

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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные базы данных»
Тема: Телеграм-бот для напоминаний преподавателям о новых записях в
таблице.

Студент гр. 0382		Санников В.А.
Студент гр. 0382	_____	Шангичев В.А.
Студент гр. 0382	_____	Азаров М.С.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург
2023

ЗАДАНИЕ

Студенты

Санников В.А.

Шангичев В.А.

Азаров М.С.

Группа 0382

Тема проекта: Телеграмм-бот для напоминаний преподавателям о новых записях в таблице.

Исходные данные: Необходимо реализовать телеграм-бота для напоминаний преподавателям о новых записях в таблице и веб-интерфейс для администрирования.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Заключение»

«Приложения»

«Литература»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 20.09.2023

Дата сдачи реферата: 00.12.2023

Дата защиты реферата: 00.12.2023

Студент гр. 0382		Санников В.А.
Студент гр. 0382		Шангичев В.А
Студент гр. 0382		Азаров М.С.
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса была поставлена задача разработать приложение, включающее в себя 2 функциональные части: телеграм-бот и веб-интерфейс для администрирования. Веб-интерфейс имеет следующий функционал: работа с данными бд, сервис с авторизацией и подобие личного кабинета для каждого администратора. В данной пояснительной записке присутствует только одна часть приложения - веб-интерфейс.

SUMMARY

As part of this course, the task was to develop an application that includes 2 functional parts: a telegram bot and a web interface for administration. The web interface has the following functionality: working with database data, a service with authorization and a kind of personal account for each administrator. This explanatory note contains only one part of the application - the web interface.

СОДЕРЖАНИЕ

	Задание	2
	Аннотация	4
1.	Введение	7
2.	Сценарии использования	8
2.1.	Макет UI	8
2.2.	Описание сценариев использования	8
2.2.1.	Добавление таблицы	8
2.2.2.	Добавление преподавателя	9
2.2.3.	Редактирование преподавателя	9
2.2.4.	Редактирование таблицы	9
2.2.5.	Редактирование профиля	9
2.2.6.	Массовый импорт/экспорт	9
2.2.7.	Поиск по дате/названию	10
3.	Модель данных	11
3.1.	Нереляционная модель данных	11
3.1.1.	Графическое представление	11
3.1.2.	Описание назначений коллекций, типов данных и сущностей	12
3.1.3.	Оценка объема информации, хранимой в модели	12
3.1.4.	Примеры запросов	13
3.2.	Реляционная модель данных	16
3.2.1.	Графическое представление	16
3.2.2.	Описание назначений типов данных и сущностей	16
3.2.3.	Оценка объема информации, хранимой в модели	17
3.2.4.	Примеры запросов	18
3.3.	Сравнение моделей	19
3.3.1.	Удельный объем информации	19
3.3.2.	Запросы по отдельным юзкейсам	20

3.3.3.	Вывод	20
4.	Разработанное приложение	21
4.1.	Описание приложения	21
4.2.	Использованные технологии	22
4.3.	Снимки экрана приложения	22
5.	Выводы	27
5.1.	Достигнутые результаты	27
5.2.	Недостатки и пути для улучшения полученного решения	27
5.3.	Будущее развитие решения	27
	Литература	28
	Приложения	29

1. ВВЕДЕНИЕ

Цель работы - разработка приложения для напоминаний преподавателям о новых записях в таблице. Приложение должно состоять из двух частей: телеграмм-бота и веб-интерфейса.

Командой было принято решение разработать веб-приложение с возможностью просматривать, добавлять, осуществлять поиск преподавателей, таблиц и логов. Также была реализована авторизация через jwtToken и подобие личного кабинета для каждого админа.

Данное приложение помогает преподавателям следить за обновлениями в таблицах, а админам регулировать работу приложения.

Качественные требования к решению: требуется разработать приложение с использованием СУБД MongoDB и реализовать развертывание приложения через docker-compose.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

