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: Веб-сервис электронного образования (EDU)

Студентка гр. 0303 _____ Костебелова Е.К.
Студентка гр. 0303 _____ Курочкина Е.А.
Студент гр. 0303 _____ Денежный А.А.
Преподаватель _____ Заславский М.М.

Санкт-Петербург
2023

ЗАДАНИЕ

Студенты

Костебелова Е. К.

Курочкина Е. А.

Денежный А. А.

Группа 0303

Тема проекта: Разработка веб-сервиса электронного образования.

Исходные данные:

Необходимо реализовать веб-сервис электронного образования для СУБД MongoDB.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 20.09.2023

Дата сдачи реферата: 18.12.2023

Дата защиты реферата: 18.12.2023

Студентка гр. 0303	_____	Костебелова Е.К.
Студентка гр. 0303	_____	Курочкина Е.А.
Студент гр. 0303	_____	Денежный А.А.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была сформулирована и утверждена индивидуальная тема – разработка веб-сервиса электронного образования для СУБД MongoDB. Во внимание будут приниматься такие аспекты как производительность и удобство разработки. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h23-edu

ANNOTATION

As part of this course, it was supposed to develop an application in a team on one of the given topics. An individual topic was formulated and approved - the development of an e-education web service for the MongoDB DBMS. Aspects such as performance and ease of development will be taken into account. You can find the source code and all additional information at the link: https://github.com/moevm/nosql2h23-edu

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

Цель работы – создать высокопроизводительное и удобное решение для веб-сервиса электронного образования.

Было решено разработать веб-приложение, которое позволит создавать курсы, добавлять с них необходимые элементы, а также удобно взаимодействовать с созданными курсами.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение с использованием СУБД MongoDB, а также с возможностью локального разворота приложения с помощью docker-compose.

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макеты UI

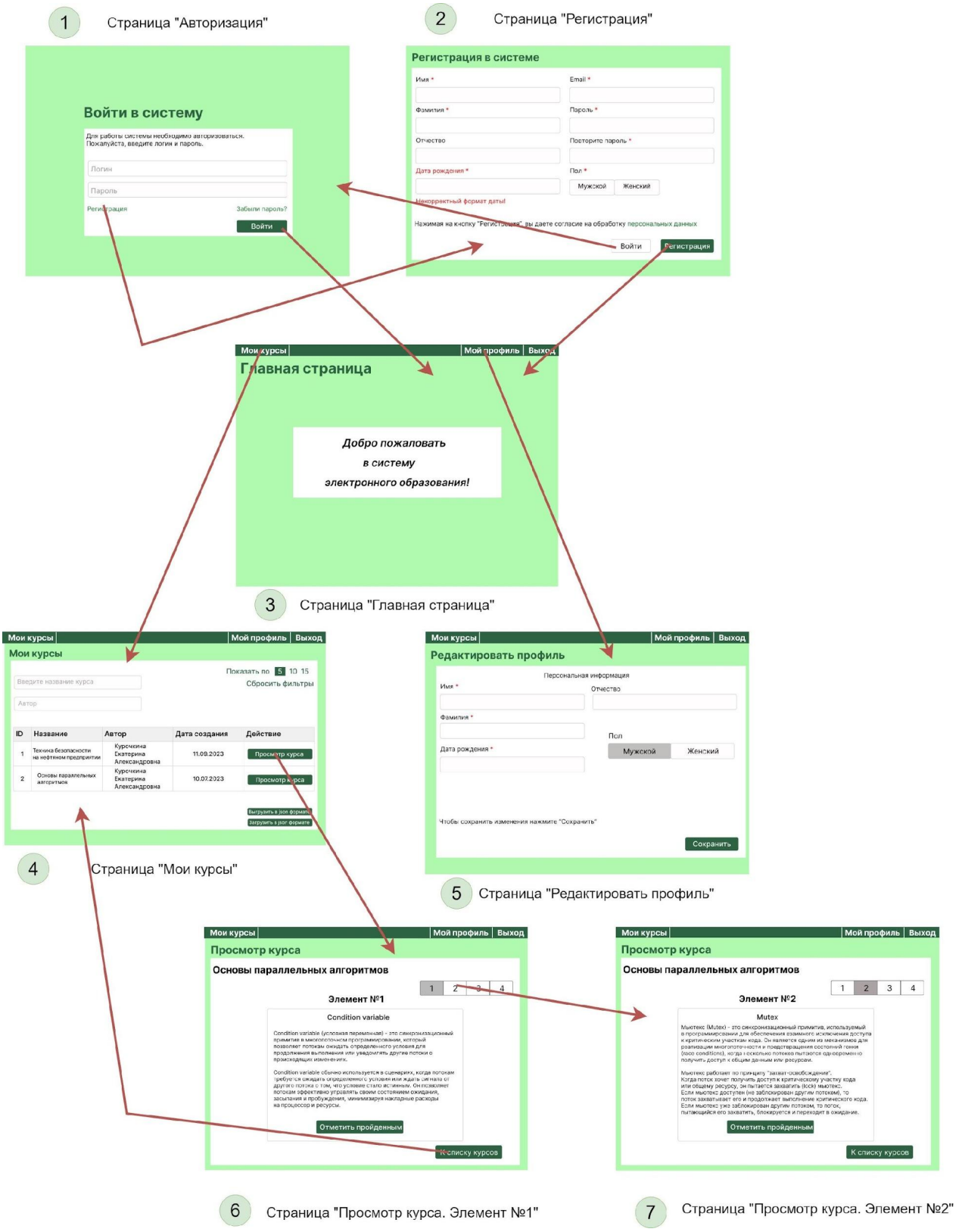

Рис. 1 – Макет UI для пользователя БЕЗ роли администратора

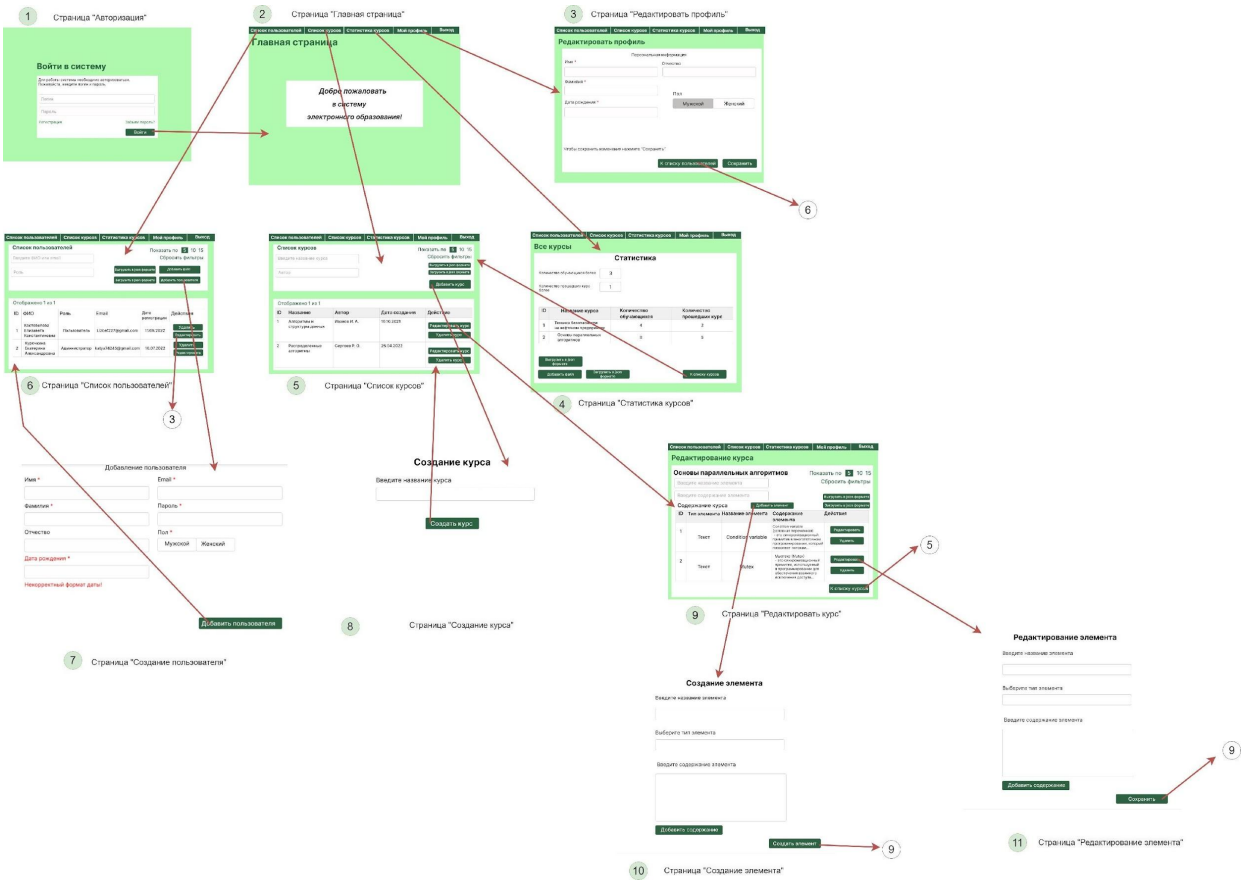

Рис. 2 – Макет UI для пользователя с ролью администратора

3.2. Описание сценариев использования

3.2.1. Сценарий использования - «Регистрация и аутентификация пользователя в системе»:

Действующее лицо: Пользователь

Основной сценарий:

1. Ранее незарегистрированный в системе пользователь переходит по ссылке "Регистрация" со страницы аутентификации.
2. Пользователь заполняет поля формы регистрации своими данными и нажимает кнопку "Зарегистрироваться".
3. В случае успешной регистрации пользователь будет перенаправлен на страницу аутентификации, где ему необходимо ввести свои данные для аутентификации в системе.

