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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Разработка программного обеспечения информационных систем»

Тема: Хранение, обработка и обновление статистики репозиториев Github

Студент гр. 5382 _____ Бачинский М.О.

Студент гр. 5382 _____ Борисовский Д.Ю.

Студентка гр. 5382 _____ Мелихова П.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2018

ЗАДАНИЕ

НА КУРСОВОЙ ПРОЕКТ

Студенты Бачинский М.О.,Борисовский Д.Ю., Мелихова П.А.

Группа 5382

Тема проекта: Хранение, обработка и обновление статистики репозиториев Github

Исходные данные:

Проект должен быть разработан с использованием базы данных MongoDB

Содержание пояснительной записки:

Содержание, Введение, Качественные требования к решению, Сценарии использования, Модель данных, Разработанное приложение, Заключение, Список использованных источников, Приложение А “Документация по сборке и развертыванию приложения”, Приложение В “Инструкция для пользователя”, Приложение С “Снимки экрана приложения”.

Предполагаемый объем пояснительной записки:

Не менее 34 страниц.

Дата выдачи задания: 13.09.2018

Дата сдачи реферата: 20.12.2018

Дата защиты реферата: 20.12.2018

Студент	_____	Бачинский М.О.
Студент	_____	Борисовский Д.Ю.
Студентка	_____	Мелихова П.А.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В курсовом проекте реализовано веб-приложение на основе БД MongoDB, с помощью которого можно отслеживать активность в репозиториях сервиса GitHub.

SUMMARY

In the course project, a web application based on MongoDB DB is implemented, with which you can track activity in the GitHub service repositories.

СОДЕРЖАНИЕ

Введение

В настоящее время студентов IT-специальностей часто объединяют по группам для выполнения курсовых проектов, с целью обучения работе в команде. Для удобства синхронизации выполненных частей проекта зачастую пользуются такими сервисами, как GitHub. Однако преподавателю сложно оценивать отдельно взятого студента из группы, поскольку не совсем ясно какую часть проекта он сделал на самом деле.
Целью данного проекта является создание веб-приложения на основе БД MongoDB, предназначенное для отслеживания прогресса выполнения конкретного проекта в выбранном курсе, а также для упрощения оценивания отдельно взятого студента, а не команды в целом.
В ходе данной работы было разработано веб-приложение на основе БД MongoDB.

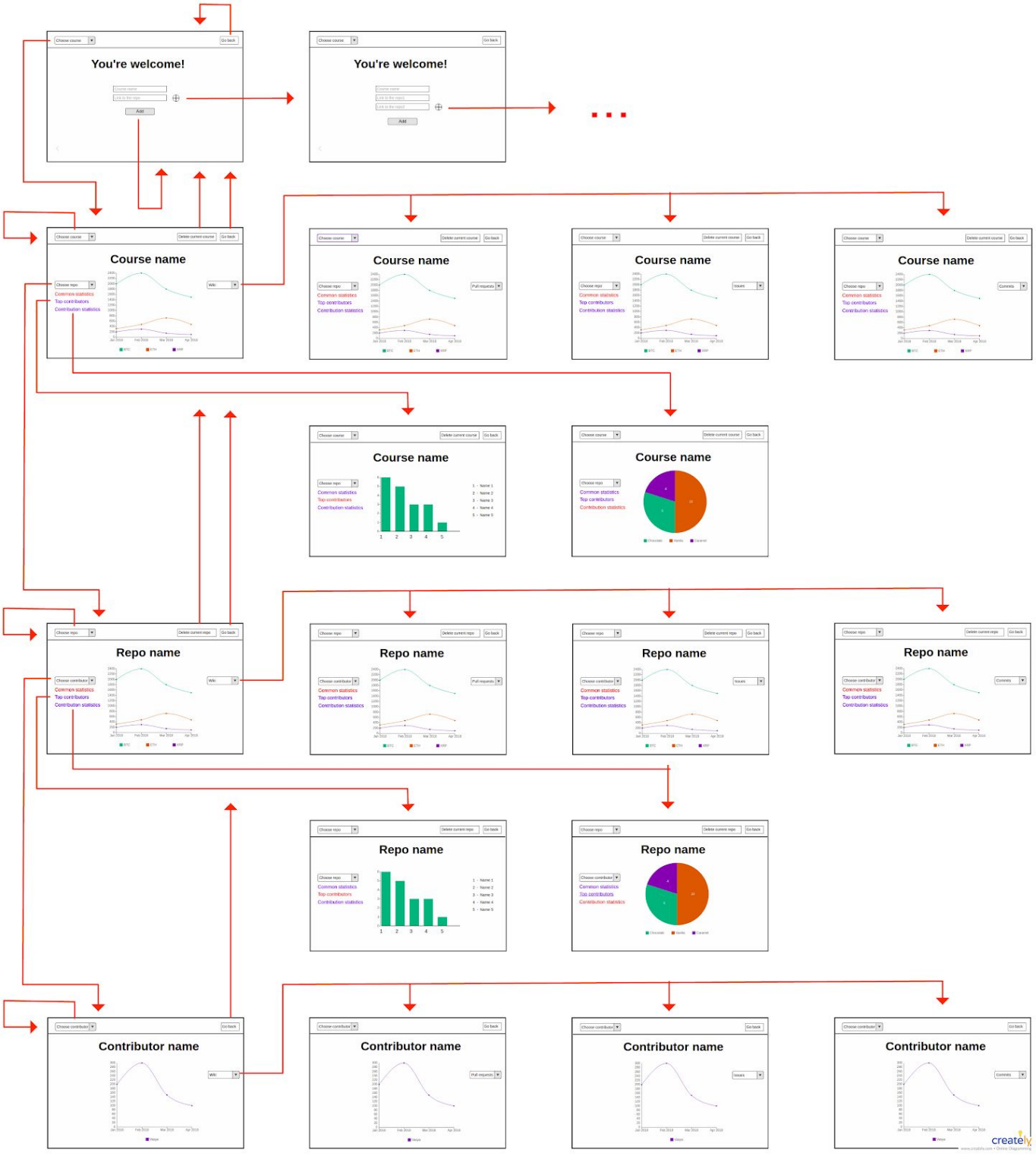
Качественные требования к решению

- Приложение должно графически выводить статистику работы в github
- Приложение должно наглядно сравнивать репозитории и контрибьюторов между собой для постановления вывода о том, кто плохо работал в команде или в курсе
- Приложение должно уметь записывать предыдущие запросы, чтобы не делать повторные запросы к Git API и экономить потребление трафика
- Приложение должно работать с документо-ориентированной базой данных
- Приложение должно уметь работать с приватными репозиториями в github
- Приложение должно уметь делать импорт/экспорт всех данных в базе данных
- Приложение должно уметь запоминать коллекцию репозиториев (далее “курсы”), чтобы пользователь не вводил данные повторно

Сценарии использования

Макет UI

Ссылка для просмотра в увеличенном виде - https://i.ibb.co/wKymsny/Mockup-and-UC.png



Сценарии использования

Use case №1: «Загрузка нового репозитория в БД»

Основной сценарий:

1. Пользователь находится на начальной странице;
2. Пользователь вводит в поле название курса;
3. Пользователь вводит ссылку на репозиторий;
4. Пользователь нажимает кнопку «add»;
5. Сервис записывает данные из репозитория в БД в существующий курс; * Или создает новый курс в БД, куда записывает данные из репозитория;
6. Переход на шаг 1.

Альтернативный сценарий:

1. Вместо шага 4 пользователь нажимает «+»;
2. Пользователь вводит ссылку на следующий репозиторий;
3. Пользователь нажимает кнопку «add»;
    - Или переход на шаг 1 альтернативного сценария.
4. Переход на шаг 1 основного сценария.

Use case №2: «Выбор курса»

Основной сценарий:

1. Пользователь находится на начальной странице;
2. Пользователь кликает по «Choose course» и выбирает из списка нужный курс;
3. Сервис отображает страницу с открытой «Common statistics» по умолчанию по коммитам;
4. Пользователь имеет возможность:
    - Ознакомиться со статистиками по курсу;
    - Удалить курс, нажав кнопку «delete course»;
    - Вернуться назад, нажав кнопку «back».

Альтернативный сценарий:

1. Пользователь удалил весь курс, нажав кнопку «delete course»;
2. Сервис отображает начальную страницу.

Если в БД не записано ни одного курса, пользователь имеет возможность создать курс, записав в него репозиторий.

Use case №3: «Выбор репозитория»

Основной сценарий:

1. Пользователь находится на странице курса;
2. Пользователь кликает по «Choose repo» и выбирает из списка нужный репозиторий;
3. Сервис отображает страницу с открытой «Common statistics» по умолчанию по коммитам;
4. Пользователь имеет возможность:
    - Ознакомиться со статистиками по репозиторию;
    - Удалить репозиторий, нажав кнопку «delete repo»;
    - Вернуться на страницу курса, нажав кнопку «back».

Альтернативный сценарий:

1. Пользователь удалил репозиторий из БД, нажав кнопку «delete repo»;
2. Сервис отображает страницу курса, из которого был удален репозиторий.

Use case №4: «Выбор контрибьютора»

Основной сценарий:

1. Пользователь находится на странице репозитория;
2. Пользователь кликает по «Choose contributor» и выбирает из списка нужного котрибьютора;
3. Сервис отображает страницу с открытой «Common statistics» по умолчанию по коммитам;
4. Пользователь имеет возможность:
    - Ознакомиться со статистиками по контрибьютору;
    - Вернуться назад, нажав кнопку «back».

Use case №5: «Просмотр статистик по курсу»

Основной сценарий:

1. Пользователь находится на странице курса с открытой «Common statistics» по умолчанию по коммитам;
2. Пользователь имеет возможность:

- Выбрать другой параметр для «Common statistics» из перечисленного ниже списка:
    - Commits
    - Pull requests
    - Issues
    - Wiki
- Выбрать другую статистику из перечисленных ниже:
    - Top contributors;
    - Contribution statistics
- Сменить курс, нажав на выпадающий список «Choose course»;
- Перейти к статистикам по репозиториям;
- Удалить весь курс из базы данных;
- Вернутся назад по кнопке «back»;

1. При выборе последнего, пользователь переходит на начальную страницу.

Альтернативный сценарий:

1. Пользователь удалил курс из базы данных;
2. Сервис отображает начальную страницу.

Use case №6: «Просмотр статистик по репозиторию»

Основной сценарий:

1. Пользователь находится на странице репозитория с открытой «Common statistics» по умолчанию по коммитам;
2. Пользователь имеет возможность:

- Выбрать другой параметр для «Common statistics» из перечисленного ниже списка:
    - Commits
    - Pull requests
    - Issues
    - Wiki
- Выбрать другую статистику из перечисленных ниже:
    - Top contributors;
    - Contribution statistics
- Сменить репозиторий, нажав на выпадающий список «Choose repo»;
- Перейти к статистикам по контрибьюторам;
- Удалить репозиторий из базы данных;
- Вернутся назад по кнопке «back»;

1. При выборе последнего пользователь переходит на страницу курса, в котором находится данный репозиторий.