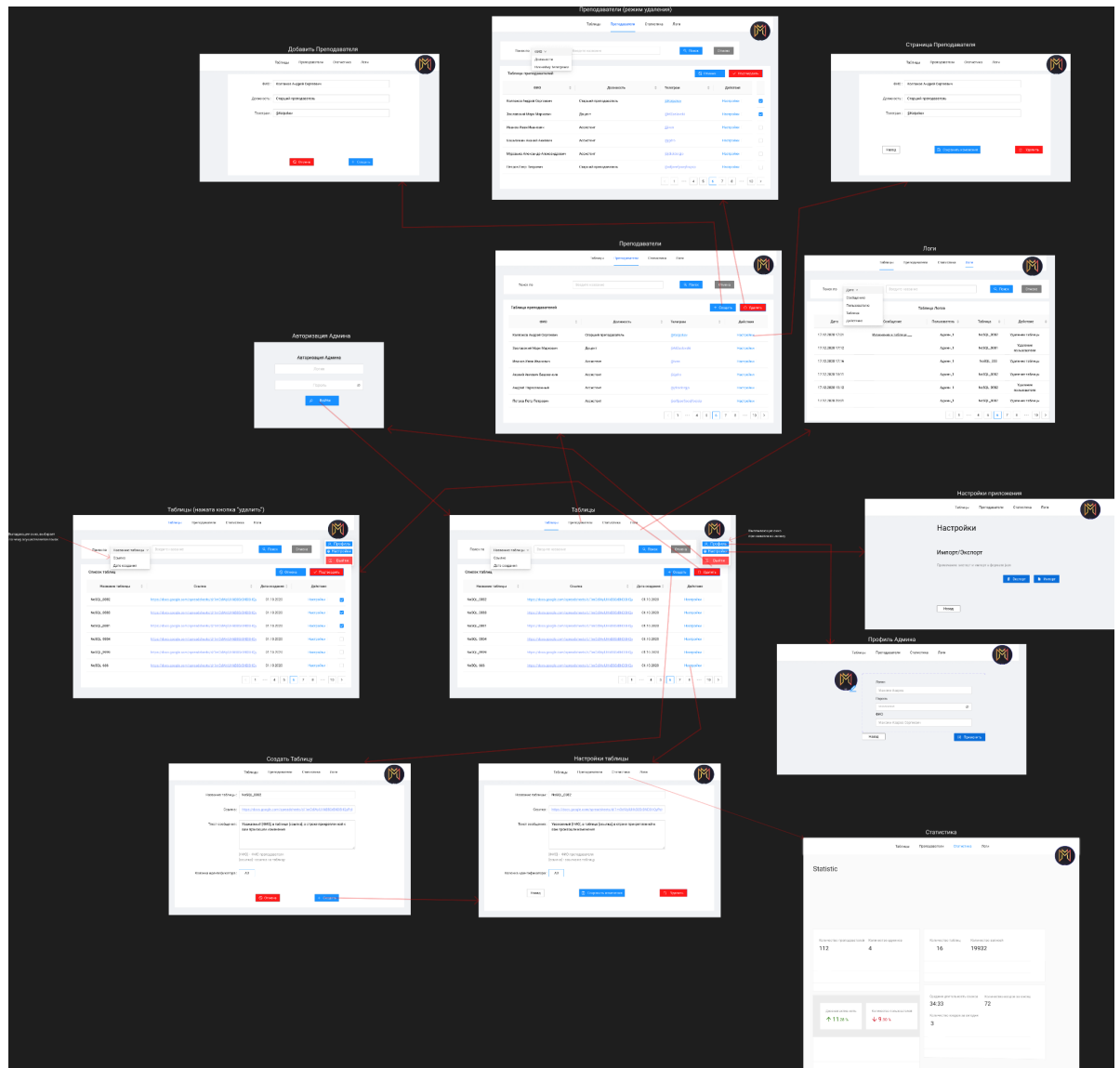


Рисунок 1 - Макет UI.

2.2. Описание сценариев использования

Во всех сценариях действующее лицо - Администратор.

2.2.1 Добавление таблицы

- При авторизации админ заходит на страницу таблиц
- Нажимает кнопку “Создать”
- Вводит данные и нажимает “Добавить”

2.2.2 Добавление преподавателя

- При авторизации админ заходит на страницу таблиц
- Далее нажимает на панели сверху “Преподаватели”
- В открывшемся окне нажимает кнопку “Создать”
- Вводит данные и нажимает “Добавить”

2.2.3 Редактирование преподавателя

- При авторизации админ заходит на страницу таблиц
- Далее нажимает на панели сверху “Преподаватели”
- В открывшемся окне нажимает кнопку “Настройки” в любой строке с преподавателем
- Вводит данные и нажимает “Сохранить”

2.2.4 Редактирование таблицы

- При авторизации админ заходит на страницу таблиц
- Нажимает кнопку “Настройки” в любой строке с таблицей
- Вводит данные и нажимает “Сохранить”

2.2.5 Редактирование профиля

- При авторизации админ заходит на страницу таблиц
- Нажимает кнопку на иконку своей фотографии в правом верхнем углу
- Во всплывающем списке нажимает кнопку “Профиль”
- Вводит данные и нажимает “Применить”

2.2.6 Массовый импорт/экспорт

- При авторизации админ заходит на страницу таблиц
- Нажимает кнопку на иконку своей фотографии в правом верхнем углу
- Во всплывающем списке нажимает кнопку “Настройки”

- Выбирает импорт и файл типа json для импорта/ выбирает экспорт и возможность скачать файл себе локально

2.2.7 Поиск по дате/названию

- При авторизации админ заходит на страницу таблиц
- Вводит интервал для поиска даты/поле для поиска по названию
- Нажимает кнопку “Поиск”
- При возможности может сбросить строки поиска на кнопку “Отмена”