4. В случае успешной аутентификации пользователь будет переадресован на главную страницу.

Альтернативный сценарий:

1. Пользователь заполнил форму регистрации некорректными данными.
2. Пользователю возвращается ошибка с указанием, какие именно поля были неверно заполнены.
3. Пользователь перезаполняет форму регистрации.

3.2.2. Сценарий использования - «Создание нового курса в системе»:

Действующее лицо: Пользователь с ролью администратора
Основной сценарий:

1. Пользователь аутентифицируется под администратором в системе на странице аутентификации.
2. Пользователь перенаправляется на главную страницу.
3. Пользователь переходит переходит на "Список курсов", где расположены все курсы в системе.
4. Пользователь нажимает кнопку "Добавить курс" и заполняет форму создания курса.
5. Пользователь перенаправляется на "Список курсов".
6. Пользователь выбирает ранее созданный им курс и попадает на страницу "Редактирования курса".
7. Пользователь по своему усмотрению добавляет неограниченное количество элементов в курс и сохраняет изменения.

Альтернативный сценарий:

1. Пользователь пытается добавить курс с названием, которое уже существует в системе.
2. Пользователь получает ошибку: "Курс с указанным вами названием уже существует".

3.2.3. Сценарий использования - «Просмотр пользователем назначенных ему курсов»:

Действующее лицо: Пользователь без роли администратора

Основной сценарий:

1. Пользователь аутентифицируется системе на странице аутентификации.
2. Пользователь перенаправляется на главную страницу.
3. Пользователь переходит на страницу "Мои курсы" и выбирает любой курс из доступных ему.
4. Пользователь перенаправляется на страницу просмотра курса.
5. Пользователь просматривает курс и отмечает изученные им элементы.
6. После полного ознакомления пользователем выбранного им курса, пользователь может выбрать следующий курс для изучения.

Альтернативный сценарий:

1. После перехода на страницу "Мои курсы" может оказаться, что пользователю не был назначен ни один курс. В таком случае стоит дождаться, пока администратор назначит курсы пользователю.

3.2.4. Сценарий использования - «Просмотр и выгрузка статистики в формате xlsx по всем курсам системы»:

Действующее лицо: Пользователь с ролью администратора

Основной сценарий:

1. Пользователь аутентифицируется под администратором в системе на странице аутентификации.
2. Пользователь перенаправляется на главную страницу.
3. Пользователь переходит на "Статистику курсов".
4. Пользователю отображается статистика по всем курсам системы.
5. Пользователь нажимает кнопку "Выгрузить статистику".
6. Система формирует xlsx файл со статистикой по всем курсам системы и браузером пользователя инициируется загрузка сформированного файла.
7. Пользователь открывает и изучает скачанный файл.

Альтернативный сценарий:

1. Пользователь ознакомился со статистикой и не захотел её выгружать.

3.2.5. Сценарий использования - «Импорт пользователей в систему при помощи json файла»:

Действующее лицо: Пользователь с ролью администратора

Основной сценарий:

1. Пользователь аутентифицируется под администратором в системе на странице аутентификации.
2. Пользователь перенаправляется на главную страницу.
3. Пользователь переходит на страницу "Списка пользователей".
4. Пользователь нажимает кнопку "Добавить файл" и добавляет json файл с данными новых пользователей.

5. Пользователь нажимает кнопку "Импортировать".
6. Новые пользователи создаются в системе и становятся доступны для просмотра на странице "Список пользователей".

Альтернативный сценарий:

1. Пользователь передал json файл с некорректными данными.
2. Пользователю возвращается ошибка: "Был передан файл с некорректными данными. Выполнить импорт не удалось".

4. МОДЕЛЬ ДАННЫХ

4.1. Нереляционные модели данных

4.1.1. Описание назначений коллекций, типов данных и сущностей

*Коллекция **users***

Данная коллекция предназначена для хранения информации о пользователях системы.

Каждый элемент коллекции обладает следующим набором свойств:

- user_id - свойство для хранения уникального идентификатора пользователя. Тип ObjectId
- email - свойство для хранения логина пользователя. Тип String.
- password - свойство для хранения зашифрованного пароля пользователя. Тип String.
- name - свойство для хранения имени пользователя. Тип String.
- surname - свойство для хранения отчества пользователя. Тип String.
- patronymic - свойство для хранения отчества пользователя. Тип String.
- date_birth - свойство для хранения даты рождения пользователя. Тип Date.
- gender - свойство для хранения пола пользователя. Тип Integer. 1 обозначает мужчину, 0 - женщину.
- role_id - свойство для хранения идентификатора роли пользователя в системе. Тип ObjectId.
- course_ids - свойство для хранения идентификаторов курсов, которые принадлежат пользователю. Свойство представляет собой массив (тип Array), каждым элементом которого является уникальный идентификатор курса (тип ObjectId).
- user_element_statistic_ids - свойство для хранения идентификаторов статистики пользователя по элементам курсов. Свойство

представляет собой массив (тип Array), каждым элементом которого является уникальный идентификатор статистики пользователя по элементу курса (тип ObjectId).

- created_at - свойство для хранения даты создания пользователя. Тип ISODate.
- updated_at - свойство для хранения даты обновления пользователя. Тип ISODate.
- deleted_at - свойство для хранения даты удаления пользователя. Тип ISODate. Свойство необходимо для реализации "мягкого" удаления.

*Коллекция **roles***

Данная коллекция предназначена для хранения информации о ролях пользователей системы.

Каждый элемент коллекции обладает следующим набором свойств:

- role_id - свойство для хранения уникального идентификатора роли. Тип ObjectId
- title - свойство для хранения названия роли. Тип String.
- user_ids - свойство для хранения идентификаторов пользователей, которые относятся к конкретной роли. Свойство представляет собой массив (тип Array), каждым элементом которого является уникальный идентификатор пользователя (тип ObjectId).
- created_at - свойство для хранения даты создания пользователя. Тип ISODate.
- updated_at - свойство для хранения даты обновления пользователя. Тип ISODate.
- deleted_at - свойство для хранения даты удаления пользователя. Тип ISODate. Свойство необходимо для реализации "мягкого" удаления.

*Коллекция **courses***

Данная коллекция предназначена для хранения информации о курсах, созданных в системе.

Каждый элемент коллекции обладает следующим набором свойств:

- course_id - свойство для хранения уникального идентификатора курса. Тип ObjectId
- title - свойство для хранения названия курса. Тип String.
- description - свойство для хранения описания курса. Тип String.
- user_id - свойство для хранения уникального идентификатора пользователя, которому принадлежит курс. Тип ObjectId.
- element_ids - свойство для хранения идентификаторов элементов, которые принадлежат курсу. Свойство представляет собой массив (тип Array), каждым элементом которого является уникальный идентификатор элемента курса (тип ObjectId).
- created_at - свойство для хранения даты создания курса. Тип ISODate.
- updated_at - свойство для хранения даты обновления курса. Тип ISODate.
- deleted_at - свойство для хранения даты удаления курса. Тип ISODate. Свойство необходимо для реализации "мягкого" удаления.

*Коллекция **elements***

Данная коллекция предназначена для хранения информации об элементах курсов.

Каждый элемент коллекции обладает следующим набором свойств:

- element_id - свойство для хранения уникального идентификатора элемента. Тип ObjectId
- type - свойство для хранения типа элемента. Тип Integer. Будут следующие типы элементов курса: "ссылка", обозначается как 0 и "текст" обозначается как 1.
- title - свойство для хранения названия элемента. Тип String.

- content - свойство для хранения содержания элемента. Тип String.
- course_id - свойство для хранения идентификатора курса, которому принадлежит конкретный элемент. Тип ObjectId.
- created_at - свойство для хранения даты создания курса. Тип ISODate.
- updated_at - свойство для хранения даты обновления курса. Тип ISODate.
- deleted_at - свойство для хранения даты удаления курса. Тип ISODate. Свойство необходимо для реализации "мягкого" удаления.

*Коллекция **user_element_statistics***

Данная коллекция предназначена для хранения информации о статистике пользователей.

