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

КУРСОВАЯ РАБОТА

по дисциплине «Введение в нереляционные СУБД»

Тема: Анализ данных «Trello» – статистика по доскам

Студент гр. 6381 _____ Мейзер Д.В.

Студентка гр. 6381 _____ Степанова С.И.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Мейзер Д.В.

Студентка Степанова С.И.

Группа 6381

Тема работы: Разработка приложения для получения статистики по доскам «Trello»

Исходные данные:

Необходимо реализовать приложение для получения статистики о выбранной доске «Trello» с заданными параметрами используя СУБД MongoDB.

Содержание пояснительной записки:

«Содержание», «Введение», «Качественные требования к решению», «Сценарий использования», «Модель данных», «Разработка приложения», «Вывод», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 13.09.2019

Дата сдачи реферата: 20.12.2019

Дата защиты реферата: 20.12.2019

Студент гр. 6381		Мейзер Д.В.
Студентка гр. 6381		Степанова С.И.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предлагалось в команде из нескольких человек разработать приложение соответствующее одной из предложенных теме. Была выбрана тема анализа данных досок «Trello» и генерация на основе этого статистики с использованием СУБД MongoDB. Также необходимо было предоставить пользователю возможность задавать различные настройки для обработки информации. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h19-trello .

SUMMARY

In this course was proposed to develop an application in a team for one of the proposed topic. We chose the theme of data analysis boards «Trello» and generation based on this statistics using the database MongoDB. It was also necessary to provide the user with the ability to set different settings for information processing. You can find the source code and all additional information here: https://github.com/moevm/nosql2h19-trello .

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время, темп жизни многих людей высок. Для того чтобы всё успевать вовремя необходимо планировать свои действия. Для этого существует множество сервисов и, в частности, «Trello». Одним из способов повышения личной эффективности является сбор и анализ статистики. Для решения данной задачи необходимо создать приложение, которое собирает данные о проделанной работе, и на их основе генерирует статистику. Также необходимо предусмотреть возможность задания настроек для генерации статистики. В ходе данной работы данная проблемы была решена путём разработки веб-приложения, которое собирает данные с заданной доски «Trello», и генерирует на их основе статистику, параметры которой также задаются пользователем.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием системы управления базами данных MongoDB, которое бы генерировало статистику выбранной пользователем доски «Trello» с заданными им же параметрами.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

Макет UI.

Макет UI представлен на рис. 1.

Рисунок 1 – Макет UI

Вид PDF-файла со статистикой представлен на рис. 2.

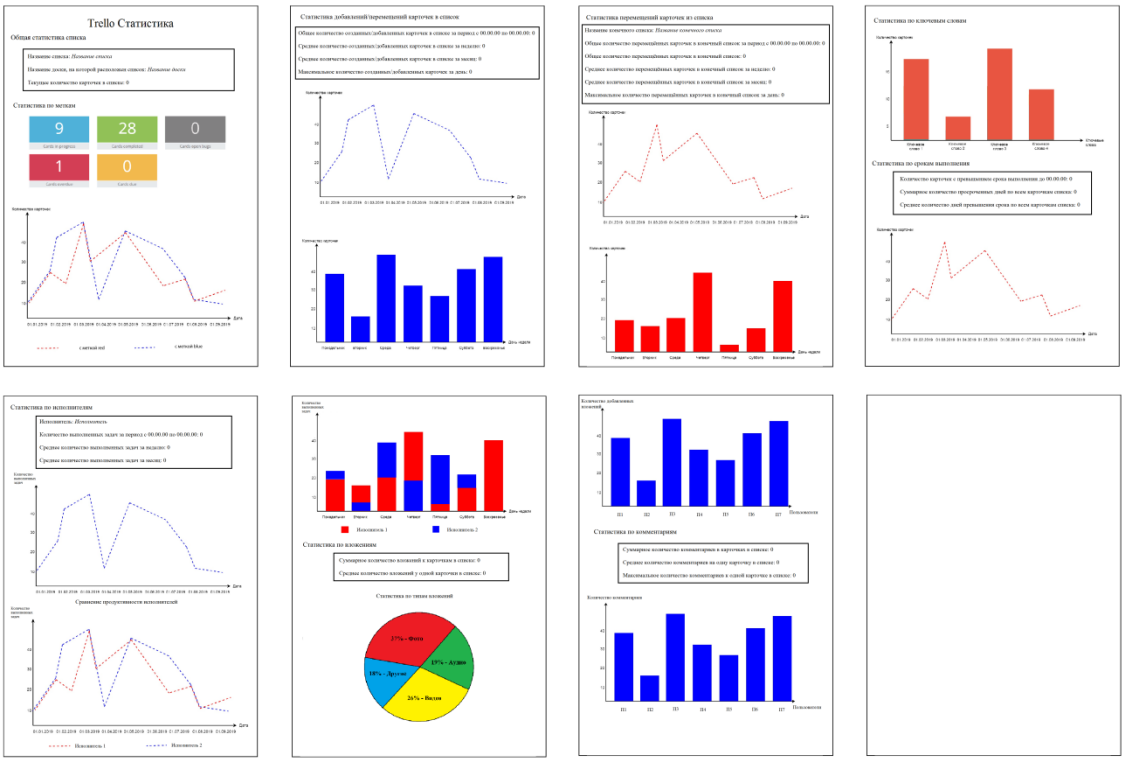

Рисунок 2 – Вид PDF-файла со статистикой

Сценарии использования.

