы оставляла ссылку на полученные результаты пользователям в вышеназванной социальной сети, что поспособствовало бы приросту новых пользователей.

При росте нагрузки приложения необходимо реализовать распараллеливание основных операций, таких как: отправка запроса к API ВК, запись в БД, обработка данных в БД. Также хорошей идеей было бы вынести серверную часть приложения с базой данных в облачные вычисления.

СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

Для данного приложения был разработан следующий макет:

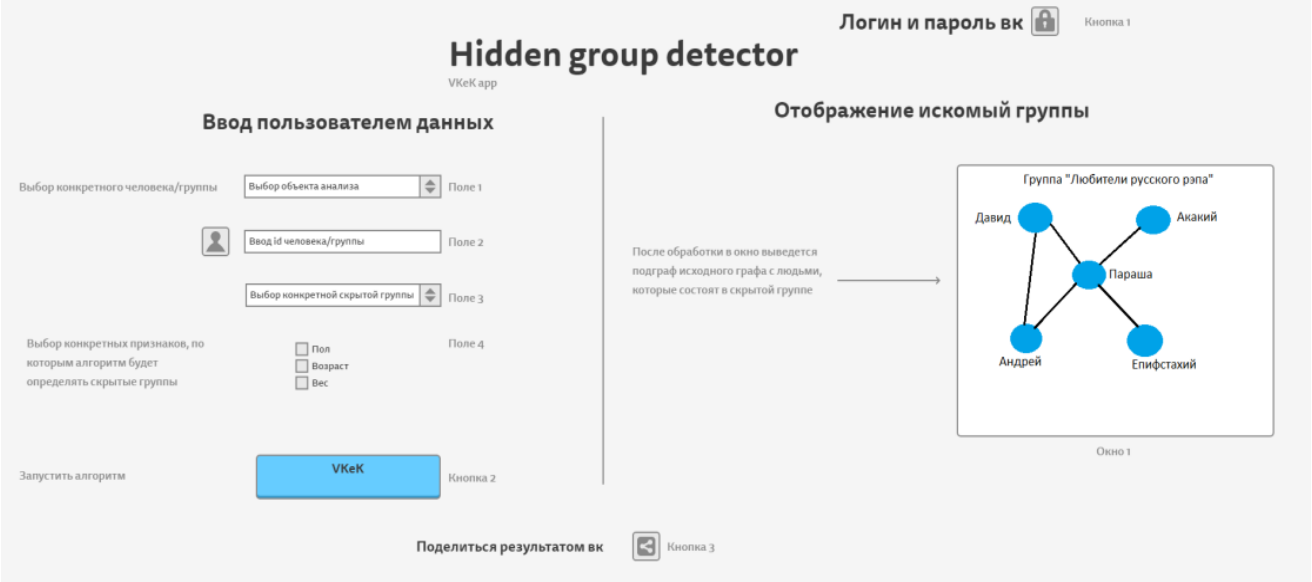


Рис.1

На макете подписаны все поля, с которыми взаимодействует пользователь. Далее описан основной сценарий использования приложения:

1. Пользователь вводит в строке в своем браузере адрес "http://www.VKeK.com" и переходит на этот сайт

2. В поле 2 "Ввод id человека/группы" пользователь вводит id человека или группы, в подписчиках и/или друзьях которого(-ой) будет производиться поиск скрытых групп.

3. В поле 3 из выпадающего списка пользователь выбирает, какую скрытую группу нужно искать программе (Например, "Фанаты немецкого кинематографа")

4. Пользователь нажимает по кнопке 2 "VKeK" для начала выполнения алгоритма

5. В окне 1 "Отображение искомой группы" рисуется граф, показывающий скрытую группу в виде графа со связи между ее участниками, либо отображается сообщение о неудаче/ошибке, если поиск скрытой групп не дал результатов

Для *импорта* информации в базу данных:

1. Пользователь вводит в строке в своем браузере адрес "http://www.VKeK.com" и переходит на этот сайт

2. В поле 2 "Ввод id человека/группы" пользователь вводит id человека или группы, в подписчиках и/или друзьях которого(-ой) будет производиться поиск скрытых групп.

3. Пользователь нажимает по кнопке 2 "VKeK" для начала выполнения алгоритма

Таким образом после получения ответа на запрос к API, который выполнит приложение на основе введенного пользователем id, данные о друзьях введенного пользователя либо об участниках группы будут записаны в БД.

Для *анализа* данных:

1. Пользователь вводит в строке в своем браузере адрес "http://www.VKeK.com" и переходит на этот сайт

2. В поле 2 "Ввод id человека/группы" пользователь вводит id человека или группы, в подписчиках и/или друзьях которого(-ой) будет производиться поиск скрытых групп.

3. Пользователь нажимает по кнопке 2 "VKeK" для начала выполнения алгоритма

Пользователь выполняет те же действия, что и при импорте, так как приложение занимается анализом данных сразу после импорта их из социальной сети ВК.

Экспорт данных нашим приложением не предусмотрен. Это сделано в целях обеспечения безопасности данных пользователей ВКонтакте - чтобы каждый желающий не смог при помощи нашего приложения выгружать огромное количество персональных данных. Приложение лишь представляет возможности анализа информации о группах людей.

