**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

**по дисциплине «Введение в нереляционные базы данных»**

**Тема: База данных статистики github репозиториев**

| Студент гр. 9381 |  | Гурин С.Н. |
| --- | --- | --- |
| Студент гр. 9381 |  | Кравченко П.В. |
| Студент гр. 9381 |  | Щеглов Д.А. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2022

**ЗАДАНИЕ**

**НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

| Студент Гурин С.Н.  Студент Кравченко П.В.  Студент Щеглов Д.А. | | |
| --- | --- | --- |
| Группа 9381 | | |
| Тема работы: База данных статистики github репозиториев | | |
| Содержание пояснительной записки:   * Аннотация * Содержание * Введение * Качественные требования к решению * Сценарии использования * Модель данных * Разработанное приложение * Заключение * Список использованных источников | | |
| Предполагаемый объем пояснительной записки:  Не менее 20 страниц. | | |
| Дата выдачи задания: 01.09.2022 | | |
| Дата сдачи реферата: 14.03.2023 | | |
| Дата защиты реферата: 14.03.2023 | | |
| Студент |  | Гурин С.Н. |
| Студент |  | Кравченко П.В. |
| Студент |  | Щеглов Д.А. |
| Преподаватель |  | Заславский М.М. |

**АННОТАЦИЯ**

В данной работе был построен макет UI для сайта фитнес-клуба и составлены сценарии использования приложения, также была построена модель данных и сформированы базы данных. В результате был разработан веб-сайт для просмотра статистики github репозиториев, выбранные пользователем.

**SUMMARY**

In this work, a user interface layout was created for a fitness club website and application use cases were compiled, a data model was created and data was generated. As a result, a website was developed to view the statistics of the github repositories selected by users.

**СОДЕРЖАНИЕ**

|  | Введение | 5 |
| --- | --- | --- |
| 1. | Качественные требования к решению | 6 |
| 2. | Сценарии использования | 7 |
| 3. | Модель данных | 9 |
| 4. | Разработанное приложение | 19 |
|  | Вывод | 22 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**ВВЕДЕНИЕ**

Цель данной работы является создание веб-приложения для просмотра статистики github репозиториев.

**1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ**

**1.1. Текущие требования**

Текущие требования к решению:

1. Веб-сайт с возможностью авторизации и регистрации.
2. Хранение баз данных статистики репозиториев в СУБД MongoDB.
3. Массовый импорт/экспорт данных в/из СУБД в формате json.
4. Сборка и развертывание приложения через docker-compose.

**2.** **СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ ПРИЛОЖЕНИЯ**

### Сценарий “Аутентификации”

##### Основной сценарий

* Пользователь вводит логин и пароль
* Нажимает кнопку войти
* Вход пользователя в систему

##### Альтернативный сценарий

* Пользователь вводит логин и пароль
* Нажимает кнопку войти
* Данные введены не верно
* Повтор сценария

### Сценарий “Регистрация”

##### Основной сценарий

* Переход в сценарий происходит в сценрии “Аутентификации”
* Пользователь вводит логин, токен и пароль
* Нажимает на кнопку “Зарегистрироваться”
* Переход в сценрий “Аутентификации”

##### Альтернативный сценарий

* Переход в сценарий происходит в сценрии “Аутентификации”
* Пользователь вводит логин, токен и пароль
* Нажимает на кнопку “Зарегистрироваться”
* Данные введены неверно

### Сценарий “Выход”

* Пользователь нажимает на соответствующую кнопку
* Пользователь переходит на страницу аутентификации

### Сценарий “Добавить репозиторий”

##### Основной сценарий

* Пользователь нажимает на кнопку “Добавить репозиторий”
* Пользователь вводит название репозитория
* Репозиторий добавлен в список репозиториев

##### Альтернативный сценарий

* Пользователь нажимает на кнопку “Добавить репозиторий”
* Пользователь вводит название репозитория
* Данные введены не верно
* Повтор сценария

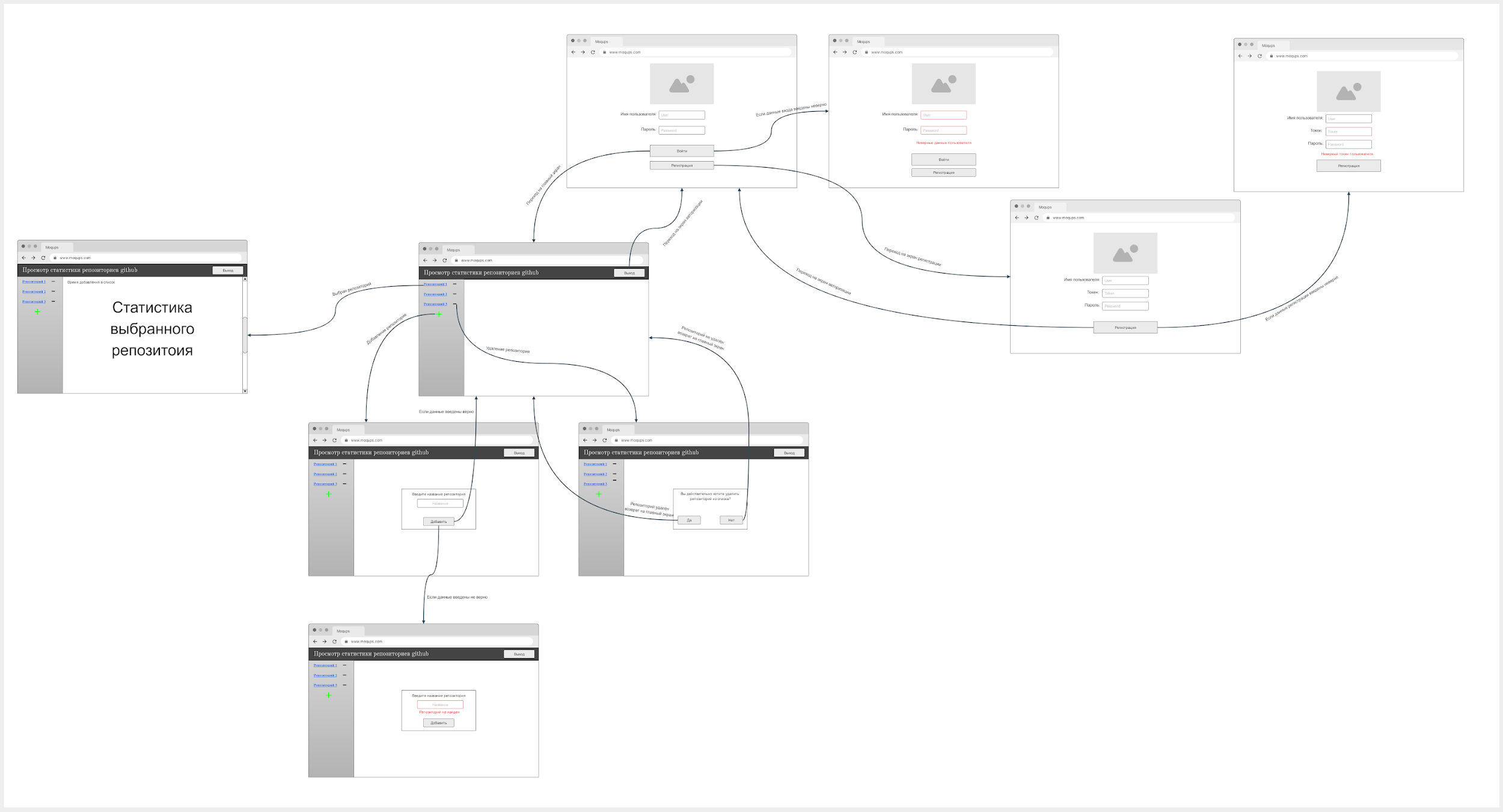
### Сценарий “Удаление репозитория”