Единственная роль в системе – пользователь. Пользователь может выбирать доску, данные которой будут загружены в базу данных, задавать настройки генерации статистики и импортировать/экспортировать данные базы данных.

Импорт данных.

Для того чтобы загрузить данные доски для анализа, пользователь должен:

1. Нажать на кнопку «Разрешить доступ» и перейти к п. 5.
2. Нажать кнопку «Обзор».
3. Выбрать файл, в котором хранятся данные о доске в формате JSON.
4. Нажать кнопку «Загрузить» и перейти к п. 11.
5. После выполнения п. 1 произойдёт перенаправление на страницу авторизации «Trello». Если пользователь не авторизован – необходимо ввести свой логин и пароль или зарегистрироваться в сервисе.
6. Нажать кнопку «Разрешить» для получения доступа к персональным данным.
7. Нажать кнопку «Далее».
8. В поле формы вставить ссылку на доску, по которой необходимо получить статистику.
9. Нажать кнопку «Далее».
10. Если была дана ссылка на несуществующую доску или у данного пользователя нет доступа к этой доске, произойдёт переход к п. 8, но в форме отобразится ошибка доступа к доске.
11. Пользователь перенаправляется на страницу настроек.

Представление данных.

Для того чтобы задать настройки для получаемой статистики, пользователь должен:

1. Выбрать в поле «Исходный список» один или несколько пунктов из

списка названий списков на заданной доске.

2. В поле «Ключевые слова» ввести ноль или более ключевых слов через пробел.

3. Выбрать в поле «Метки» один или несколько пунктов из списка меток заданной доски.

4. Выбрать в поле «Исполнитель» один или несколько пунктов из списка исполнителей заданной доски.

5. Задать дату с помощью выпадающего календаря в поле «Срок выполнения до» до которой необходимо учитывать просроченные задачи.

6. Задать даты с помощью выпадающего календаря в полях «Период с» и «по» в интервале между которыми должны происходит действия, вошедшие в статистику.

7. Если необходимо получить статистику по вложениям, поставить галочку в поле «Учитывать наличие вложений».

8. Если необходимо получить статистику по комментариям, поставить галочку в поле «Учитывать наличие комментариев».

Анализ данных.

Для того чтобы сгенерировать статистику по полученным данным с заданными настройками, пользователь должен:

1. Нажать кнопку «Получить статистику».

2. Если диапазон дат был введён неверно, форма с настройками отобразится вновь с объявлением об ошибке. Необходим выставить корректный диапазон дат и перейти к п. 1.

3. Пользователь будет перенаправлен на страницу получения сгенерированной статистики.

4. Нажать кнопку «Скачать статистику» для загрузки файла со статистикой на компьютер.

5. Открыть загруженный файл со статистикой.

Экспорт данных.

Для того чтобы экспортировать содержимое базы данных, пользователь должен:

1. Нажать кнопку «Выгрузить БД».
2. Открыть загруженный на компьютер файл, в котором содержимое базы данных представлено в формате JSON.

Дополнительные сценарии использования.

Для данного решения в большей степени будут использоваться операции чтения, так как запись в базу данных осуществляется только один раз за сеанс использования, а чтение происходит многократно для отображения списков с данными доски на странице настроек, а так же в ходе генерации статистики для получения необходимых данных.

3. МОДЕЛЬ ДАННЫХ

Нереляционная модель данных.

Графическое представление модели данных изображено на рис. 3.