МОДЕЛЬ ДАННЫХ

Для нашего приложения была разработана следующая схема модели данных:

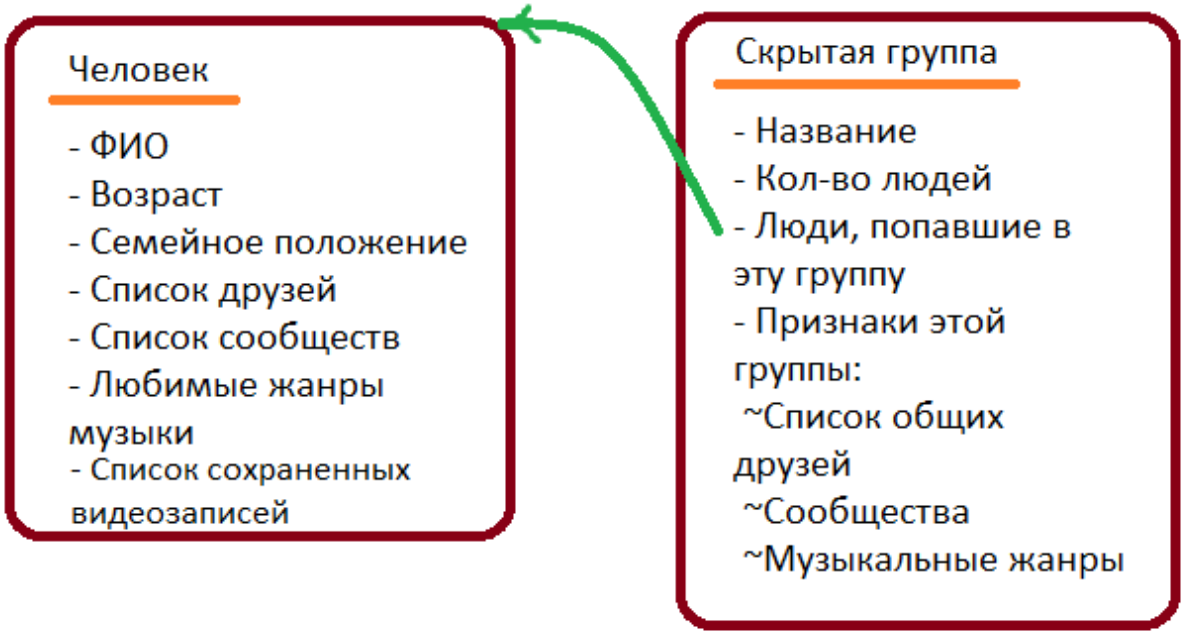


Рис.2

Из нее следует, что в базе данных хранятся данные о все пользователях группы/человека, которых мы анализируем, но те, которые являются участниками выбранной скрытой группы имеют дополнительную «связь» принадлежности к этой скрытой группе (В следующий разделах будет показано, как это хранится в базе данных).

Описание назначения коллекций модели данных

Данная модель данных включает в себя 3 сущности - Человек, Скрытая группа и Выборка

Она содержит такие коллекции, как:

Коллекции, относящиеся к скрытой группе:

- Список людей, определенных в данную скрытую группу

Коллекции, относящиеся к конкретному человеку:

- Список сообществ, на которые подписан человек
- Количественные признаки (друзья, подписчики и т.д.)
- Другая информация о пользователях

Коллекции, относящиеся к выборке:

- Список людей, принадлежащих к данной выборке

Описание модели данных

Источник данных

API ВКонтакте

Структура данных

Данные представлены в виде ненаправленного графа, узлы которого содержат следующие данные:

- ФИО пользователя
- id пользователя
- возраст
- список сообществ, на которые подписан пользователь
- список любимых жанров музыки (исполнителей/групп)
- информация по сохраненным видеозаписям

Ребра в графе эквивалентны отношению "дружба и/или схожесть по критериям". Отношение дружбы: многие-ко-многим

Исходный формат данных

JSON-файл

Запросы к NoSQL-модели

- Запрос на создание узла, описывающего конкретного человека

```
CREATE (n: User {user_id: 8430389, first_name: 'Deadmouse', last_name:
```

```
'Lmao’, bdate: D.M.YYYY})
CREATE (k: Community {comm_id:19774, comm_name: 'Keaboard'})
CREATE (m: Hidden_groups {hidden_group_name: 'Любители музыки',
count_of_users: 100})
```

Запрос на сравнение нескольких узлов по выбранному признаку

```
MATCH (n: User {first_name: 'Deadmouse'} RETURN n;
```

- Запрос на добавление нового узла пользователя в "Скрытую группу"

```
MATCH (User {user_id: 8430389}), (Hidden_groups {hidden_group_name:
'Любители музыки'})
MERGE (User)-[r:Composed]->(Hidden_groups)
```

- Запрос на добавление связи между определённым пользователем и сообществом:

```
CREATE (n: User {user_id: 8430389})-[:Following]->(k:
Community{comm_id:19774})
```

- Запрос на выборку данных из "Скрытой группы"

```
MATCH (n:User)-->(m:Hidden_groups {hidden_group_name: 'Любители
музыки'}) RETURN n;
```

Описание типов используемых данных

Типы данных, используемые для описания человека:

- "_id", int - id человека
- "firstName", String - имя
- "secName", String - фамилия
- "midName", String - отчество
- "birth", String - дата рождения
- "status", bool - находится ли человек в отношениях
- "friends", array - массив друзей
- "communities", array - массив id сообществ
- "music", array - массив любимых жанров музыки
- "videos", array - массив названий сохраненных видеозаписей человека

Типы данных, используемые для описания сообщества:

- "name", String - название группы
- "quantity", double - количество людей, попавших в скрытую группу
- "members", array - массив людей, попавших в эту скрытую группу
- "commonFriends", array - массив общих друзей
- "commonCommunities", array - массив схожих сообществ
- "commonMusic", array - массив схожих любимых жанров музыки

Оценка удельного объема информации, хранимой в модели (сколько потребуется памяти, чтобы сохранить)

Типы данных, используемые для описания человека:

- "_id", 4 байта
- "firstName", средняя длина имени 8 символов ~ 8 байт

- "secName", средняя длина фамилии 10 символов ~ 10 байт
- "midName", средняя длина отчества 13 символов ~ 13 байт
- "birth", формат данных дд/мм/гг - 8 байт
- "status", занят или свободен - 1 байт
- "friends", в среднем человек имеет 75 друзей в вк ~ 4 * 75 байт
- "communities", в среднем человек периодически посещает 10 одних и тех же сообществ ~ 4 * 10 байт
- "music", человек слушает обычно от 3 до 10 различных жанров музыки ~ 1 * 10 * 75 байт
- "videos", предположим, что у среднестатистического человека в сохраненных видеозаписях присутствует 30 видео ~ 1 * 15 * 30 байт