* Пользователь нажимает на кнопку “Удалить репозиторий” рядом с выбранным репозиторием
* Пользователю выводится диалоговое окно “Удалить репозиторий или нет”
* Репозиторий удален из списка репозиториев (Если пользователь согласился)
* Репозиторий не удален из списка репозиториев (Если пользователь отказался)

### Сценарий “Просмотр статистики”

* Пользователь нажимает на уже добавленный репозиторий
* Пользователю предоставляется статистика выбранного репозитория

**Use Case диаграмма**



# 3. МОДЕЛЬ ДАННЫХ

# Схема БД и список сущностей

База данных содержит информацию о репозиториях, коммитах, пуллреквестах и Issue. Ниже приведен список полей. ### User - Username - Password - Token - Repos

### Repo

* Name
* Issues
* Commits
* Pull\_Requests

### Commit

* Id
* Author\_Name
* Author\_login
* Author\_email
* Data\_and\_time
* Changed\_files
* Message

### Pull\_Request

* Id
* Title
* State
* Commit\_into
* Creator\_name
* Creation\_date
* Creator\_login
* Creator\_email
* Changed\_files
* Comments
* Merger\_login
* Merger\_email

### Issue

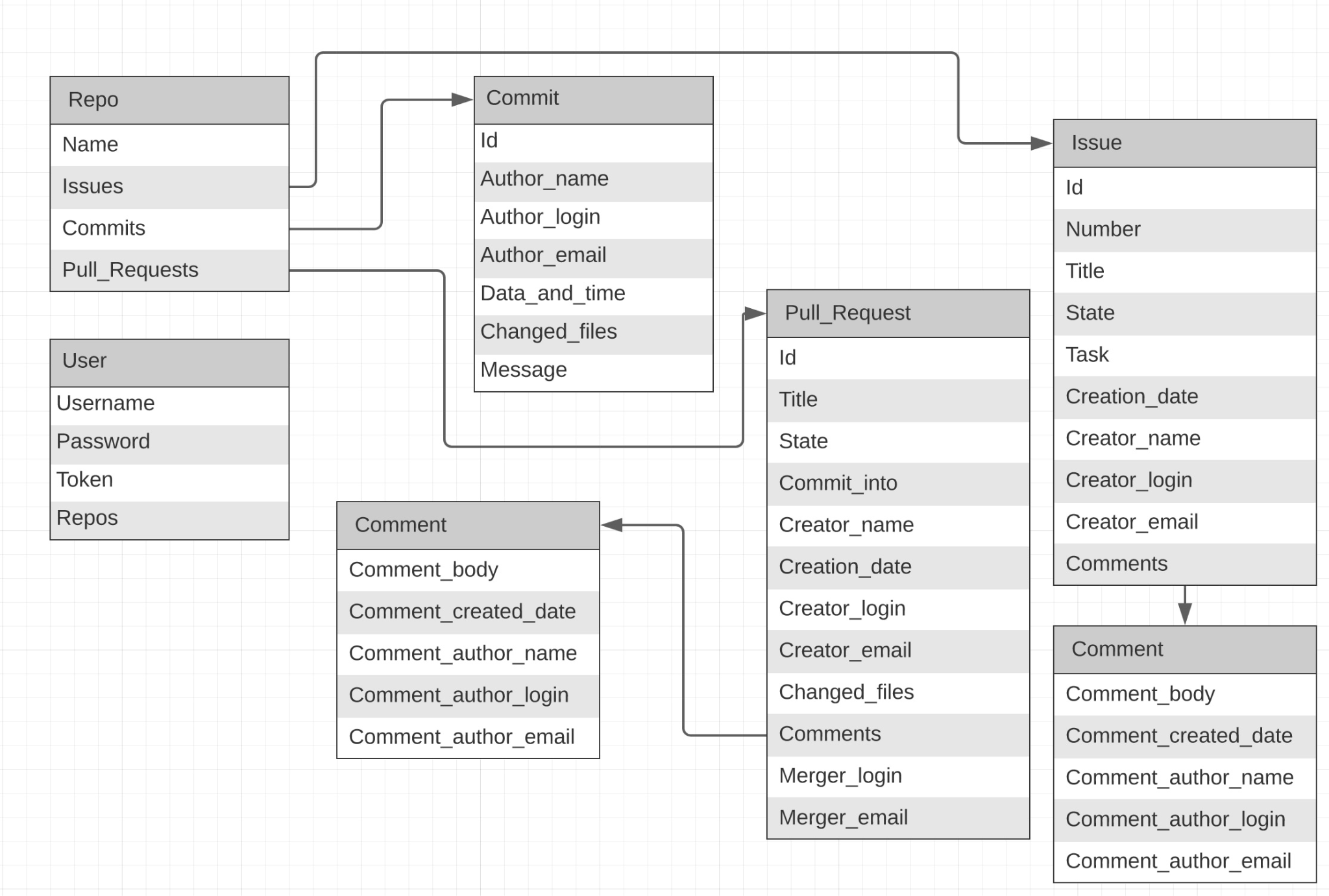
* Id
* Number
* Title
* State
* Task
* Creation\_date
* Creator\_name
* Creator\_login
* Creator\_email
* Changed\_files
* Comments