Каждый элемент коллекции обладает следующим набором свойств:

- user_element_statistic_id - свойство для хранения уникального идентификатора элемента. Тип ObjectId
- points - свойство для хранения количества очков, которое получил пользователь по элементу курса. Тип Double
- weight - свойство для хранения множителя очков. Тип Double
- user_id - свойство для хранения уникального идентификатора пользователя, которому принадлежит статистика. Тип ObjectId.
- element_id - свойство для хранения уникального идентификатора элемента курса, которому принадлежит статистика. Тип ObjectId.
- created_at - свойство для хранения даты создания курса. Тип ISODate.
- updated_at - свойство для хранения даты обновления курса. Тип ISODate.
- deleted_at - свойство для хранения даты удаления курса. Тип ISODate. Свойство необходимо для реализации "мягкого" удаления.

*Коллекция **assignments***

Данная коллекция предназначена для хранения информации о назначениях пользователей на курсы.

Каждый элемент коллекции обладает следующим набором свойств:

- assignment_id - свойство для хранения уникального идентификатора элемента. Тип ObjectId
- assignment_author_id - свойство для хранения уникального идентификатора пользователя, создал назначение. Тип ObjectId.
- user_id - свойство для хранения уникального идентификатора пользователя, которому принадлежит назначение. Тип ObjectId.
- course_id - свойство для хранения уникального идентификатора курса, которому принадлежит назначение. Тип ObjectId.
- created_at - свойство для хранения даты создания курса. Тип ISODate.
- updated_at - свойство для хранения даты обновления курса. Тип ISODate.
- deleted_at - свойство для хранения даты удаления курса. Тип ISODate. Свойство необходимо для реализации "мягкого" удаления.

4.1.2. Оценка удельного объема информации, хранимой в модели

За основной элемент примем курс, его количество будет N

Наличие курса (140 байт).

Наличие автора курса (264 байта (пользователь) + 88 байт (роль))

Наличие ученика курса (264 байта (пользователь) + 88 байт (роль)).

Наличие назначение ученика на курс (72 байт).

Наличие элемента курса (292 байта).

Наличие статистики ученика по элементу курса (76 байт).

Итого получается N * 1284 байт

4.1.3. Примеры запросов к модели

Поиск пользователя.

- Поиск с фильтром:

```
db.users.find({name: "Анастасия", surname: "Губкина", patronymic: "Сергеевна"})
```

- Для отображения страницы пользователя требуется вся информация о пользователи:

```
db.users.find({user_id: 654fa8293976d4e72e0299b1})
```

Добавления пользователя на курс.

- Если пользователь еще не был добавлен на курс:

```
db.assignments.updateOne(
  {assignment_id: 654fa8293976d4e72e0299c2},
      {
    $addToSet: {
        user_id: 654fa8293976d4e72e0299b1,
     course_id: 654fa8293976d4e72e0299k1
    }
  }
)
```

Изменение статистики пользователя на курсе

- Изменение очков за курс у пользователя:

```
db.user_element_statistics.updateOne(
  {user_element_statistic_id : 1654fa8293976d4e72e0299u1},
      {
    $set: {
     points: 40
      }
  }
```

)

Удаления пользователя из курса

- Удаление пользователя из курса:

```
db.assignments.deleteOne(
  {assignments_id: 1654fa8293976d4e72e0299p1}
)
```

4.1.4. Графическое представление модели

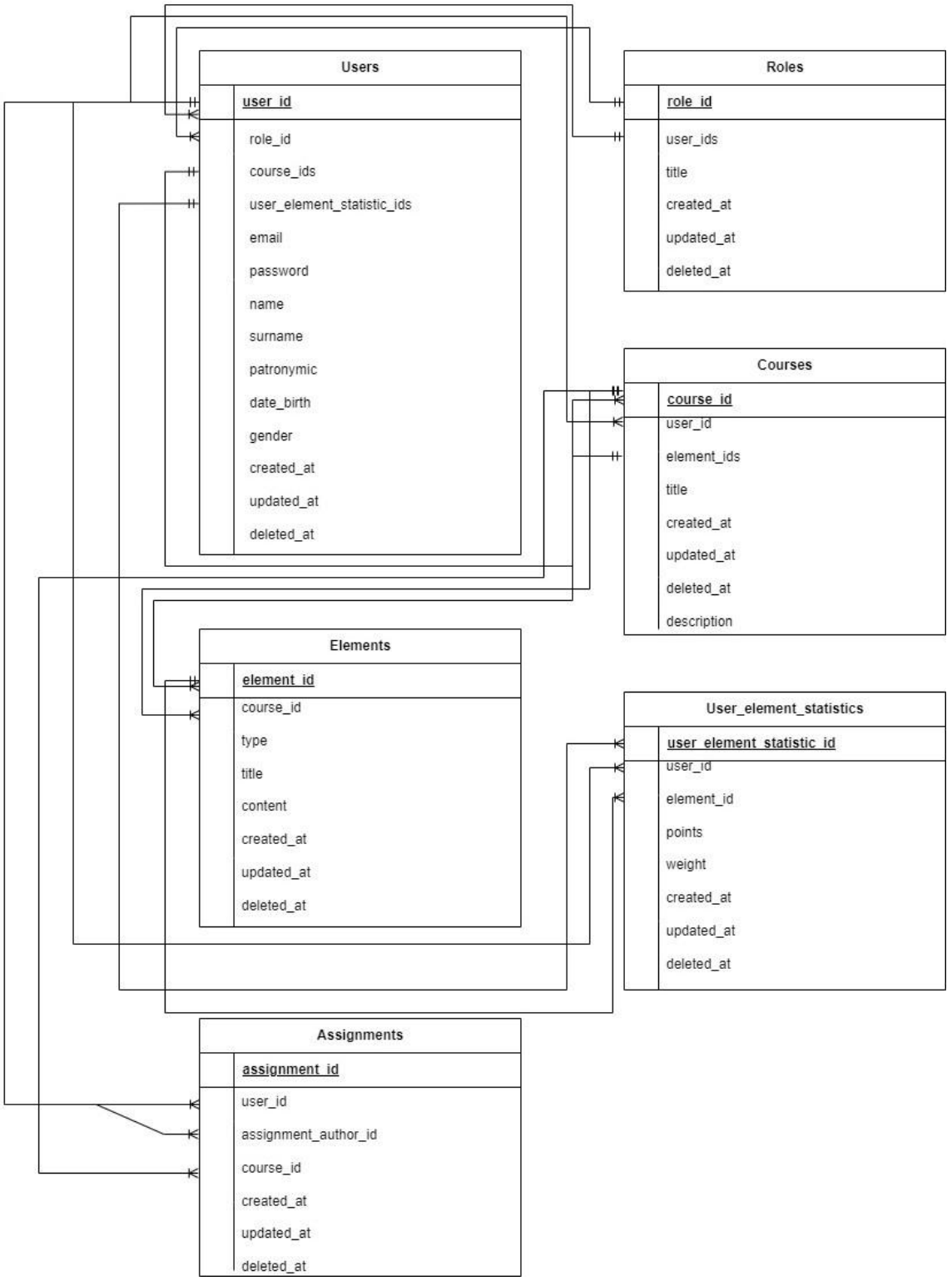

Рис. 3 - Модель нереляционной БД

4.2. Реляционные модели данных

4.2.1. Описание назначений коллекций, типов данных и сущностей

*Таблица **users***

Данная таблица предназначена для хранения информации о пользователях системы. Каждая запись в таблице обладает следующими полями:

- user_id - поле для хранения уникального идентификатора пользователя. Тип Integer.
- email - поле для хранения логина пользователя. Тип Varchar(40).
- password - поле для хранения зашифрованного пароля пользователя. Тип Varchar(40).
- name - поле для хранения имени пользователя. Тип Varchar(40).
- surname - поле для хранения отчества пользователя. Тип Varchar(40).
- patronymic - поле для хранения отчества пользователя. Тип Varchar(40).
- role_id - поле для хранения идентификатора роли пользователя в системе. Тип Integer. Ссылается на поле role_id из таблицы roles.
- date_birth - поле для хранения даты рождения пользователя. Тип Date.
- gender - поле для хранения пола пользователя. Тип Integer. 1 обозначает мужчину, 0 - женщину.
- created_at - поле для хранения даты создания пользователя. Тип Timestamp.
- updated_at - поле для хранения даты обновления пользователя. Тип Timestamp.
- deleted_at - поле для хранения даты удаления пользователя. Тип Timestamp. Свойство необходимо для реализации "мягкого" удаления.

*Таблица **roles***

Данная таблица предназначена для хранения информации о ролях пользователей системы.

Каждая запись в таблице обладает следующими полями:

- role_id - поле для хранения уникального идентификатора роли. Тип Integer
- title - поле для хранения названия роли. Тип Varchar(40).
- created_at - поле для хранения даты создания пользователя. Тип Timestamp.
- updated_at - поле для хранения даты обновления пользователя. Тип Timestamp.
- deleted_at - поле для хранения даты удаления пользователя. Тип Timestamp. Свойство необходимо для реализации "мягкого" удаления.

*Таблица **courses***

Данная таблица предназначена для хранения информации о курсах, созданных в системе.