Типы данных, используемые для описания сообщества:

- "name", длина названия группы ~ 10 байт
- "quantity", 8 байт
- "members", предположим, что мы рассматриваем группы из 100 человек ~ 4 * 100 байт
- "commonFriends", предположим, что в скрытой группе 20%-ов людей окажутся общими друзьями ~ 4 * 25 байт
- "commonCommunities", предположим, что в схожее сообщество попадают такие группы, на которые подписаны хотя бы 5 человек целевой аудитории, и таких групп на группу из 100 человек будет 15 ~ 4 * 15 байт
- "commonMusic", предположим, что в скрытой группе по музыкальному вкусу будет 2-3 общих музыкальных вкуса ~ 1 * 3 * 10 байт

Реляционная модель данных SQL

Ниже мы рассмотрели вариант использования реляционной модели данных для нашего приложения.

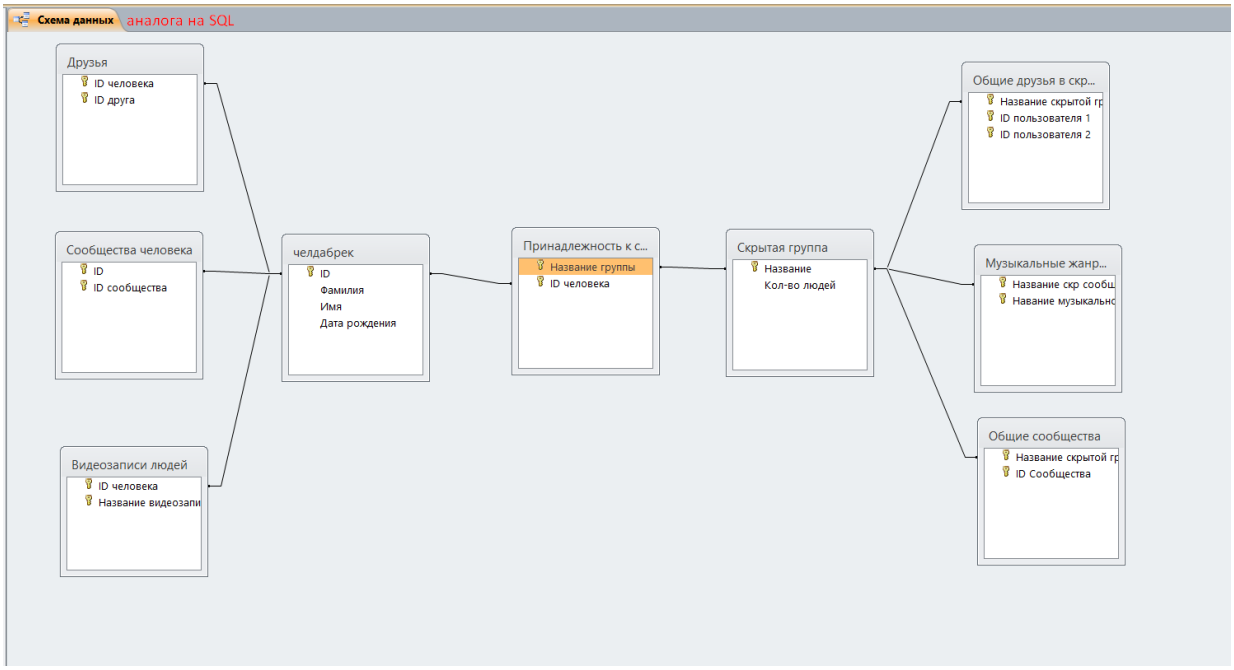


Рис.3

На рис.3 представлена схема модели данных для SQL-решения.

Описание назначений таблиц, типов данных и сущностей

Реляционную модель составляют следующие таблицы:

Человек:

- "ID" - длинное целое (ключ)
- "Имя" - varchar(20)
- "Фамилия" - varchar(20)
- "Дата рождения" - Дата/время

Друзья человека:

- "ID человека" - длинное целое
- "ID друга" - длинное целое

Ключом является данная пара значений

Сообщества человека:

- "ID человека" - длинное целое
- "ID сообщества" - длинное целое

Ключом является данная пара значений

Скрытая группа:

- "Название" - varchar(20), ключ
- "Кол-во людей" - длинное целое

Общие друзья в скрытой группе:

- "Название скрытой группы" - varchar(20)
- "ID пользователя 1" - длинное целое
- "ID пользователя 2" - длинное целое

Ключом является данная тройка значений

*Музыкальные жанры в скрытой груп*пе:

- "Название скрытой группы" - varchar(20)
- "Название музыкального жанра" - varchar(10)

Ключом является данная пара значений

Общие сообщества:

- "Название скрытой группы" - varchar(20)
- "Название сообщества" - длинное целое

Ключом является данная пара значений

Оценка удельного объема информации, хранимой в модели

N - кол-во людей в выборке.