### Comment

* Comment\_body
* Comment\_created\_date
* Comment\_author\_name
* Comment\_author\_login
* Comment\_author\_email

# Нереляционная модель данных

## Графическое представление



Оценка удельного объема информации, хранимой в модели Пусть в базе данных N1 пользователей N2 репозиториев N3 пулреквестов N4 коммитов и N5 Issue . Максимальные размеры полей документов: ### User - UserName: 128 bytes - Password: 128 bytes - Token: 93 bytes - Repos: 760 bytes

Чистый объём документа user: 1109 bytes

Фактической объём документа user: 1109 bytes

### Repo

* Name: 256 bytes
* Issues: 760 bytes
* Commits: 760 bytes
* Pull\_Requests: 760 bytes

Чистый объём документа Repo: 2536 bytes

Фактической объём документа Repo: 2536 bytes

### Commit

* Id: 8 bytes
* Author\_Name :128 bytes
* Author\_login: 128 bytes
* Author\_email: 128 bytes
* Data\_and\_time: 20 bytes
* Changed\_files: 760 bytes
* Message: 408 bytes

Чистый объём документа commit: 1572 bytes

Фактической объём документа commit: 1580 bytes

### Pull\_Request

* Id: 8 bytes
* Title: 256 bytes
* State: 100 bytes
* Commit\_info: 100 bytes
* Creator\_name: 128 bytes
* Creation\_date: 20 bytes
* Creator\_login: 128 bytes
* Creator\_email: 128 bytes
* Changed\_files: 760 bytes
* Comments: 1404 bytes
* Merger\_login: 128 bytes
* Merger\_email: 128 bytes

Чистый объём документа Pull\_Request: 4684 bytes

Фактической объём документа Pull\_Request: 4692 bytes

### Issue

* Id: 8 bytes
* Number: 8 bytes
* Title: 256 bytes
* State: 100 bytes
* Task: 1000 bytes
* Creation\_date: 20 bytes
* Creator\_name: 128 bytes
* Creator\_login: 128 bytes
* Creator\_email: 128 bytes
* Changed\_files: 760 bytes
* Comments: 1404 bytes

Чистый объём документа Issue: 3932 bytes

Фактической объём документа Issue: 3940 bytes

### Comment

* Comment\_body: 1000 bytes
* Comment\_created\_date: 20 bytes
* Comment\_author\_name: 128 bytes
* Comment\_author\_login: 128 bytes
* Comment\_author\_email: 128 bytes

Чистый объём документа Comment: 1404 bytes

Фактической объём документа Comment: 1404 bytes

## Избыточность модели

Пусть в базе данных N1=5 пользователей, N2=10 репозиториев, N3=100 пулреквестов, N4=120 коммитов, N5=10 Issue и N6=200 комментариев

Vclear = N1\*1109 + N2\*2536 + N3\*4684 + N4\*1164+N5\*3932 + N6\*1404= 5\*1109 + 10\*2536 + 100\*4684 +120\*1164 + 10\*3932 + 200\*1404 = 959105 bytes

Vreal = N1\*1109 + N2\*2536 + N3\*4692 + N4\*1172+N5\*3940 + N6\*1404= 5\*1109 + 10\*2536 + 100\*4692 +120\*1172 + 10\*3940 + 200\*1404 = 960945 bytes

Избыточность: Vreal / Vclear = 960945/959105 = 1,00191846

## Направление роста модели при увеличении количества объектов каждой сущности

Размер базы данных растет линейно по каждому параметру.

## Запросы к модели, с помощью которых реализуются сценарии использования

### Добавление

collection.new\_commit({ “Id”:1, “Author\_Name”:“Pavel”, “Author\_login”:“cirilohysP”, “Author\_email”:“realmathsmc@gmail.com”, “Data\_and\_time”:“2022-12-12 19:58:48”, “Changed\_files”:“,Новая папка (2)/README.md” })

### Удаление

collection.delete\_commit(‘id’ : ‘1’)

### Обновление

collection.update\_commit({“\_id” : “1”}, {“$set” : {“Author\_Name” :“Pavel Kravchenko”}})

### Поиск по полю

collection.find\_commit({“Author\_login” : “Dunkel”})