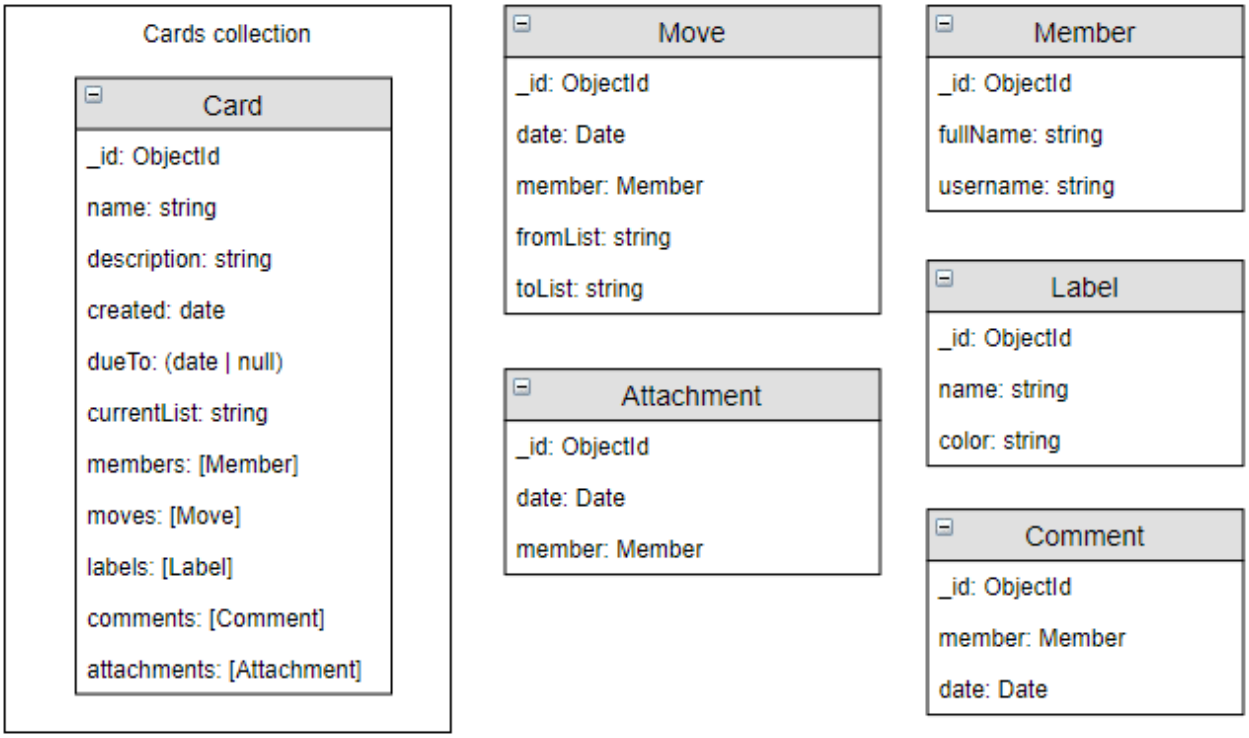

Рисунок 3 – Графическое представление модели данных

Коллекция "Cards" предназначена для хранения карточек. В каждой карточке, помимо сведений о самой карточке, хранятся массивы следующих документов:

- Member – хранит сведения о пользователях, назначенных на данную карточку,
- Move – хранит историю перемещений карточки; внутри себя содержит документ Member,
- Label – метки, прикрепленные к карточке,
- Comment – комментарии, написанные к карточке; содержит внутри себя документ Member,
- Attachment – вложения, прикрепленные к карточке; содержит документ Member.

Пусть на 1 символ строки приходится 4 Б, на дату – 8 Б, на уникальный идентификатор – 12 Б. Оценка размера каждого из типов данных в байтах с исключением вложенных документов:

- Card – 12 + 4C1 + 4C2 + 8 + 8 + 4C3 = 28 + 4(C1 + C2 + C3):
  - _id: ObjectId – 12,
  - name: string – если средняя величина имени карточки равна C1 символам, то 4C1,
  - description: string – если средняя величина описания карточки равна C2, то 4C2,
  - created: Date – 8,
  - dueTo: Date – 8,
  - currentList: string – если средняя длина названия списка равна C3, то 4C3.
- Member – 12 + 56 + 32 = 100:
  - _id: ObjectId – 12,
  - fullName: string – основываясь на том, что средняя длина полного имени - 14 символов, 14 * 4 = 56,
  - username: string – если длина пользовательского имени в среднем равна 8 символам, то 8 * 4 = 32.
- Move – 12 + 8 + 4C3 + 4C3 = 20 + 8C3:
  - _id: ObjectId – 12,
  - date: Date – 8,
  - fromList: string – 4C3,
  - toList: string – 4C3.
- Label – 12 + 24 + 20 = 56:
  - _id: ObjectId – 12,
  - name: string – пусть средняя длина имени метки равна 6 - 6 * 4 = 24,
  - color: string – пусть средняя длина имени цвета равна 5 - 5 * 4 = 20.
- Comment – 12 + 8 = 20:
  - _id: ObjectId – 12,
  - date: Date – 8.
- Attachment – 12 + 8 = 20:
  - _id: ObjectId – 12,
  - date: Date – 8.

Есть предположить, что C1 = 30, C2 = 300, C3 = 20, то расчетная формула "чистого" объема данных будет равна:

1428N + 100membersN + 180movesN + 56labelsN + 20(commentsN + attachmentsN) = N(1428 + 100membersAVG + 180movesAVG + 56labelsAVG + 20(commAVG + attachAVG)), где N - количествокарточек.

Приняв membersAVG = 5, movesAVG = 3, labelsAVG = 2, commAVG = 6.5, attachAVG = 0.5, былполучен"чистый"размерБДвзависимостиотколичествакарточек N: pureSize = N(1428 + 500 + 540 + 112 + 140) = 2720N

Был оценен размер каждого документа с учетом других вложенных документов:

- Member – 100,
- Move – 20 + 8C3 + 100 = 120 + 8C3 = 280:
  - member: Member – 100.
- Label – 56,
- Comment – 20 + 100 = 120:
  - member: Member – 100.
- Attachment – 20 + 100 = 120:
  - member: Member – 100.
- Card – 1428 + 100membersAVG + 280movesAVG + 56labelsAVG + 120(commAVG + attachAVG):
  - members: [Member] – если на одну карточку приходится в среднем membersAVG участников, то 100membersAVG,
  - moves: [Move] – если на одну карточку в среднем приходится movesAVG перемещений, то (120 + 8C3)movesAVG,
  - labels: [Label] – если на одну карточку приходится в среднем labelsAVG меток, то 56labelsAVG,
  - comments: [Comment] – всреднем commAVG комментариев - 120commAVG,
  - attachments: [Attachment] – всреднем attachAVG вложений - 120attachAVG.

