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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Нереляционные базы данных»

Тема: Сервис редактирования и составления расписания в ВУЗе

Студентки гр. 0382 _____ Деткова А.С.

_____ Рубежова Н.А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

ЗАДАНИЕ

Студентки
Деткова А.С.
Рубежова Н.А.
Группа 0382

Тема работы: Сервис редактирования и составления расписания в ВУЗе.

Исходные данные:
Задача – сервис, позволяющий принимать пожелания от преподавателей, принимать данные об аудиториях и их вместительности, данные учебных планов (предметы / курсы / группы), данные о студентах (группы, количество человек в них, направления, к которым относятся группы). Необходимые фичи – автоматическое составление расписания, визуализация и редактирование, логика согласования расписания (преподаватель высказал пожелание, администратор поправил, преподаватель одобрил), анализ составленного расписания. Пользователи – администраторы, преподаватели, студенты.

Содержание пояснительной записки:
«Содержание»
«Введение»
«Качественные требования к решению»
«Сценарии использования»
«Модель данных»
«Разработка приложения»
«Заключение»
Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студентки гр. 0382	_____	Деткова А.С.
	_____	Рубежова Н.А.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках курса «Нереляционные базы данных» предполагалась разработка веб-приложения в команде на одну из тем с использованием нереляционной базы данных. Было предложено несколько тем с разными БД. Была выбрана тема разработки сервиса автоматического составления и редактирования расписания в ВУЗе с использование СУБД MongoDB. Найти исходный код, дополнительную информацию, Wiki-страницы и информацию по сборке можно в репозитории проекта: https://github.com/moevm/nosql2h23-schedule.

SUMMARY

As part of the course "Non-relational databases", it was supposed to develop a web application in a team on one of the topics using a non-relational database. Several topics with different databases were proposed. The topic of developing a service for automatic scheduling and editing at the university using the MongoDB database was chosen. You can find the source code, additional information, Wiki pages and assembly information in the project repository: https://github.com/moevm/nosql2h23-schedule.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

a. Актуальность решаемой проблемы

В сети Интернет стремительно растет объем данных, которые хранятся и обрабатывается веб-приложениями, а также растет число пользователей информационных приложений и сервисов в сети Интернет. Такое явление создает проблему эффективного представления информации, обеспечивающего быстрый доступ и надёжность хранения, целостность данных. Следовательно, реляционные базы данных не справляются с нагрузками, актуальными в настоящее время. Большинство разработчиков нуждаются в поиске новых путей для хранения и масштабирования огромных массивов данных. Это обуславливает актуальность построения и применения моделей представления данных, отличных от реляционных.

b. Постановка задачи

Тема: Сервис редактирования и составления расписания в ВУЗе.

Задача: Сервис, позволяющий принимать пожелания от преподавателей, принимать данные об аудиториях и их вместительности, данные учебных планов (предметы / курсы / группы), данные о студентах (группы, количество человек в них, направления, к которым относятся группы). Необходимые фичи – автоматическое составление расписания, визуализация и редактирование, логика согласования расписания (преподаватель высказал пожелание, администратор поправил, преподаватель одобрил), анализ составленного расписания. Пользователи – администраторы, преподаватели, студенты.

c. Предлагаемое решение

В качестве решения предлагается написание веб-приложения. Серверная часть которого будет написана на технологии Java Spring Boot 2.7.24, СУБД – MongoDB, клиентская часть – Vue.js + TypeScript.

d. Качественные требования к решению

Требуется разработать веб-приложение, в качестве СУБД в котором будет использована MongoDB.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