*Человек (54 * N байт):*

- "ID" - длинное целое, 4 байта
- "Имя" - varchar(20), 20 байт
- "Фамилия" - varchar(20), 20 байт
- "Дата рождения" - Дата/время, 10 байт

*Друзья человека (8 * 75 * N байт):*

- "ID человека" - длинное целое, 4 байта
- "ID друга" - длинное целое, 4 байта

Ключом является данная пара значений

*Сообщества человека (8 * 10 * N байт):*

- "ID человека" - длинное целое, 4 байта
- "ID сообщества" - длинное целое, 4 байта

Ключом является данная пара значений

Скрытая группа (24 байта - для 1-ой скрытой группы):

- "Название" - varchar(20), 20 байт
- "Кол-во людей" - длинное целое, 4 байта

*Общие друзья в скрытой группе (28 * 20 * 5 байт, из расчета, что 20%-ов скрытой группы являются общими друзьями, и общим другом считается тот человек, который дружит не менее чем с 5-ью членами выборки):*

- "Название скрытой группы" - varchar(20), 20 байт
- "ID пользователя 1" - длинное целое, 4 байта
- "ID пользователя 2" - длинное целое, 4 байта

Ключом является данная тройка значений

*Музыкальные жанры в скрытой группе (30 * 3 байт):*

- "Название скрытой группы" - varchar(20), 20 байт

- "Название музыкального жанра" - varchar(10), 10 байт

Ключом является данная пара значений

*Общие сообщества (24 * 15 байт):*

- "Название скрытой группы" - varchar(20), 20 байт
- "Название сообщества" - длинное целое, 4 байта

Ключом является данная пара значений

Запросы на SQL

1)(Добавление информации о человеке)

INSERT "ID","фамилия","имя","Дата рождения" IN человек INSERT (в таблицу друзья) INSERT (в таблицу сообщества человека) INSERT (в таблицу видеозаписи)

2) (Добавление скрытой группы)

INSERT "название_скрытой_группы", "кол-во человек" IN Скрытые_группы INSERT (в таблицу общие друзья) INSERT (в таблицу Муз жанры) INSERT (в таблицу видеозапии)

3)(Поиск людей по определенному признаку)

SELECT * FROM человек WHERE (SELECT * FROM Видеозаписи LEFT JOIN человек ON челдабрек.ID = Видеозаписи.id).название_видеозаписи = "Котики"

4) (Добавление определенного человека к скрытой группе)

INSERT "Название", "ID_человека" IN Принадлежность_к_скрытым_группам

5) (Получение информации о скрытой группе и ее признаках)

```
SELECT * FROM Скрытые_группы, Общие_друзья LEFT OUTER JOIN Общие_друзья ON Скрытые_группы.название = Общие_друзья.название, Музыкальные_жанры_скр_сообщества LEFT OUTER JOIN Музыкальные_жанры_скр_сообщества ON Скрытые_группы.название = Музыкальные_жанры_скр_сообщества .название, Общие_сообщества LEFT OUTER JOIN Общие_сообщества ON Скрытые_группы.название = Общие_сообщества.название GROUP BY Скрыты_группы.название (все данные по скрытой группе)
```

6) (Получение информации о людях в определенном сообществе)

```
SELECT * FROM Принадлежность_к_скр_сообществам WHERE Принадлежность_к_скр_сообществам.Название_группы="Любое_название"
```

Вывод

- Запросы на SQL-языке являются более громоздкими, чем на языке *Neo4j Cypher*.
- В реализации реляционной модели данных пришлось бы создать слишком большое количество таблиц для связи всей информации
- В SQL-реализации пришлось бы выделить 54*N + 600*N + 80*N + 24 + 2800 + 90 + 360 = 734*N + 3274*M байта, где N - количество целевой выборки, M - количество скрытых групп.
- В то же время в NoSQL-реализации пришлось бы выделить (4+8+10+8+1+300+40+750+450)*N + 18+400+100+60+30 = 1571*N + 608*M байт, где N - количество целевой выборки, M - количество скрытых групп.

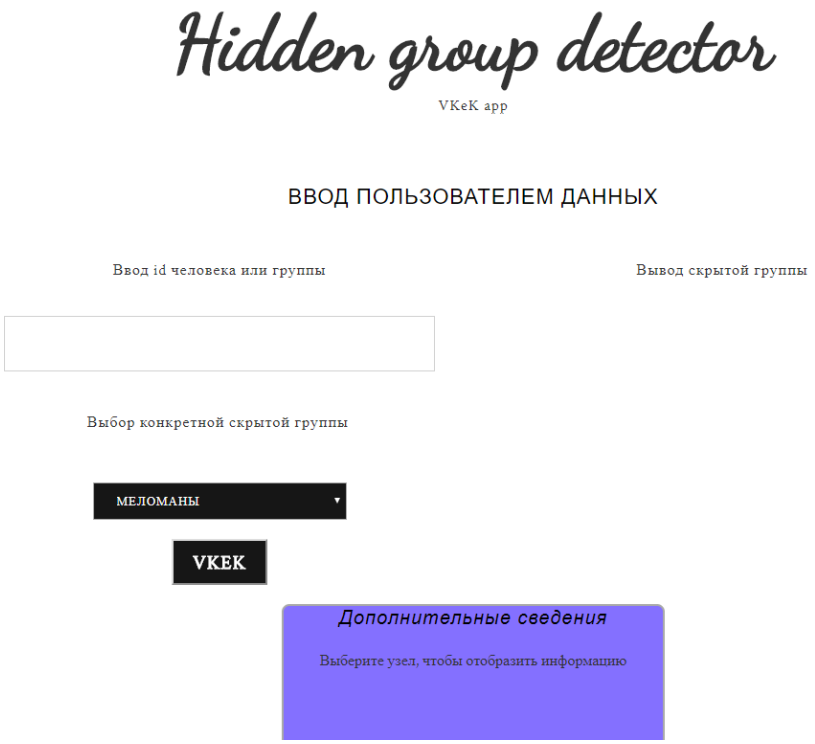
- Можно сделать вывод, что при хранении данных о большом количестве скрытых групп NoSQL реализация выигрывает по количеству занимаемой памяти.
- для рассматриваемой задачи NoSQL модель данных предпочтительнее SQL модели в плане сложности запросов к БД и требуемой для хранения данных памяти

РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

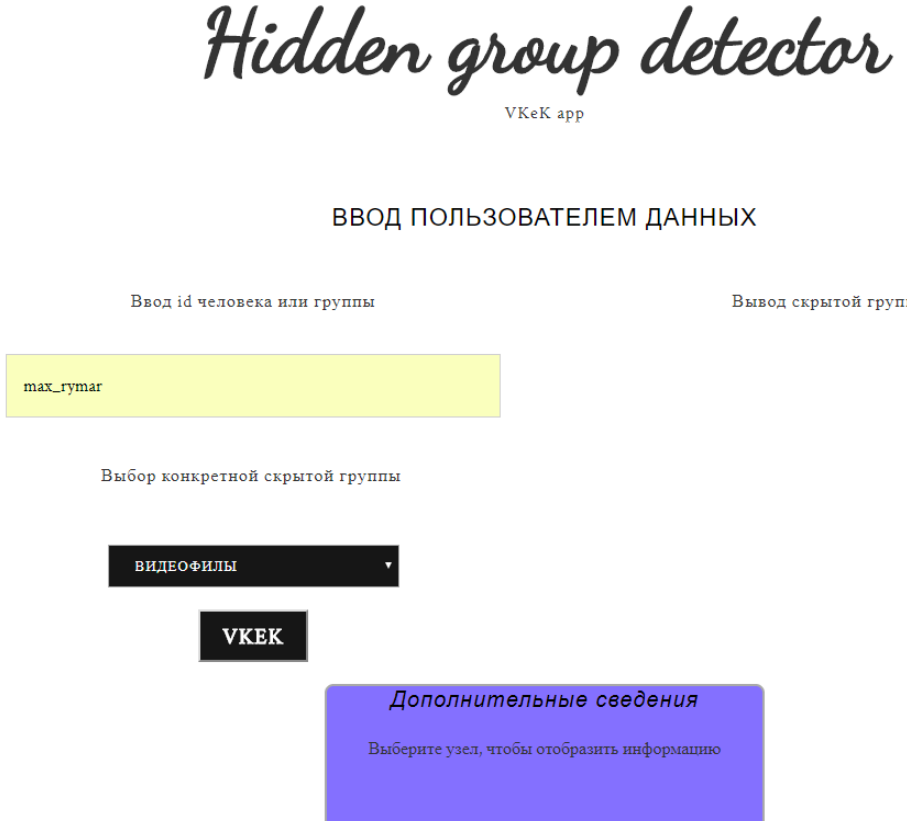
Краткое описание

Разработанное приложение представляет собой интернет-сайт, на который может зайти пользователь и воспользоваться им. Пользователю необходимо ввести id пользователя/группы для начала работы с программой, также необходимо выбрать скрытую группу, для анализа данных приложением.

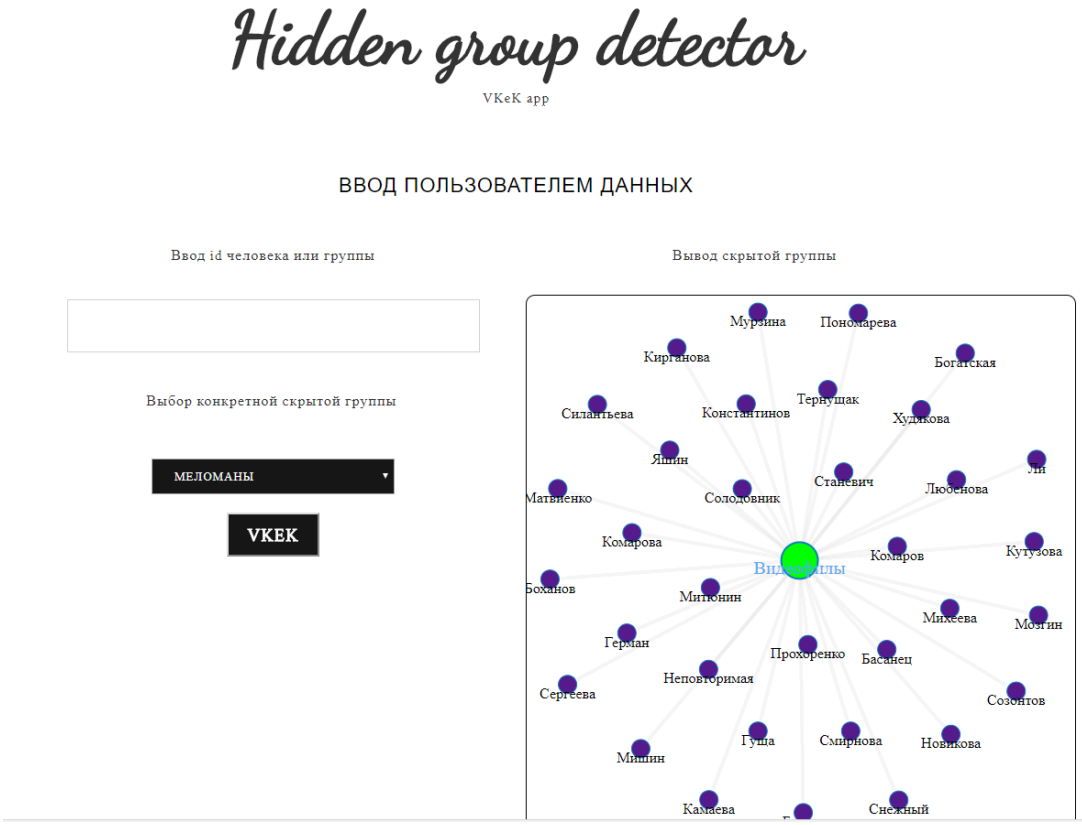
Схема экранов приложения и/или схема интерфейса командной строки