Такимобразом, фактическийразмерБДможнооценитькак N(1428 + 100membersAVG + (120 + 8C3)movesAVG + 56labelsAVG + 120(commAVG + attachAVG)), где N - числокарточек.

Используя средние значения сущностей, приходящихся на каждую карточку, использованные при расчете "чистого" объема данных, было получено значение размера фактического объема БД в зависимости от количества карточек:

size = N(1428 + 500 + 840 + 112 + 840) = 3720N

Запросы к модели, с помощью которых реализуются сценарии использования:

- Число карточек, помещенных в список List за период с Date1 по Date2:

```
cardsNumber = db.cardsCollection.count_documents({
'moves.toList': List,
'moves.date': {'$gte': Date1, '$lte': Date2 }
});
```

- Число комментариев, написанных пользователем с именем Member за период с Date1 по Date2:

```
commentsNumber = collection.aggregate([
{ '$project': {
"count": { "$size": { # We put in field 'count' size of
'$filter': { # Filtered 'comments' array
'input': '$comments',
# Elements of which matches this conditions:
'cond': {'$and': [
{'$eq': ['$$this.member.fullName', Member]},
{'$gte': ['$$this.date', Date1]},
{'$lt': ['$$this.date', Date2]},
]}
}
}}
}},
{ '$group': { # Then group all documents
'_id': None, # In one
# 'comments': {'$push': '$count'},
'number': {'$sum': '$count'}, # And sum of counts in each document is a new
field 'number'
}},
])
commentsNumber = commentsNumber.next()['number'];
```

Аналогмоделиданныхдля SQL СУБД.

Графическое представление для реляционной модели данных показано на рис. 4.