a. Макет UI

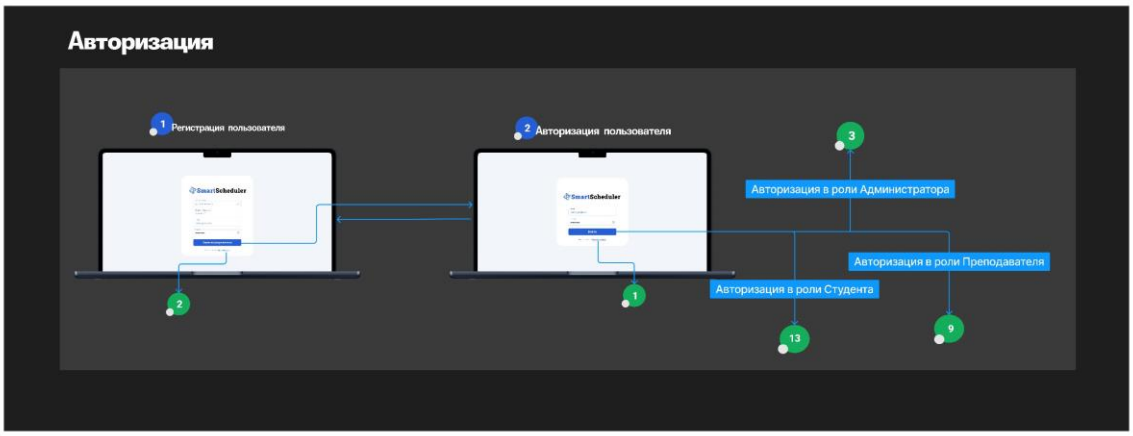


Рисунок 1 – Экраны авторизации и регистрации

Рисунок 2 – Экраны для пользователя с ролью Администратор

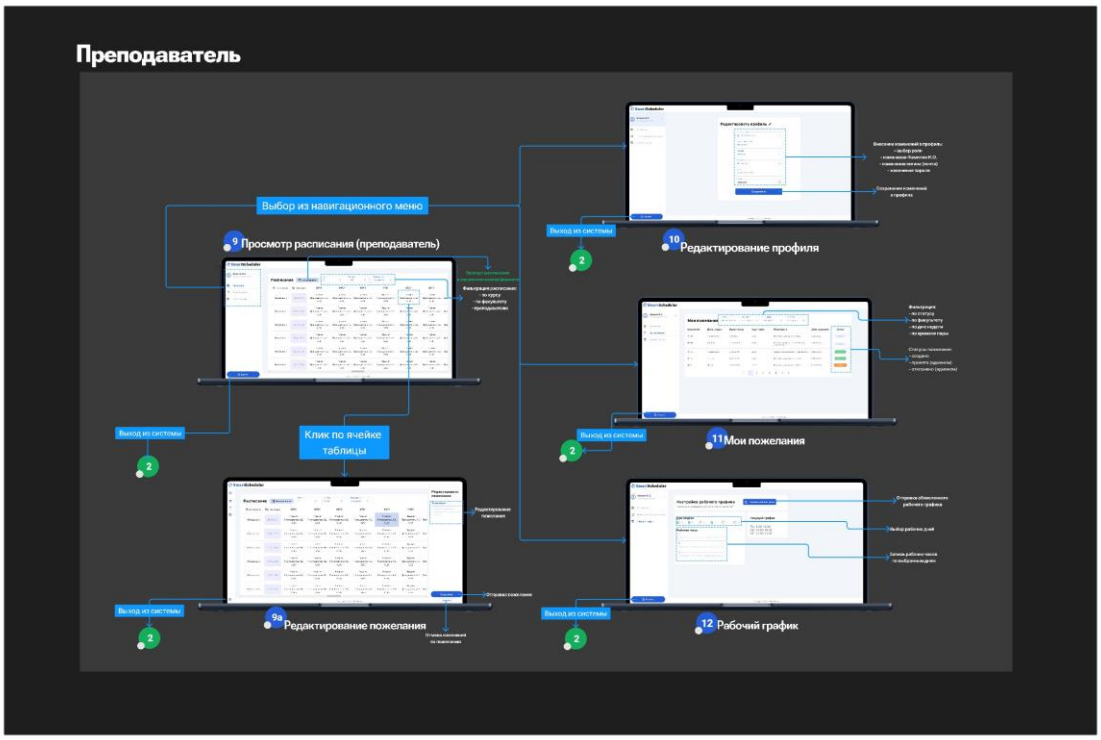


Рисунок 3 – Экраны для пользователей с ролью Преподаватель

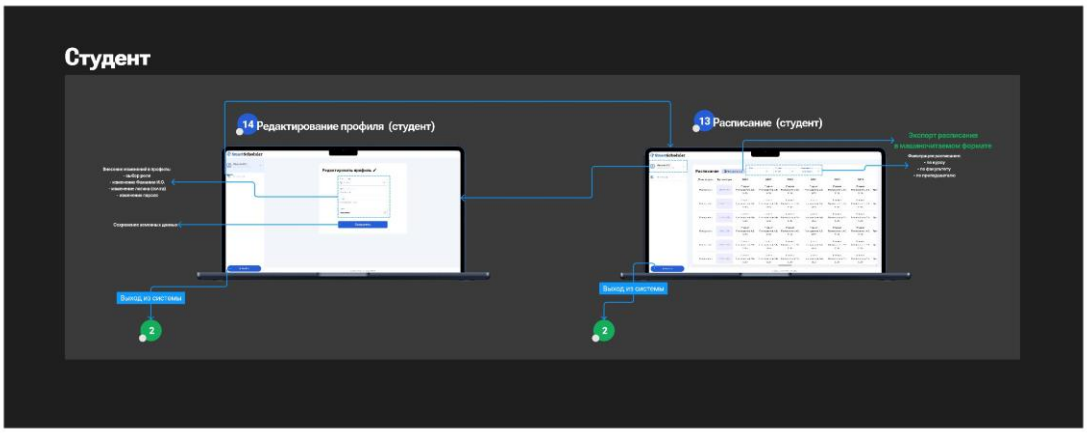


Рисунок 4 – Экраны для пользователей с ролью Студент

b. Сценарии использования приложения

<u>Сценарий использования – «Просмотр расписания».</u>

Действующее лицо: Пользователь, Администратор или Преподаватель.

Сценарий:

1. Лицо проходит авторизацию. *Ответ системы*: Загрузка навигационной панели с открытой вкладкой "Расписание".

2. Лицо выбирает курс и факультет *или* Лицо выбирает курс, факультет и преподавателя. *Ответ системы*: Вывод расписания согласно выбранным фильтрам.

Результат: Лицо изучает информацию о расписании.

Альтернатива:

1. Если расписание еще не составлено, то отображается "Расписание пока не составлено :(".
2. Лицо может перейти на доступные вкладки в навигационной панели.

Сценарий использования - "Загрузка учебного плана".

Действующее лицо: Администратор.

Сценарий:

1. Лицо авторизовывается. *Ответ системы:* Загрузка навигационной панели с открытой вкладкой "Расписание".
2. Лицо выбирает в левом меню вкладку "Загрузка учебного плана". *Ответ системы:* Загрузка вкладки "Загрузка учебного расписания".
3. Лицо добавляет выбирает факультет, прикрепляет файл расписания и жмет кнопку "Обновить уч. план". *Ответ системы:* Отправляет файл расписания на сервер и обновляет "Историю загрузок" на вкладке.

Результат: Загружены учебные планы.

Альтернатива:

1. Лицо может перейти на другую вкладку навигационной панели.
2. В случае ошибки при загрузке лицу будет сообщено о проблеме.

Сценарий использования - "Отправка пожеланий".

Действующее лицо: Преподаватель.

Сценарий:

1. Лицо авторизовывается. *Ответ системы:* Загрузка навигационной панели с открытой вкладкой "Расписание".
2. Лицо выбирает курс и факультет или Лицо выбирает курс, факультет и преподавателя. *Ответ системы:* Вывод расписания согласно выбранным фильтрам.
3. Лицо нажимает на ячейку, где он указан преподавателем. *Ответ системы:* Отображение панели редактирования справа.
4. Лицо записывает текст с пожеланием по смене аудитории / времени / дню недели / преподавателю / предмету и нажимает кнопку "Сохранить". *Ответ системы:* Отправляет запрос на сервер, где создает заявку по пожеланию.

Результат: Создает заявку по пожеланию преподавателя по изменению ячейки расписания.

Альтернатива:

1. Если расписание еще не составлено, то отображается "Расписание пока не составлено :(".
2. Отображение информации о невозможности создания заявки.
3. На шаге 4 возможно нажатие "Отменить", что закроет панель редактирования справа.
4. Возможно нажатие кнопки "Мои пожелания" на навигационной панели.
5. Возможно нажатие кнопки "Рабочий график" на навигационной панели.
6. Возможно нажатие кнопки "Выход" на навигационной панели.

Сценарий использования - "Обработка пожеланий".

Действующее лицо: Администратор.

Сценарий:

1. Лицо авторизовывается. *Ответ системы:* Загрузка навигационной панели с открытой вкладкой "Расписание".

2. Лицо дублирует текущую вкладку. *Ответ системы:* Открывается новое окно с навигационной панелью с открытой вкладкой "Расписание".

3. Лицо нажимает на кнопку "Пожелания преподавателей" на навигационной панели. *Ответ системы:* Загрузка вкладки "Пожелания преподавателей".

4. Лицо запоминает пожелание преподавателя, возвращается к окну с открытой вкладкой "Расписание", выбирает нужный курс, факультет (и если нужно - преподавателя). *Ответ системы:* Вывод расписания согласно выбранным фильтрам.

5. Лицо нажимает на нужную ячейку. *Ответ системы:* Отображение панели редактирования справа.

6. Лицо изменяет параметры ячейки согласно пожеланию преподавателя и жмет кнопку "Сохранить". *Ответ системы:* Открытие диалогового окна с кнопками "Удовлетворить" и "Отклонить".

7.1. Лицо жмет кнопку "Удовлетворить". *Ответ системы:* Отправляет запрос на сервер, где вносит изменение в расписание и меняет статут заявки преподавателя. Закрытие панели редактирования справа.

7.2. Лицо жмет кнопку "Отклонить". *Ответ системы:* Отправляет запрос на сервер, где меняет стату заявки преподавателя. Закрытие панели редактирования справа.

Результат: Изменение статуса заявки по пожеланию преподавателя по изменению ячейки расписания и, возможно, внесение изменений в расписание.

Альтернатива:

1. Если расписание еще не составлено, то на вкладке "Расписание" отображается "Расписание пока не составлено :(".

2. Если заявок от преподавателей нет, то на вкладке "Пожелание преподавателей" отображается "Пожеланий нет :)".

3. На шаге 6 возможно нажатие "Отменить", что закроет панель редактирования справа.

4. Возможно нажатие кнопки "Загрузка уч. плана" на навигационной панели.