### Сортировка

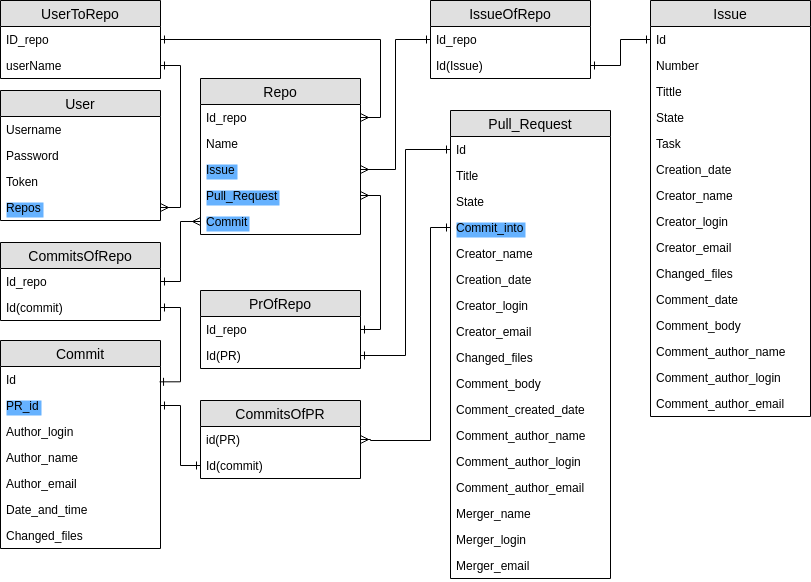
collection.find().sort(“Author\_Name”)

### Поиск по нескольким полям

collection.find\_commit({“Author\_Name”:“Pavel”, “Changed\_files”:“,Новая папка (2)/README.md”})

# Аналог модели данных для SQL СУБД

## Графическое представление



*ModelData*

### User

* UserName:128 bytes
* Password: 128 bytes
* Token: 93 bytes

Чистый объём документа user: 349 bytes

Фактической объём документа user: 349 bytes

### Repo

* Name: 256 bytes
* Id: 8 bytes Чистый объём документа Repo: 256 bytes

Фактической объём документа Repo: 264 bytes

### Commit

* Id: 8 bytes
* Author\_Name: 128 bytes
* Author\_login: 128 bytes
* Author\_email: 128 bytes
* Data\_and\_time: 20 bytes
* Changed\_files: 760bytes

Чистый объём документа commit: 1164 bytes

Фактической объём документа commit: 1172 bytes

### Pull\_Request

* Id:8 bytes
* Title:256 bytes
* State: 100 bytes
* Creator\_name: 128 bytes
* Creation\_date: 20 bytes
* Creator\_login: 128 bytes
* Creator\_email: 128 bytes
* Changed\_files: 760bytes
* Comment\_body: 1000 bytes
* Comment\_created\_date: 20 bytes
* Comment\_author\_name: 128 bytes
* Comment\_author\_login: 128 bytes
* Comment\_author\_email: 128 bytes
* Merger\_name: 128 bytes
* Merger\_login: 128 bytes
* Merger\_email: 128 bytes

Чистый объём документа Pull\_Request: 3308 bytes

Фактической объём документа Pull\_Request: 3316 bytes

### Issue

* Id:8 bytes
* Number:8 bytes
* Title:256 bytes
* State:100 bytes
* Task: 1000 bytes
* Creation\_date:20 bytes
* Creator\_name:128 bytes
* Creator\_login:128 bytes
* Creator\_email:128 bytes
* Changed\_files:760 bytes
* Comment\_date:20 bytes
* Comment\_body:1000 bytes
* Comment\_author\_name:128 bytes
* Comment\_author\_login:128 bytes
* Comment\_author\_email:128 bytes

Чистый объём документа Issue: 3932 bytes

Фактической объём документа Issue: 3940 bytes

### IssueOfRepo

* Id\_repo: 8 bytes
* Id(Issue):8 bytes

Чистый объём документа IssueOfRepo: 0 bytes

Фактической объём документа IssueOfRepo: 16 bytes

### UserToRepo

* ID\_repo:8 bytes
* userName:8 bytes

Чистый объём документа UserToRepo: 0 bytes

Фактической объём документа UserToRepo: 16 bytes

### CommitsOfRepo

* Id\_repo:8 bytes
* Id(commit):8 bytes

Чистый объём документа CommitsOfRepo: 0 bytes

Фактической объём документа CommitsOfRepo: 16 bytes

### PrOfRepo

* Id\_repo:8 bytes
* Id(PR):8 bytes

Чистый объём документа PrOfRepo: 0 bytes

Фактической объём документа PrOfRepo: 16 bytes

### CommitsOfPR

* id(PR):8 bytes
* Id(commit):8 bytes

Чистый объём документа PrOfRepo: 0 bytes

Фактической объём документа PrOfRepo: 16 bytes

## Оценка удельного объема информации, хранимой в модели

Пусть у нас есть N1=5 пользователей каждый работает в N2=10 репозиториях (все во всех) Имеется по N4=120 коммитов, N3=100 PR и N5=10 Issue

Vclear = N1\*349 + N2\*256 + N3\*3308 + N4\*1164+N5\*3932 = 5\*349 + 10\*256 + 100\*3308 +120\*1164 + 10\*3932 = 514105 bytes

Vreal = N1\*349 + N2\*264 + N3\*3316 + N4\*1172+N5\*3940+N1\*N2\*16+N3\*N2\*16+N4\*N2\*16+N5\*N2\*16+N4\*16= 5\*349 + 10\*264 + 100\*3316 +120\*1172 + 10\*3940 +5\*10\*16+100\*10\*16+120\*10\*16+10\*10\*16+120\*16= 555545 bytes

Тогда по сравнению с нереаляционной мы задействуем на N2\*(N1+N3+N4+N5)\*(8+8)+N4\*(8+8) = 39520 bytes на свяжуюшие таблицы, но сэкономим на Полях перечисления N1\*(1109-349) + N2\*(2544-264) + N3\*(3416-3316) + N4\*(1172-1172)+ N5\*(3940-3940) = 5\*760 + 10\*2280 + 100\*100 + 120\*0 + 10 \*0 = 36600 bytes.

Разница фактического веса моделей при заданных данных равна: 555545 - 552625 = 2920 bytes

## Избыточность модели

Избыточность: Vreal / Vclear = 555545/514105 = 1,08060610187

## Направление роста модели при увеличении количества объектов каждой сущности

Размер базы данных растет линейно по каждому параметру.