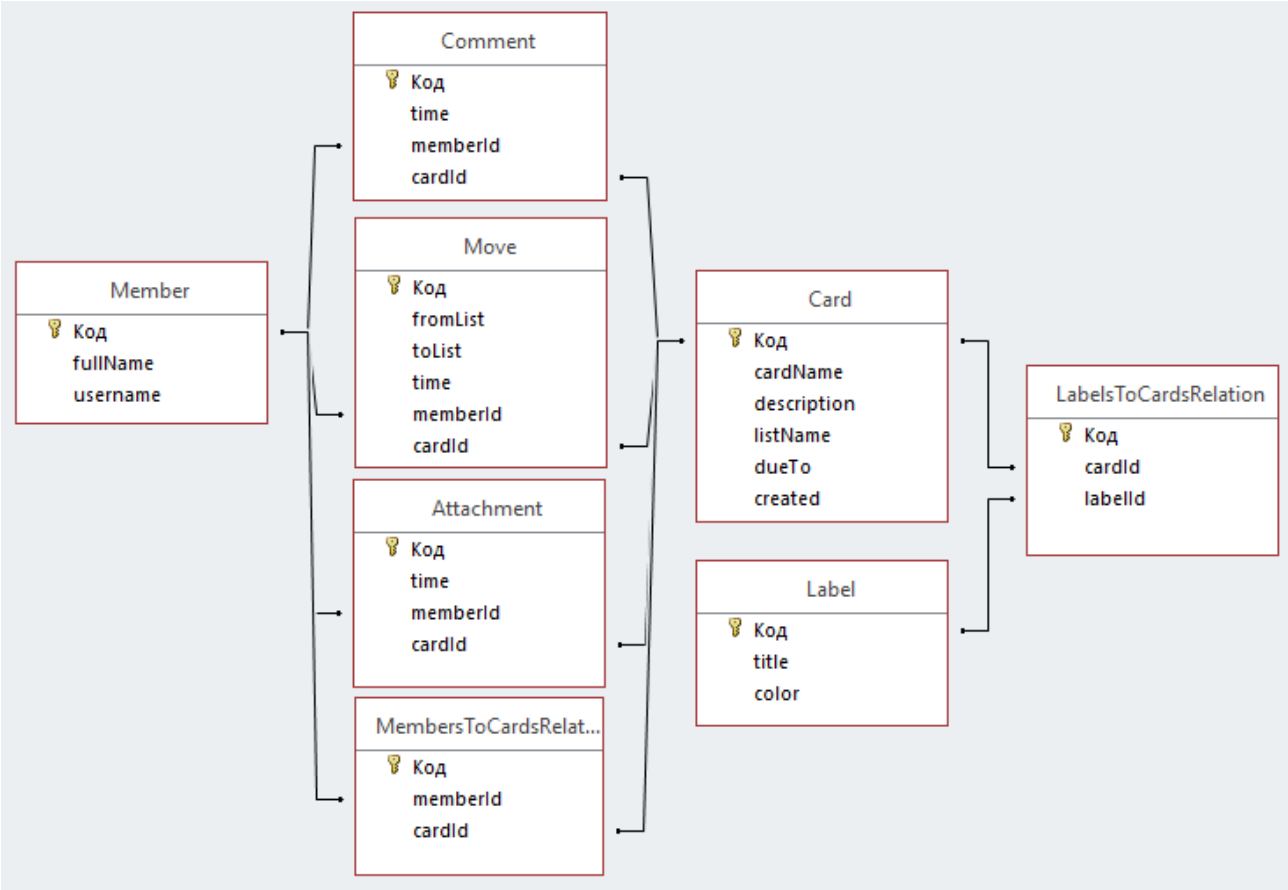

Рисунок 4 – Графическое представление реляционной модели

Таблицы Move, Card, Member, Label, Attachment, Comment хранят данные о сущностях базы данных.

Поле memberId в таблицах Move, Attachment, Comment реализует связь один-ко-многим с таблицей Card.

Таблицы MembersToCardsRelation и LabelsToCardsRelation реализуют связь многие-ко-многим таблицы Card с таблицами Members и Labels соответственно.

Был оценен средний размер одной записи для каждой из таблиц с учетом и без учета затрачиваемой памяти на связи между сущностями:

- Member -- Код(4Б) + fullName(4Б*14) + userName(4Б*8) = 92Б (88Б)
- Comment -- Код(4Б) + time(8Б) + memberId(4Б) + cardId(4Б) = 20Б (8Б)
- Move -- Код(4Б) + fromList(4Б*20) + toList(4Б*20) + time(8Б) + memberId(4Б) + cardId(4Б) = 180Б (168Б)
- Attachment -- Код(4Б) + time(8Б) + memberId(4Б) + cardId(4Б) = 20Б (8Б)
- Card -- Код(4Б) + cardName(4Б*30) + description(4Б*300) + listName(4Б*20) + dueTo(8Б) + created(8Б) = 1420Б (1416Б)
- Label -- Код(4Б) + title(4Б*6) + color(4Б*5) = 48Б (44Б)

- LabelsToCardsRelation -- 12Б (0Б)
- MembersToCardsRelation -- 12Б (0Б)

Пусть карточек - N, на каждую карточку приходится в среднем commAVG=6.5 комментариев, movesAVG=3 перемещений, attachAVG=0.5 вложений, labelsAVG=2 меток и membersAVG=5 пользователей, тогда "чистый" объем:

pureSize = 1416N + 6.5N*8 + 3N*168 + 0.5N*8 + 2N*44 + 5N*88 = 2504N

Фактический объем - чистый объем + связи между сущностями:

size = 2504N + 6.5N*12 + 3N*12 + 0.5N*12 + 2N*4 + 5N*4 + 2N*12 + 5N*12 = 2736N

Запросы к модели, содержащиеся в сценарии использования приложения:

- Число карточек, помещенных в список List за период с Date1 по Date2:

```
SELECTCount(Card.cardName) AS [Cards count], Move.Time, Move.Time,
Move.toList
FROM Card INNER JOIN Move ON Card.Код=Move.cardId
GROUP BYMove.toList
HAVING ((Move.toList=[List]) AND (Move.Time>=[Date1] AndMove.Time<[Date2]));
```

- Число комментариев, написанных пользователем с именем Name за период с Date1 по Date2:

```
SELECTCount(Comment.Код) AS [Количество], Member.fullName
FROM Member INNER JOIN Comment ON Member.Код=Comment.memberId
WHEREComment.Time<[Date2] AndComment.Time>=[Date1]
GROUP BYMember.fullName
HAVINGMember.fullName=[Name];
```

Сравнениемоделей.

SQL при прочих равных требует меньше памяти - в среднем 2736N у SQL против 3720N у Mongo, где N - количество карточек в базе данных.

Для каждого из приведенных примеров потребовалось по 1 запросу как в SQL, так и в Mongo. В каждом из запросов у SQL задействовано 2 таблицы, так как данные хранятся разрозненно, а в Mongo - 1 коллекция, так как данные хранятся не в нормальной форме и вложены друг в друга.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

Краткое описание.

Разработанное веб-приложение позволяет загрузить данные выбранной пользователем доски сервиса «Trello» с помощью API или с помощью файла с данными в формате JSON, задать списки на доске, ключевые слова, метки карточек, исполнителей задач, предельный срок выполнения задач и период, за который необходимо получить статистику. Данные доски после передачи загружаются в базу данных, а при генерации статистики получаются из базы данных посредством запросов. Полученная информация размещается в PDF-файл и отдаётся для скачивания пользователю.

Схема экранов приложения.

Экраны приложения и переходы между ними отображены на рис. 5.