5. Возможно нажатие кнопки "Выход" на навигационной панели.

Сценарий использования - "Изменение рабочего графика".

Действующее лицо: Преподаватель.

Сценарий:

1. Лицо авторизовывается. *Ответ системы:* Загрузка навигационной панели с открытой вкладкой "Расписание".

2. Лицо нажимает на кнопку "Рабочий график" на навигационной панели. *Ответ системы:* Загрузка вкладки "Рабочий график".

3. Лицо выбирает рабочие дни, указывает в соответствующие поля рабочие часы и жмет кнопку "Обновить раб. график". *Ответ системы:* отправляет запрос на сервер с новым рабочий графиком.

Результат: Изменение рабочего графика преподавателя.

Альтернатива:

1. Отображение информации о невозможности изменения рабочего графика.

2. Лицо может выбрать другую вкладку на навигационной панели, что отменит выбранные изменения.

3. Возможно нажатие кнопки "Мои пожелания" на навигационной панели.

4. Возможно нажатие кнопки "Расписание" на навигационной панели.

5. Возможно нажатие кнопки "Выход" на навигационной панели.

3. МОДЕЛЬ ДАННЫХ

a. Нереляционная модель данных (MongoDB)

- **Графическое представление**

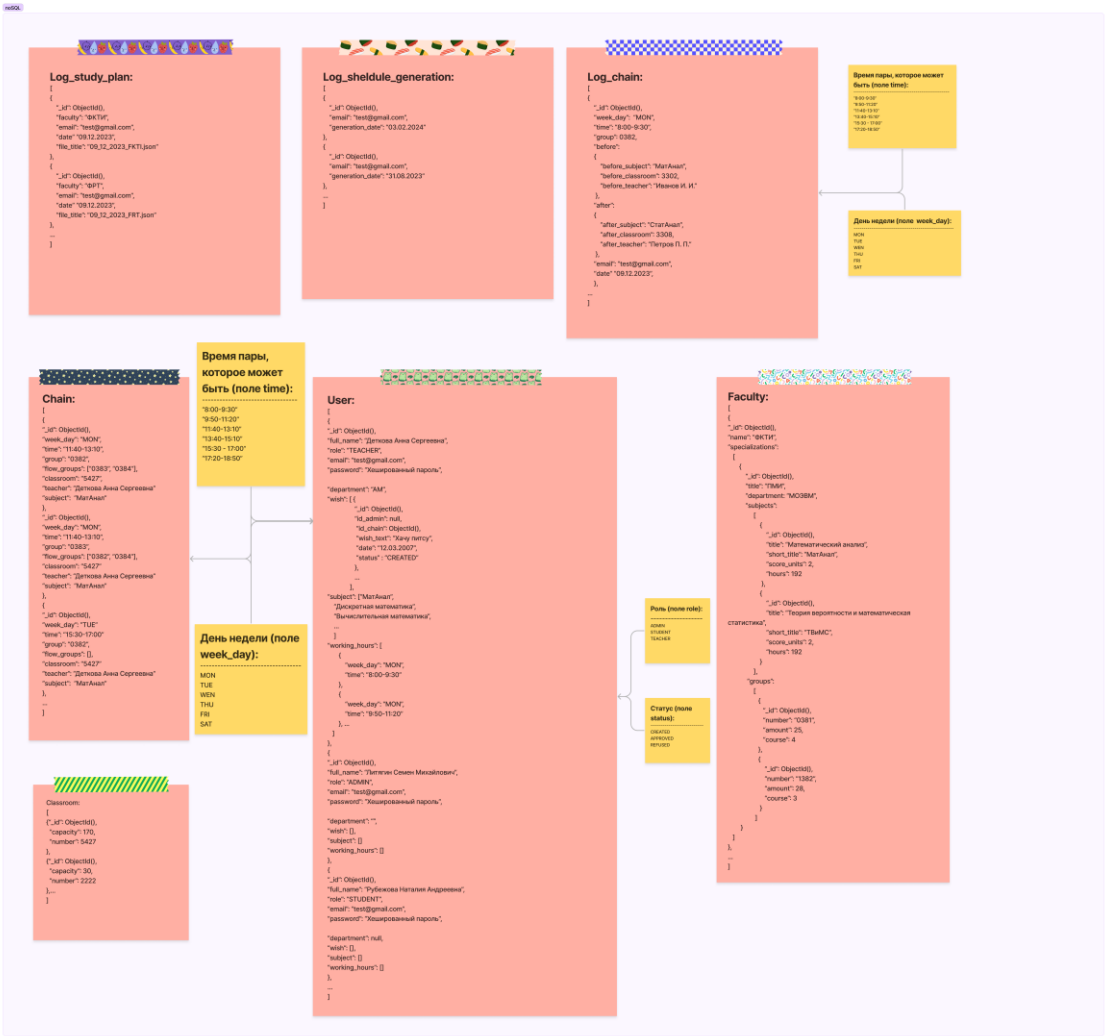


Рисунок 5 – Графическое представление нереляционной базы данных

- **Описание назначений коллекций, типов данных, сущностей**

База данных содержит 7 коллекций:

Коллекция "User":

Хранит информацию о пользователях, у каждого пользователя может быть одна из трех ролей (ROLE_ADMIN, ROLE_TEACHER, ROLE_STUDENT). Если пользователь является преподавателем, то для него есть доп. поля, которые не являются пустыми, у остальных типов информации там нет.

- _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
- full_name - полное имя пользователя ФИО **V = 100*2=200 байт**, тип String.
- role - роль пользователя (Админ - ADMIN, Студент - STUDENT, Преподаватель - TEACHER), **V=2*7=14 байт**, тип String.
- email - почта пользователя (используется как логин для входа в систему) **V = 50*2=100 байт**, тип String.
- password - пароль пользователя для входа в личный кабинет **V = 20 байт**, тип String.
- department - кафедра, на который работе преподаватель, это поле непустое, когда роль Юзера преподаватель **V=2*10=20 байт**, тип String.
- wish - пожелания преподавателя, это поле непустое, когда роль Юзера преподаватель. Представляет собой массив документов, тип Array(ObjectWish). **V=146Nw=146*4=584 байт**, Nw~4 (примерно 4 пожеланий для одного преподавателя). Каждое пожелание - это объект, у которого есть своя структура:
  - _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
  - id_admin - Id администратора, который обрабатывает пожелание преподавателя **V=12 байт**, может быть null, если статус пожелания - создано, тип ObjectId().
  - id_chain - Id ячейки таблицы расписания, к которой создано пожелание **V=12 байт**, тип ObjectId().
  - wish_text - сам текст пожелания, например, «Изменить аудиторию с 5427 на 5425» **V=2*50=100 байт**, тип String.

- date - дата создания пожелания **V=8 байт**, тип Date().*
- status - статус пожелания. Может быть Создано (CREATED), Удовлетворено (APPROVED), Отклонено (REFUSED) **V=8*2=16 байт**, тип String.