Альтернативный сценарий:

1. Пользователь удалил репозиторий из базы данных;
2. Сервис отображает страницу курса, из которого был удален репозиторий.

Use case №7: «Просмотр статистик по контрибьютору»

Основной сценарий:

1. Пользователь находится на странице контрибьютора с открытой «Common statistics» по умолчанию по коммитам;
2. Пользователь имеет возможность:

- Выбрать другой параметр для «Common statistics» из перечисленного ниже списка:
    - Commits
    - Pull requests
    - Issues
    - Wiki
- Сменить контрибьютора, нажав на выпадающий список «Choose countributor»;
- Перейти к статистикам по репозиториям;

- Вернутся назад по кнопке «back»;

Альтернативный сценарий:

1. Пользователь нажал кнопку «back»;
2. Сервис отображает страницу репозитория, в котором был выбран данный контрибьютор.

Вывод о том, какие операции (чтение или запись) будут преобладать для вашего решения.

В решении будут преобладать операции чтения, потому как операция записи используется только при обновлении данных курса, то есть единожды при открытии статистики курса, в то время как операции чтения будут использоваться при открытии каждого нового графика (для курса/репозитория/контрибьютора, для сущности коммита/пулл реквеста/исьюс).

Модель данных

Нереляционная модель данных

- courses - коллекция для хранения курсов
    - _id: ObjectId - id курса
    - lastUpdate : Date - дата последнего обновление БД
    - name: String - имя курса
    - repositories - список репозиториев
        - _id: ObjectId - id репозитория
        - name: String - имя репозитория
        - owner: String - владелец репозитория
        - contributors - список контрибьюторов
            - name: String - логин контрибьютора [использовать как id]
            - commits - список коммитов
                - date: Date - дата коммита [использовать как id]
            - issues - список исьюс
                - date: Date - дата исью [использовать как id]
            - pullRequests - список пр
                - date: Date - дата пр [использовать как id]

Размеры полей NoSQL

- courses - коллекция для хранения курсов, где id задается автоматически базой данных
    - _id: ObjectId [12 bytes]
    - lastUpdate : Date [8 bytes]
    - name: String [пусть 18 символов - 18 bytes]
    - repositories [в среднем N]
        - _id: ObjectId [12 bytes]

- name: String [пусть 18 символов - 18 bytes]
- owner: String [пусть 18 символов - 18 bytes]
- contributors [в среднем K]
    - name: String [пусть 18 символов - 18 bytes]
    - commits [в среднем P]
        - date: Date [8 bytes]
    - issues - список исьюс [в среднем P]
        - date: Date [8 bytes]
    - pullRequests - список пр [в среднем P]
        - date: Date [8 bytes]

В среднем одна запись в коллекции занимает (((P * 24) + 18) * K + 48) * N+ 38

Запросы NoSQL

Имя запроса	Запрос	Количество запросов

Сохранение нового курса в бд
 	
Сохраняем новый курс, в котором пользователь указал
1) Имя “N_C” (поле name)
2) Для каждого репозитория:
a) Название “N_R” (поле name)
b) Владелец “O” (поле owner)
Поля “_id”, “repositories._id” и “lastUpdate” генерируются автоматически
	Добавляем новый пустой документ в базу данных
	База возвращает id, который мы возвращаем клиенту
	
	db.courses.insert({
	“_id”: new ObjectdId(),
	“lastUpdate”: “D”,
	“name”: “N_C”,
	“repositories”: [
	{
	“_id”: new ObjectId(),
	“name”: “N_R”,
	“owner”: “O”,
	“contributors”: []
	},
	…
	]
	})
		1

Запрос всех коммитов (Аналогично, для исьюс, пр) у курса с id : N	db.courses.find({ “_id”: ObjectId(N) })	1

Обновление данных - добавление новых коммитов (Аналогично, для исьюс, пр)
 	
Запрос для всех обновленных репозиториев
от 1 до repoCount
для всех обновленных контрибьюторов в репозитории
от 1 до contributorCount
 	
В запросе указываются дата коммита “D”,
id текущего репозитория “currentRepoId”,
name текущего контрибьютора “currentContributorName”
 	
+
 	
Запрос для обновления даты последнего обновления “lastUpdate” с датой “N_D” у курса с id “N”
	db.course.update(
	{},
	{
	$addToSet: {
	
	“repositories.$[i].contributors.$[j].commits”: {
	$each: {
	{
	“_id”: “D”
	}
	}
	}
	}
	},
	{
	upsert: true,
	arrayFilters: [
	{
	“i._id”: ObjectId(currentRepoId)
	},
	{
	“j._id”: “currentContributorName”
	}
	]
	}
	)
	
	db.course.update(
	{
	“_id”: ObjectId(N)
	},
	{
	“lastUpdate”: “N_D”
	}
	)
		Количество_обновленных_репозиториев
		*
		Количество_обновленных_контрибьюторов_в_каждом_репозитории
		+
		1 запрос для обновления даты последнего обновления

Запрос коммитов одного репозитория с id N из курса с id K (Аналогично, для исьюс, пр)	//Немного подправил с ObjectId
	db.courses.find({
	“_id”: ObjectId(N),
	“repositories._id”: ObjectId(K)
	})
		1

Запрос коммитов одного контрибьютора с name N из репозитория с id R из курса с id K (Аналогично, для исьюс, пр)	db.courses.find({ “_id”: ObjectId(N), “repositories._id”: ObjectId(K) “repositories.contributors._id”: “N” })	1

Добавление нового репозитория с названием “N_R”, и владельцем “O” в курс с id “N”
 	
Данные по контрибьюторам нового репозитория (name, commits, issues, pullReuests) запрашиваются через github API
	db.course.update(
	{
	“_id”: ObjectId(N)
	},
	{
	$addToSet: {
	“repositories: {
	“_id”: new ObjectId(),
	“name”: “N_R”,
	“owner”: “O”,
	“contributors”: [
	{
	“name”: “contributorName”
	“commits”: [...]
	“issues”: [...]
	“pullRequests”: [...]
	},
	….
	]
	}
	}
	},
	{
	upsert: true,
	}
	)
		1

Удаление курса с id N	db.course.remove(
	{
	“_id”: ObjectId(N)
	}
	)
		1

Удаление репозитория с id N в курсе с id K	db.course.update({ “_id”: ObjectId(N)	1

	},
	{
	$pull: {
	“repositories”: {
	“_id”: ObjectId(K),
	}
	}
	}
	)	

Реляционная модель данных

Таблица items (commits, prs, issues)			
Имя	Тип	Размер, в байтах	Примечание
{items}_id	длинное целое	4	ключ
contributor_id	длинное целое	4	связь с таблицей contributor
дата	date	10	

Таблица contributor (контрибьютор)			
Имя	Тип	Размер, в байтах	Примечание
contributor_id	длинное целое	4	ключ
rep_id	длинное целое	4	связь с таблицей reps
name	varchar(20)	20	

Таблица reps (репозиторий)			
Имя	Тип	Размер, в байтах	Примечание
rep_id	длинное целое	4	ключ
course_id	длинное целое	4	связь с таблицей courses

name	varchar(20)	20	
owner	varchar(20)	20	

Таблица courses (курс)			
Имя	Тип	Размер, в байтах	Примечание
course_id	длинное целое	4	ключ
name	varchar(20)	20	
update_date	date	10	

Для хранения одного документа с N репозиториями, K контрибьюторами в среднем в каждом, каждый из которых совершил в среднем P коммитов, P issues и P pull requests, необходимо: (((P * 56) + 28) * K + 48) * N+ 34

Цифра получилась больше, чем в базе данных, основанных на mongo из-за хранения лишних повторяющихся id.

Запросы SQL

Имя запроса	Запрос	Количество запросов
Сохранение нового курса в бд	INSERT INTO *Courses* () VALUES()	1
Запрос всех коммитов (Аналогично, для исьюс, пр) у курса с id : N	Получить все репозитории select rep_id from reps where course_id = ... Получить все контрибьюторы select contributor_id from contributors where rep_id in (...) Получить все коммиты select commit_id from commits where contributor_id in (...)	3
Обновление данных -	1) update date UPDATE Courses SET date = … WHERE	4

добавление новых коммитов (Аналогично, для исьюс, пр)	course_id = ...; 2) add commits INSERT INTO *COMMITS ()* VALUES() 3) add prs also INSERT INTO *PRS ()* VALUES() 4) add issues also INSERT ISSUES *PRS ()* VALUES()	
Запрос коммитов одного репозитория с id N из курса с id K (Аналогично, для исьюс, пр)	1) Получить все контрибьюторы select contributor_id from contributors where rep_id = ... 2) Получить все коммиты select commit_id from commits where contributor_id in (...)	2
Запрос коммитов одного контрибьютора (Аналогично, для исьюс, пр)	Получить все коммиты select commit_id from commits where contributor_id =...	1
Добавление репозитория в курс	INSERT INTO *Courses* () VALUES()	1
Удаление репозитория из курса	1) Удаление репозитория (1) DELETE 2) Получение всех id и удаление всех контрибьюторов (1 + 1) SELECT & DELETE 3) Получение всех id и удаление всех коммитов, пр и исьюс (1+1) SELECT & DELETE	5
Удаление курса	1) Удаление курса (1) DELETE 2) Получение всех id и удаление репозиториев всех (1 + 1) SELECT & DELETE 3) Получение всех id и удаление всех контрибьюторов (1 + 1) SELECT & DELETE 4) Получение всех id и удаление всех коммитов, пр и исьюс (1+1) SELECT & DELETE	7

Единственное преимущество реляционной БД над документированной базой - запрос на обновление из-за отсутствия некоторых команд в MongoDB. В остальном документированная база выигрывает. Все сценарии в документированной базе работает за один запрос, в SQL это невозможно из-за того, что база состоит из нескольких таблиц.

Вывод по сравнению моделей данных

Для хранения одного документа в реляционной БД с N репозиториями, K контрибьюторами в среднем в каждом, каждый из которых совершил в среднем P коммитов, P issues и P pull requests, необходимо: (((P * 56) + 28) * K + 48) * N+ 34. В среднем одна запись в коллекции в нереляционной БД занимает (((P * 24) + 18) * K + 48) * N+ 38.
Структуры данных имеющих несколько уровней сложности лучше реализовывать в документо-ориентированных базах данных, так как выгоднее по памяти, так как нет дублирования полей id. И в некоторых случаях выгоднее по количеству операций, когда идет речь о нескольких уровнях сложности.

Разработанное приложение

Краткое описание

Данное приложение выводит статистику по работе в github. Основная цель - сравнение между собой репозиториев, участников или курсов (набор однотипных репозиториев). Статистика строится для трех типов работы в github - коммиты, пулл реквесты, исьюс. Приложение сохраняет данные на основе предыдущих запросов, чтобы заново не загружать те же самые данные. Приложение имеет функционал логина в github, чтобы работать с приватными репозиториями. Данное приложение умеет импортировать и экспортировать данные в базу данных в формате JSON.