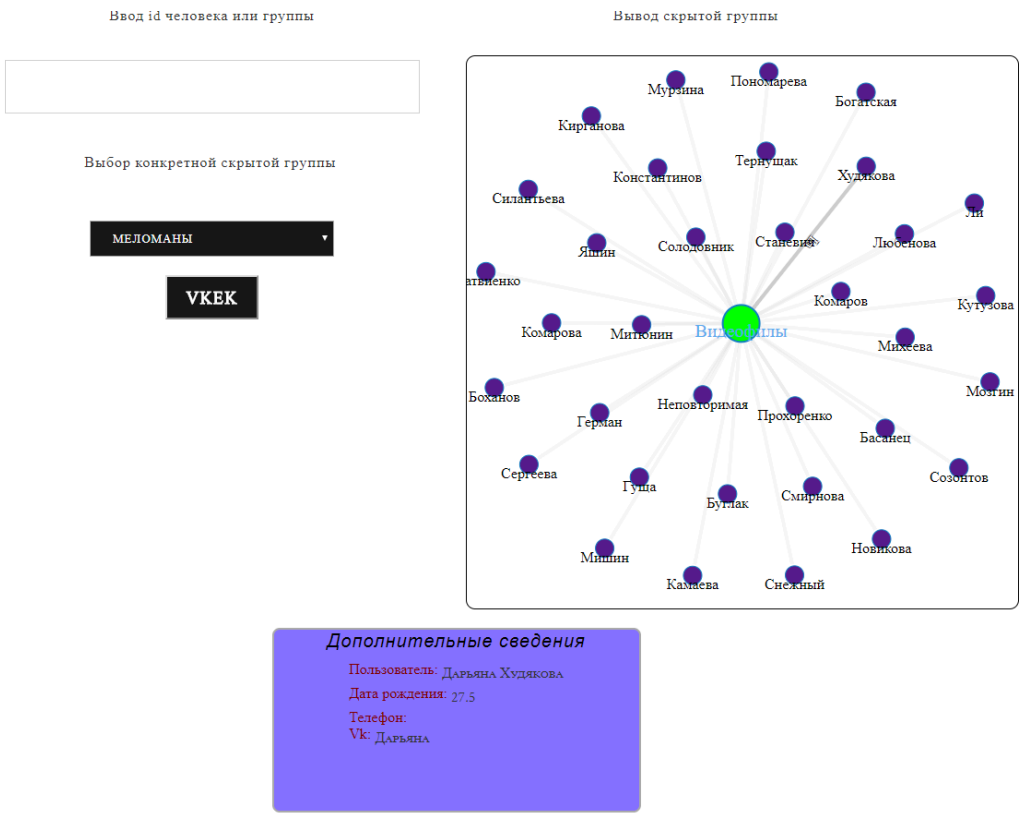
Стартовый экран приложения



Стартовый экран приложения с введенными пользователем данными



Приложение после обработки данных



Вывод дополнительной информации по клику на узел

Использованные технологии

Для разработки приложения были использованы следующие технологии:

- Backend

Были использован язык программирования *php* в виду хорошей его адаптации для серверных приложений и нереляционная СУБД *Neo4j*, так как графовое представление пользователей в социальной сети является наглядной демонстрацией связей.

- Frontend

Были использованы такие технологии, как *html, CSS, javascript*. Также была использована библиотека *Alchemy.js* для рисования графа, который предоставляет демонстрацию данных пользователю.

Ссылки на Приложение

https://github.com/moevm/nosql-2017-hidden_group_detector/wiki - страница проекта на github.com.

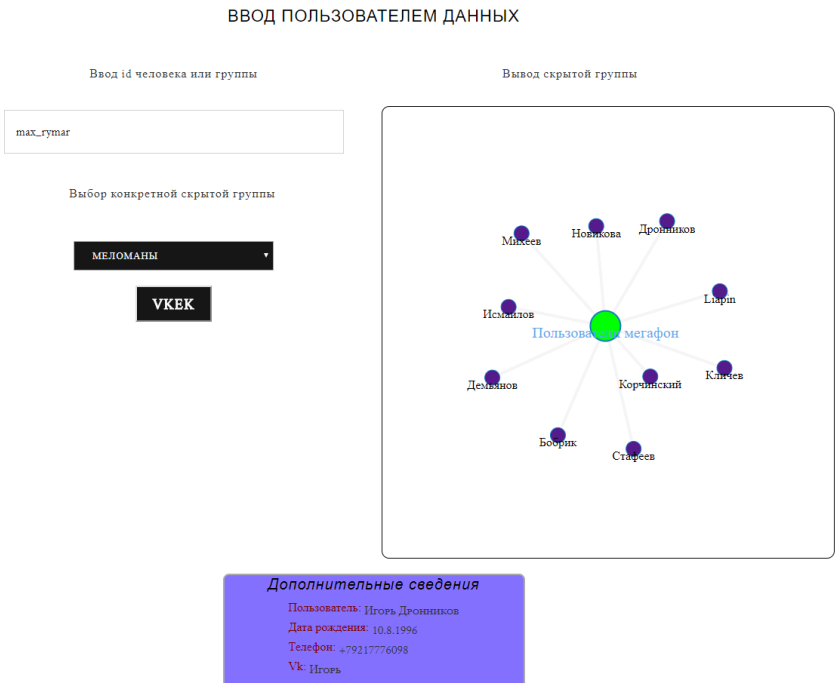
Пример детализации попадания пользователя в скрытую группу

Рассмотрим простой пример попадания человека в одну из реализованных скрытых группу.

