

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей  
Кафедра информатики  
Дисциплина «Модели данных и системы управления базами данных»

**ОТЧЕТ**  
к лабораторной работе №6  
на тему:  
**«РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С  
ИСПОЛЬЗОВАНИЕМ БАЗЫ ДАННЫХ»**  
БГУИР 6-05-0612-02

Выполнил студент группы 353503  
КОХАН Артём Игоревич  
РЯЗАНЦЕВ Алексей Владимирович

---

(дата, подпись студента)

Проверил ассистент каф. информатики  
КОЖЕМЯКО Евгения Александровна

---

(дата, подпись преподавателя)

Минск 2025

## **1 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

Реализация запросов по журналированию событий, реализация интерфейсов приложения. Результат работы: набор SQL-запросов с описанием, иллюстрации проволочных интерфейсов.

## **2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**

Система журналирования событий является важным компонентом современных приложений, обеспечивающим отслеживание действий пользователей и системных процессов. В базах данных журналирование реализуется через создание специализированных таблиц для записи событий, которые фиксируют информацию о действиях пользователей, изменениях данных и системных операциях.

Основой для журналирования служит оператор INSERT, который добавляет записи в таблицу логов. Каждая запись обычно содержит идентификатор пользователя, тип выполненного действия, временную метку, описание события и дополнительную контекстную информацию. Для классификации событий по уровням важности используются ENUM-типы или справочные таблицы, определяющие категории событий такие как информационные, предупреждения и ошибки.

Реализация пользовательских интерфейсов приложения строится на основе проволочных прототипов (wireframes), которые представляют собой схематичное отображение интерфейса без детализации визуального оформления.

События журналирования могут включать авторизацию пользователей, выполнение критических операций с данными, ошибки ввода и другие значимые действия. Собранная информация используется для анализа работы системы, устранения проблем и улучшения пользовательского опыта.

### 3 ВЫПОЛНЕНИЕ РАБОТЫ

Разработка пользовательского интерфейса приложения велась на основе проволочных прототипов (wireframes), которые представляют собой схематичное отображение структуры и основных элементов интерфейса без детальной проработки визуального оформления. В рамках работы был создан проволочный интерфейс в Swagger (OpenAPI) для документирования и тестирования REST API.



Рисунок 3.1 – Swagger

Этот интерфейс описывает все доступные эндпоинты, их параметры, форматы запросов и ответов, а также позволяет интерактивно тестировать методы API непосредственно из браузера. Swagger-интерфейс включает описание операций для всех сущностей системы: управления пользователями, работы с абонементами, бронирования шкафчиков и записи на персональные тренировки. Для каждого эндпоинта определены HTTP-методы (GET, POST, PUT, DELETE), ожидаемые параметры запроса, структуры тел запросов и ответов, а также возможные коды HTTP-ответов. Такой подход обеспечивает четкую спецификацию API и упрощает интеграцию фронтенд-части приложения с бэкенд-сервисами.

This screenshot displays a portion of the Swagger UI interface showing a list of API operations. It is organized into three main sections corresponding to different controllers:

- personal-training-controller**: Contains five operations: GET /training/{id} (blue), PUT /training/{id} (orange), DELETE /training/{id} (red), GET /training (light blue), and POST /training (green).
- membership-controller**: Contains five operations: GET /membership/{id} (blue), PUT /membership/{id} (orange), DELETE /membership/{id} (red), GET /membership (light blue), and POST /membership (green).
- locker-controller**: Contains one operation: GET /locker/{id} (blue).

Each operation is represented by a colored button (GET, PUT, DELETE) followed by its specific endpoint path. The buttons are arranged vertically for each controller, with expand/collapse arrows to the right of each group.

Рисунок 3.2 – Часть списка возможных запросов в swagger

В качестве примера воспользуемся GET-запросом получения всех возможных тренировок в системе.