Схема экранов приложения и/или схема интерфейса командной строки

Главный экран

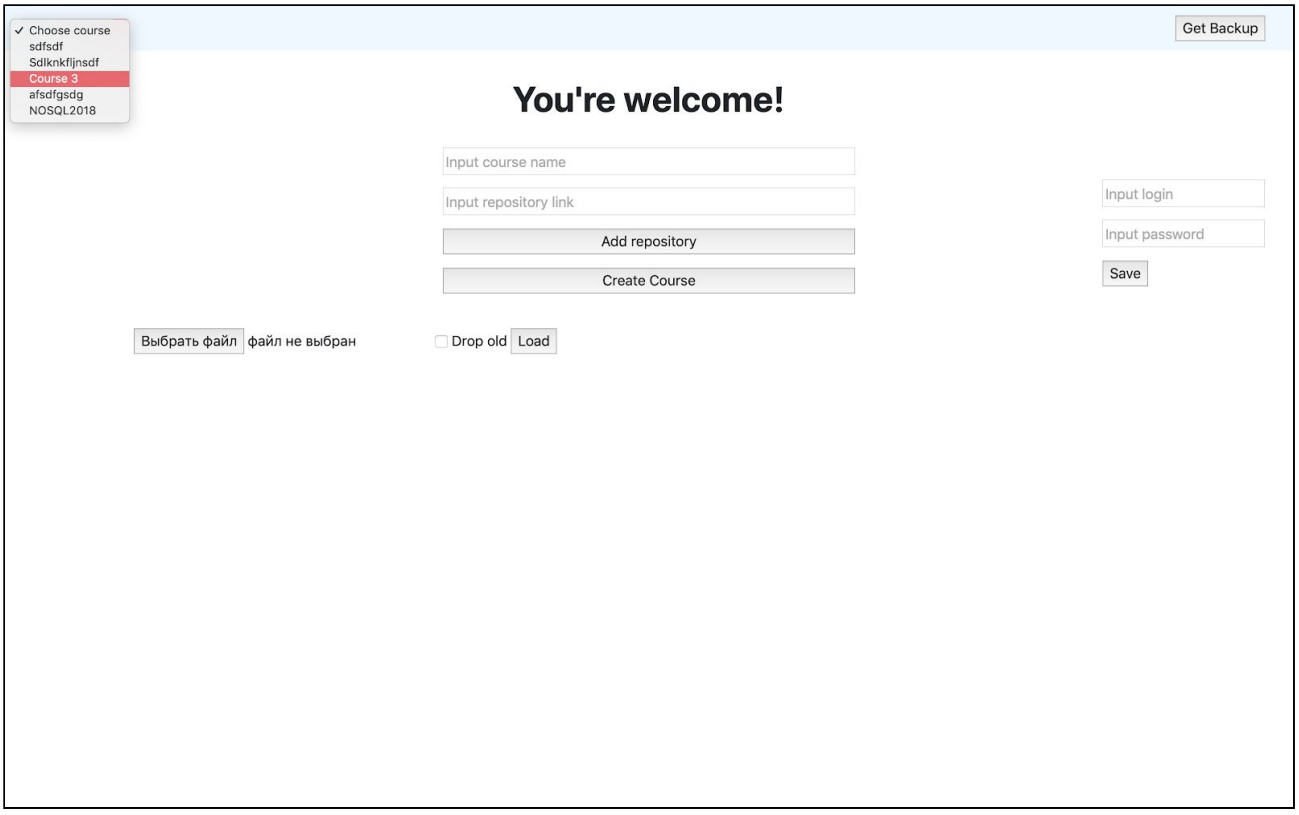


Рис. 1. Главный экран

На данном рисунке изображен главный экран, на котором вы можете добавить новый курс, залогиниться, сделать бэкап базы данных, провести импорт данных и выбрать существующий курс из списка, чтобы перейти на экран курса.

Экран курса и разные графики

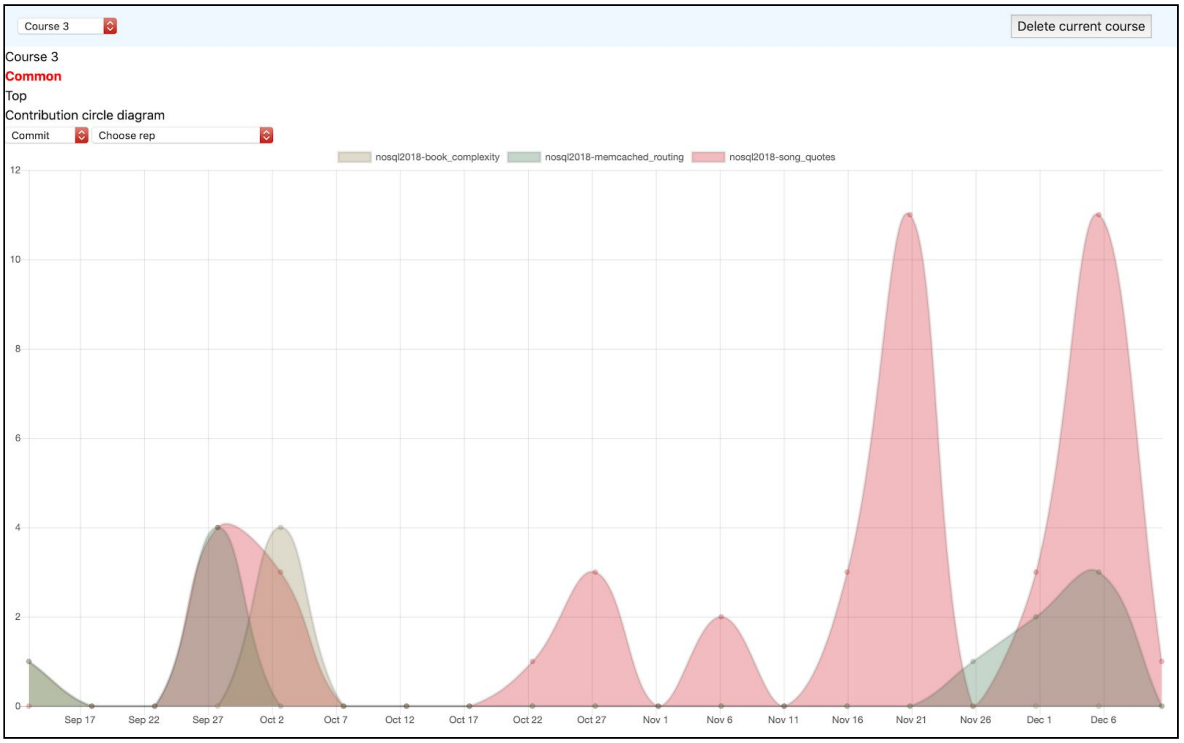


Рис. 2. Экран курса. График частоты коммитов

Рис. 3. Экран курса. График лучших участников.

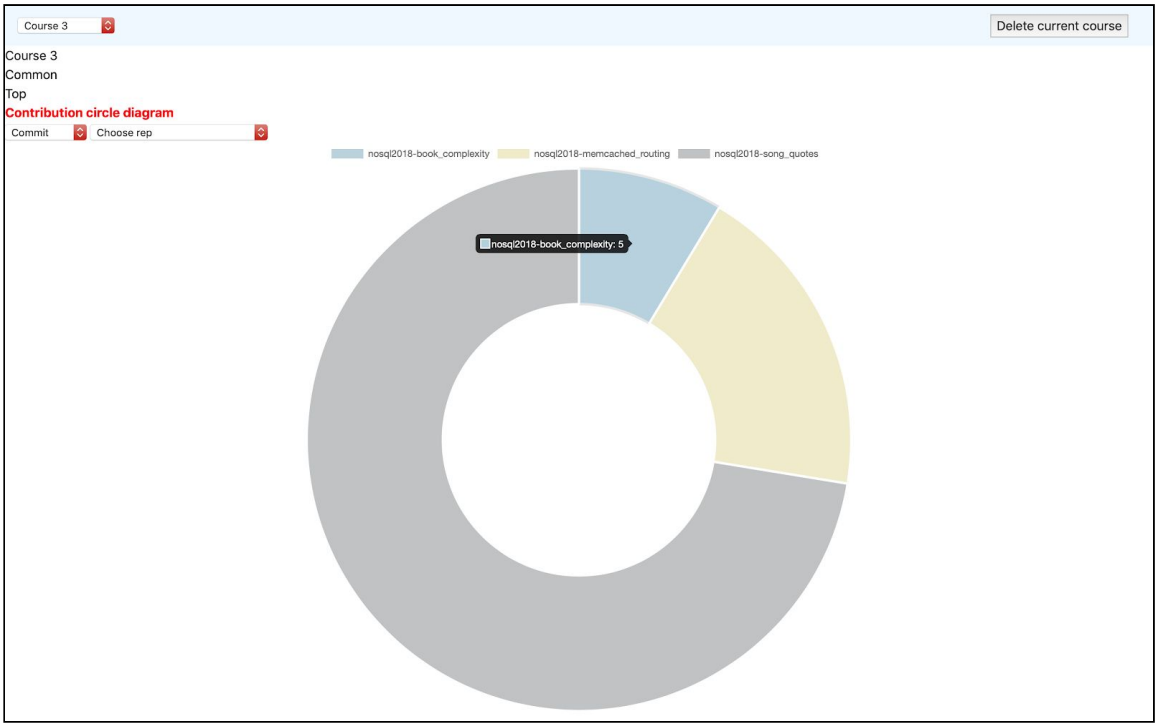


Рис. 4. Экран курса. График сравнения участников.

На данных экранах расположена вся статистика о курсе: график частоты коммитов, диаграмма лучших участников среди всех в курсе и пропорциональное распределение работы по репозиториям. На данном экране можно удалить текущий курс, перейти на экран репозитория или выбрать другую сущность для сравнения групп, например пулл реквесты.