## Запросы к модели, с помощью которых реализуются сценарии использования

### Добавление

INSERT INTO Commit VALUES (…);

### Удаление

DELETE FROM Commit WHERE id == 1;

### Обновление

UPDATE commit SET Author\_Name = Pavel Kravchenko WHERE id == 1;

### Поиск по полю

SELECT \* FROM commit WHERE Author\_login = Dunkel;

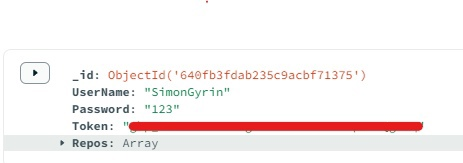
### Сортировка

SELECT \* FROM commit ORDERED BY Author\_Name ASC;

### Поиск по нескольким полям

SELECT \* FROM commit WHERE Author\_Name = Pavel AND Changed\_files=“Новая папка (2)/README.md”

## Пример представления данных в базе данных

## Сравнение моделей

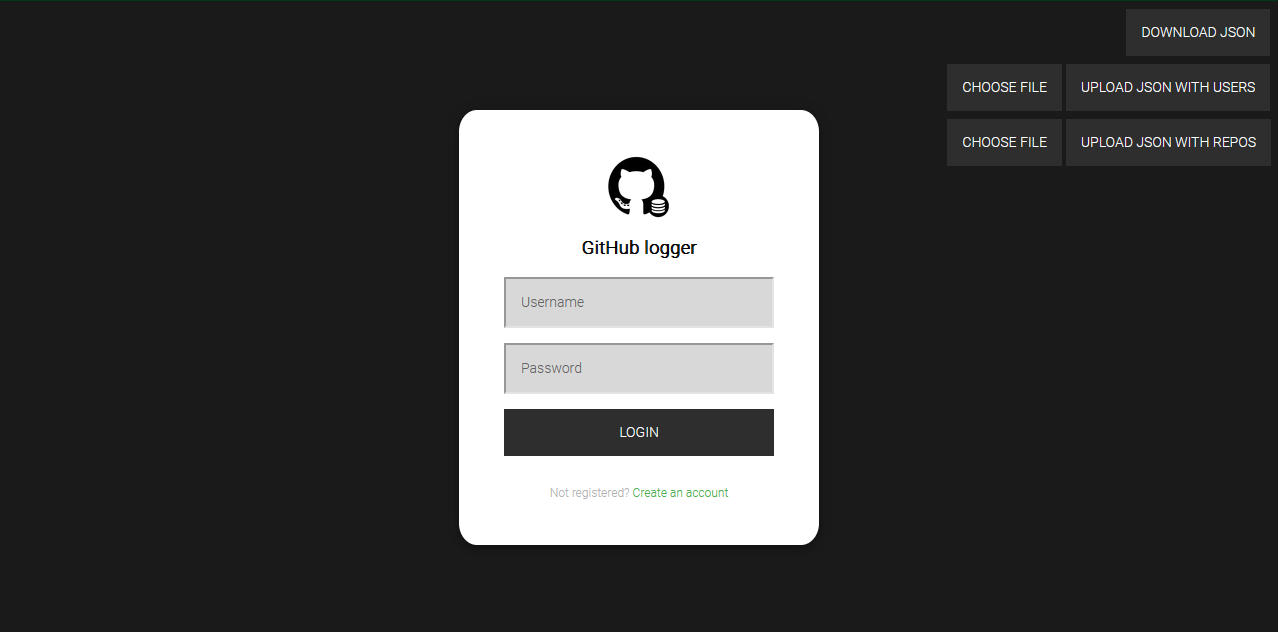
В данной модели обнаружено преимущество нереляционной модели все связи с отсутствием накладных данных на таблицы связи, которые покрываются списками в одной сущности.