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных

3.1.1. Графическое представление

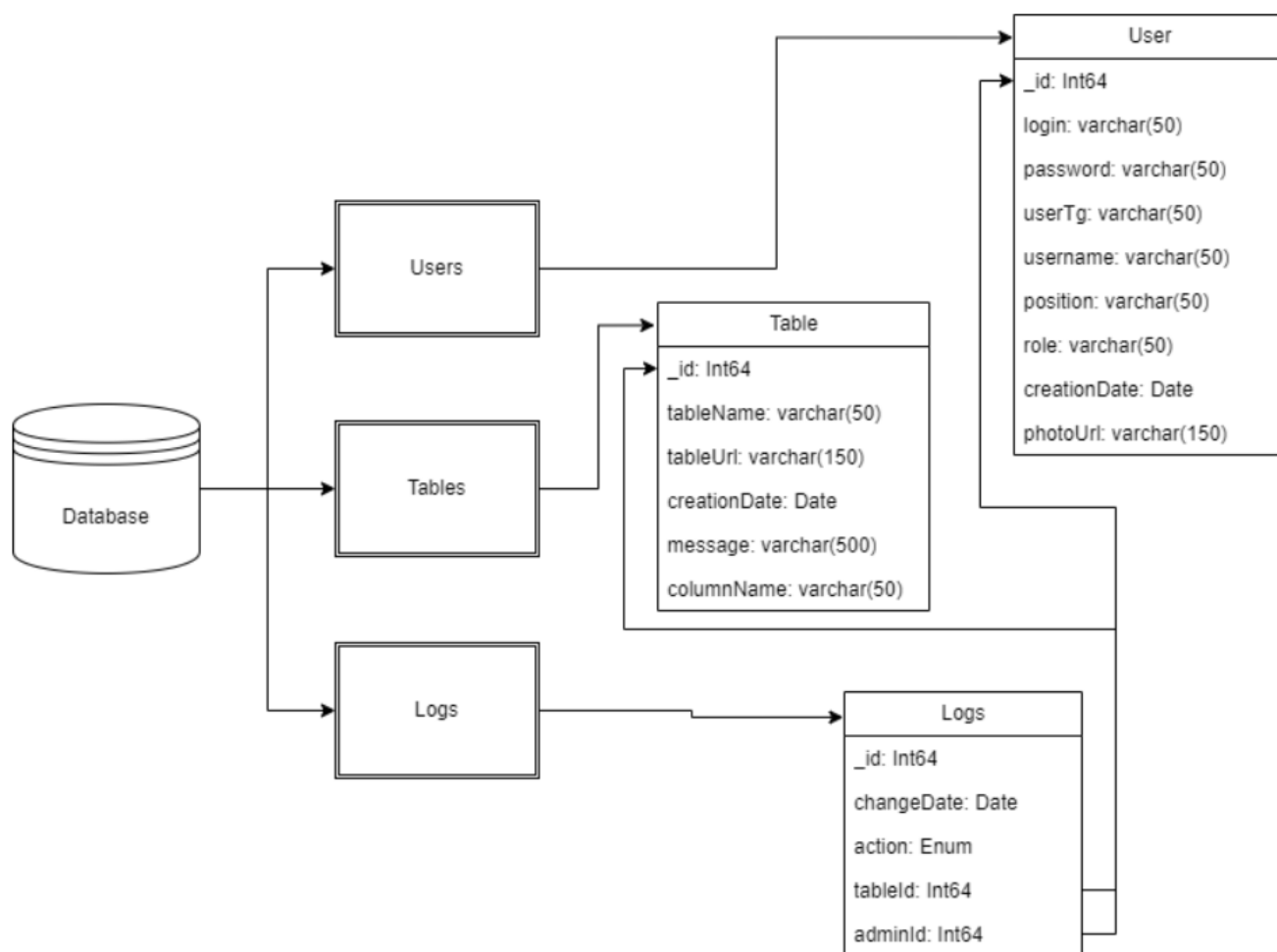


Рисунок 2 - Модель нереляционной БД

3.1.2. Описание модели

Коллекция "User" - соответствует сущности админа и преподавателя:

- _id - уникальный идентификатор
- login - логин админа (null, если роль учителя)
- password - пароль админа (null, если роль учителя)
- photoUrl - ссылка на фото в облаке (null, если роль учителя)
- userTg - телеграмм преподавателя (null, если роль админа)
- position - должность (null, если роль админа)

- username - ФИО преподавателя (null, если роль админа)
- role - роль пользователя (Админ, Преподаватель)
- creationDate - дата регистрации пользователя в систему

Коллекция "Logs":

- _id- уникальный идентификатор
- changeDate - время лога
- message - сообщение лога
- userId - _id админа, которым были произведены изменения (в зависимости от лога , может быть NULL)
- tableId - _id таблицы , в которой произошли изменения (в зависимости от лога , может быть NULL)
- action - enum (перечисление), обозначающее действие

Коллекция "Table" - соответствует сущности google таблиц :

- _id- уникальный идентификатор
- tableName - имя таблицы
- tableUrl - ссылка на таблицу
- creationData - дата создания таблицы
- message - текст уведомления в телеграме
- columnName - номер колонки в которой должны быть идентификаторы

3.1.3. Оценка объема информации, хранимой в модели

Рассчитаем объем каждой сущности:

- User: $8 + 50 * 8 = 408$ байт
- Table: $8 + 50 + 150 + 8 + 500 + 50 = 768$ байт
- Logs: $8 + 8 + 1 + 8 + 8 = 33$ байт.