Экран репозитория

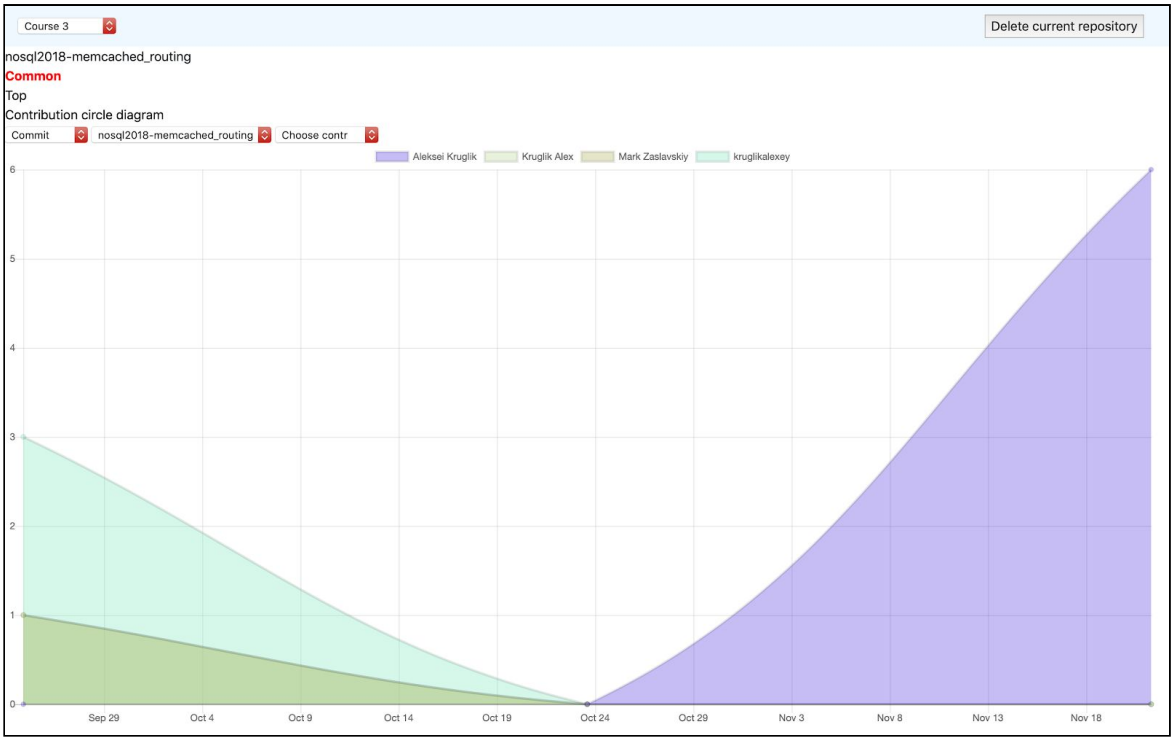


Рис. 5. Экран репозитория. График частоты коммитов.

Данный экран схож по функционалу с экраном курса. Здесь можно наблюдать аналогичную предыдущему экрану статистики. На этом экране можно перейти на экран участника или удалить текущий репозиторий из курса.

Экран участника

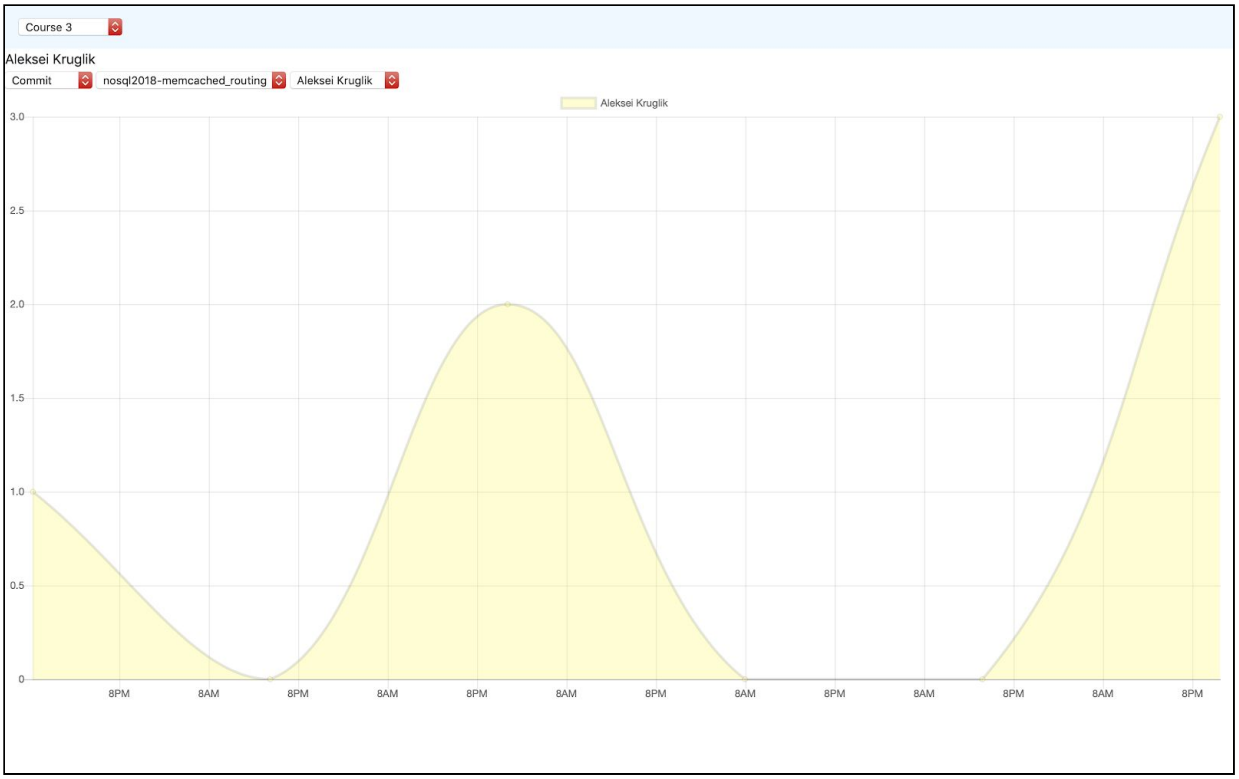


Рис. 6. Экран участника.

На текущем экране присутствует возможность просмотреть статистику по частоте коммитов одного участника в группе. Из этого экрана есть возможность только перейти на экран назад, выбрав другой репозиторий, курс или участника из списков.

Использованные технологии

Приложение состоит из двух частей: бэкенд и фронтэнд. На бэкенде происходит взаимодействие с базой данных и идет обработка данных, полученных из БД; а также присутствует логика по работе с Git API для получения анализируемых данных с Github. Полученные из базы данных данные передаются на фронтэнд, конвертируются в формат, необходимый для отрисовки, и далее по ним рисуются графики.
Интерфейс взаимодействия между Git API и бэкэндом приложения - REST по протоколу HTTPS.
Интерфейс взаимодействия между бэкэндом приложения и фронтендом приложения - REST по протоколу HTTP.

Бэкенд

Java/Kotlin - языки программирования, использованные в написании сервера.
Spring Framework - фреймворк, созданный для облегчения написания веб-приложений. В данном приложении используются модуль Boot для упрощенного развертывания системы, модуль Data для создания упрощенных GET-запросов к базе данных и модель MVC для настройки веб-приложения.
MongoDB - документированная база данных, взятая за основу и для изучения в данной работе. База имеет одну коллекцию документов.

Фронтэнд

Javascript - языки программирования, использованные в написании UI слоя приложения
JQuery - библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML
Chart.Js - библиотека, которая отвечает за отрисовку всех графиков в приложении.

Ссылки на Приложение

https://github.com/moevm/nosql2018-github_stats - исходный код приложения

Выводы

Достигнутые результаты

В ходе работы было разработано веб-приложение, позволяющее хранить, обрабатывать и обновлять статистику Github-репозиториев в документоориентированной базе данных MongoDB. Приложение позволяет просматривать статистику по курсу, состоящему из нескольких репозиториев; по одному репозиторию а также по одному контрибьютору репозитория. В качестве анализируемых сущностей используются коммиты (Commits), пулл реквесты (Pull Requests) и исьюсы (Isues). Приложение позволяет строить графики частоты анализируемх сущностей, графики лучших участников и графики сравнения участников по курсу и репозиторию, а также только графики графики частоты анализируемх сущностей по контрибьютору. В качестве дополнительного функционала приложение позволяет экспортировать и импортировать данные из базы данных и в базу данных соответственно; сохранять авторизационные данные Github-аккаунта в текущую в веб-сессию.

Недостатки и пути для улучшения полученного решения

В качестве недостатков можно выявить следующие моменты:

- В случае, когда для анализа рассматриваются “большие” репозитории (подразумеваются репозитории, имеющие количество коммитов/пулл реквестов/исьюсов более 1000) приложение начинает очень сильно проседать по производительности и зависать. В основном это связано с тем, что Git API позволяет за один раз запросить не более 100 сущностей. Соответственно когда идет речь о

больших масштабах данных, необходимо выполнять большое количество запросов для того чтобы сохранить все эти данные. Из-за этого очень сильно ухудшается производительность приложения. Также это связано с некоторыми не очень грамотными решениями в ходе разработки приложения, в частности с необходимостью делать запросы для каждого контрибьютора с целью попытки установления его имени и фамилии. Данный функционал необходим для того, чтобы исключить дублирование одинаковых контрибьюторов в репозиториях.
Путем для улучшения решения может являться оптимизация и рефакторинг не очень хорошо продуманных решений на стороне бэкэнда, однако в связи с ограничением Git API на запрос сущностей глобально проблему производительности на больших масштабах репозиториев это не решит.

- Отсутствие некоторых из заявленных на стадии дизайна частей функционала из-за нехватки рабочих ресурсов и неграмотного расчёта времени. В частности отказ от реализации анализа сущности Wiki-страницы репозитория, что связано с отсутствием простого API для получения оной. А также отсутствие некоторых элементов навигирования по приложению, например кнопки “Назад” для возвращения с графиков по репозиториям к графикам по курсу и для возвращения с графиков по контрибьюторам к графикам по репозиторию.
  Путем решения может быть более тщательное продумывание дизайна а также реализация отсутствующего функционала.

Будущее развитие решения

Решение можно развить

- усовершенствовав его дизайн;
- добавив в приложение возможность анализировать более широкий спектр сущностей;
- добавив в приложение возможность строить более широкий спектр графиков;
- добавив в приложение личный кабинет для того чтобы была возможность разделять курсы по пользователям приложения. Также это помогло бы решить проблему с доступом к приватным репозиториям.

Список используемых источников

1. GitHub REST API v3 URL: https://developer.github.com/v3/
2. MongoDB 4.0 Manual URL: https://docs.mongodb.com/manual/
3. Spring Boot Guide URL: http://spring.io/projects/spring-boot
4. Spring Data MongoDB - Reference Documentation URL: https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/
5. MongoDB Java Driver Documentation URL: http://mongodb.github.io/mongo-java-driver/3.9/
6. Chart.js Documentation URL: https://www.chartjs.org/docs/latest/

Приложение A. Документация по сборке и развертыванию приложения

https://github.com/moevm/nosql2018-github_stats/blob/master/RUN_GUIDE.md - документация по сборке и развертыванию приложения

Приложение B. Инструкция для пользователя

Главный экран

Для того, чтобы попасть на главный экран запущенного на локальном сервере приложения необходимо перейти в браузере по адресу http://localhost:8090/.

Примечание:

Для стабильной и корректной работы приложения очень рекомендуется в первую очередь сохранить в сессию авторизационные данные Github-аккаунта. Данная рекомендация связана с техническими ограничениями, налагаемыми на Git API, а также с доступом к данным приватных репозиториев.

После попадания на стартовую страницу, пользователю предлагается выбор следующих действий:

1. Добавление в базу данных нового курса
   Для того, чтобы добавить в базу данных новый курс, необходимо указать в поле с надписью “Input course name” название создаваемого курса, затем в следующем поле с надписью “Input repository link” указать ссылку на Github-репозиторий, который необходимо добавить в данный курс. Для добавления в курс большего количества репозиториев необходимо нажать кнопку “Add repository”. После указания ссылок на все желаемые репозитории необходимо нажать на кнопку “Create course”.