**4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ**

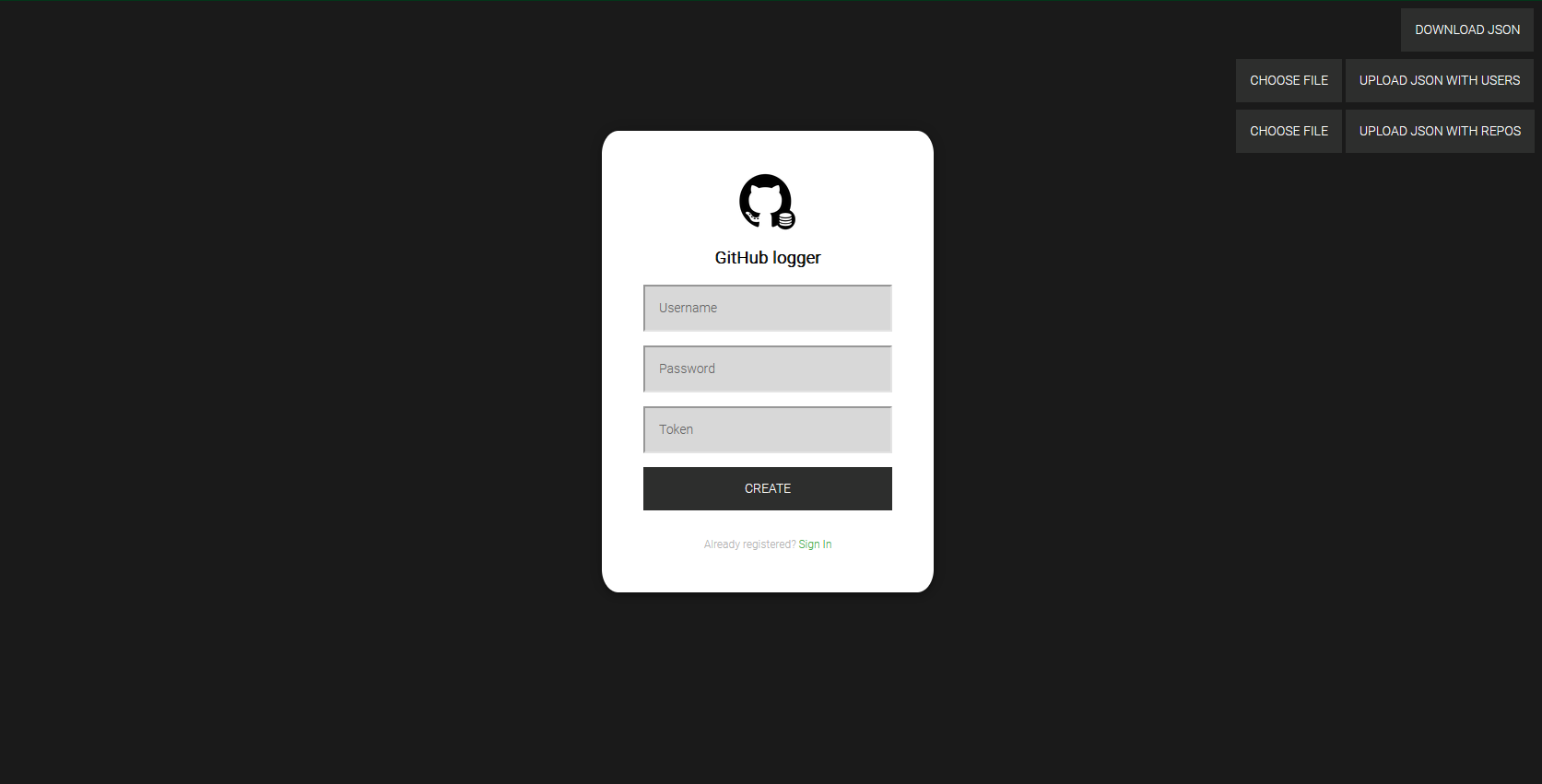
**Краткое описание**

Приложение реализовано с использованием Python библиотеки Flask, для работы с MongoDB была использована библиотека pymongo.