Каждая запись в таблице обладает следующими полями:

- course_id - поле для хранения уникального идентификатора курса. Тип Integer
- title - поле для хранения названия курса. Тип Varchar(40).
- description - свойство для хранения описания курса. Тип Varchar(40).
- course_author_id - поле для хранения уникального идентификатора пользователя, которому принадлежит курс. Тип Integer. Данное поле ссылается на поле user_id из таблицы users.
- created_at - поле для хранения даты создания курса. Тип Timestamp.
- updated_at - поле для хранения даты обновления курса. Тип Timestamp.
- deleted_at - поле для хранения даты удаления курса. Тип Timestamp. Свойство необходимо для реализации "мягкого" удаления.

*Таблица **elements***

Данная таблица предназначена для хранения информации об элементах курсов.

Каждая запись в таблице обладает следующими полями:

- element_id - поле для хранения уникального идентификатора элемента. Тип Integer

- element_type_id - поле для хранения типа элемента. Тип Integer. Ссылается на element_type_id из таблицы element_types.
- title - поле для хранения названия элемента. Тип Varchar(40).
- content - поле для хранения содержания элемента. Тип Text.
- course_id - поле для хранения идентификатора курса, которому принадлежит конкретный элемент. Тип Integer. Ссылается на поле course_id из таблицы courses.
- created_at - поле для хранения даты создания курса. Тип Timestamp.
- updated_at - поле для хранения даты обновления курса. Тип Timestamp.
- deleted_at - поле для хранения даты удаления курса. Тип Timestamp. Свойство необходимо для реализации "мягкого" удаления.

*Таблица **element_types***

Данная таблица предназначена для хранения информации о типах элементов курса.

Каждая запись в таблице обладает следующими полями:

- element_type_id - поле для хранения уникального идентификатора типа элемента. Тип Integer
- name - поле для хранения названия типа элемента. Тип Varchar(40).
- created_at - поле для хранения даты создания курса. Тип Timestamp.
- updated_at - поле для хранения даты обновления курса. Тип Timestamp.

*Таблица **user_element_statistics***

Данная таблица предназначена для хранения информации о статистике пользователей.

Каждая запись в таблице обладает следующими полями:

- user_element_statistic_id - поле для хранения уникального идентификатора элемента. Тип Integer
- points - поле для хранения количества очков, которое получил пользователь по элементу курса. Тип Double

- weight - поле для хранения множителя очков. Тип Double.
- user_id - поле для хранения уникального идентификатора пользователя, которому принадлежит статистика. Тип Integer. Ссылается на поле user_id из таблицы users.
- element_id - поле для хранения уникального идентификатора элемента курса, которому принадлежит статистика. Тип Integer. Ссылается на поле element_id из таблицы elements.
- created_at - поле для хранения даты создания курса. Тип Timestamp.
- updated_at - поле для хранения даты обновления курса. Тип Timestamp.
- deleted_at - поле для хранения даты удаления курса. Тип Timestamp. Свойство необходимо для реализации "мягкого" удаления.

*Таблица **assignments***

Данная таблица предназначена для хранения информации о назначениях пользователей на курсы.

Каждая запись в таблице обладает следующими полями:

- assignment_id - поле для хранения уникального идентификатора элемента. Тип Integer
- assignment_author_id - поле для хранения уникального идентификатора пользователя, который создал назначение. Тип Integer. Ссылается на поле user_id из таблицы users.
- user_id - поле для хранения уникального идентификатора пользователя, которому принадлежит назначение. Тип Integer. Ссылается на поле user_id из таблицы users.
- course_id - поле для хранения уникального идентификатора курса, которому принадлежит назначение. Тип Integer. Ссылается на поле course_id из таблицы courses.
- created_at - поле для хранения даты создания курса. Тип Timestamp.
- updated_at - поле для хранения даты обновления курса. Тип Timestamp.

- deleted_at - поле для хранения даты удаления курса. Тип Timestamp. Свойство необходимо для реализации "мягкого" удаления.

4.2.2. Оценка удельного объема информации, хранимой в модели

За основной элемент примем курс, его количество будет N
Наличие курса (112 байта).
Наличие автора курса (244 байта (пользователь) + 68 байт (роль)) Наличие ученика курса (244 байта (пользователь) + 68 байт (роль)).
Наличие назначение ученика на курс (40 байт).
Наличие элемента курса (276 байта).
Наличие типа элемента курса (60 байт).
Наличие статистики ученика по элементу курса (52 байта).
Итого получается N * 1164 байт

4.2.3. Пример запросов

Поиск пользователя

- Для отображения страницы пользователя требуется вся информация о пользователи:
  SELECT * FROM users
  WHERE user_id = 5

Добавления пользователя на курс

- Если пользователь еще не был добавлен на курс:
  INSERT INTO assignments(user_id, course_id)
  VALUES (5, 20);

Изменение статистики пользователя на курсе

- Изменение очков за курс у пользователя:

UPDATE user_element_statistics

SET points = 40

WHERE user_element_statistic_id = 13

Удаления пользователя из курса

- Удаление пользователя из курса:

  DELETE FROM assignments

  WHERE assignment_id = 33

4.2.4. Графическое представление модели

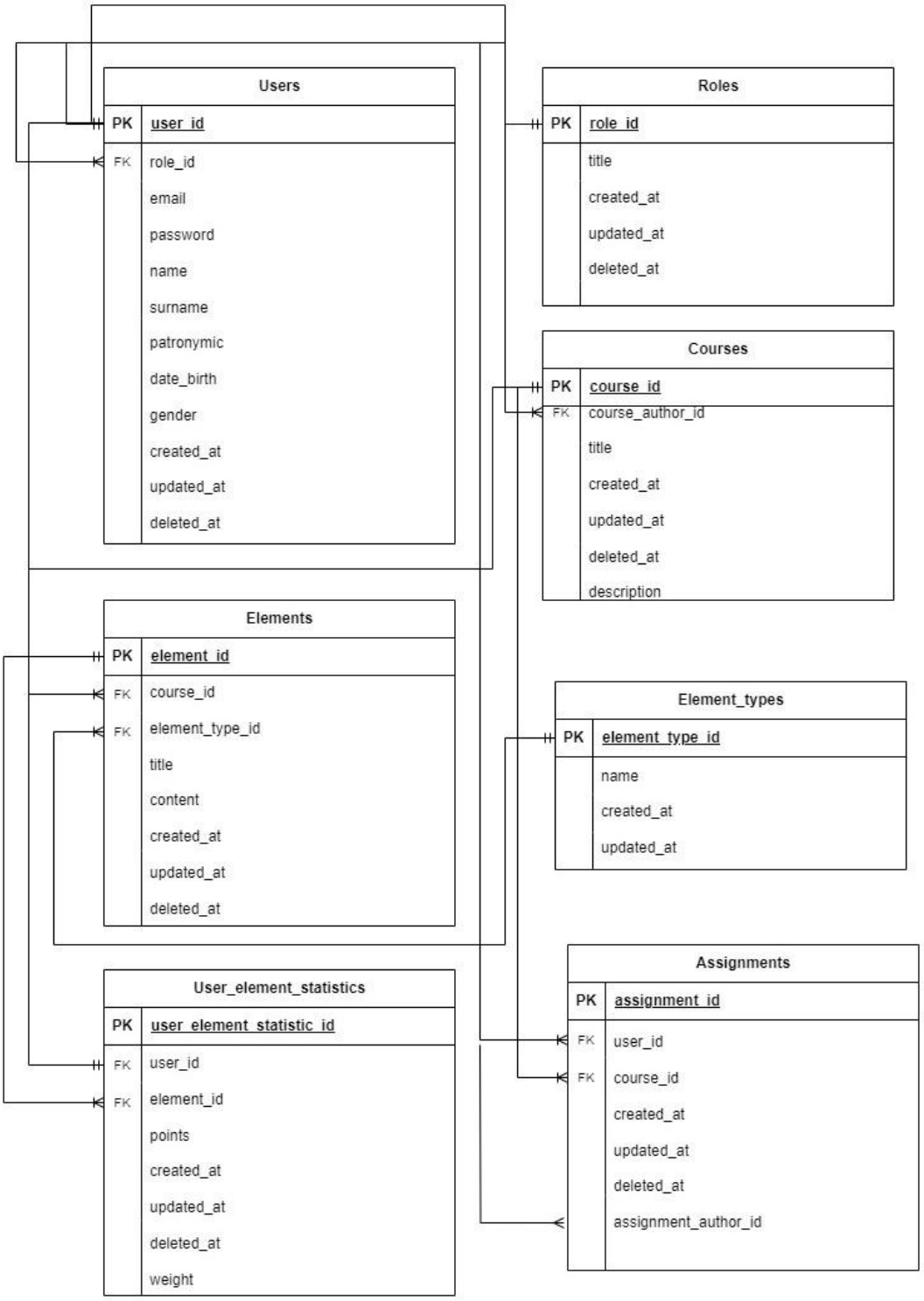

Рис. 4 - Модель реляционной БД

4.3. Сравнение моделей

- Удельный объем информации

За основной элемент был принят курс, его количество N.