Итого: **146 байт**

- subject - предметы, которые ведет преподаватель, это поле непустое, когда роль Юзера преподаватель, представляет собой список строк **V=(20*2)Nsub=20*2*3=120 байт**, Nsub~3 (примерно 3 предмета, которые ведет один преподаватель), тип Array(String).
- working_hours - рабочие часы преподавателя, это поле непустое, когда роль Юзера преподаватель, представляет собой список объектов - пара в определенное время и день недели, преподаватель выбирает именно пары, которые он сможет вести, каждый объект состоит из двух полей: week_day (день недели, может быть пн - MON, вт - TUE, ср- WEN, чт- THU, пт - FRI, сб - SAT, тип String, **V=2*3=6 байт**) и time (время пары, может быть 1ая пара - “8:00-9:30”, 2ая пара - “9:50-11:20”, 3я пара - “11:40-13:10”, 4ая пара - “13:40-15:10”, 5ая пара - “15:30 - 17:00”, 6ая пара - “17:20-18:50”, тип String, **V=2*11=22 байт**), всего **V=28Nwh=28*10=280 байт**, Nwh~10 (преподаватель за неделю ведет по 10 пар - оценка сверху), тип Array(ObjectWorkingHours).

Всего для одного объекта: **1098 байт**.

Коллекция "Chain":

Объект для ячейки расписания, ячейка характеризуется днем недели, временем пары и группой - строки и столбцы таблицы.

- _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
- week_day - день недели, может быть пн - MON, вт - TUE, ср- WEN, чт- THU, пт - FRI, сб - SAT, тип String, **V=2*3=6 байт**.

- time - время пары, может быть 1ая пара - “8:00-9:30”, 2ая пара - “9:50-11:20”, 3я пара - “11:40-13:10”, 4ая пара - “13:40-15:10”, 5ая пара - “15:30 - 17:00”, 6ая пара - “17:20-18:50”, тип String, **V=2*11=22 байт**.
- group - группа, пара которой проходит, **V=4*2=8 байт**, тип String.
- flow_groups - группы, с которыми данная пара является потоковой, представляет собой массив, **V=8Nfl=8*6=48 байт**, Nfl~6 (6 групп может быть на потоковой паре), тип Array(String).
- classroom - аудитория, где проходит пара, **V=4*2=8 байт**, тип String.
- teacher - ФИО преподавателя, **V = 100*2=200 байт**, тип String.
- subject - предмет (короткое название, например, ТВиМС), **V=20*2=40 байт**, тип String.

Всего для одного объекта: **240 байт**.

Коллекция "Faculty":

Объект описывает учебный план на семестр для факультета, внутри содержится информация о направлениях факультета, для каждого наравления есть информация о группах и учебных предметах.

- _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
- name - краткое название факультета (ФКТИ, ФРТ и так далее), **V=4*2=8 байт**, тип String.
- specializations - специальности факультета (например, ПМИ и ПИ), является списком объектов - специальность. **V=1218Nsp=1218*10=12180 байт**, Nsp~10 (10 специальностей на факультете). Тип Array(ObjectSpesialization). Каждый объект имеет следующую структуру:
    - _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
    - title - краткое название специальности (ПМИ, ПИ, РСиК и так далее), **V=4*2=8 байт**, тип String.

- department - кафедра, на который обучается группа (могут быть ситуации, когда на разных кафедрах есть направления с одним названием), **V=2*10=20 байт**, тип String.
- subjects - предметы, изучаемые студентами на данной специальности в текущем семестре, является списком объектов - предмет. **V=136Nsubj=136*8=1088 байт**, Nsubj~8 (8 предметов изучается в семестре). Тип Array(ObjectSubject). Каждый объект имеет следующую структуру:
  - _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
  - title - название предмета (Теория вероятности и математическая статистика), **V=50*2=100 байт**, тип String.
  - short_title - краткое название предмета (ТВиМС), **V=10*2=20 байт**, тип String.
  - score_units - зачетные единицы по предметы (2), **V=2 байт**, тип Integer.
  - hours - количество часов на изучение предмета в семестре (192), **V=2 байт**, тип Integer.
    Итого: **136 байт**
- groups - группы, которые находятся на данном направлении, является списком объектов - группа. **V=22Ngr=22*5=110 байт**, Ngr~5 (5 групп на данном направлении). Тип Array(ObjectGroup). Каждый объект имеет следующую структуру:
  - _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
  - number - номер группы, **V=4*2=8 байт**, тип String.
  - amount - количество студентов в группе, **V=2 байт**, тип Integer.

- course - курс, на котором находится группа, **V=2 байт**, тип Integer.

  Итого: **22 байт**

Итого: **1218 байт**

Всего для одного объекта: **12200 байт**.

Коллекция "Classroom":

Объект описывает одну аудиторию в университете.

- _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
- capacity - вместительность аудитории (170), **V=2 байт**, тип Integer.
- number - номер аудитории (5427), **V=2 байт**, тип Integer.

Коллекция "Log_sheldule_generation":

Объект логгирует генерацию расписания.

- _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
- email - email Админа, сгенерировавшего расписание, **V = 50*2=100 байт**, тип String.
- generation_date - дата генерации расписания, **V=8 байт**, тип Date().

  Всего для одного объекта: **120 байт**.

Коллекция "Log_study_plan":

Объект логгирует загрузку учебных планов на семестр.

- _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
- faculty - название факультета, для которого загружается учебный план, **V=4*2=8 байт**, тип String.
- email - email Админа, загрузившего учебный план, **V = 50*2=100 байт**, тип String.
- date - дата загрузки учебного плана, **V=8 байт**, тип Date().
- file_title - имя файла с учебным планом, **V=50*2=100 байт**, тип String.

Всего для одного объекта: **228 байт**.

Коллекция "Log_chain":

Объект логгирует изменения ячеек с расписанием.

- _id - уникальный идентификатор документа **V=12 байт**, тип ObjectId().
- week_day - день недели, может быть пн - MON, вт - TUE, ср- WEN, чт- THU, пт - FRI, сб - SAT, тип String, **V=2*3=6 байт**.
- time - время пары, может быть 1ая пара - “8:00-9:30”, 2ая пара - “9:50-11:20”, 3я пара - “11:40-13:10”, 4ая пара - “13:40-15:10”, 5ая пара - “15:30 - 17:00”, 6ая пара - “17:20-18:50”, тип String, **V=2*11=22 байт**.
- group - группа, пару которой редактируем, **V=4*2=8 байт**, тип String.
- before - информация о том, какая пара, в какой аудитории и с каким преподавателем были до внесения изменений, **V=360 байт**, хранит в себе объект с тремя полями:
    - before_subject - пара, которая была до изменений, **V=60*2=120 байт**, тип String.
    - before_classroom - аудитория, в которой была назначена пара до изменений, **V=60*2=120 байт**, тип String.
    - before_teacher - преподаватель, которой был назначен до изменений, **V=60*2=120 байт**, тип String.
      Итого: 360 байт. **V=200*2=400 байт**, тип String.
- after - информаця о том, какая пара, в какой аудитории и с каким преподавателем стали после внесения изменений, **V=360 байт**, хранит в себе объект с тремя полями:
    - after_subject - пара, которая стала после изменений, **V=60*2=120 байт**, тип String.
    - after_classroom - аудитория, в которой стала назначена пара после изменений, **V=60*2=120 байт**, тип String.

- after_teacher - преподаватель, которой стал назначен после изменений, **V=60*2=120 байт**, тип String.
  Итого: 360 байт. **V=200*2=400 байт**, тип String.