2. Экспорт данных, хранящихся в базе данных

   Для того, чтобы экспортировать данные, хранящиеся в базе данных, необходимо нажать на кнопку “Get Backup” в правом верхнем углу, после чего пользователю будет предложено сохранить файл в формате JSON с содержимым базы данных в желаемое расположение.

3. Импорт данных в базу данных
   a. С возможностью удаления данных, хранящихся в базе

Для того, чтобы импортировать данные из файла в формате JSON с содержимым базы данных с возможностью удаления данных, уже хранящихся в базе, необходимо нажать на кнопку “Выберите файл”, затем указать путь к файл и выбрать нужный файл. После этого необходимо поставить галочку рядом с надписью “Drop old” и нажать на кнопку “Load”.

b. С возможностью сохранения данных, хранящихся в базе

Для того, чтобы импортировать данные из файла в формате JSON с содержимым базы данных с возможностью сохранения данных, уже хранящихся в базе, необходимо нажать на кнопку “Выберите файл”, затем указать путь к файл и выбрать нужный файл. После этого необходимо нажать на кнопку “Load”.

4. Сохранение данных Github-аккаунта в текущую веб-сессию

Для того, чтобы сохранить авторизационные данные от Github-аккаунта в текущую веб-сессию необходимо заполнить логин и пароль в полях с надписями “Input login” и “Input password”, расположенных в правой части страницы, после чего нажать на кнопку “Save”, расположенную под вышеупомянутыми полями (Кнопка доступна только если авторизационные данные в текущей сессии еще не сохранены).

5. Удаление данных Github-аккаунта из текущей веб-сессии

Для того, чтобы удалить авторизационные данные от Github-аккаунта из текущей веб-сессии необходимо нажать на кнопку “Clear” под полями, отвечающими за ввод авторизационных данных (Кнопка доступна только если авторизационные данные уже сохранены в текущей сессии).

6. Просмотр статистики курсов, уже хранящихся в базе данных
Для того чтобы просмотреть статистику курсов, уже хранящихся в базе данных, необходимо выбрать желаемый курс из выпадающего списка “Choose course” в левом верхнем углу.

Экран с графиками

Для того, чтобы попасть на экран с графиками, необходимо на главной странице

A. Создать новый курс
B. Выбрать для просмотра один из имеющихся курсов

После попадания на экран с графиками пользователь может наблюдать график частоты коммитов для репозиториев, хранящихся в данном курсе. Далее на выбор пользователю предлагаются следующие действия:

1. Выбор других анализируемых объектов для просмотра статистики (коммиты/пулл реквесты/исьюсы)

   Для того, чтобы выбрать объекты, по которым будет показана статистика, необходимо выбрать желаемый объект из выпадающего списка с надписью “Commit/Pull Request/Issue” (в зависимости от выбранного на данный момент объекта)

2. Просмотр графиков одного из репозиториев, входящих в курс

   Для того, чтобы просмотреть статистику по одному из репозиториев, входящих в текущий курс, необходимо выбрать желаемый репозиторий из выпадающего списка с надписью “Choose rep” . Далее пользователь сможет просматривать аналогичную курсу статистику но только внутри репозитория между его контрибьюторами. После включения статистики по одному из репозиториев для пользователя также становятся доступны следующие действия:

   A. Просмотр графиков одного из контрибьюторов, ходящих в репозиторий

      Для того, чтобы просмотреть статистику по одному из контрибьюторов, входящих в выбранный репозиторий, необходимо выбрать желаемого контрибьютора из выпадающего списка с надписью “Choose contr”. Далее пользователь сможет просматривать аналогичную репозиторию статистику внутри контрибьютора между его коммитами/пулл реквестами/исьюсами за исключением того, что теперь доступен только график частоты коммитов.

   B. Удаление текущего репозитория из курса

      Для того, чтобы удалить текущий репозиторий из курса, необходимо нажать на кнопку “Delete current repository”, после чего репозиторий удалится из базы данных и пользователь попадет на страницу графиков для курса, в котором хранился репозиторий.

3. Выбор другого типа графика

   Для того, чтоб выбрать другой тип графика (График частоты коммитов/График лучших участников/График сравнения участников) необходимо выбрать соответствующую опцию (Common/Top/Contribution Circle Diagramm).

4. Удаление текущего курса из базы данных

Для того, чтобы удалить текущий курс из базы данных, необходимо нажать на кнопку “Delete current course”, после чего курс удалится из базы данных и пользователь попадет на стартовую страницу

5. Возвращение на стартовую страницу

   Для того, чтобы вернуться на стартовую страницу, необходимо выбрать в выпадающем списке в левом верхнем углу опцию “Choose course”

6. Выбор другого курса для просмотра его статистики

   Для того, чтобы просмотреть статистику другого курса, необходимо выбрать в выпадающем списке в левом верхнем углу жел