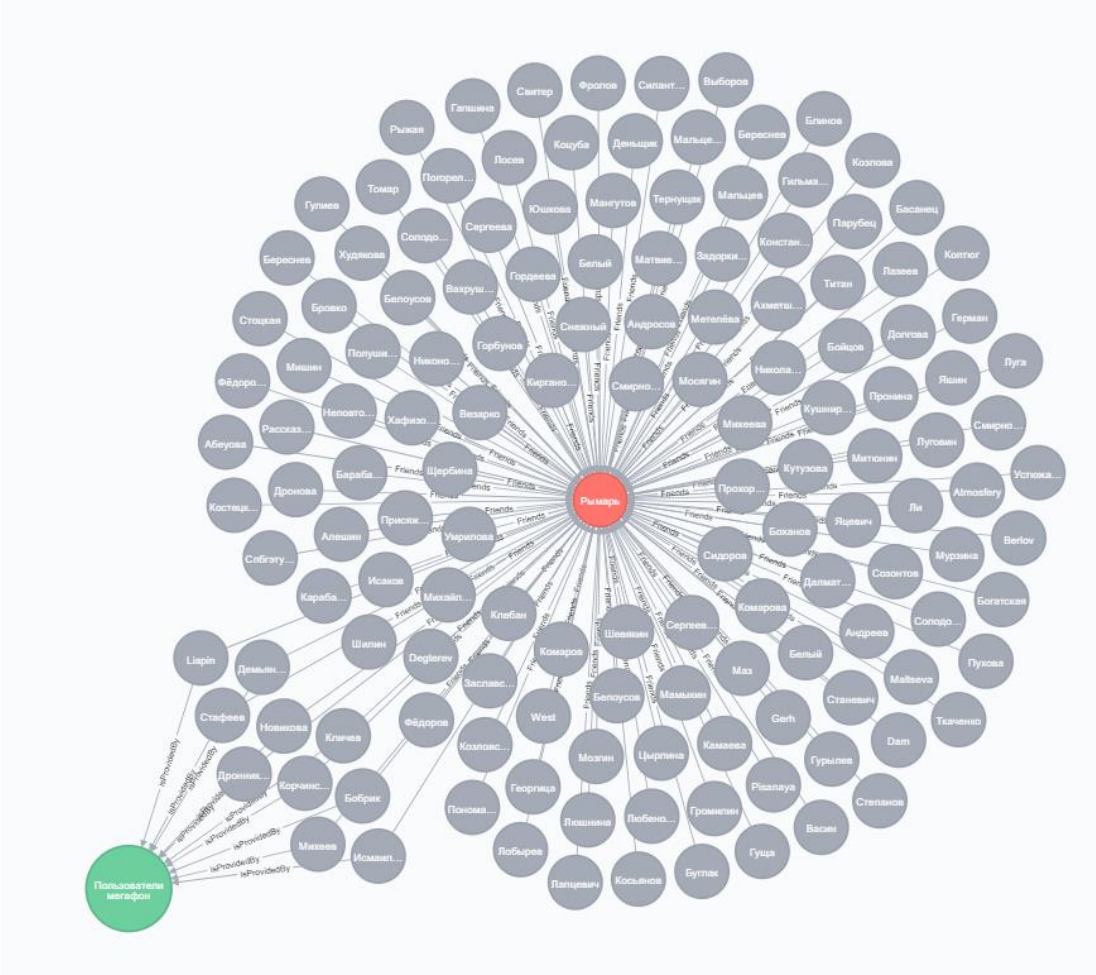
В качестве целевой группы возьмем группу «Пользователи Мегафон». Чтобы определить, пользуется ли человек услугами того или иного оператора, нужно посмотреть первые 3 цифры его номера телефона (не исключая первую 7-ку) и сравнить их с кодами существующих операторов. Для этого оператора «Мегафон» используется регулярное выражение '^.[9](([2][0-9])|([3][0-9])).$'

Следовательно, чтобы найти «Пользователей Мегафон» в указанной группе людей, с помощью запроса к API ВКонтакте нужно:

- Получить персональную информацию о пользователе
- Выделить из нее поле «телефон»
- Записать его в БД
- Далее, для каждого пользователя сравнить его номер телефона с заданным регулярным выражением, и при совпадении – добавить его в указанную группу.



Результат обработки «Пользователей мегафона»



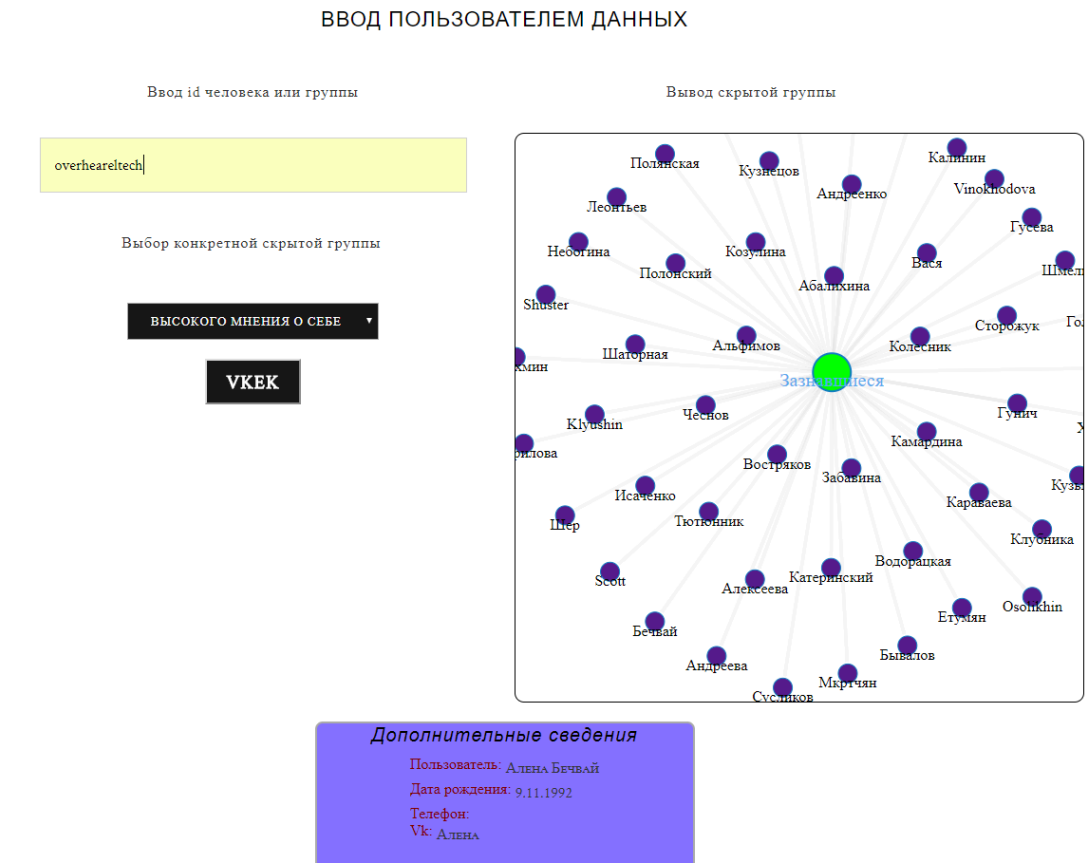
Демонстрация представления данных в БД

Рассмотрим более сложный пример попадания человека в одну из реализованных скрытых группу.