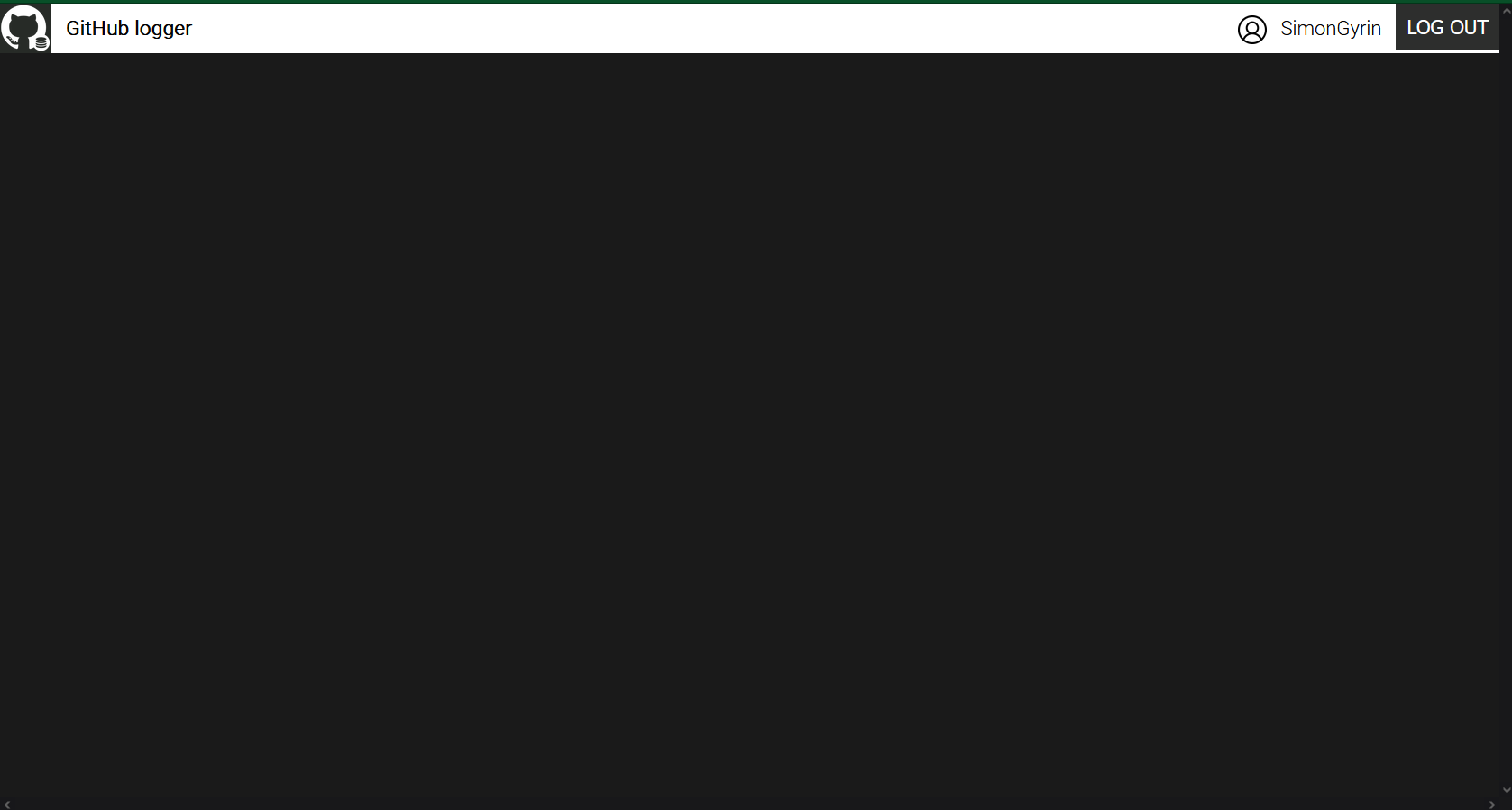
**Схема экранов приложения**

****

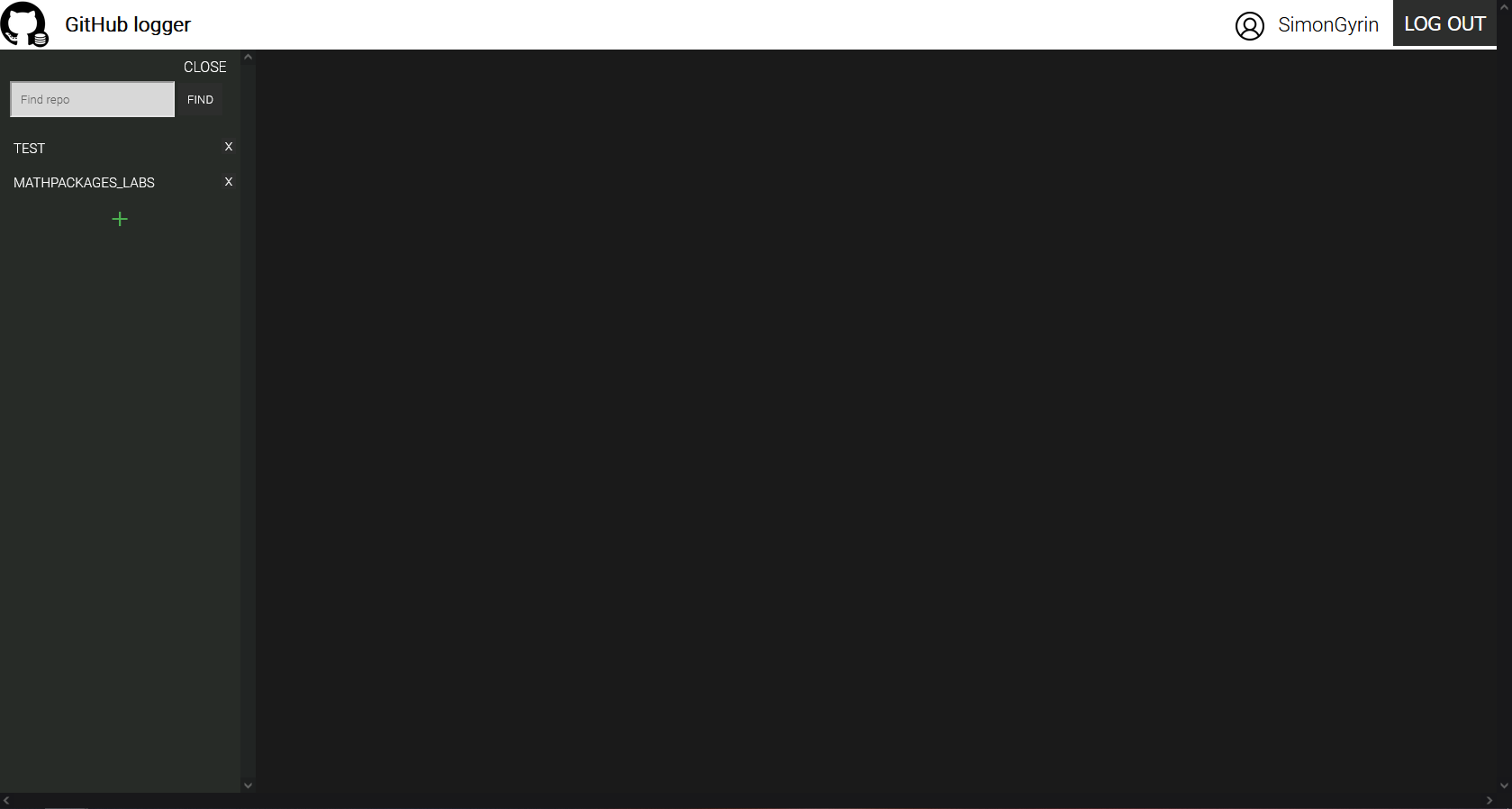
Экран входа

****

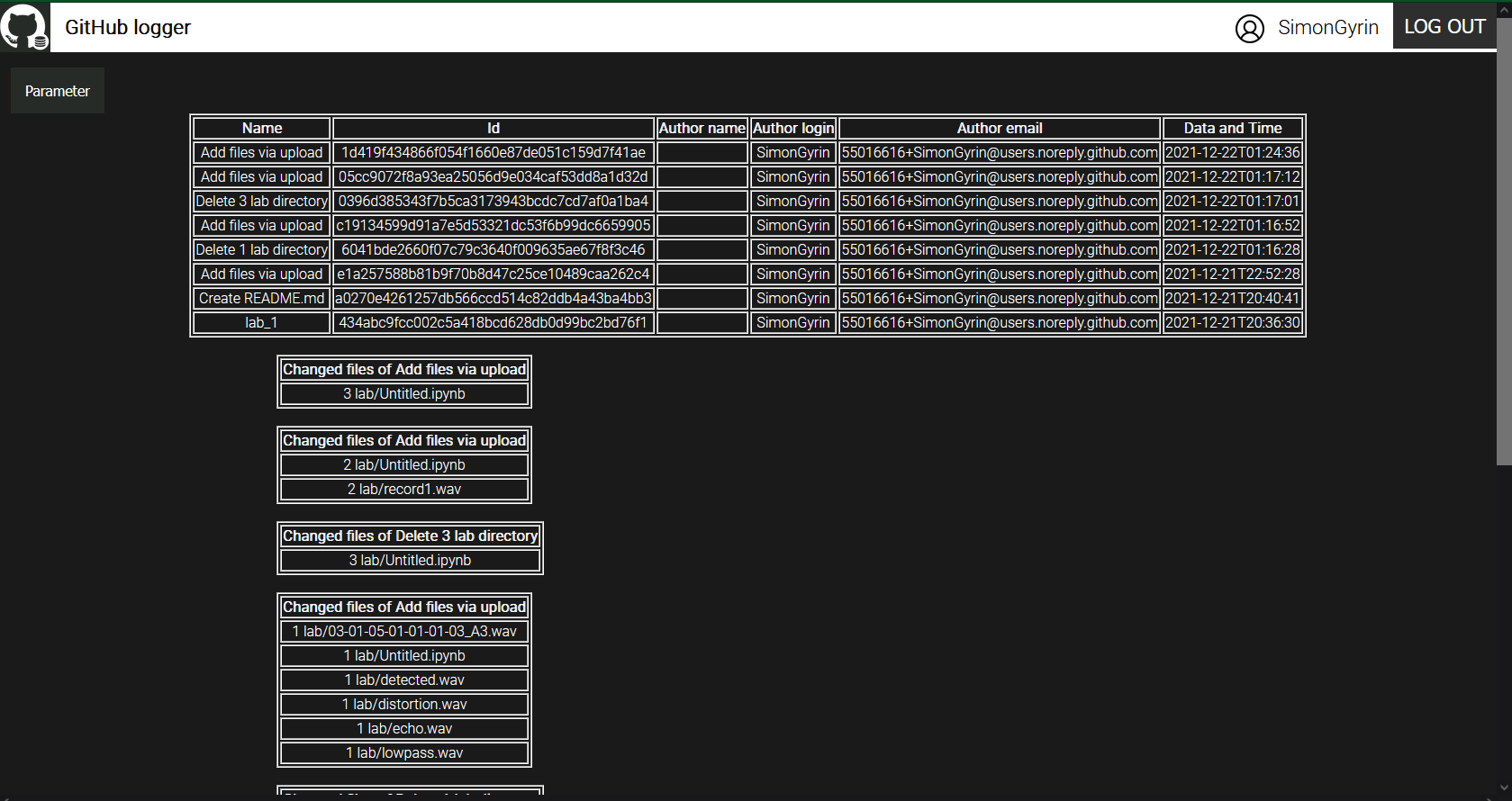
Экран регистрации



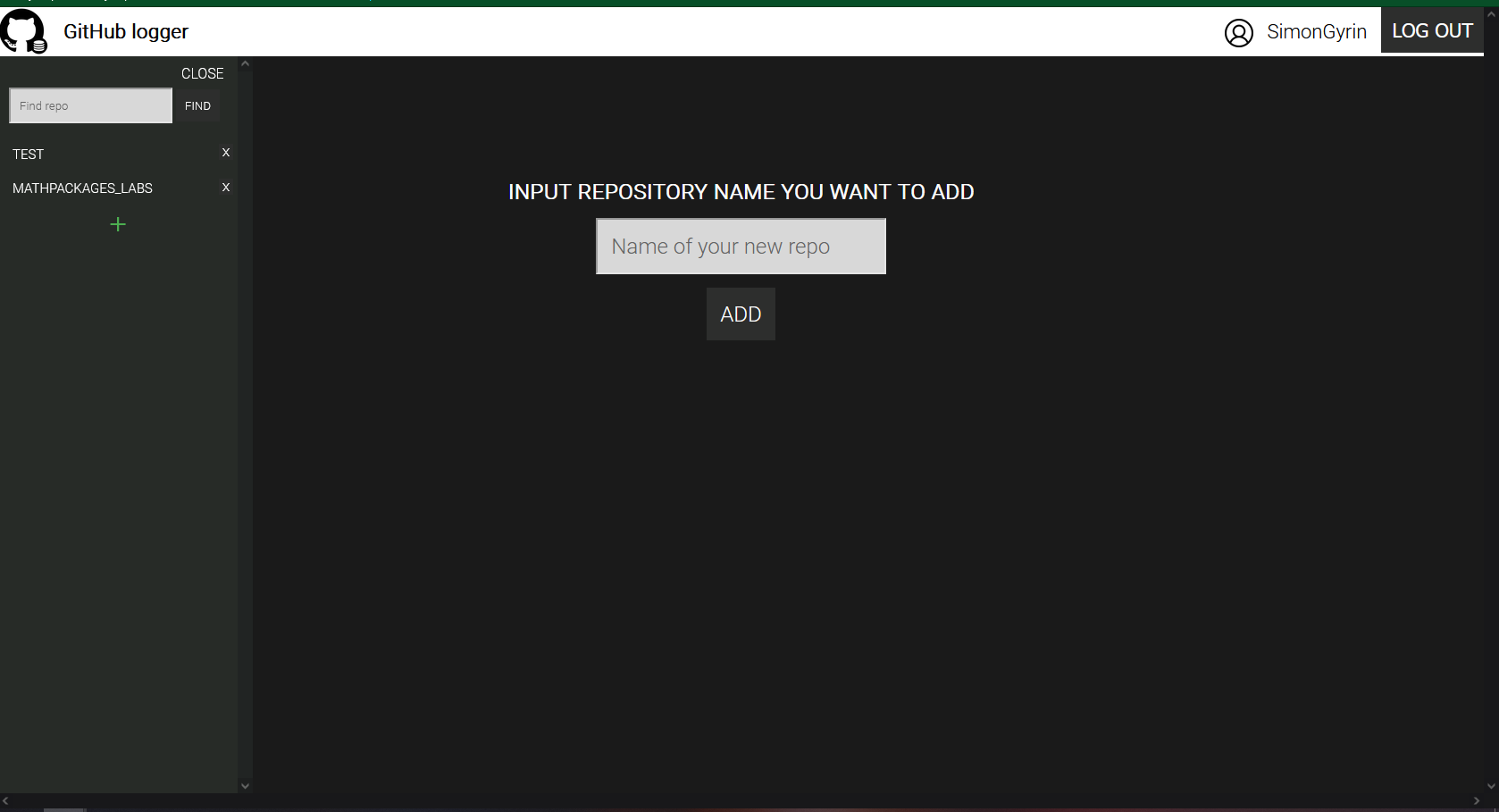
Главный экран



Экран выбора репозитория



Экран статистики коммитов в выбранном репозитории



Экран добавления репозитория

**Использованные технологии**

СУБД: MongoDB

Backend: pymongo, Flask

Frontend: HTML, js, jquery

**Ссылка на приложение**https://github.com/moevm/nosql2h22-github

**ВЫВОД**

В ходе выполнения задачи было реализовано web-приложение просмотра статистики git репозиториев. Были изучены методы обращения с нереляционными СУБД в частности с MongoDB.