- email - email Админа, изменившего расписание, **V = 50*2=100 байт**, тип String.
- date - дата изменения ячейки, **V=8 байт**, тип Date().

  Всего для одного объекта: **932 байт**.

- **Оценка удельного объема информации, хранимой в модели**

Посчитаем общий объем информации для N_u - количество пользователей, N_c - количество ячеек расписания, N_f - количество факультетов, N_{lsp} - количество логов учебного плана, N_{lsg} - количество логов генерации расписания, N_{lc} - количество логов изменения ячеек таблицы:

$V(N_u, N_c, N_f, N_{lsp}, N_{lsg}, N_{lc}) = 1098N_u + 240N_c + 12200N_f + 120N_{lsg} + 228\ N_{lsg} + 932N_{lc}$.

Выразим объем информации, хранимой в модели, через переменную N - количество пользователей, среди которых в среднем 1/5 - это преподаватели. Также в университете ЛЭТИ имеется 8 факультетов, в среднем на каждом факультете по 50 групп, значит количество ячеек с расписанием = 50 * 15 = 750 (у каждой группы на неделе по ~15 пар), количество логов - по 100 на каждую коллекцию:

V(N) = 1098N + 240*750 + 12200*8 + 120*100 + 228*100 + 932*100 = 1098N + 405600.

- **Запросы к модели, с помощью которых реализуются сценарии использования**

1. Поиск юзера при авторизации:

```
db.User.findOne({email:email, password: password})
```

Была использована 1 коллекция. Количество запросов - 1.

2. Просмотр страницы с расписанием (с фильтрами 4 курс и ФКТИ):

```
db.Chain.aggregate([
  // Соединяем с коллекцией Faculty по полю group
  {
    $lookup: {
      from: "Faculty",
      localField: "group",
      foreignField: "specializations.groups.number",
      as: "faculty"
    }
  },
  // Разворачиваем массив faculty
  { $unwind: "$faculty" },

 // Фильтруем по курсу и факультету
 {
    $match: {
      "faculty.specializations.groups.course": 4,
      "faculty.name": "ФКТИ"
    }
 },
 // Проекция для выбора нужных полей
 {
    $project: {
      week_day: 1,
      time: 1,
      group: 1,
      flow_groups: 1,
      classroom: 1,
      teacher: 1,
      subject: 1
    }
  }
]);
```

Было использовано 2 коллекции. Количество запросов - 1 (поиск списка групп для данных фильтров, вывод ячеек расписания для полученных групп).

3. Загрузка учебного плана:

```
db.Faculty.insertOne({"name": "ФКТИ", "specializations": [{"title": "ПМИ", "subjects": [{"title": "Математический анализ", "short_title": "МатАнал", "score_units": 2, "hours": 192}, {"title": "Теория вероятности и математическая статистика", "short_title": "ТВиМС", "score_units": 2, "hours": 192}], "groups": [{"number": "0381", "amount": 25, "course": 4}, {"number": "1382", "amount": 28, "course": 3}]})
```

Была использована 1 коллекция. Количество запросов - 1.

4. Получить пожелания преподавателей:

```
db.User.aggregate([
    // Фильтрация пользователей с ролью Админ (2)
    { $match: { role: 2 } },
```

```
        // Разворачиваем массив wish
        { $unwind: "$wish" },

        // Соединяем с коллекцией Chain по полю id_chain
        {
            $lookup: {
                from: "Chain",
                localField: "wish.id_chain",
                foreignField: "_id",
                as: "chain"
            }
        },

        // Соединяем с коллекцией User по полю id_admin
        {
            $lookup: {
                from: "User",
                localField: "wish.id_admin",
                foreignField: "_id",
                as: "admin_full_name"
            }
        },

        // Форматируем вывод, выбираем нужные поля
        {
            $project: {
                full_name: 1,
                wish: {
                    wish_text: "$wish.wish_text",
                    date: "$wish.date",
                    enum_status: "$wish.enum_status",
                    chain: {
                        $arrayElemAt: ["$chain", 0] // Получаем первый
элемент массива chain
                    },
                    admin_full_name: {
                        $arrayElemAt:    ["$admin_full_name",    0]    //
Получаем первый элемент массива admin
                    }
                }
            }
        }
    ]);
```

Было использовано 2 коллекции. Количество запросов - 1 (присоединение двух таблиц по id админа и chain, запрос на выборку пожеланий всех преподавателей).

5. Обработка пожелания (установка статуса 1 - удовлетворено для пожелания)

```
var yourwishId = ObjectId("yourWishId");
```

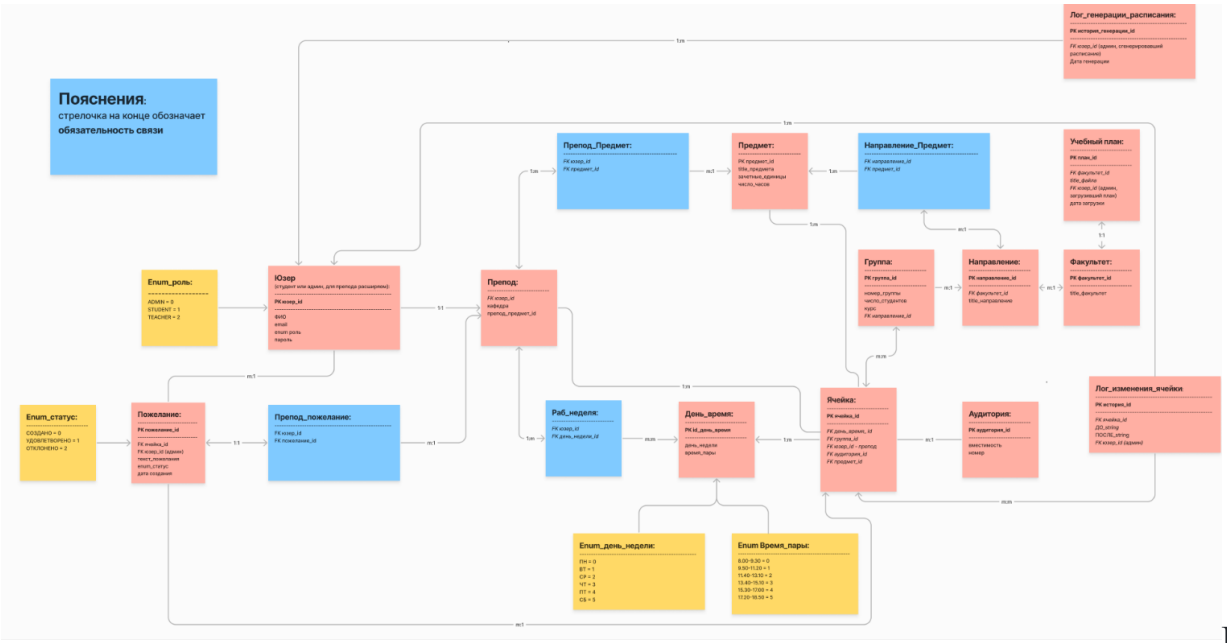
```
var yourUserId = db.User.findOne({email:email, password: password});

db.User.updateOne(
   { "wish._id": yourWishId },
   { $set: {"wish.$.enum_status": 1, "wish.$.id_admin": yourUserId }}
);
```

Было использовано 2 коллекции. Количество запросов - 2 (поиск юзера, обновление пожелания).

b. Аналог модели данных для SQL СУБД

- **Графическое представление**



Р

Рисунок 6 – Графическое представление реляционной базы данных

- **Описание назначений коллекций, типов данных, сущностей**

База данных содержит 17 таблиц:

Таблица "Юзер":

Общая информация о пользователе.