Полученная формула в NoSQL: N * 1284

Полученная формула в SQL: N * 1164

Отношение между NoSQL и SQL объемами данных равно: (N * 1284) / (N * 1164) = 1,103

Отношение между чистым NoSQL и SQL объемами данных равно: (N * 1092) / (N * 1092) = 1,0

- Запросы по отдельным use case

Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД и прочих параметров

- Поиск пользователя: количество запросов одинаково - 2 запроса
- Добавления пользователя на курс: количество запросов одинаково - 3 запроса
- Изменение статистики пользователя на курсе: количество запросов одинаково - 2 запроса
- Удаления пользователя из курса: количество запросов одинаково - 2 запроса

- Количество задействованных коллекций

- В SQL задействовано 7 сущностей, а в NoSQL задействовано 6 коллекций.

4.4. Вывод

Таким образом, NoSQL отличный выбор, поскольку система выигрывает по памяти, в одном из UseCase по количеству запросов, а также используется меньше коллекций, чем сущностей.

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

5.1. Краткое описание

Back-end представляет из себя приложение, разработанное на языке программирования PHP и в фреймворке Laravel. Приложение основано на Model — View — Controller архитектуре. MVC – это шаблон проектирования, который построен на основании принципа сохранения представления данных. Согласно этому принципу, данные хранятся отдельно от методов, взаимодействующих с этими данными.
За основную базу данных взята MongoDB, настроено взаимодействие между PHP и базой данных.
Сервис позволяет создавать курсы, добавлять в них элементы, а также создавать и назначать пользователей на курсы. Для каждого элемента, курса и пользователя были разработаны формы редактирования и возможность корректного удаления.
Также на страницах просмотра списков пользователей и курсов были реализованы фильтры для упрощения поиска необходимой информации.

Front-end основан на языке шаблонов предоставляемым фреймворком Laravel – Blade.
Blade – это простой, но мощный движок шаблонов. В отличие от некоторых шаблонизаторов PHP, Blade не ограничивает нас в использовании обычного "сырого" кода PHP в шаблонах. Шаблоны Blade могут быть возвращены из маршрутов или контроллера с помощью глобального помощника view.
Также front-end основан на Server-Side Rendering. Мы предоставляем пользователю уже готовую страницу со всеми стилями и данными, когда он делает запрос внутри нашего приложения.

5.2. Использованные технологии

БД: MongoDB

Back-end: PHP, Laravel framework

Front-end: HTML, CSS, PHP.