Приложение C. Снимки экрана приложения

Главный экран

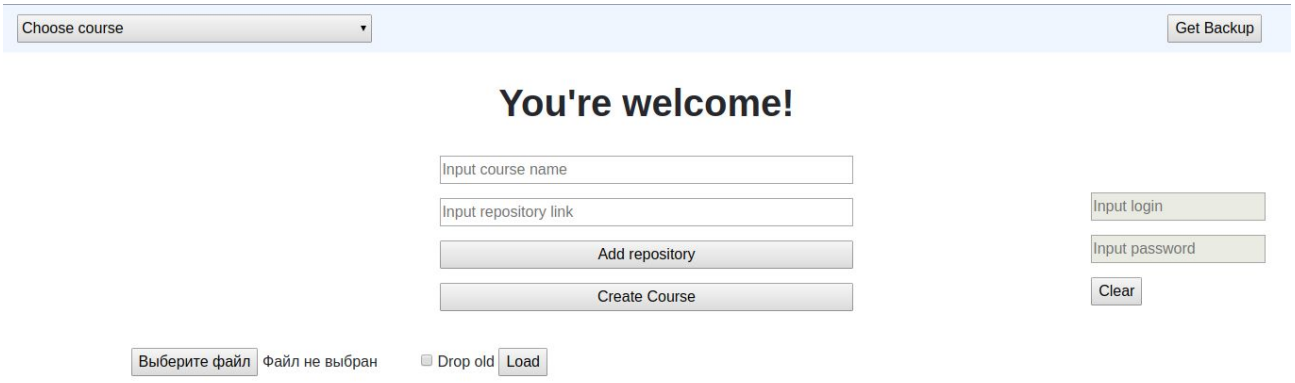


Рис. 7 Главный экран

Экран с графиками

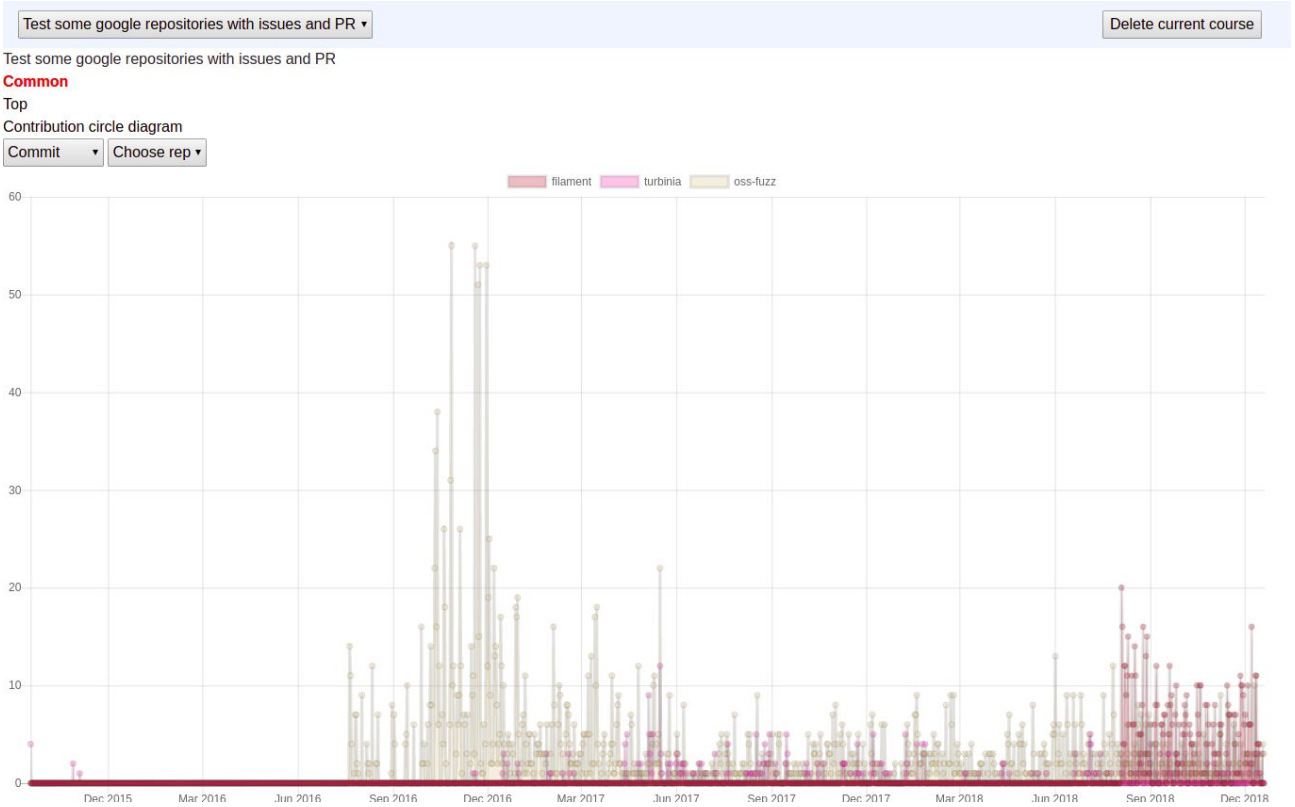


Рис. 8 График частоты коммитов по курсу

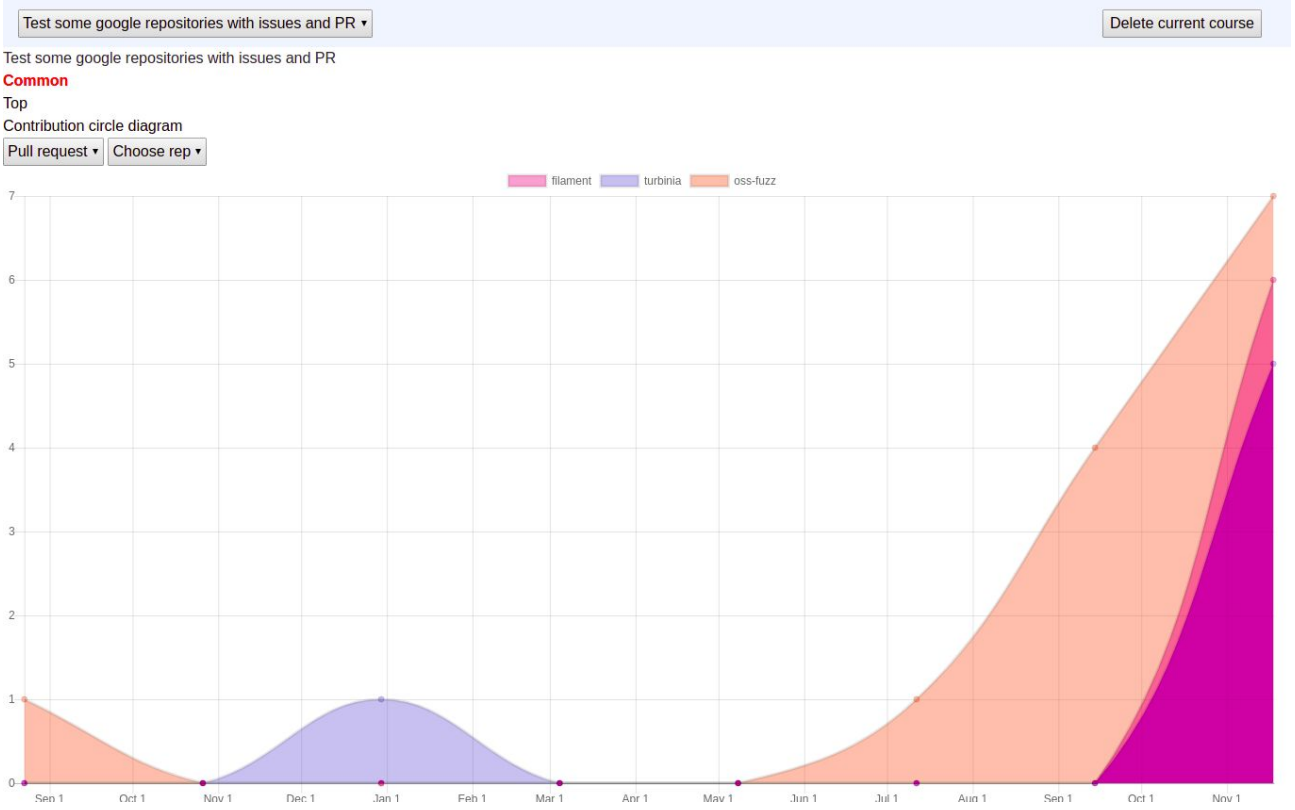


Рис. 9 График частоты пулл реквестов по курсу

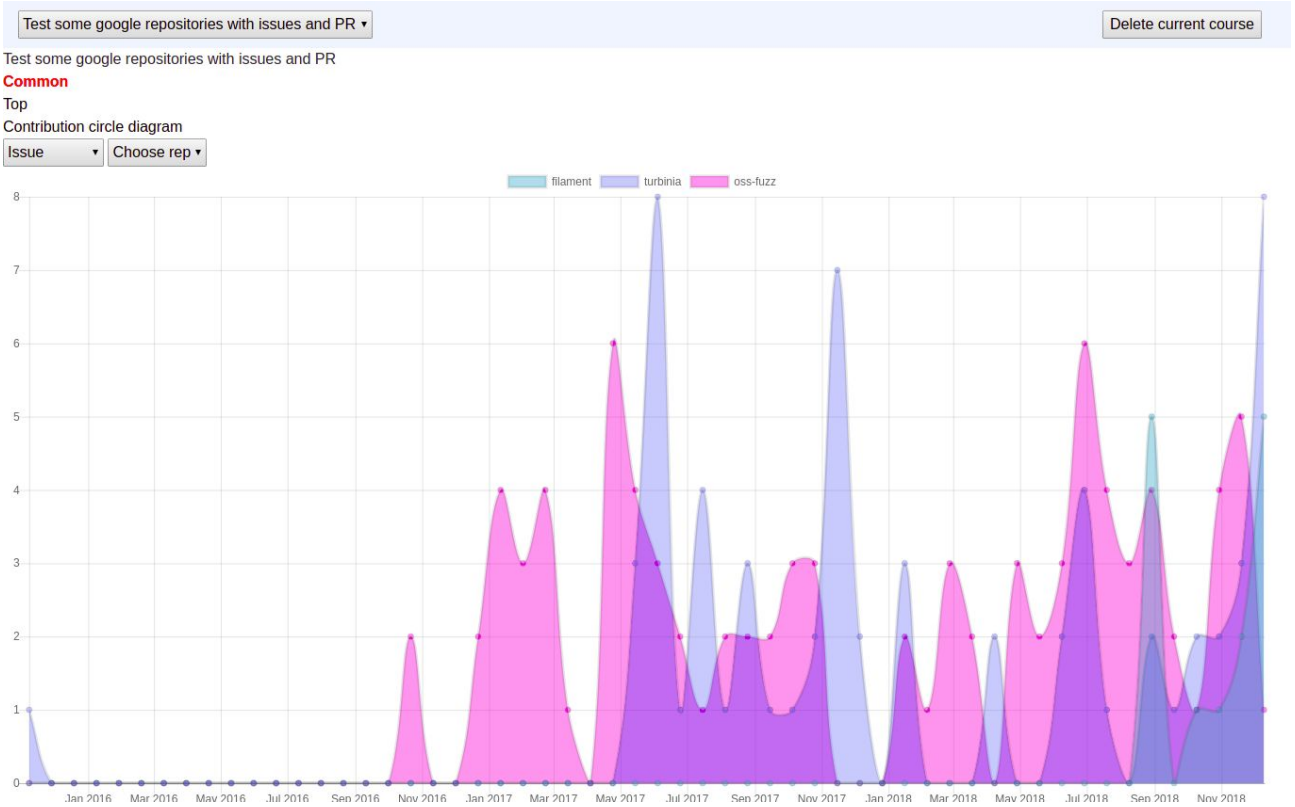


Рис. 10 График частоты исьюсов по курсу