Будем считать, что **один из пяти** пользователей является админом.
Рассчитаем объем хранимых данных для преподавателя и админа:

В случае добавления нового админа:

- необходимо сохранить сущность User: 408 байт
- действия админа будут сохраняться (в течении определенного времени). Положим, что в базу логов до удаления будут добавляться дополнительно пять записей, т. е. 5 сущностей Logs: $5 * 33 = 165$ байт

Итого: $408 + 165 = 573$ байт

В случае добавления нового преподавателя:

- необходимо сохранить сущность User: 408 байт
- Пусть в среднем преподаватель ведет два предмета. Для учета сдачи работ потребуются две сущности типа Table: $2 * 768 = 1536$ байт

Итого: $408 + 1536 = 1944$ байт

Т. к. один из пяти пользователей является админом:

$$V(N) = N * (0.2 * 573 + 0.8 * 1944) = 1669.8 * N$$

3.1.4. Примеры запросов

Запрос на добавление Teacher:

```
db.Users.insertOne({  
    teacherTg: "@teacher",  
    degree: "Старший преподаватель",  
    teacherName: "Иванов Иван Иванович"  
})
```

Запрос на добавление Table:

```
db.Tables.insertOne({
    tableName: "Сдача лабораторных работ по
МатАнализу, группа 0382",
    tableUrl: "https://googledocs/...",
    creationDate: "03.03.2023",
    message: "Доброго времени суток! В таблице
https://googledocs/... были сделаны изменения.",
    columnName: "A1",
})
```

Запрос на добавление Admin:

```
db.Users.insertOne({
    login: "admin",
    password: "123",
    photoUrl: "https://photo",
})
```

Запрос на добавление Logs:

```
db.Logs.insertOne({
    changeDate: "03.03.2023:15:14",
    action: "Админ admin добавил пользователя Иванов
И. И. в список преподавателей.",
    tableId: "234343",
    adminId: "88839483",
})
```

Поиск логов по действию:

```
db.Logs.find(
```

```
        {action: "Админ admin добавил пользователя  
Иванов И. И. в список преподавателей."}  
    )
```

Поиск логов по админу:

```
db.Logs.find(  
    {adminId: "88839483"}  
)
```

Поиск логов по админу и действию:

```
db.Logs.find(  
    {action: "Админ admin добавил пользователя  
Иванов И. И. в список преподавателей.",  
    adminId: "88839483"}  
)
```

Поиск таблиц по названию:

```
db.Tables.find(  
    {tableName: "Сдача лабораторных работ по  
МатАнализу, группа 0382"}  
)
```

Поиск таблиц по ссылке:

```
db.Tables.find(  
    {tableUrl: "https://googledocs/..."}  
)
```