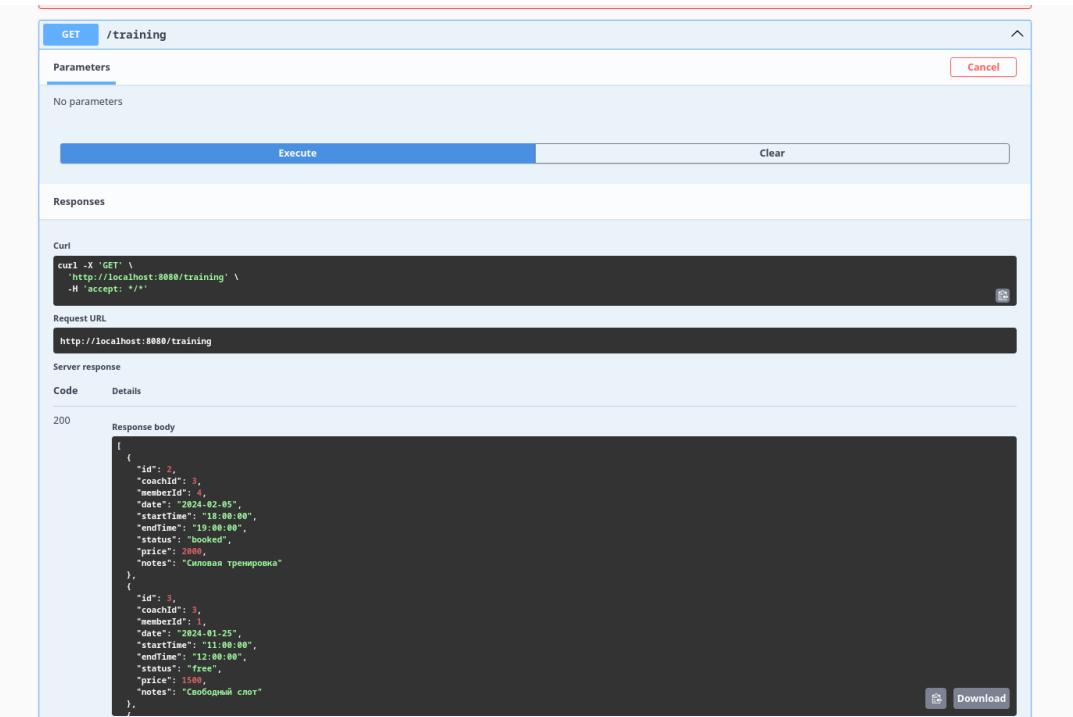


Рисунок 3.3 – GET-запрос тренировок спортивного зала

Реализация журналирования событий, представленная в листинге 3.1. Для записи событий в систему используется оператор INSERT, который добавляет записи в таблицу user\_log. Данный запрос параметризован и принимает следующие значения: идентификатор пользователя (person\_id), тип выполненного действия (action\_type), описание события (action\_description), временная метка (action\_time), IP-адрес (ip\_address) и уровень важности события (severity). Особенностью реализации является использование приведения типа ?::log\_severity\_enum для поля severity, что обеспечивает соответствие значения предопределенному ENUM-типу в базе данных, который классифицирует события по уровням важности такие как информационные, предупреждения и ошибки. Для извлечения записей журнала используется SELECT-запрос, который получает все поля таблицы user\_log с сортировкой по времени события в порядке убывания (ORDER BY action\_time DESC), что позволяет анализировать последние события в первую очередь.

Раз уж мы познакомились со swagger, то в качестве примеры выполним GET-запрос для получения логов со стороны админа в swagger.

The screenshot shows a REST API documentation interface. At the top, there is a blue header bar with the method 'GET' and the endpoint '/admin/users/logs'. Below this, there is a 'Parameters' section which is currently empty. To the right of the parameters is a red 'Cancel' button. In the center, there is a large 'Execute' button in a blue bar and a 'Clear' button. Below these buttons is a 'Responses' section. Under 'Responses', there is a 'Curl' block containing a command to run a curl request to the specified endpoint. Below the curl command is a 'Request URL' field containing the URL. Under 'Server response', there are two tabs: 'Code' and 'Details'. The 'Code' tab is selected, showing the status code 200. The 'Details' tab is also present. Below the status code, there is a 'Response body' block containing a JSON array of log entries. Each entry includes fields like 'id', 'person\_id', 'action\_type', 'action\_description', 'action\_time', 'ip\_address', and 'severity'. The last entry in the array has a 'Download' button next to it. At the bottom of the 'Responses' section, there is a 'Response headers' section.

Рисунок 3.4 – GET-запрос логов

### Листинг 3.1 – Журналирование событий

```
INSERT INTO user_log (person_id, action_type, action_description, action_time,
ip_address, severity)
VALUES (?, ?, ?, ?, ?, ?::log_severity_enum)

SELECT id, person_id, action_type, action_description, action_time, ip_address,
severity FROM user_log ORDER BY action_time DESC
```

## **ВЫВОД**

В рамках работы были созданы механизмы записи и отслеживания событий системы, а также разработан проволочный интерфейс с помощью Swagger для документирования и тестирования API.

Реализованная система журналирования обеспечивает надежную фиксацию всех значимых событий через оператор INSERT в таблицу user\_log. Каждая запись журнала содержит идентификатор пользователя, тип действия, описание события, временную метку, IP-адрес и уровень важности. Использование ENUM-типа log\_severity\_enum позволяет классифицировать события по уровням важности (информационные, предупреждения, ошибки), что упрощает последующий анализ и фильтрацию записей.

Для извлечения записей журнала реализован SELECT-запрос с сортировкой по времени события в порядке убывания, что обеспечивает удобный просмотр последних событий системы.

Разработанный проволочный интерфейс в Swagger предоставляет полную документацию REST API системы, включая описание всех эндпоинтов для управления пользователями, абонементами, бронированием шкафчиков и персональными тренировками. Интерфейс позволяет интерактивно тестировать каждый метод API, проверять форматы запросов и ответов, а также анализировать коды HTTP-ответов.