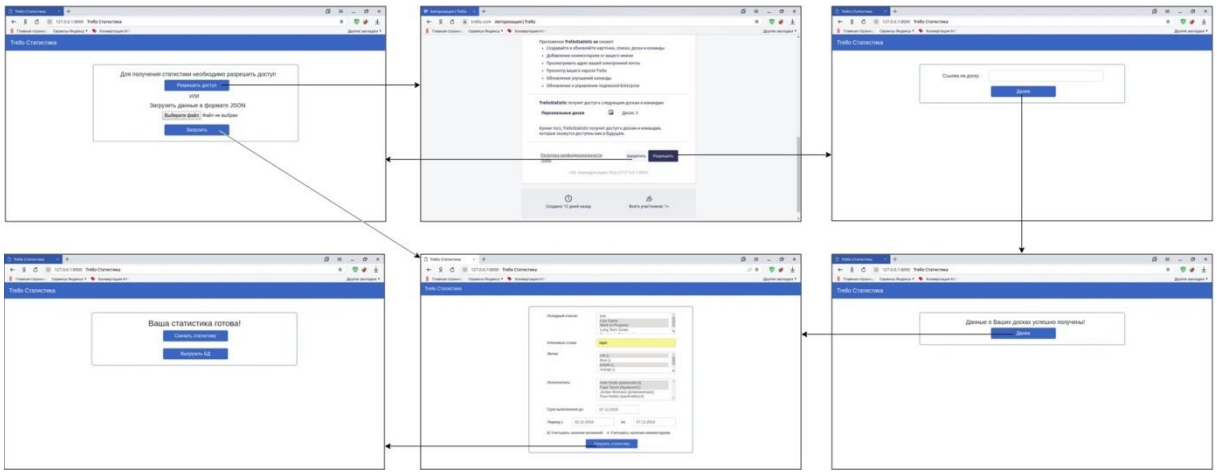

Рисунок 5 – Схема экранов приложения

Использованные технологии.

БД – MongoDB; back-end – Python 3.7 (библиотеки django, pymongo, json, matplotlib, numpy, fpdf и другие);front-end – HTML5, CSS, JavaScript (Bootstrap 4).

Ссылки на Приложение.

Ссылка на исходный код приложения на GitHub – https://github.com/moevm/nosql2h19-trello .

5. ВЫВОДЫ

Достигнутые результаты.

В ходе выполнения задания было разработано приложение, которое генерирует статистику на основе выбранной пользователем доски сервиса «Trello» с помощью API или с помощью файла с данными в формате JSON, размещается в PDF-файл и отдаётся для скачивания пользователю. Также у пользователя есть возможность задавать списки на доске, ключевые слова, метки карточек, исполнителей задач, предельный срок выполнения задач и период, по которому необходимо получить статистику.

Недостатки и пути для улучшения полученного решения.

На данный момент загрузка больших досок при использовании ссылки происходит медленно, так как для каждой карточки посылается запрос на сервер Trello.

Возможные пути решения данной проблемы:

- Выполнение запроса для нескольких карточек одновременно.
- Импорт доски из экспортированного ранее файла.

Будущее развитие решения.

В дальнейшем планируется разработка, и включение разработанного приложения в раздел «Улучшения» сервиса «Trello».

6. ПРИЛОЖЕНИЯ

Документация по сборке и развертыванию приложения.

1. Скачать проект из репозитория с помощью команды «git clone https://github.com/moevm/nosql2h19-trello».

2. Перейти в виртуальное окружение с использованием команды «source venv/bin/activate».

3. Перейти в папку nosql с помощью команды «cd ./nosql».

4. Запустить файл run.py командой «python3 run.py».

5. Открыть приложение в браузере по адресу «http://127.0.0.1:8000».

Инструкция для пользователя.

Подробная инструкция пользователя описана в разделе «Сценарии использования».

Снимки экрана приложения.

Вид стартового экрана приложения представлен на рис. 6.

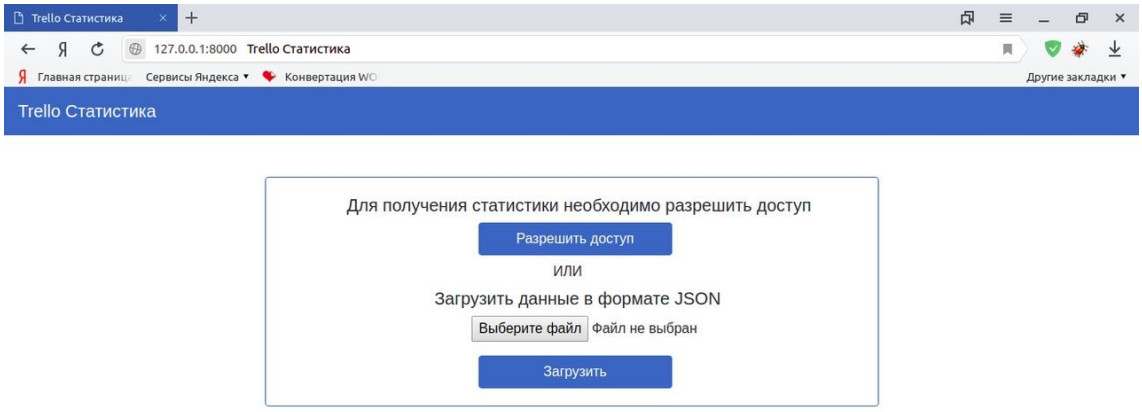

Рисунок 7 – Стартовая страница

После нажатия кнопки «Разрешить доступ» пользователь перенаправляется на страницу доступа к персональным данным «Trello» представлен на рис. 7.