3.2. Реляционная модель данных

3.2.1. Графическое представление

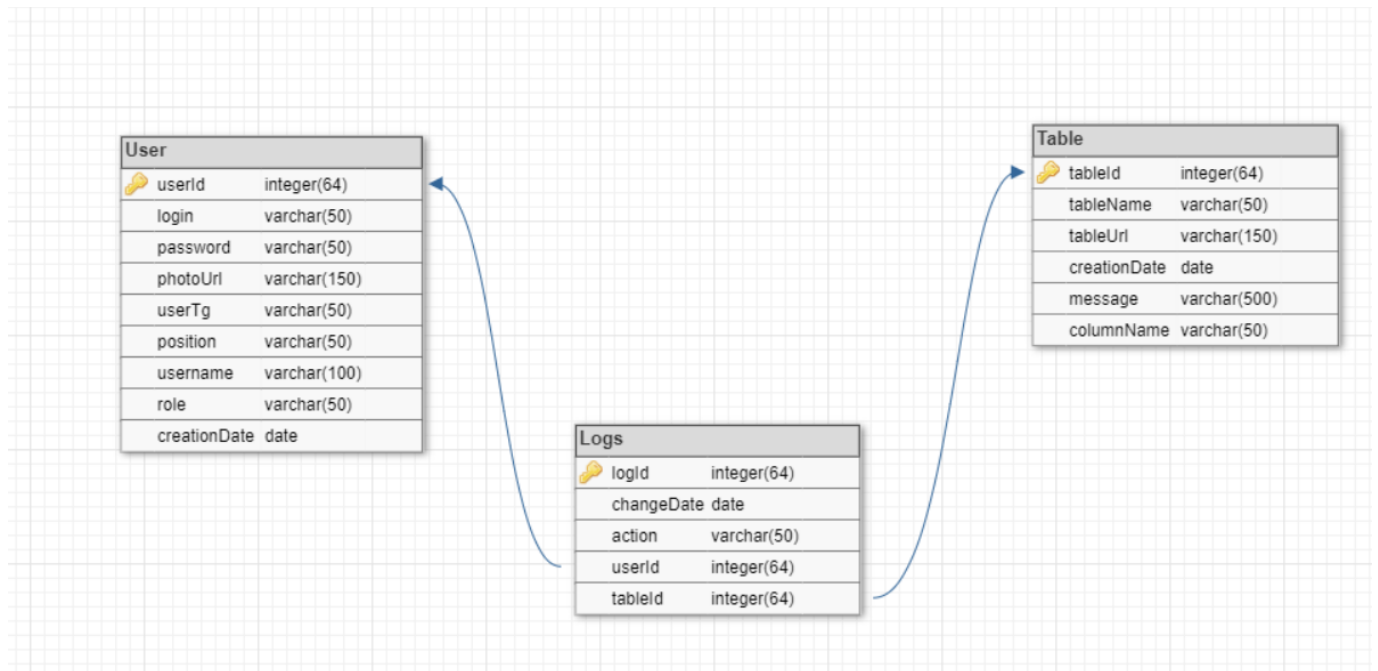


Рисунок 3 - Модель реляционной БД

3.2.2. Описание назначений типов данных и сущностей

Таблица "User":

- userId - уникальный идентификатор
- login - логин админа (null, если роль учителя)
- password - пароль админа (null, если роль учителя)
- photoUrl - ссылка на фото в облаке (null, если роль учителя)
- userTg - телеграмм преподавателя (null, если роль админа)
- position - должность (null, если роль админа)
- username - ФИО преподавателя (null, если роль админа)
- role - роль пользователя (Админ, Преподаватель)
- creationDate - дата регистрации пользователя в систему

Таблица "Logs":

- logId - уникальный идентификатор

- `changeDate` - время лога
- `action` - содержание лога
- `userId` - каким админом были произведены изменения (в зависимости от лога , может быть NULL, связь один ко многим)
- `tableId` - в какой таблице произошли изменения (в зависимости от лога , может быть NULL, связь один ко многим)

Таблица "Table":

- `tableId` - уникальный идентификатор
- `tableName` - имя таблицы
- `tableUrl` - ссылка на таблицу
- `creationData` - дата создания таблицы
- `message` - текст уведомления в телеграмме
- `columnName` - номер колонки в которой должны быть идентификаторы

3.2.3. Оценка объема информации, хранимой в модели

Таблица "User":

- `userId`: V=8b
- `login`: V=200b
- `password`: V=200b
- `photoUrl`: V=400b
- `creationDate`: V=3b
- `userTg`: V=200b
- `position`: V=200b
- `username`: V=400b
- `role`: V=200b

Таблица "Logs":

- logId: V=8b
- changeDate: V=3b
- action: V=200b
- adminId: V=8b
- tableId: V=8b

Таблица "Table":

- tableId: V=8b
- tableName: V=200b
- tableUrl: V=600b
- creationData: V=3b
- message: V=2000b
- columnName: V=200b

Тогда объем данных в БД: $V=1811*Nu + 227*Nl + 3011*Ntb$

3.2.4. Примеры запросов

Запрос на добавление преподавателя в User:

```
INSERT teacher(userTg, position1, username, role)
VALUES (userTg1, position1, username1, role1);
```

Запрос на добавление Table:

```
INSERT table(tableName, tableUrl, creationData,
message, columnName)
VALUES (tableName1, tableUrl1, creationData1,
message1, columnName1);
```

Запрос на добавление админа в User:

```
INSERT admin(login, password, photoUrl, role)
VALUES (login1, password1, photoUrl1, role1);
```

Запрос на добавление Logs:

```
INSERT logs(changeDate, action, userId, tableId)
VALUES (changeDate1, action1, userId1, tableId1);
```

Поиск логов по действию:

```
SELECT * FROM LOGS WHERE action = action1;
```

Поиск логов по админу:

```
SELECT * FROM LOGS WHERE userId = id;
```

Поиск логов по админу и действию:

```
SELECT * FROM LOGS WHERE userId = id AND action =
action1;
```

Поиск таблиц по названию:

```
SELECT * FROM LOGS WHERE tableName = tableName1;
```

3.3. Сравнение моделей

3.3.1. Удельный объем информации

В связи с тем что архитектурном плане модели реляционной и нереляционной бд очень схожи , достаточно сравнить удельное значение каждого объекта.

В нереляционной :

- User: 408 байт
- Table: 768 байт
- Logs: 33 байт.

В реляционной :

- User: 1811 байт

- Table: 3011 байт
- Logs: 227 байт.

Сравнивая данные значения можем сделать вывод, что реляционная модель занимает значительно больше памяти, при одинаковом количестве содержащихся объектов в бд.

3.3.2. Запросы по отдельным юзкейсам

Количество запросов для обеих моделей во всех юзкейсах совпадает и равно 1, для всех юзкейсов описанных в сценарии.

Хотя все же небольшое различие есть. Если мы рассмотрим запрос, не входящий в сценарий (но возможно который, будет использоваться в рабочем приложении). Например: получение данных “пользователя” или “гугл таблицы” соответствующих конкретному “логу”. То для такого случая нереляционной модели понадобится 2 запроса , а реляционной только 1.