- PK юзер_id - уникальный идентификатор пользователя, первичный ключ, **V=12 байт**, тип BigInt().
- ФИО - полное имя пользователя ФИО **V = 100*2=200 байт**, тип String.
- enum роль - роль пользователя (Админ, Студент, Поеподаватель), представляет собой одно из значений Enum’а **V=2 байт**, тип Integer.

- email - почта пользователя (используется как логин для входа в систему) **V = 50*2=100 байт**, тип String.
- пароль - пароль пользователя для входа в личный кабинет **V = 20 байт**, тип String.

Всего для одного объекта: **V=334 байт**.

Таблица "Препод":

Общая информация о пользователе с ролью преподаватель (TEACHER = 2).

- FK юзер_id - уникальный идентификатор преподавателя, вторичный ключ, **V=12 байт**, тип BigInt().
- кафедра - кафедра, на который работе преподаватель, **V=2*10=20 байт**, тип String.
- препод_предмет_id - идентификатор из таблицы отношения преподаватель-предмет, **V=12 байт**, тип BigInt().

Всего для одного объекта: **V=44 байт**.

Таблица "Пожелание":

Пожелания преподавателя для ячеек таблицы расписания.

- PK пожелание_id - уникальный идентификатор пожелания, первичный ключ, **V=12 байт**, тип BigInt().
- FK ячейка_id - уникальный идентификатор ячейки расписания, вторичный ключ, **V=12 байт**, тип BigInt().
- FK юзер_id (админ) - уникальный идентификатор пользователя-администратора, вторичный ключ, **V=12 байт**, тип BigInt().
- текст_пожелания - сам текст пожелания, например, «Изменить аудиторию с 5427 на 5425» **V=2*50=100 байт**, тип String.
- enum_статус - статус пожелания, одно из чисел из Enum. Может быть Создано, Удовлетворено, Отклонено, **V=2 байт**, тип Integer.

- дата создания - дата создания пожелания **V=8 байт**, тип Date().

Всего для одного объекта: **V=146 байт**.

Таблица "Препод-пожелание":

Связь преподавателя и его пожеланий.

- FK юзер_id - уникальный идентификатор пользователя-преподавателя, вторичный ключ, **V=12 байт**, тип BigInt().
- FK пожелание_id - уникальный идентификатор пожелания, вторичный ключ, **V=12 байт**, тип BigInt().

Всего для одного объекта: **V=24 байт**.

Таблица "День-время":

Связь дня и времени пары (учавствует в ячейке расписания и при планировании рабочего графика преподавателя).

- PK id_день_время - уникальный идентификатор, первичный ключ, **V=12 байт**, тип BigInt().
- день_недели - день недели пары, одно из чисел из Enum, **V=2 байт**, тип Integer.
- время_пары - время проведения пары, одно из чисел из Enum, **V=2 байт**, тип Integer.

Всего для одного объекта: **V=16 байт**.

Таблица "Раб_неделя":

Связь преподавателя и его рабочих часов (время, когда преподаватель может вести пару). Рабочий график.

- FK юзер_id - уникальный идентификатор пользователя, вторичный ключ, V=12 байт, тип BigInt().
- FK день_недели_id - уникальный идентификатор, вторичный ключ, V=12 байт, тип BigInt().

Всего для одного объекта: **V=24 байт**.

Таблица "Предмет":

Описание учебного предмета.

- PK предмет_id - уникальный идентификатор предмета, первичный ключ, **V=12 байт**, тип BigInt().
- title_предмета - название предмета, **V=50*2=100 байт**, тип String.
- зачетные_единицы - зачетные единицы по предметы, **V=2 байт**, тип Integer.
- число_часов - количество часов на изучение предмета в семестре (192), **V=2 байт**, тип Integer.

Всего для одного объекта: **V=116 байт**.

Таблица "Препод-предмет":

Связь между преподавателем и предметами, которые он ведет.

- FK юзер_id - уникальный идентификатор пользователя, вторичный ключ, **V=12 байт**, тип BigInt().
- FK предмет_id - уникальный идентификатор предмета, вторичный ключ, **V=12 байт**, тип BigInt().

Всего для одного объекта: **V=24 байт**.

Таблица "Ячейка":

Описывает ячейку таблицы с расписанием.

- PK ячейка_id - уникальный идентификатор ячейки, первичный ключ, **V=12 байт**, тип BigInt().
- FK день_время_ id - уникальный идентификатор, вторичный ключ, **V=12 байт**, тип BigInt().
- FK группа_id - уникальный идентификатор группы, вторичный ключ, **V=12 байт**, тип BigInt().

- FK юзер_id - препод - уникальный идентификатор пользователя-преподавателя, вторичный ключ, **V=12 байт**, тип BigInt().
- FK аудитория_id - уникальный идентификатор аудитории, вторичный ключ, **V=12 байт**, тип BigInt().
- FK предмет_id - уникальный идентификатор предмета, вторичный ключ, **V=12 байт**, тип BigInt().

Всего для одного объекта: **V=72 байт**.

Таблица "Аудитория":

Описание аудиторий, доступных в университете.

- PK аудитория_id - уникальный идентификатор аудитории, первичный ключ, **V=12 байт**, тип BigInt().
- вместимость - количество человек, что может влезь в аудиторию, **V=2 байт**, тип Integer.
- номер - номер аудитории, **V=2 байт**, тип Integer.

Всего для одного объекта: **V=16 байт**.

Таблица "Группа":

Описание группы.

- PK группа_id - уникальный идентификатор группы, первичный ключ, **V=12 байт**, тип BigInt().
- номер_группы - номер, **V=2 байт**, тип Integer.
- число_студентов - количество человек в группе, **V=2 байт**, тип Integer.
- курс - курс обучения, **V=2 байт**, тип Integer.
- FK направление_id - уникальный идентификатор направления, вторичный ключ, **V=12 байт**, тип BigInt().

Всего для одного объекта: **V=30 байт**.

Таблица "Направление":

Описание направлений.

- PK направление_id - уникальный идентификатор направления, первичный ключ, **V=12 байт**, тип BigInt().
- FK факультет_id - уникальный идентификатор факультета, вторичный ключ, **V=12 байт**, тип BigInt().
- title_направление - краткое название специальности (ПМИ, ПИ, РСиК и так далее), **V=4*2=8 байт**, тип String.

Всего для одного объекта: **V=32 байт**.

Таблица "Направление_Предмет":

Связь направления и предмета.

FK направление_id - уникальный идентификатор направления, вторичный ключ, V=12 байт, тип BigInt().

FK предмет_id - уникальный идентификатор предмета, вторичный ключ, V=12 байт, тип BigInt().

Всего для одного объекта: V=24 байт.

Таблица "Факультет":

Описание факультета.

- PK факультет_id - уникальный идентификатор факультета, первичный ключ, **V=12 байт**, тип BigInt().
- title_факультет - краткое название факультета (ФКТИ, ФРТ и так далее), **V=4*2=8 байт**, тип String.