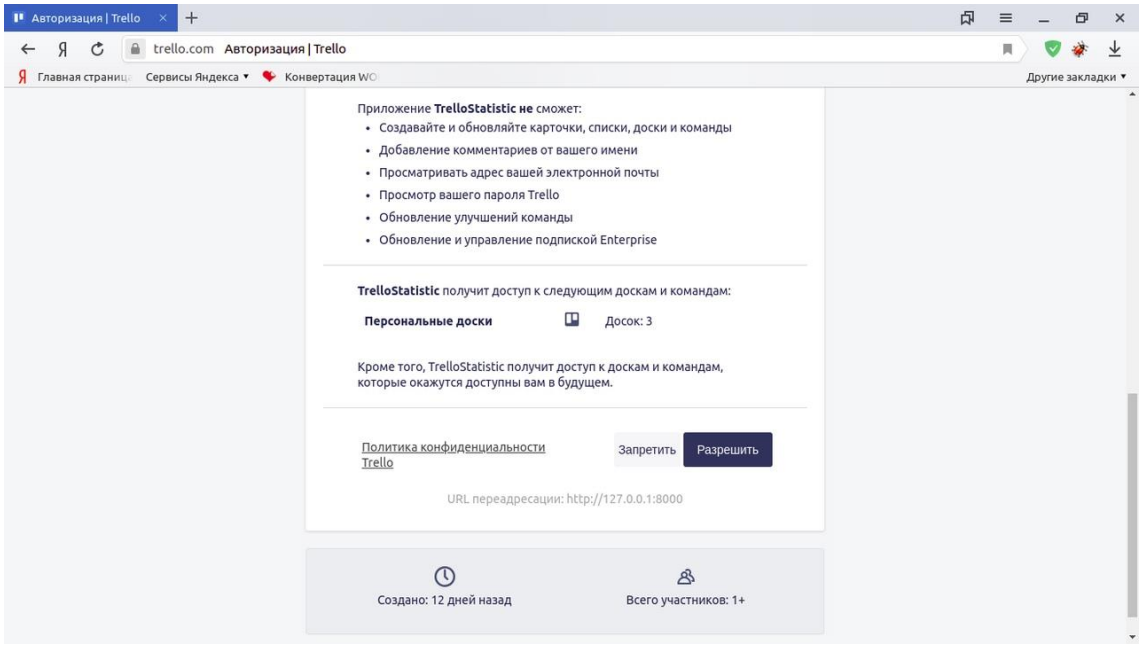

Рисунок 7 – Авторизация в «Trello»

После авторизации, пользователь перенаправляется на страницу, представленную на рис. 8.

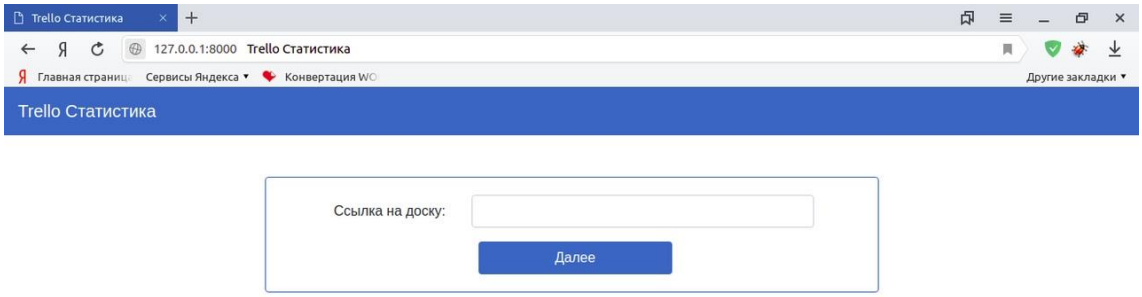

Рисунок 8 – Страница с полем для ссылки на доску

После нажатия на кнопку «Далее» в случае успешной загрузки данных, пользователь перенаправляется на страницу, представленную на рис. 9.

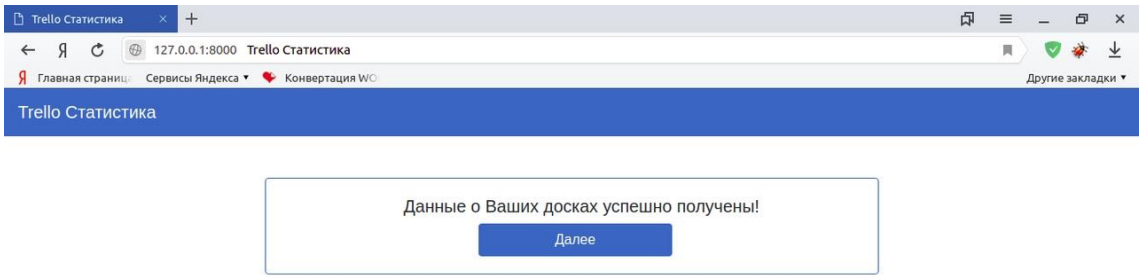

Рисунок 9 – Страница успешной загрузки данных

После нажатия на кнопку «Далее» на странице успешной загрузки или в случае загрузки данных из файла, пользователь перенаправляется на страницу задания настроек статистики (см. рис. 10).