3.3.3. Вывод

Если рассматривать по кол-ву запросов, то не имеет особой разницы использование реляционной бд или нереляционной модели бд. Так происходит в связи с тем , что в нереляционной модели связи были реализованы с помощью id документов в других коллекциях , а не с помощью денормализации (в противном случае память занимаемая нереляционной моделью была намного больше). Заметим , что если бы функционал бд (описанный в сценарии) включал в себя использование связей между объектом Log и Admin то кол-во запросов для нереляционной модели было больше чем реляционной.

Если рассматривать по кол-ву занимаемой памяти то нереляционная бд занимает меньше места . Скорее всего это связано с особенностями реализации хранения элементарных типов в каждой бд.

Таким образом можно сделать вывод, что для данного проекта лучше использовать нереляционную бд для хранения данных.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Описание приложения

Веб-интерфейс представляет собой приложение, состоящее из двух частей: back-end и front-end.

Back-end написан на fastapi Python 3.11, который представляет собой асинхронный REST API сервис для просмотра, изменения, редактирования данных БД. В данном API была реализована авторизация через jwtToken (для запросов к БД проверяется header запроса, для получения страницы проверяется cookie).

Для Front-end использовался шаблонизатор Jinja, jQuery для запросов к API и html. Реализован через Server-Side Rendering (пользователь получает уже готовую страницу со всеми данными и стилями, когда делает запрос внутри приложения).

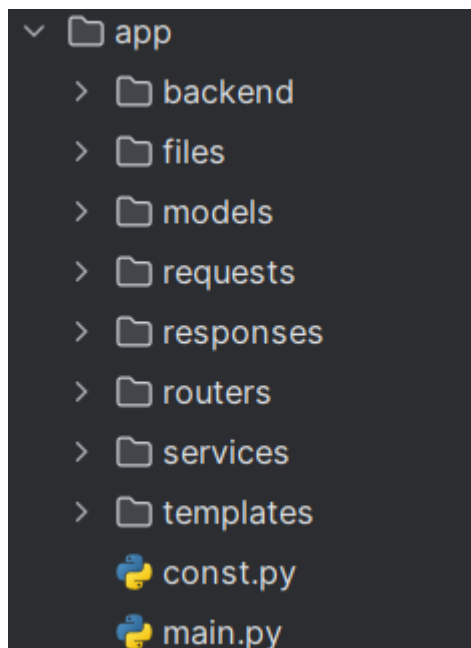


Рисунок 4 - Архитектура веб-приложения.

Как видно из рис.4, архитектура состоит из корневой папки app, в которой располагаются важные модули. Пример выполнения запроса будет выглядеть следующим образом: приходит запрос по эндпоинту и необходимый роутер его

обрабатывает routers, далее идет работа с телом запроса в services, далее сервис запрашивает данные из БД (если необходимо) backend/bd.py.

Шаблоны хранятся в папке templates, responses/requests - это модели для ответа/запроса соответственно, files для экспортированных данных БД, models - это entity БД, точка входа в приложение - main.py.

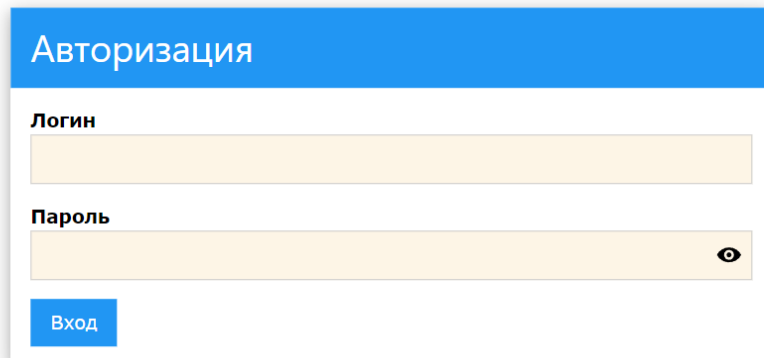
4.2. Используемые технологии

БД: MongoDB (pymongo)

Back-end: fastapi, Python 3.11

Front-end: Jinja2Templates, html

4.3. Снимки экрана приложения



Авторизация

Логин

Пароль

Вход

Рисунок 5 - Окно авторизации.

Таблицы

Преподаватели

Статистика

Логи

Поиск

Название таблицы:

Введите данные

Дата создания (после):

Введите данные

Поиск

Отмена

Ссылка:

Введите данные

Дата создания (до):

Введите данные

Список таблиц

Создать

Удалить

Название таблицы	Ссылка	Дата создания	Действия
ИДЗ 099	https://docs.google.com/spreadsheets/d/1mOdWylUHK88SrBND8HQyPck2po8QYd...	23.04.2012	Настроить
ИДЗ 666	https://docs.google.com/spreadsheets/d/1mOdWylUHK88Shsidgdodigdiodyhido...	23.04.2012	Настроить

<

1

2

>

Рисунок 6 - Окно таблиц.

Таблицы

Преподаватели

Статистика

Логи

Редактирование таблицы

Название таблицы

ИДЗ 099

Ссылка

https://docs.google.com/spreadsheets/d/1mOdWylUHK88SrBND8HQyPck2po8QYdHQG_27Rvh2jU/edit?r

Текст уведомления

Произошли изменения в таблице "ИДЗ 0999"

Колонка идентификатора

A3

Сохранить

Рисунок 7 - Окно редактирования таблицы.