Рис. 11 График лучших участников по курсу

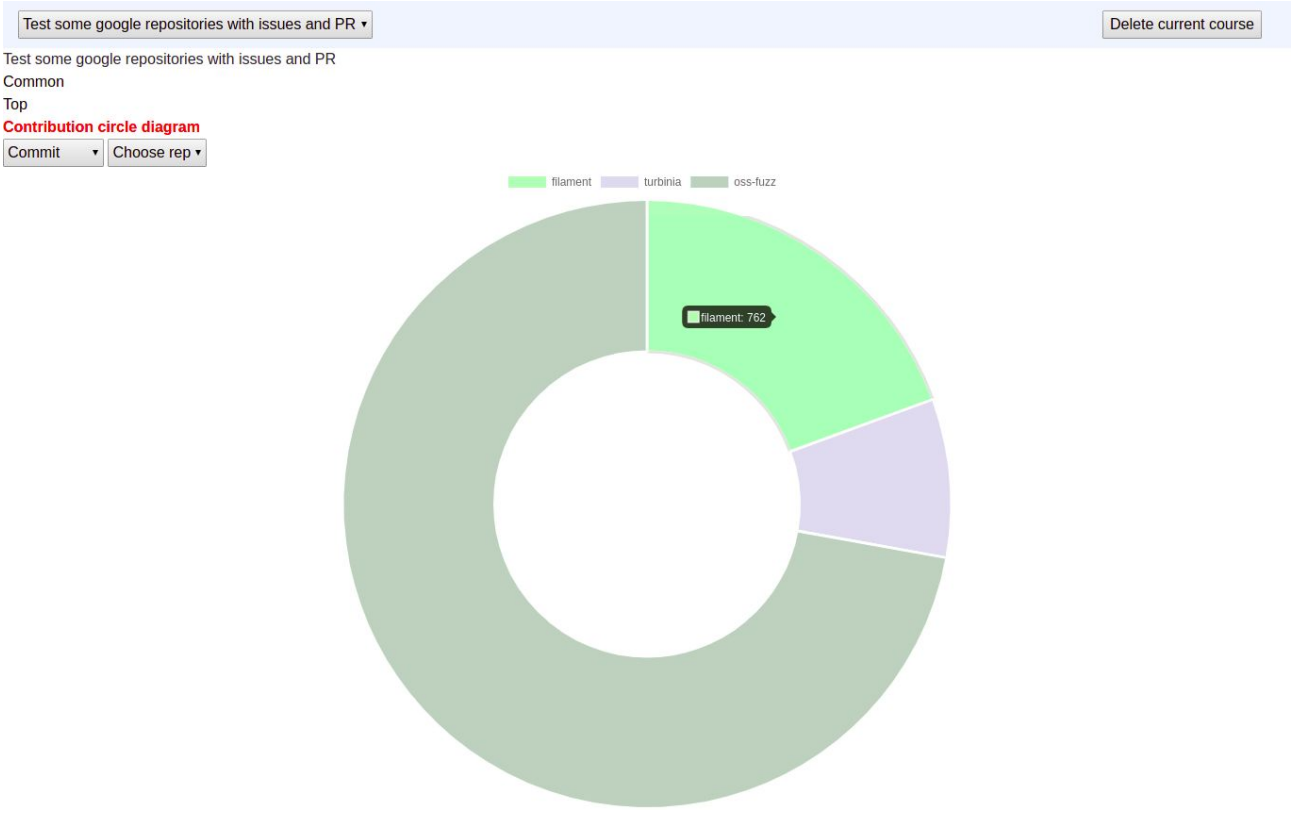


Рис. 12 График сравнения участников по курсу

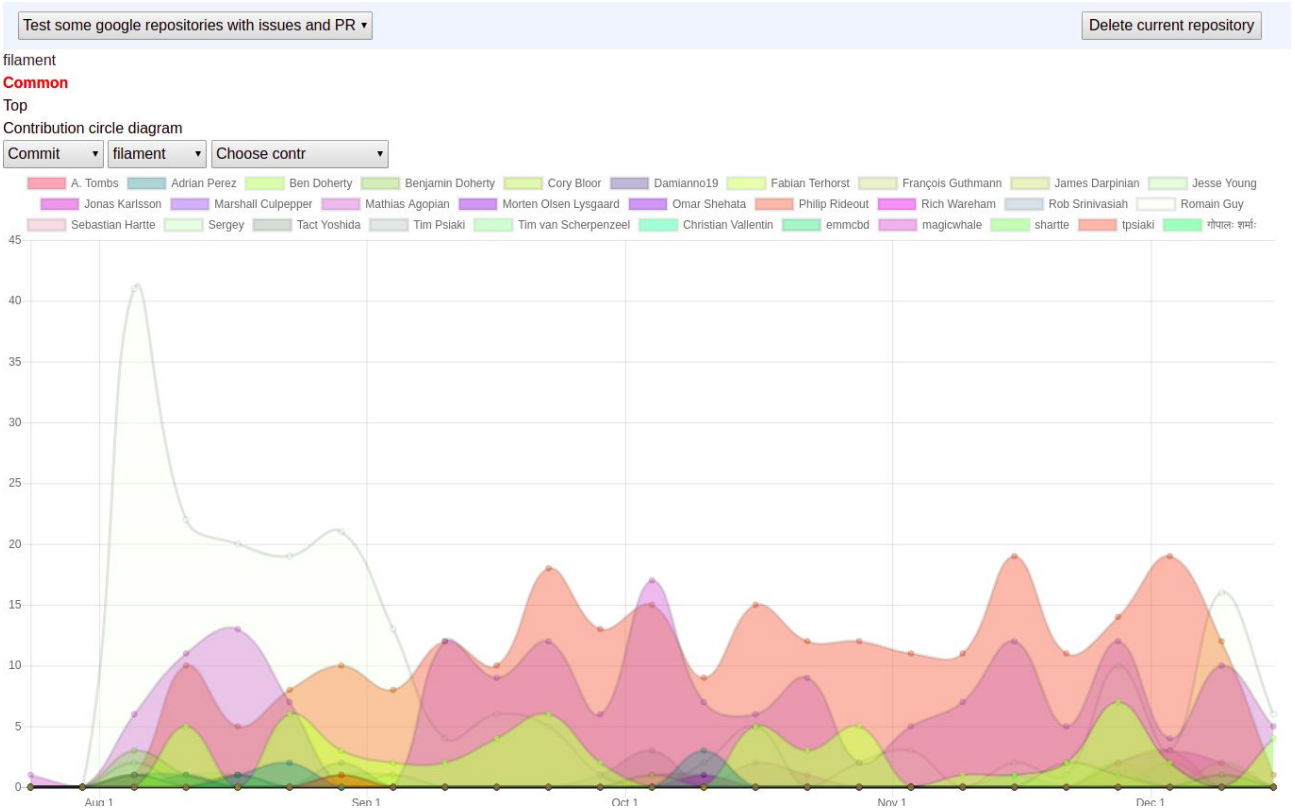


Рис. 13 График частоты коммитов по репозиторию

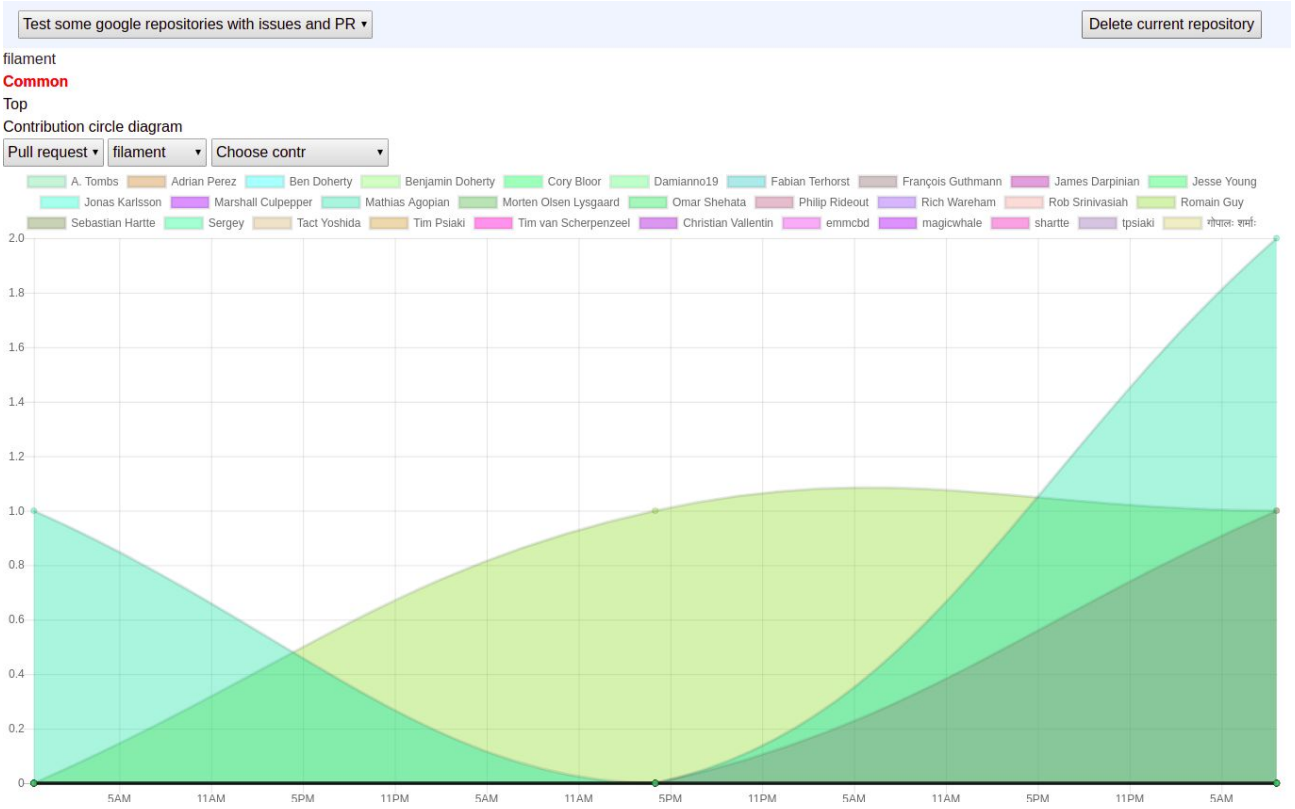


Рис. 14 График частоты пулл реквестов по репозиторию

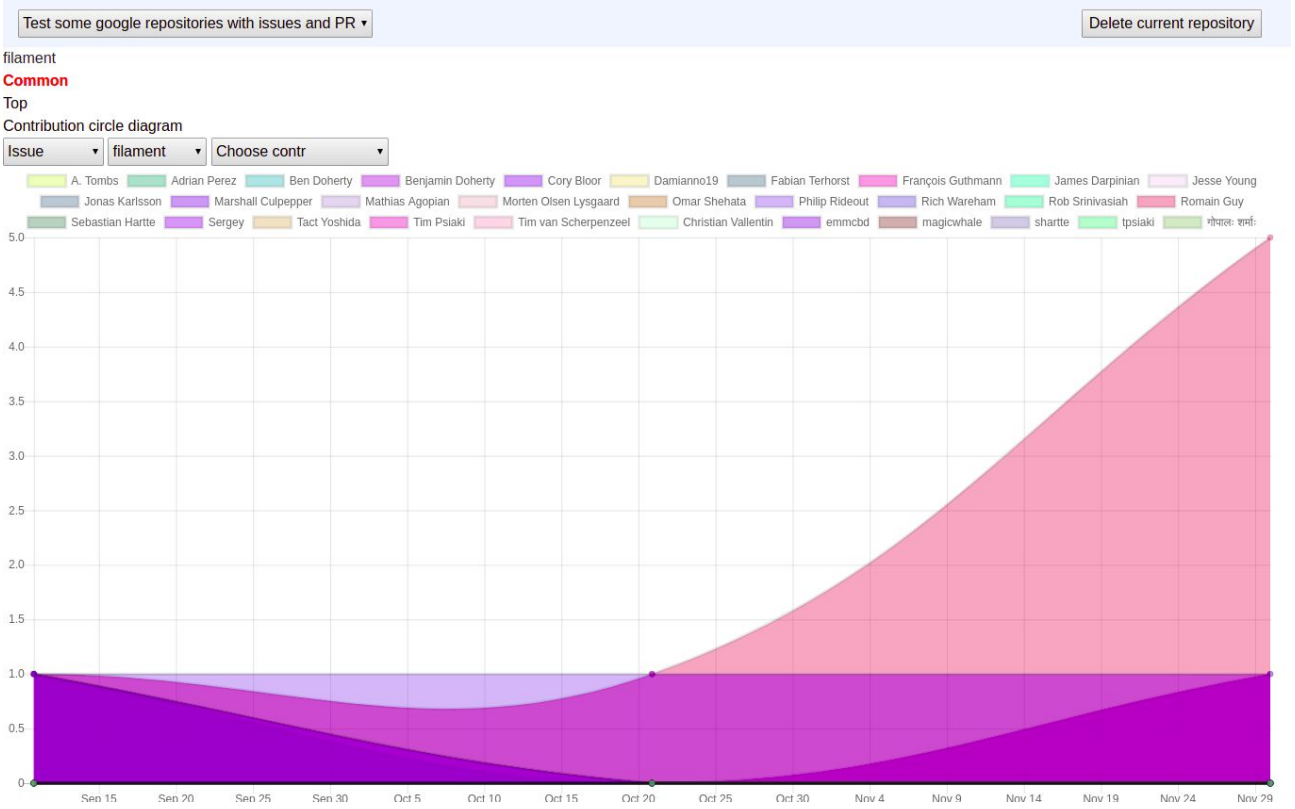


Рис. 15 График частоты исьюсов по репозиторию