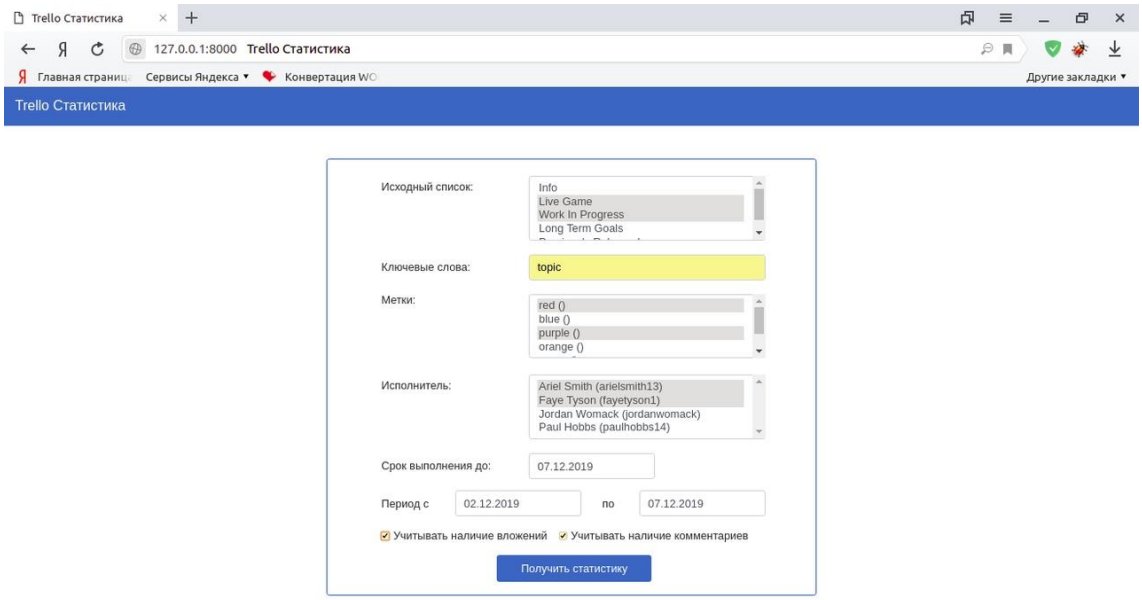

Рисунок 10 – Страница задания настроек статистики

После нажатия кнопки «Получить статистику» пользователь

перенаправляется на страницу представленную на рис. 11. На данной странице пользователь может скачать сгенерированную статистику в виде PDF-файла и/или экспортировать содержимое базы данных.

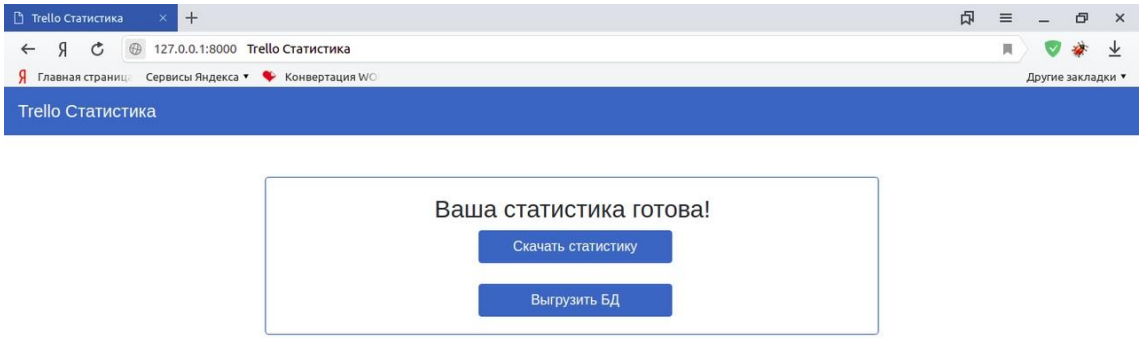

Рисунок 11 – Страница экспорта статистики и/или данных из базы данных

ЗАКЛЮЧЕНИЕ

В ходе выполнения задания было разработано приложение, которое генерирует статистику на основе выбранной пользователем доски сервиса «Trello» с помощью API или с помощью файла с данными в формате JSON, размещается в PDF-файл и отдаётся для скачивания пользователю. Также у пользователя есть возможность задавать списки на доске, ключевые слова, метки карточек, исполнителей задач, предельный срок выполнения задач и период, по которому необходимо получить статистику.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB. URL: https://www.mongodb.com/ .
2. Документация Python. URL: https://www.python.org/ .
3. Документация Django. URL: https://www.djangoproject.com/ .
4. Документация Bootstrap. URL: https://getbootstrap.com/ .
5. Документация Matplotlib. URL: https://matplotlib.org/ .