Таблицы

Преподаватели

Статистика

Логи

Поиск

ФИО:

Введите данные

Telegram ник:

Введите данные

Поиск

Отмена

Должность:

Введите данные

Таблица преподавателей

Создать

Удалить

ФИО	Должность	Telegram	Действия
Максим Максимыч	ректоратник	@tg_max	Настроить
Вадим Александрович	Преподаватель ИТ	@arofeoz	Настроить

<

1

2

>

Рисунок 8 - Окно преподавателей.

Таблицы

Преподаватели

Статистика

Логи

Редактирование преподавателя

ФИО преподавателя

Максим Максимыч

Должность

ректоратник

Телеграмм

@tg_max

Сохранить

Рисунок 9 - Окно редактирования преподавателя.

Таблицы

Преподаватели

Статистика

Логи

Поиск

Таблицы:

Введите данные

Действие:

Введите данные

Дата создания (после):

Введите данные

Пользователь:

Введите данные

Сообщение:

Введите данные

Дата создания (до):

Введите данные

Поиск

Отмена

Таблица Логов

Дата	Сообщение	Действие	Таблица	Пользователь
23.04.2012	Была добавлена таблица апофис	Добавление таблицы	ИДЗ 099	-
23.04.2012	Была удалена таблица с учителями	Удаление таблицы	ИДЗ 099	vasia

«

1

2

3

»

Рисунок 10 - Окно логов.

Таблицы

Преподаватели

Статистика

Логи

Добавление новой таблицы

Название таблицы

Введите название новой таблицы...

Ссылка

Введите ссылку на новую таблицу...

Текст уведомления

Введите текст уведомления для преподавателя, связанного с новой таблицей...

Колонка идентификатора

Введите идентификатор колонки с отслеживаемыми изменениями...

Добавить

Рисунок 11 - Окно добавления таблицы.

The screenshot shows a web application interface with a blue header bar containing navigation links: 'Таблицы', 'Преподаватели', 'Статистика', and 'Логи'. On the right side of the header is a user profile icon. The main content area is titled 'Добавление нового преподавателя'. It contains three input fields: 'ФИО преподавателя' with placeholder text 'Введите ФИО нового преподавателя...', 'Должность' with placeholder text 'Введите должность нового преподавателя...', and 'Telegram' with placeholder text 'Введите текст уведомления для преподавателя, связанного с новой таблицей'. Below these fields is a blue button labeled 'Добавить'.

Рисунок 12 - Окно добавления преподавателя.

The screenshot shows a web application interface with a blue header bar containing navigation links: 'Таблицы', 'Преподаватели', 'Статистика', and 'Логи'. On the right side of the header is a user profile icon. The main content area displays a modal window titled 'Профиль админа'. It contains three input fields: 'Логин' with the value 'vasia', 'Новый пароль' (password field with an eye icon), and 'ФИО' with the value 'Василий Иванович Пупкин'. At the bottom of the modal are two buttons: 'Назад' and 'Применить'.

Рисунок 13 - Профиль админа.

The screenshot shows a web application interface with a blue header bar containing navigation links: 'Таблицы', 'Преподаватели', 'Статистика', and 'Логи'. On the right side of the header is a user profile icon. The main content area is titled 'Импорт/Экспорт'. Below the title is a note: 'Примечание: экспорт и импорт в формате json'. At the bottom are two buttons: 'Импорт' and 'Экспорт'.

Рисунок 14 - Окно импорта/экспорта.

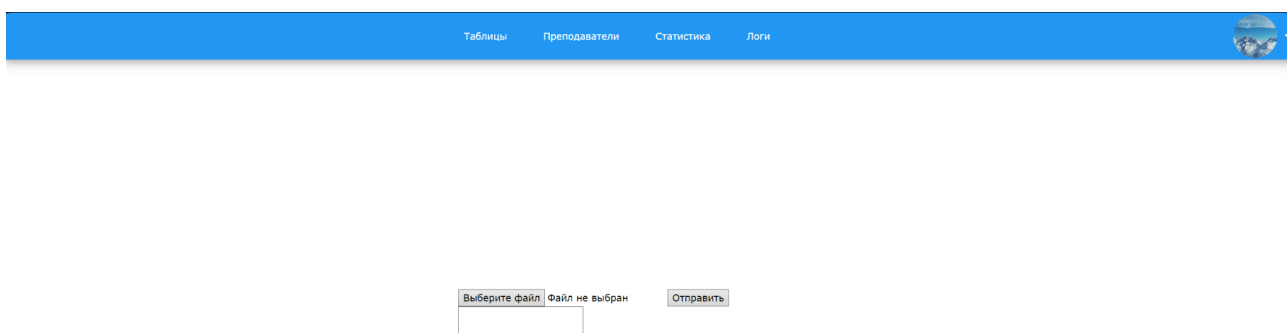


Рисунок 15 - Окно выбора файла для импорта.