5.3. Снимки экрана приложения

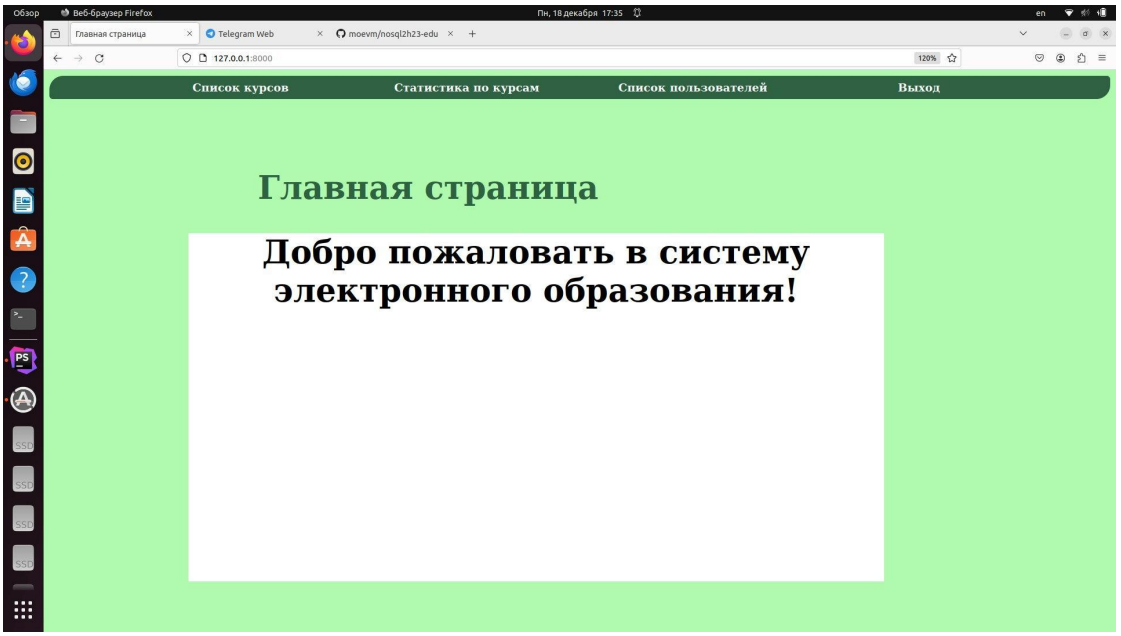

Рис. 5 – Главная страница

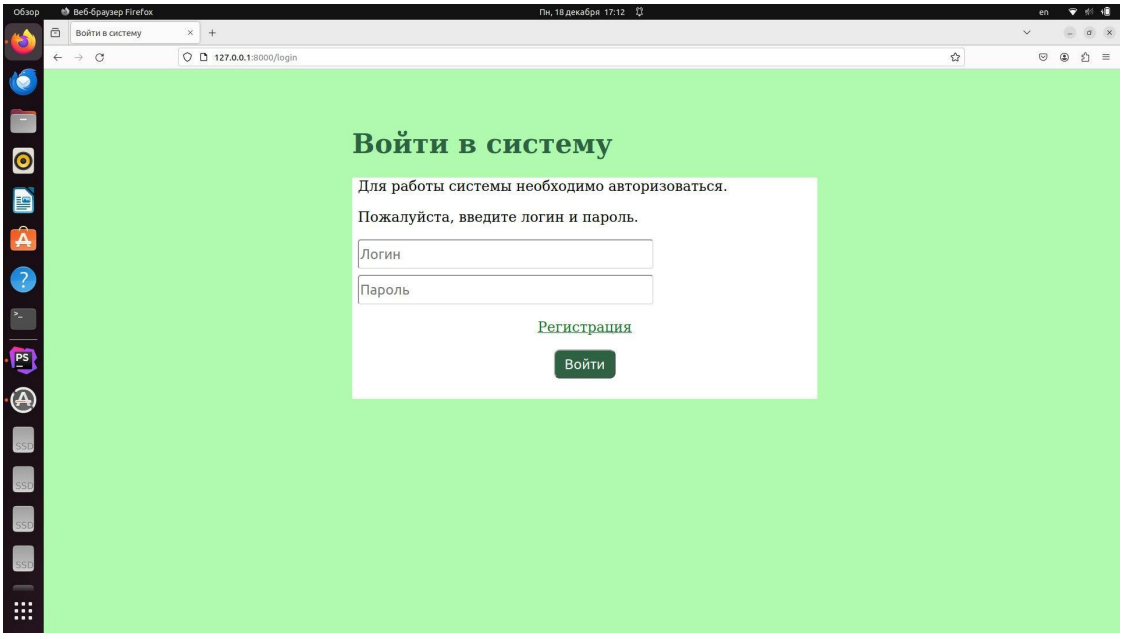

Рис. 6 – Страница Авторизации

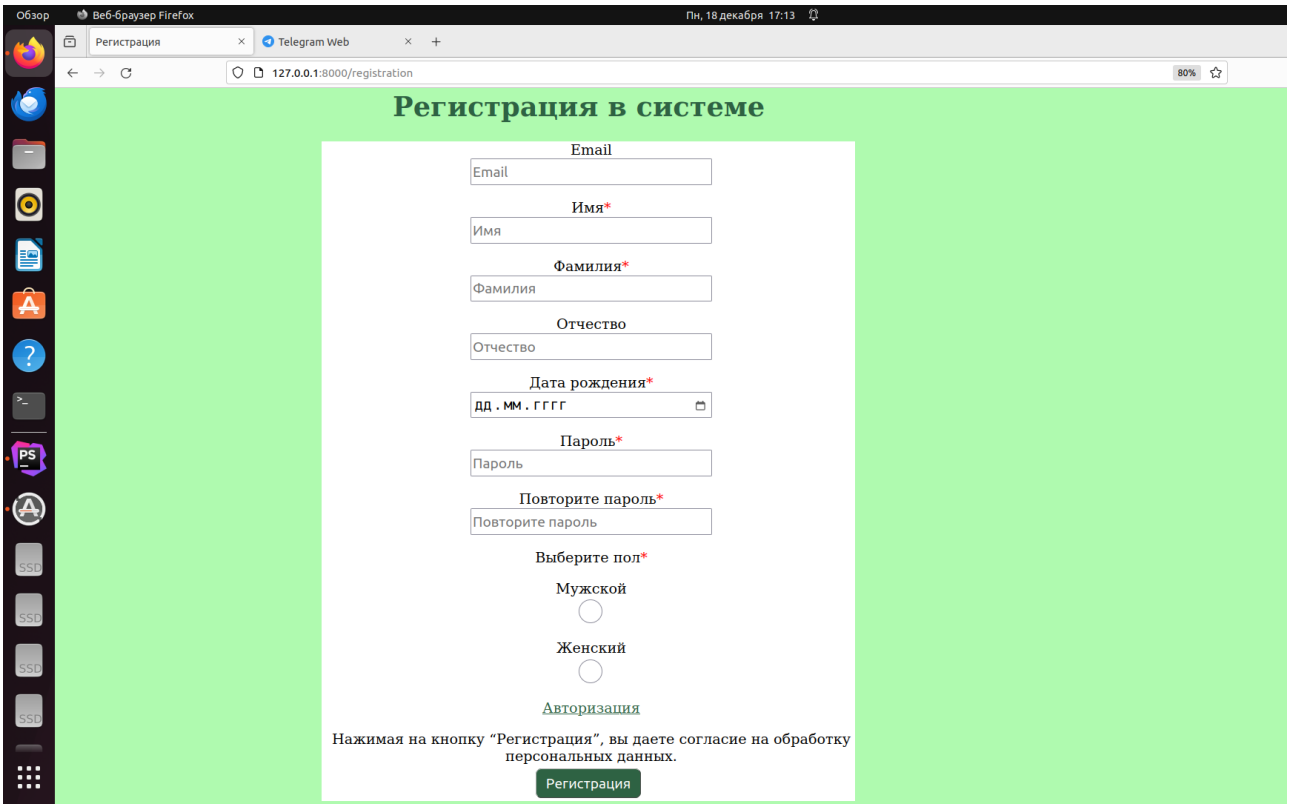

Рис. 7 – Страница регистрации

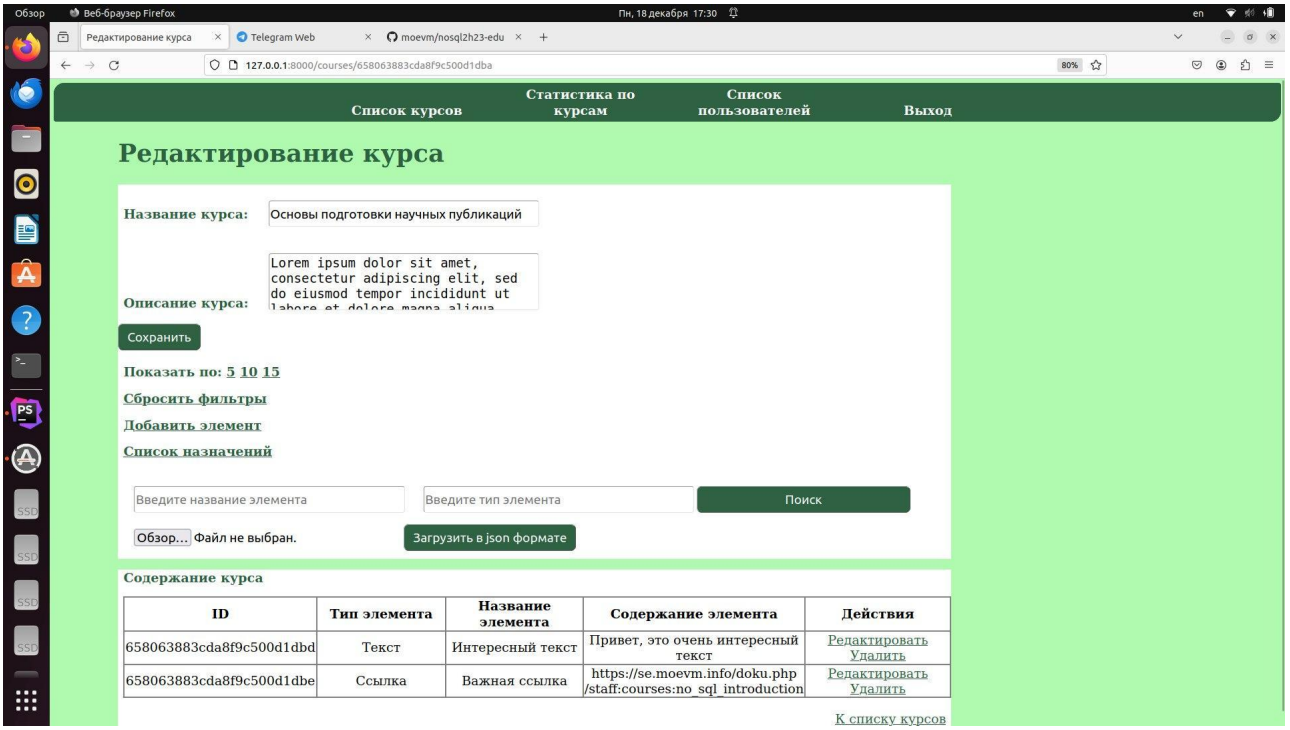

Рис. 8 – Страница редактирования курса

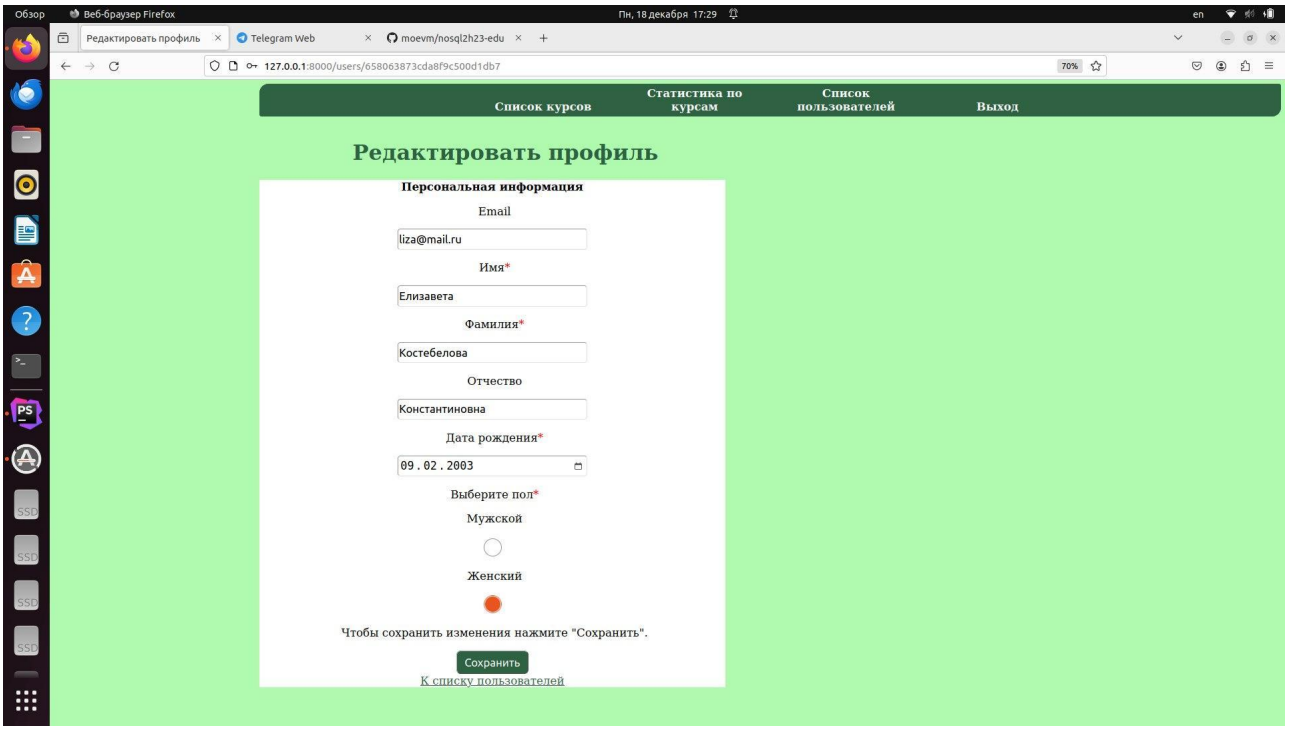

Рис. 9 – Страница редактирования профиля

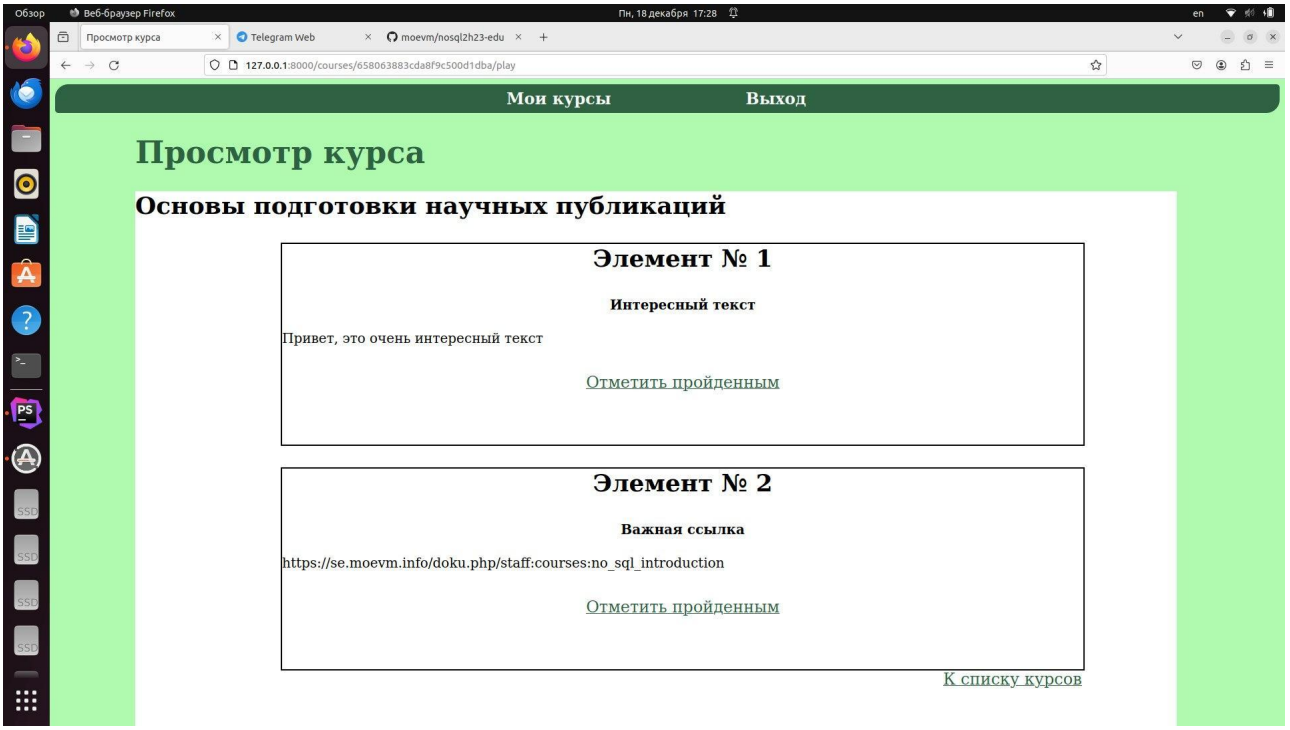

Рис. 10 – Страница просмотра курса

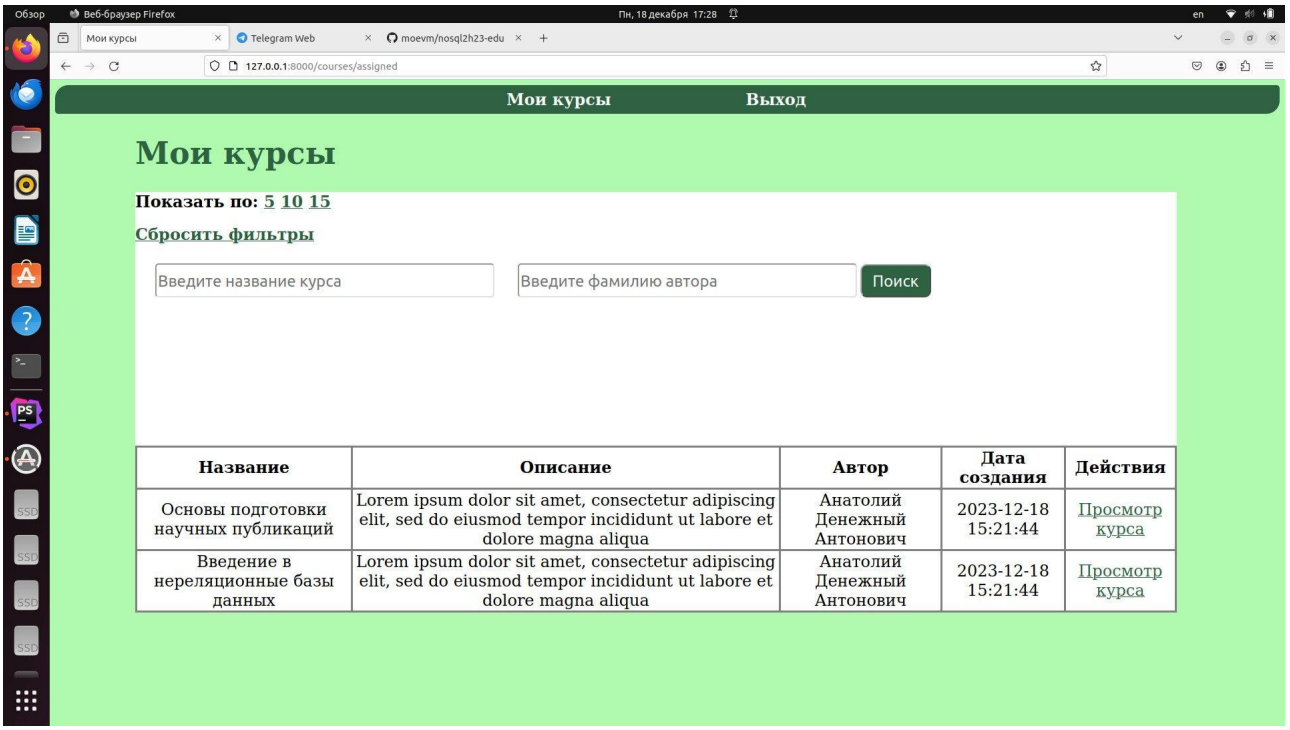

Рис. 11 – Страница моих курсов

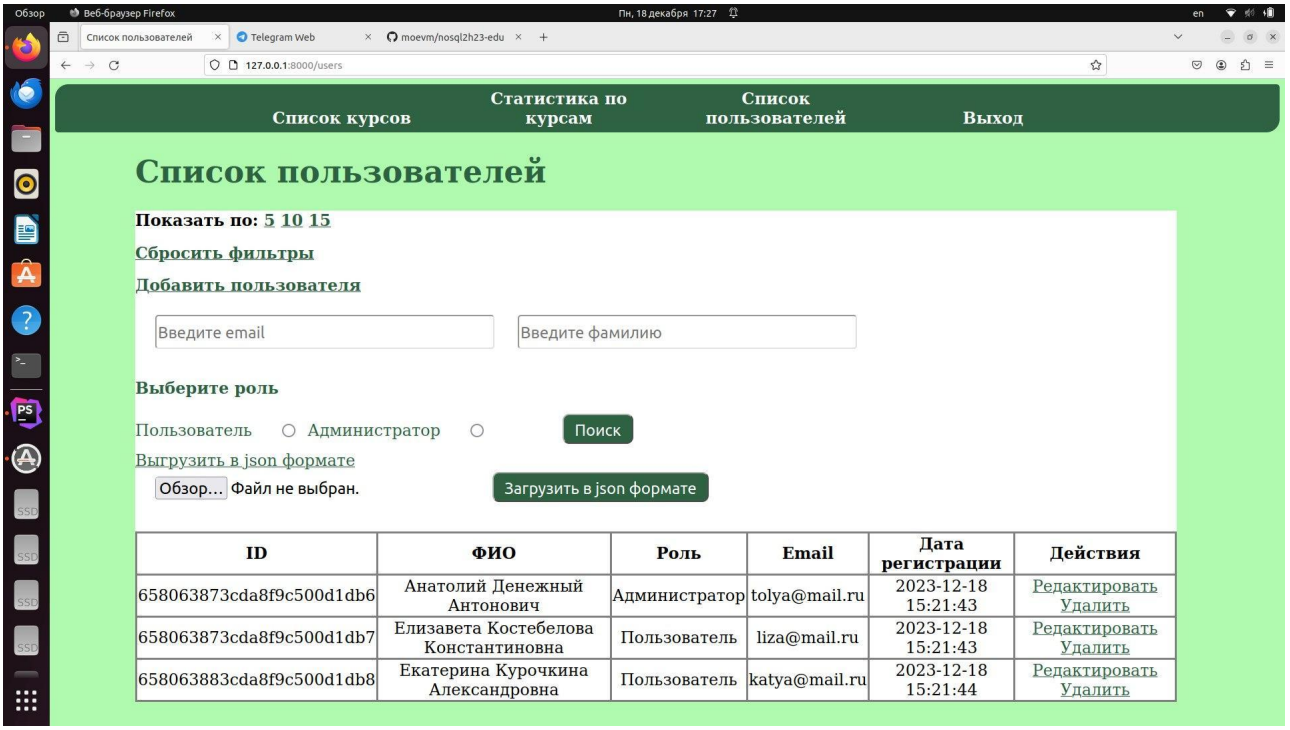

Рис. 12 – Страница списка пользователей

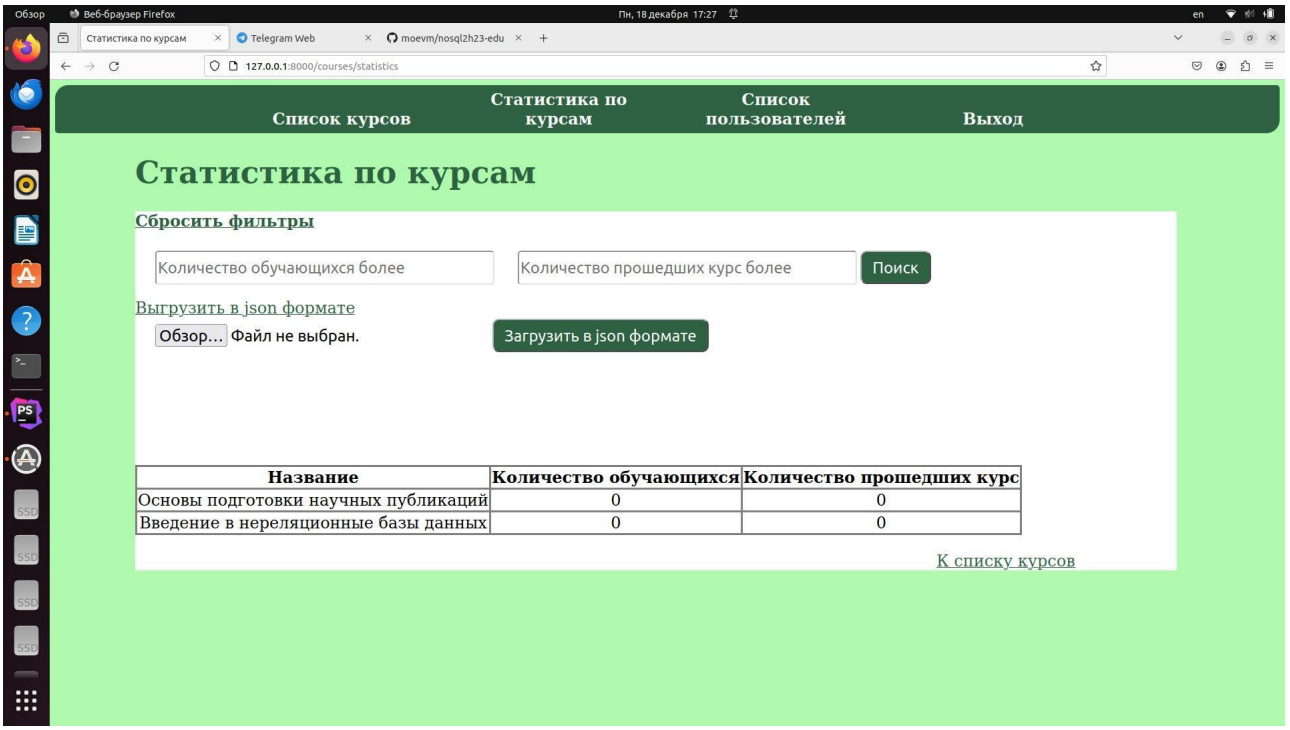

Рис. 13 – Страница статистики

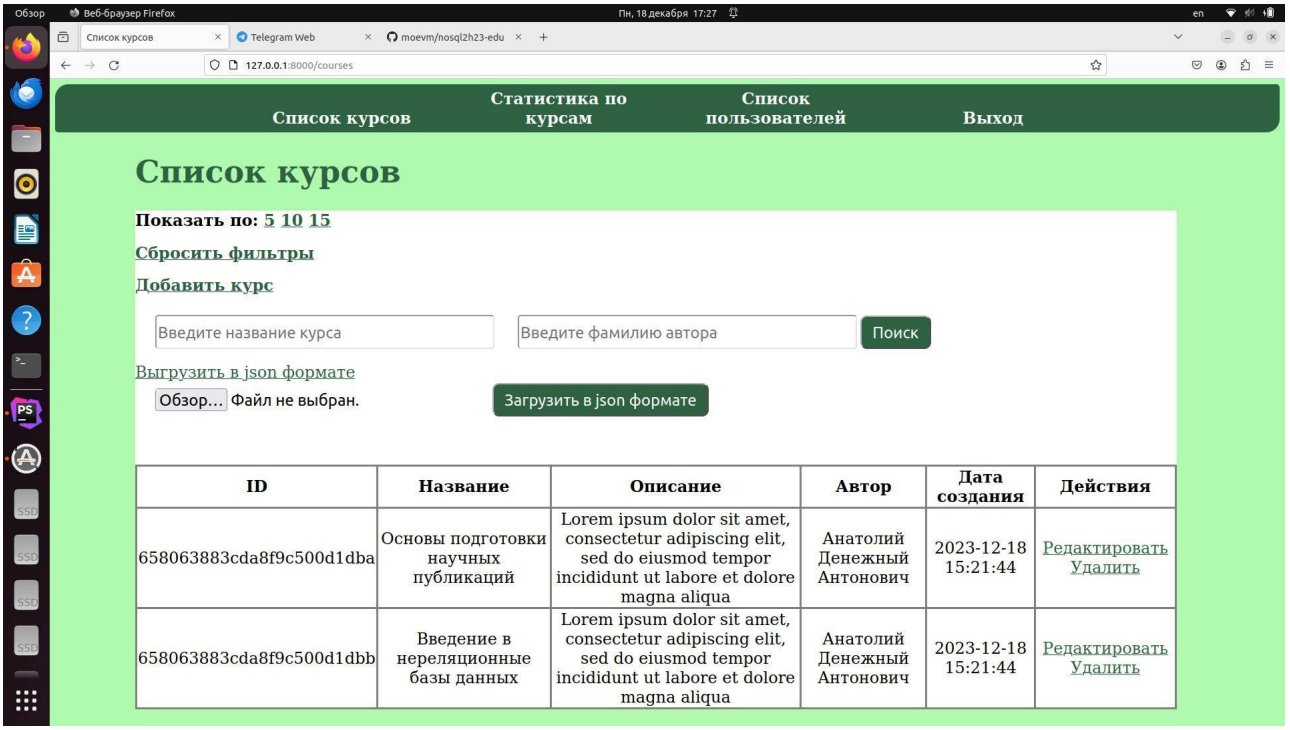

Рис. 14 – Страница списка курсов

6. ВЫВОД

6.1. Достигнутые результаты

В ходе работы было разработано приложение веб-сервиса электронного образования с использованием нереляционной базы данных MongoDB. Была реализована вся заявленная функциональность: корректная работа с MongoDB, управление курсами, их элементами и пользователями. Добавлена возможностью локального разворота приложения с помощью docker-compose.

6.2. Недостатки и пути для улучшения полученного решения