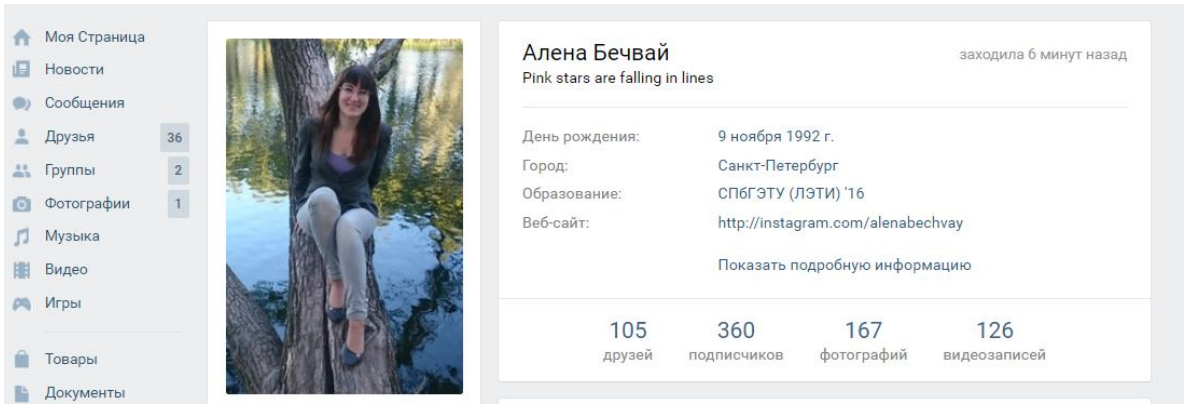
В качестве целевой группы возьмем «Зазнавшихся». В такой группе могут состоять люди, у которых отношение числа друзей к отношению числа подписчиков меньше коэффициента 0.7. Такое соотношение может говорить о том, что человек популярен в обществе, но по каким-то соображением отклоняет получаемые заявки в друзья, оставляя людей в группе своих подписчиков.

Следовательно, чтобы найти «зазнавшихся» в указанной группе людей, с помощью запроса к API ВКонтакте нужно:

- Получить персональную информацию о пользователе
- Выделить из нее поля «кол-во подписчиков» и «кол-во друзей»
- Записать их в БД
- Далее, для каждого пользователя сравнить отношение числа друзей к количеству подписчиков, и если оно будет подходить под выбранный критерий – добавить пользователя в группу «Зазнавшиеся»



Вывод «Зазнавшихся»



Пользователь

При расчете, становится понятно, что алгоритм сработал верно:

105/360 = 0.29, что больше, чем 0.7

ВЫВОДЫ

Достигнутые результаты

На данный момент реализован полноценный программный продукт, который позволяет найти скрытые группы людей в социальной сети Вконтакте. В приложении реализовано взаимодействие с API Вконтакте с помощью авторизации посредством «токена». Полученные данные записываются в графовую базу Neo4j, а затем, на основании алгоритмов поиска скрытых групп, данные выгружаются, проходят процесс обработки и предоставляются для демонстрации пользователю. Таким образом, на основе полученных данных были реализованы алгоритмы поиска различных групп:

- Абоненты Мегафона
- Пользователи iOS, Windows, Android
- Видеофилы
- Меломаны
- Люди с высоким чувством собственной важности

Недостатки и пути для улучшения полученного решения

К сожалению, требуемое количество запросов, отсылаемых на сервер Вконтакте, очень велико, а API Вконтакте дает разрешение с ключом доступа пользователя на обращение к API не чаще 3 запросов в секунду. Так как логика данного приложения подразумевает вызов множества методов один за другим, в дальнейшем практически все запросы будут передаваться в методе *execute:* специальном методе, предоставляемом API, который позволяет совершать до 25 обращений к разным методам в рамках одного запроса.

Будущее развитие решения

По результатам исследования можно сказать, что приложение имеет практически неограниченный потенциал по анализу и выявлению скрытых групп. В будущем в приложение можно добавить функцию поиска людей по общим интересам, то есть таким, который не позволяет обычный фильтр поиска Вконтакте, например, «поиск девушек, которые обожают котиков».

Проект ставит перед собой следующие основные задачи по улучшению разработанного ПО:

- Расширение поиска скрытых групп, благодаря разработке новых алгоритмов
- Аренда хостинга для размещения сайта в сети интернет
- Улучшение как front-end’a, так и back-end’a приложения
- Добавление обратной связи с пользователями для улучшения работы программы

ЗАКЛЮЧЕНИЕ

В итоге проделанной работы по курсу получилось приложение, способное получать данные извне, записывать их, хранить и анализировать на предмет принадлежности к определенной группе (кластеру). Таким образом были закреплены все знания, который были получены в ходе прохождения заданий по предмету «Разработка ПО ИС», а также освоена работы с СУБД Neo4j, получены навыки работы с языком запросов Cypher, базовые навыки работы с языком php,

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Описание API Вконтакте // API Вконтакте. URL: https://vk.com/dev/methods (дата обращения: 10.12.2017).

2. Описание методов Neo4j // Neo4j. URL: https://neo4j.com/docs/operations-manual/3.3/ (дата обращения: 01.12.2017).

3. Описание возможностей php //URL: http://php.net/manual/ (дата обращения: 25.11.2017).