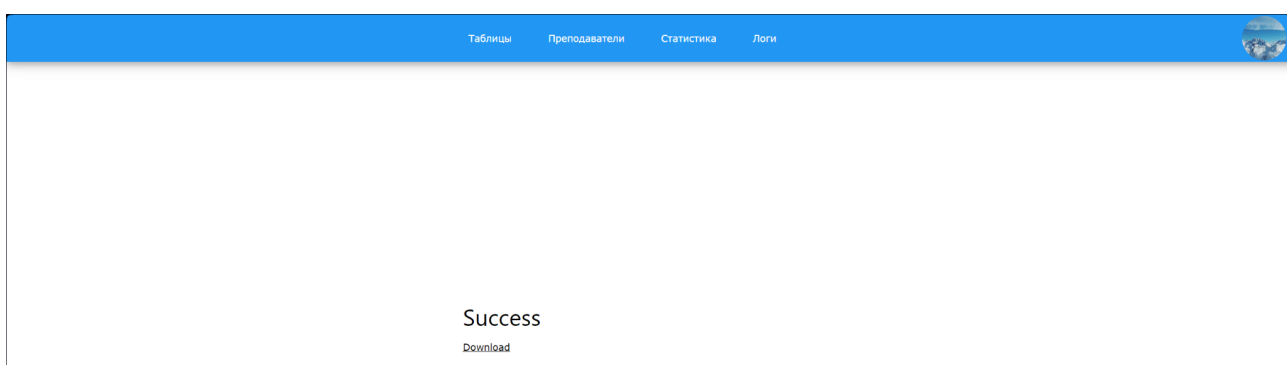


Рисунок 16 - Окно успешного экспорта и возможность скачать файл.

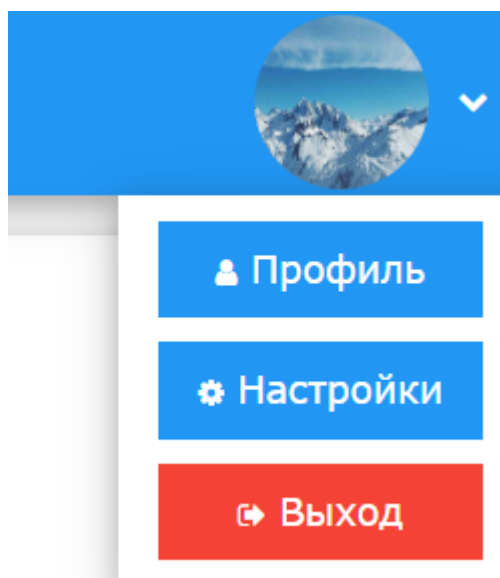


Рисунок 17 - Всплывающее меню.

5. ВЫВОДЫ

5.1. Достигнутые результаты

В ходе работы над проектом был создан веб-сайт администратора, содержащий инструменты для ведения учета преподавателей, управления списком таблиц с информацией о сдаче студентами контрольных заданий по предмету, аффилированному с преподавателем(ями). Также была создана страница с записями о действиях администраторов (“логи”), относящихся к добавлению/удалению/редактированию данных проекта. Кроме того, пользователь веб-сайта обладает возможностью импорта/экспорта данных в JSON-формате. Была проведена тщательная работа по выбору и реализации стилей: каждая веб-страница предстает перед пользователем в изящной комбинации цветов российского триколора.

5.2. Недостатки и пути для улучшения полученного решения

Полученное решение имеет ряд недостатков.

Во-первых, формы добавления/удаления/редактирования таблиц/преподавателей не проходят валидации, соразмерной особенностям проекта: в поле ссылки на гугл таблицу может быть введена ссылка на фишинговый сайт или на таблицу, содержащую компрометирующую информацию. Поля должности, ФИО, телеграмм-тега также не проходят проверок на корректность.

Вторым направлением для улучшения проекта может стать добавление возможности для более гибкого контроля рассылок уведомлений. В текущей версии проекта выбирается единственная версия уведомления на одну таблицу при изменениях в колонке с указанным идентификатором.

5.3. Будущее развитие решения

Следующим этапом в развитии проекта является реализация рассылки уведомлений преподавателям при изменениях в таблице с помощью GoogleTablesAPI. Затем, после работы над вышеуказанными недостатками, в качестве основной задачи по улучшению решения можно выбрать создание страницы с подробной статистикой по данным проекта.

ЛИТЕРАТУРА

1. Ссылка на github проекта: <https://github.com/moevm/nosql2h23-tg-notify>
2. Документация MongoDB: <https://www.mongodb.com/docs/>
3. Документация по работе с MongoDB на Python:
<https://pymongo.readthedocs.io/en/stable/>
4. Документация fastapi: <https://fastapi.tiangolo.com>
5. Документация Docker: <https://docs.docker.com>

ПРИЛОЖЕНИЯ

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

1. Склонировать репозиторий
2. Зайти в папку app
3. Заполнить .env файл
4. Ввести команду `docker-compose up -d`
5. Зайти по ссылке в браузере: <http://127.0.0.1:8000/page/auth> (порты могут быть другие)