Рис. 16 График лучших участников по репозиторию

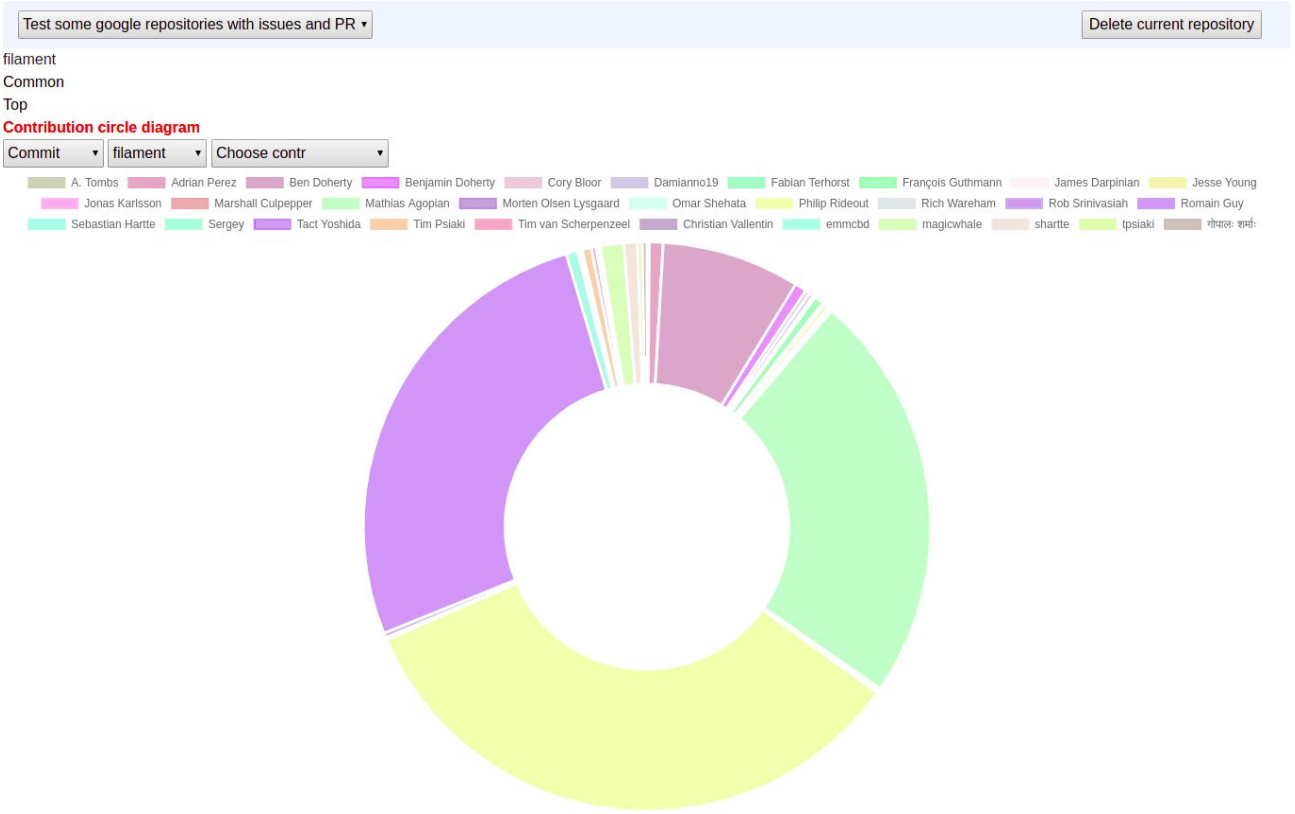


Рис. 17 График сравнения участников по репозиторию

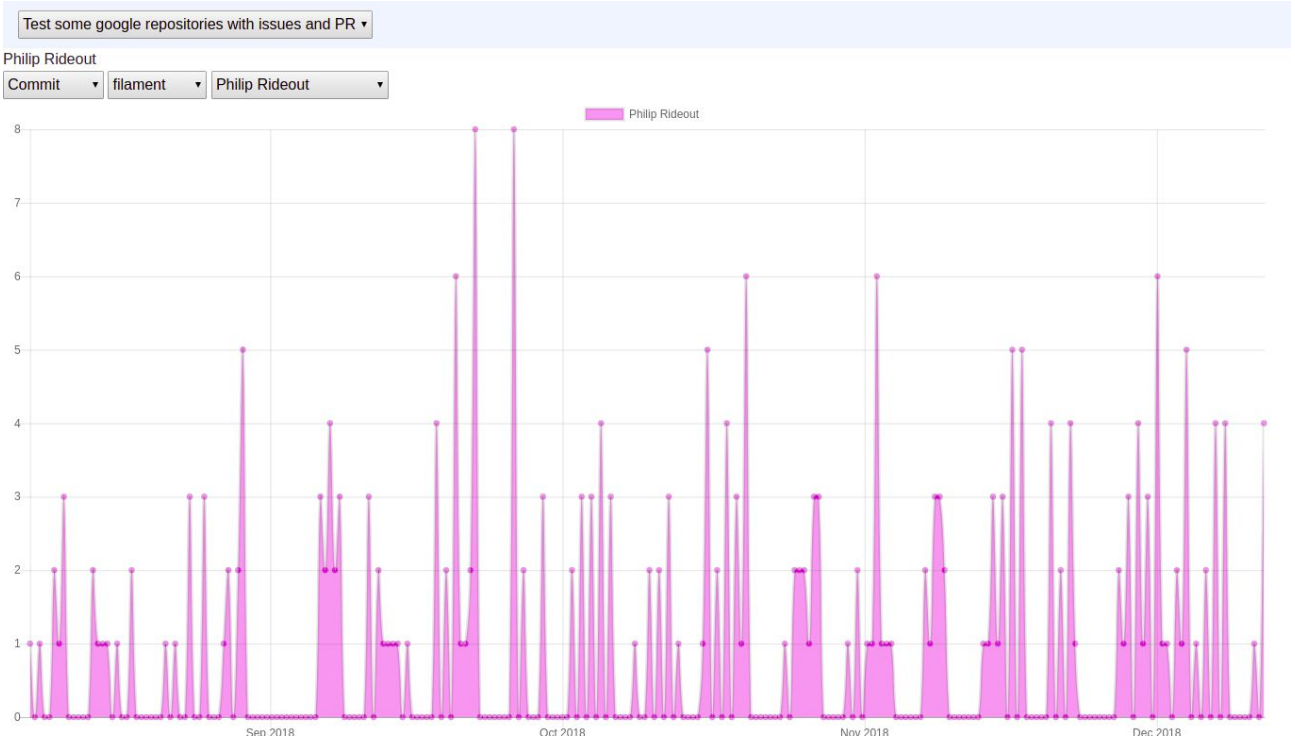


Рис. 18 График частоты коммитов по контрибьютору

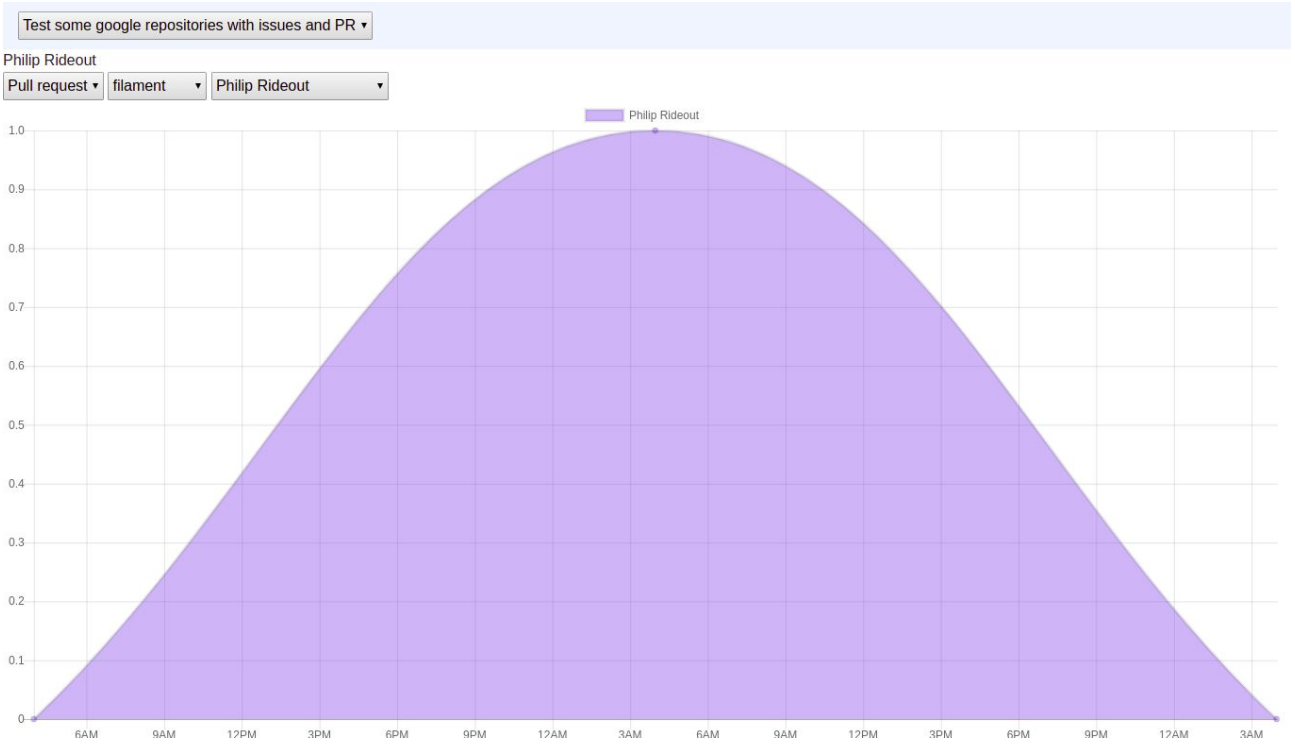


Рис. 19 График частоты пулл реквестов по контрибьютору

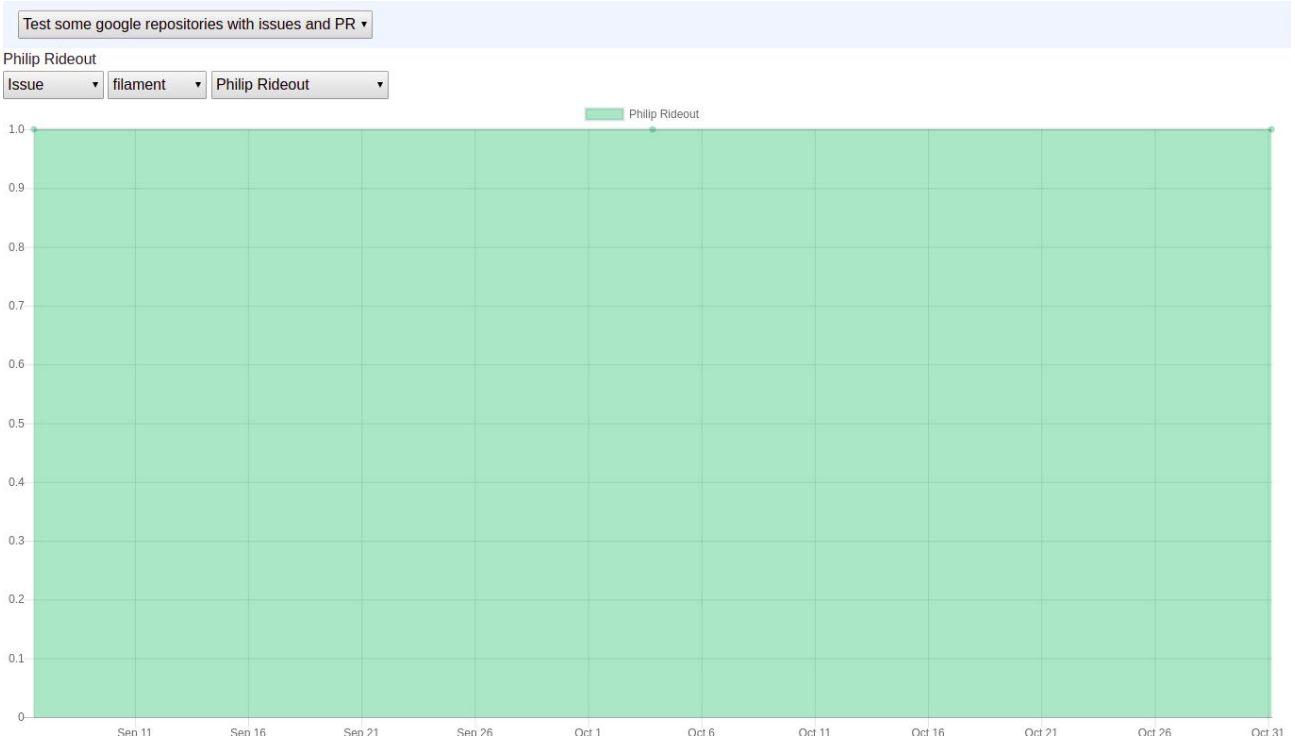


Рис. 20 График частоты исьюсов по контрибьютору