Всего для одного объекта: **V=20 байт**.

Таблица "Учебный план":

Описание всех учебных планов, загруженных в системе.

- PK план_id - уникальный идентификатор плана, первичный ключ, **V=12 байт**, тип BigInt().
- FK факультет_id - уникальный идентификатор факультета, вторичный ключ, **V=12 байт**, тип BigInt().
- title_файла - имя файла с учебным планом, **V=50*2=100 байт**, тип String.
- FK юзер_id (админ, загрузивший план) - уникальный идентификатор юзера, вторичный ключ, **V=12 байт**, тип BigInt().
- дата загрузки - дата загрузки учебного плана, **V=8 байт**, тип Date().

Всего для одного объекта: **V=144 байт**.

Таблица "Лог_изменения_ячейки":

Логирование об изменениях ячейки.

- PK история_id - уникальный идентификатор записи истории лога, первичный ключ, **V=12 байт**, тип BigInt().
- FK ячейка_id - уникальный идентификатор ячейки, к которой применялись изменения, вторичный ключ, **V=12 байт**, тип BigInt().
- ДО_string - информация в ячейке до изменений, **V=50*2=100 байт**, тип String.
- ПОСЛЕ_string - информация в ячейке после изменений, **V=50*2=100 байт**, тип String.
- FK юзер_id (админ) - уникальный идентификатор юзера, вторичный ключ, **V=12 байт**, тип BigInt().

Всего для одного объекта: **V=236 байт**.

Таблица "Лог_генерации_расписания":

Логирование генераций расписания.

- PK история_генерации_id - уникальный идентификатор записи истории лога, первичный ключ, **V=12 байт**, тип BigInt().

•FK юзер_id (админ, сгенерировавший расписание) - уникальный идентификатор юзера, вторичный ключ, **V=12 байт**, тип BigInt().

•Дата генерации - дата геенрации учебного расписания, **V=8 байт**, тип Date().

Всего для одного объекта: **V=32 байт**.

•**Оценка удельного объема информации, хранимой в модели**

Выразим объем информации, хранимой в модели, через переменную N - количество пользователей, среди которых в среднем 1/5 - это преподаватели. Также в университете ЛЭТИ имеется 8 факультетов, в среднем на каждом факультете по 50 групп, значит количество ячеек с расписанием = 50 * 15 = 750 (у какждой группы на неделе по ~15 пар), количество логов - по 100 на каждую коллекцию:

V(N) = 558N + 300112

•**Запросы к модели, с помощью которых реализуются сценации использования**

1. Поиск юзера при авторизации:

```
SELECT * FROM User WHERE email = 'ваш_email' AND password = 'ваш_хешированный_пароль' LIMIT 1;
```

Была использована 1 коллекция. Количество запросов - 1.

2. Просмотр страницы с расписанием (с фильтрами 4 курс и ФКТИ):

```
SELECT Ячейка.*
FROM Ячейка
JOIN Группа ON Ячейка.группа_id = Группа.группа_id
JOIN Направление ON Группа.направление_id = Направление.направление_id
JOIN Факультет ON Направление.факультет_id = Факультет.факультет_id
WHERE Группа.курс = 4 AND Факультет.title_факультет = 'ФКТИ';
```

Было использовано 3 коллекции. Количество запросов - 1.

3. Загрузка учебного плана:

```
-- Вставляем данные в таблицу Факультет
INSERT INTO Факультет (name)
VALUES ('ФКТИ');

-- Получаем идентификатор добавленного факультета
DECLARE @faculty_id INT;
SET @faculty_id = SCOPE_IDENTITY();

-- Вставляем данные в таблицу Специализация
INSERT INTO Специализация (факультет_id, title)
VALUES (@faculty_id, 'ПМИ');

-- Получаем идентификатор добавленной специализации
DECLARE @specialization_id INT;
SET @specialization_id = SCOPE_IDENTITY();

-- Вставляем данные в таблицу Предмет
INSERT INTO Предмет (специализация_id, title, short_title, score_units, hours)
VALUES
    (@specialization_id, 'Математический анализ', 'МатАнал', 2, 192),
    (@specialization_id, 'Теория вероятности и математическая статистика', 'ТВиМС', 2, 192);

-- Вставляем данные в таблицу Группа
INSERT INTO Группа (специализация_id, number, amount, course)
VALUES
    (@specialization_id, '0381', 25, 4),
    (@specialization_id, '1382', 28, 3);
```

Было использовано 4 коллекции. Количество запросов - 4.

4. Получить пожелания преподавателей:

```
SELECT
    Пожелание.пожелание_id,
    Пожелание.ячейка_id,
    Пожелание.юзер_id AS админ_id,
    Пожелание.текст_пожелания,
    Пожелание.enum_статус,
    Пожелание.дата_создания,
    Препод.юзер_id AS препод_id,
    Препод.кафедра,
    Препод.препод_предмет_id,
    Ячейка.день_время_id,
    Ячейка.группа_id,
    Ячейка.юзер_id AS препод_в_ячейке_id,
    Ячейка.аудитория_id,
    Ячейка.предмет_id
FROM
    Пожелание
JOIN
    Ячейка ON Пожелание.ячейка_id = Ячейка.ячейка_id
LEFT JOIN
    Препод ON Ячейка.юзер_id = Препод.юзер_id;
```

Было использовано 3 коллекции. Количество запросов - 1.

5. Обработка пожелания (установка статуса 1 - удовлетворено для пожелания)

```
UPDATE Пожелание SET enum_статус = 1 WHERE пожелание_id = 'yourWishId' AND юзер_id = 'yourUserId';
```

Было использовано 2 коллекции. Количество запросов - 2.

c. Сравнение моделей

В общем случае количество запросов оказалось больше для SQL-модели, но при прочих равных данные в NoSQL модели занимают больше памяти. Во всех (кроме первого - по поиску юзера) запросах у SQL использовано несколько таблиц, так из-за особенности хранения данных в SQL-модели таблиц оказалось

больше, а также существуют связи между таблицами; в Mongo - 1-2 коллекции, так как данные вложены друг в друга. Хотя память, необходимая для MongoDB, увеличивается из-за дублирования данных, но устройство такой модели повышает производительность. В сценариях, где важно минимизировать использование памяти, рекомендуется предпочесть SQL-модель. Однако в нашем случае, где ключевыми факторами являются скорость выполнения и эффективная обработка запросов от пользователей, более предпочтительным вариантом будет использование нереляционной модели.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

a. Краткое описание (архитектура, модули)

Серверная часть приложения представляет из себя REST API web-приложение, реализована авторизация, аутентификация пользователя, реализована одна роль – ROLE_ADMIN, добавлены тестовые данные в базу данных (пользователи, группы, расписание), написаны контроллеры для получения расписания с учетом фильтрации (курс, факультет, преподаватель).

Клиентская часть представляет собой web-приложение, которое использует API сервера и отображает данные для пользователя. Реализованы формы регистрации и аутентификации, главная таблица расписания и фильтры для отображения данных.

Клиент и сервер общаются по средством обмена JSON-объектами, настроена авторизация через JWT токены, настроены CORS.