Местами фильтрация происходит не на уровне базы данных (запроса, во время получения данных), а уже получается постфильтрация на уровне php после получения данных. Плохо это тем, что на больших данных наша конструкция будет показывать медленные результаты или вовсе утечку данных. Путь для улучшения – перенести фильтрацию на уровень запроса в базу данных в момент получения данных.

Наличие повторяющихся кусков кода как внутри back-end, так и внутри front-end. Огромный недостаток состоит в том, что при внесении каких-то изменений в повторяющиеся куски – необходимо много раз копировать изменения и вставлять в другие файлы, что увеличивает объём работы в разы. Путь для улучшения – вынести эти куски кода в отдельные файлы и далее их переиспользовать их через подключение.

6.3. Будущее развитие решения

Улучшение front-end вёрстки, поскольку большей части страниц необходим обновлённый дизайн и более качественный пользовательский интерфейс.

Улучшение фильтрации, перенос её на уровень запроса в базу данных в момент получения данных.

Вынесение часто повторяющихся кусков кода в отдельные файлы, чтобы переиспользовать файлы и ускорить внесение изменений в код.

7. ПРИЛОЖЕНИЯ

7.1. Документация по сборке и развертыванию приложения

1. Склонировать репозиторий (указан в списке литературы)
2. Зайти в папку ./nosql2h23-edu/
3. Установить docker и docker-compose с помощью следующих команд:
   - sudo apt update
   - sudo apt install docker-ce -y
   - sudo systemctl status docker
   - sudo apt-get install docker-compose
4. Запустить локальный разворот приложения через docker-compose с помощью следующих команд:
   - sudo docker-compose build --no-cache
   - sudo docker-compose up –d
5. Перейти в любом удобном браузере по адресу: 127.0.0.1

7.2. Инструкция для пользователя

1. Используя свой email и пароль войти в систему на странице авторизации ИЛИ зарегистрироваться на странице регистрации, после чего авторизоваться.
2. Перейти на страницу «Мои курсы»
3. Выбрать один из назначенных на Вас курсов с помощью ссылки «Просмотр курса»
4. Приступить к изучению материала
5. Повтор пунктов 2-4

8. ЛИТЕРАТУРА

1. Ссылка на github проекта: https://github.com/moevm/nosql2h23-edu/
2. Документация MongoDB: https://docs.mongodb.com/manual/
3. Документация PHP: https://www.php.net/docs.php
4. Документация Laravel: https://laravel.com/docs/10.x
5. Примеры для front-разработки: https://getbootstrap.com/docs/5.3/examples/