8. Выгрузка из БД integrationlog (логи)

Integrationlog – это журнал, который фиксирует события, ошибки и статусы процессов, связанных с интеграцией различных систем, приложений или сервисов. Такой журнал помогает отслеживать обмен данными, диагностировать проблемы и обеспечивать стабильность работы интеграционных решений.

В настоящем веб-приложении взаимодействие фронтенда и бекенда осуществляется по протоколу HTTPS на базе фреймворка Javalin.

Для сохранения сведений о запросах и ответах бекенда используется два класса: HttpLoggingFilter и HttpLogRepository – из пакета logging.

Для экспорта данных используется метод export класса ExportCSV.

Чтобы включить ведение журнала http-запросов, нужно настроить работу фреймворка Javalin, выполнив следующие методы класса Javalin:

```
app.before(new HttpLoggingFilter(logRepository));
app.after(ctx -> HttpLoggingFilter.logResponse(ctx, logRepository));
app.exception(Exception.class, (e, ctx) -> {
        ctx.status(500);
        ctx.json(Map.of("error", e.getMessage()));
        ctx.attribute("errorMessage", e.getMessage());
    });
```

Сообщение о выполнении запросов сохраняется в базу данных в таблицу http_logs в следующем формате:

```
id                  - номер записи
user_id             - идентификатор пользователя
request_method      - метод запроса
request_path        - эндпоинт
request_ip          - ip-адрес
request_user_agent  - агент
request_timestamp   - дата и время запроса
response_status     - статус ответа
response_time_ms    - длительность выполнения запроса
error_message       - сообщение об ошибке
created_at          - временная метка создания записи
```

Сведения об ответе сервера помещаются в ту же запись, в которой содержатся сведения о запросе, путём сопоставления метода, эндпоинта и временной метки запроса.

Содержимое таблицы http_logs в формате CSV можно получить путём выполнения GET-запроса к серверу "/api/get_logs" с токеном авторизации, указав имя файла для выгрузки и имя таблицы в json-параметре tableName:

```
curl.exe -v -X GET http://localhost:7070/api/get_logs -H "Authorization: Bearer [token] "
-H "Content-Type: application/json"  -d '{\"tableName\":\"http_logs\"}' -o data.csv
```

9. Выгрузка аудита БД

Аудит базы данных — это регистрация событий, связанных с изменениями в базе данных, когда фиксируется кем инициировано изменение, когда изменение произошло, новое и предыдущее состояние данных.

В настоящем веб-приложении поддерживается аудит таблиц, содержащих сведения о пользователях приложения и их транзакциях:

- users;
- transactions

Сведения об изменении данных пользователя сохраняются в таблицу users_audit. Фиксируется внесение новых записей и изменение существующих записей в таблице users (выполнение DML команд INSERT и UPDATE).

Сведения об изменении данных о транзакциях пользователя сохраняются в таблицу transactions_audit. Фиксируется внесение новых записей, изменение существующих записей и удаление записей в таблице transactions (выполнение DML команд INSERT, UPDATE и DELETE).

Общая структура таблиц users_audit и transactions_audit следующая:

```
{ id } { тип изменения } { кто и когда внёс изменения } { старое и новое значение }
```

Отслеживание изменений в таблицах осуществляется средствами СУБД PostgreSQL, путём настройки триггеров на события создания новой записи, обновления записи и удаления записи.

При срабатывании триггера вызывается выполнение соответствующей триггерной функции.

Содержимое таблиц users_audit и transactions_audit в формате CSV можно получить путём выполнения GET-запроса к серверу "/api/get_logs" с токеном авторизации, указав имя файла для выгрузки нужной таблицы и имя таблицы в json-параметре tableName: users_audit или transactions_audit соответственно.

```
curl.exe -v -X GET http://localhost:7070/api/get_logs -H "Authorization: Bearer [token] "
-H "Content-Type: application/json"  -d '{\"tableName\":\"users_audit\"}' -o data.csv
```

Тестирование

Проверка логирования HTTP-запросов, проверка срабатывания триггеров и заполнения таблиц аудита БД выполнено по следующему сценарию.

1. Регистрация нового пользователя
   - эндпоинт /api/register
   - в таблицу http_logs вносится запись о запрос, метод POST
   - в таблицу БД users_audit – запись о выполнении INSERT
2. Логин зарегистрированного пользователя
   - эндпоинт /api/login
   - в таблицу http_logs вносится запись о запросе, метод POST
3. Изменение данных о пользователе – установка нового имени
   - эндпоинт /api/set_full_name
   - в таблицу http_logs вносится запись о запросе, метод PATCH
   - в таблицу БД users_audit – запись о выполнении UPDATE
4. Создание новой транзакции
   - эндпоинт /api/new_transaction
   - в таблицу http_logs вносится запись о запросе, метод POST
   - в таблицу БД transactions_audit – запись о выполнении INSERT
5. Редактирование существующей транзакции
   - эндпоинт /api/update_transaction
   - в таблицу http_logs вносится запись о запросе, метод PATCH
   - в таблицу БД transactions_audit – запись о выполнении UPDATE
6. Удаление транзакции
   - эндпоинт /api/delete_transaction
   - в таблицу http_logs вносится запись о запросе, метод DELETE
   - в таблицу БД transactions_audit – запись о выполнении DELETE, при этом триггер срабатывает до выполнения SQL-запроса и триггерная функция изменяет статус транзакции на "Платеж удален" без физического удаления самой записи о транзакции из таблицы transactions.

Эндпоинты сделаны для тестирования, в окончательную версию приложения можно не включать.