Серверная часть собирается в докере на порте 8080, клиентская локально – на 8081.

b. Использованные технологии

Технологии серверной части: Java Spring Boot 2.7.24, Spring Security JWT.

Технологии клиентской части: Vue.js + TypeScript, формы и компоненты написаны с использованием UI-кита Vuetify.

c. Снимки экрана приложения

Рисунок 7 - Страница авторизации пользователя.

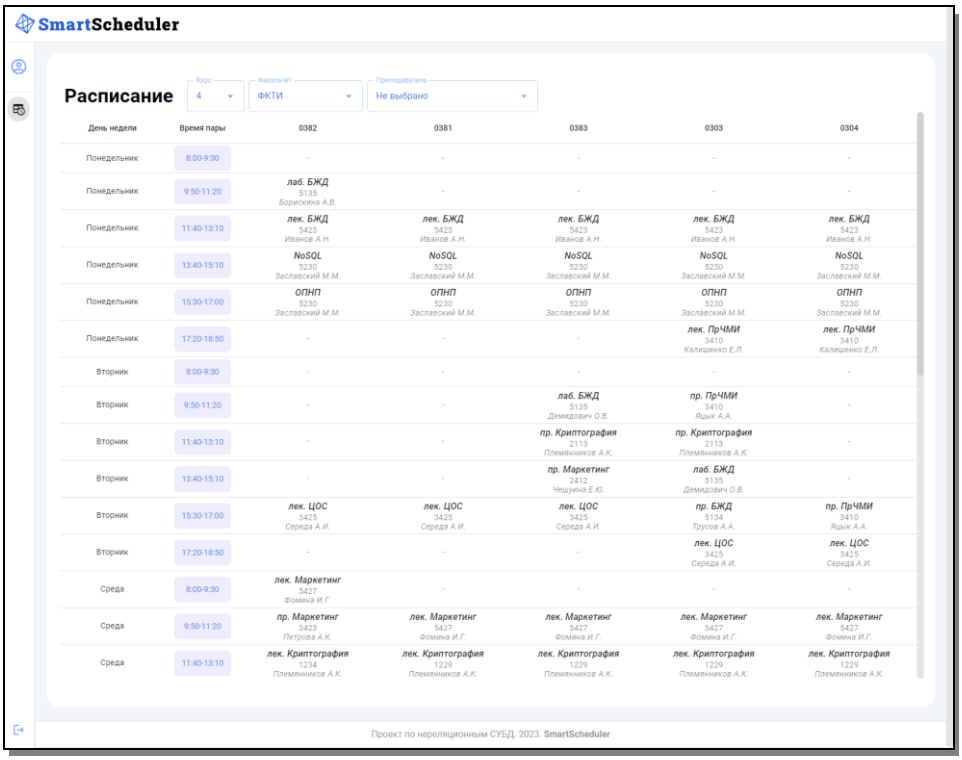


Рисунок 8 – Просмотр расписания и фильтрация по полям.

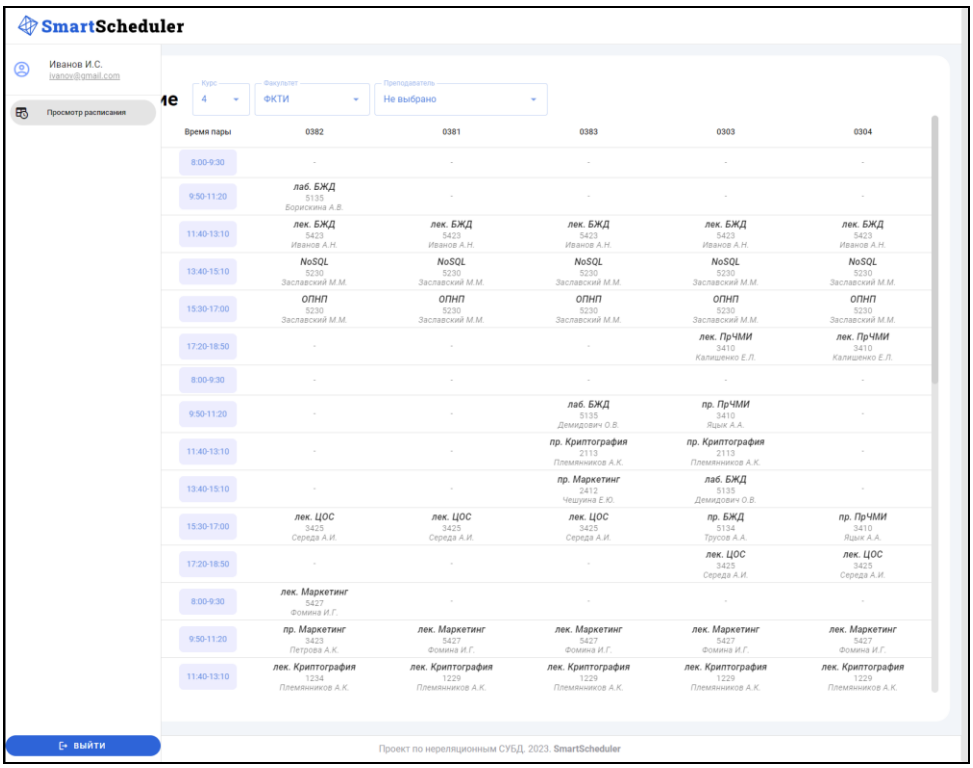


Рисунок 9 – Навигационная панель с информацией о пользователе.

5. ВЫВОДЫ

Результаты.

В ходе работы было разработано веб-приложение для просмотра расписания в вузе, в которой в качестве СУБД использована MongoDB. Пользователи могут авторизоваться в системе и получить доступ к приятному интерфейсу для просмотра расписания в вузе в виде таблицы и для поиска информации о паре с помощью удобной фильтрации.

Недостатки и пути улучшения полученного решения.

Для прототипа Хранение и Представления был разработан функционал для пользователя-администратора. Разграничение функционала по нескольким ролям обеспечит разный уровень доступа к разным частям приложения, а также персонализированный интерфейс. Например, администратор может иметь полный доступ ко всем функциям, в то время как студент ограничен в правах.

Масштабирование и будущее развитие приложения.

Добавление возможности генерации расписания. Был представлен прототип Хранение и Представление, а поскольку в приложении используется микросервисная архитектура, при масштабировании есть возможность добавить независимый модуль генерации расписания на любом языке программирования и связать его с сервером, что позволит расширить возможности приложения.

6. ПРИЛОЖЕНИЯ

Документация по сборке и развертыванию приложения:

1. В папке */backend* в терминале запустить команду

   ```
   docker compose up
   ```

2. В папке /frontend в терминале запустить команду

   ```
   npm run serve
   ```

3. Веб-приложение открыть в браузере по адресу:

   http://localhost:8081

Инструкция для пользователя – авторизоваться под именем:

```
Логин: ivanov@gmail.com
Пароль: 1
```

7. ЛИТЕРАТУРА

1. Ссылка на репозиторий: https://github.com/moevm/nosql2h23-schedule
2. Документация MongoDB: https://docs.mongodb.com/manual/
3. Настройка CORS в Spring: https://www.baeldung.com/spring-cors
4. Документация UI-фреймворка Vuetify: https://vuetifyjs.com/en/getting-started/installation/
5. Документация Vue.js: https://vuejs.org/api/