**Требования к сервису “Активности пользователей”**

Данный сервис сохранять историю активности пользователей в работе с сервисом, позволяет отправлять различные и получать данные о различных действиях пользователя, и метрик сервисов и приложений.

Собранные метрики истории можно будет мониторить получаю информацию из собранных запросов.

**Отправители метрик:**

Авторизованный участник курса,

Авторизованный преподаватель курса,

Сторонние сервисы.

**Пользователи:**

Сотрудники компании, собирающие метрики (аналитики, разработчики).

1. **Доступность**
2. Необходимо обеспечить непрерывный мониторинг работоспособности и доступности системы: настройка оповещений команды разработки о возникающих ошибках работы системы.
3. В случае отключения системы должен быть предусмотрен автоматический перезапуск системы
4. Система должна быть доступна как минимум на 95% круглосуточно.
5. Плановое обслуживание по резервному копированию данных не должно влиять на доступность системы.
6. В случае проведения плановых работ уведомление пользователей не предусмотрено.
7. Максимальное время ожидания ответа равно 30 секундам, по истечению времени пользователю должен быть возвращен ответ с кодом 500 “Попробуйте запрос позже”.
8. **Целостность**
9. Должна быть реализована поддержка транзакционных операций по записи, обновлению и удалению данных в системе.
10. База данных должна содержать ограничения уникальности, обязательности наличия значения, ограничения длин для всех строковых полей.
11. Система должна проверять идентификаторы курса и пользователя на существование в соответствующих сервисах.
12. Каждый день с 03:00-04:00 автоматически должно производиться резервное копирование базы данных с дальнейшим сохранением копии в двух хранилищах, расположенных в разных датацентрах.
13. **Совместимость**
14. Взаимодействие клиента с системой должно производиться с использованием http/https протокола.
15. Прием и отправка ответа от сервера производится по REST.
16. Версия базы данных sqllite 3.9.0.
17. **Производительность**
18. Среднее время получения информации сохраненных метриках, после отправки в сервис 5 мин.
19. **Надежность**
20. Критическая неисправность системы (Например: отсутствие возможность получения информации о собранных метриках или отсутствие возможности отправить метрику) должна быть исправлена в течение 12 часов с момента обнаружения неисправности.
21. Среднее время восстановления не должно превышать (сумма времени простоя за период, деленная на количество инцидентов) 1 час в день.
22. Средняя наработка на отказ (чем больше времени между отказами, тем надежнее система. Общее время работы, деленное на количество сбоев) не должна быть меньше 23 часов - 1 сбой в день длительностью 1 час.
23. **Устойчивость**
24. Система должна оповещать пользователей о невозможности работы с системой при ошибках соединения с БД или сторонними системами и отправлять сотрудникам, компании текст c причиной поломки.
25. С неавторизированного участника курса метрики не собираются
26. **Удобство использования**
27. Содержать комментарии к столбцам таблиц и таблицам в базе данных.
28. Система должна предоставлять описание каждого метода API с телом запроса и ответа.
29. **Безопасность**
30. Работа с БД осуществляется с помощью пользователя, у которого есть доступ на выполнение SELECT, UPDATE, DELETE операций к таблицам “Истории активности пользователя”.

**Имеющий права участник компании может:**

* - создать запись с информацией о метрике
* **createAllert** - создать запись с информацией об аллерте
* **createError** - создать запись с информацией об ошибке
* **getErrorHistoryRecordByCriticalLevel** – можно получить информацию о записи по критичности ошибки
* **getHistoryRecordByActionName** - можно получить информацию по имени действия
* **getHistoryRecordByUserId** – можно получить информацию о записи по пользовательскому id

1. **Защита**
2. Должна быть реализована защита от DDoS атак: пользователь может делать не более 3 запросов в секунду на получение сведений из сервиса от одного ip адреса.
3. **Эффективность**

В системе должно быть предусмотрено кеширование обратной связи из бд на срок до 5 минут или при устаревании данных (произошло обновление обратной связи).

1. **Возможность модификации**
2. Архитектура системы должна быть представлена трехуровневой архитектурой. На программном уровне должны быть выделены слои: взаимодействие с БД, слой бизнес-логики, слой представления данных.
3. Внесение изменений в базы данных должно проводиться с использованием миграций для возможности приведения хранилища в предыдущее состояние.
4. **Переносимость**
5. В новой системе должны быть установлены следующие программы: cmake, git
6. Для операционных систем Windows и Linux должны быть представлены файлы для скачивания библиотек sqllite3, moongoes через компонент cmake.
7. Система должна предоставлять возможность сборки docker-образа для его дальнейшего запуска на сервере.
8. **Масштабируемость**
9. Система должна предусматривать возможность балансировки нагрузки при обращении к сервисам, которые предоставляют информацию о пользователях и курсах.
10. В системе должна быть предусмотрена балансировка нагрузки при обращении к API получению информации об активности.
11. **Проверяемость и тестируемость**

Вся бизнес-логика должна быть покрыта unit-тестами. Работа контроллеров также должна быть покрыта интеграционными тестами. Тесты должны запускаться при операции слияния ветки, на которой велась разработка в основную ветку. Выполнение тестов должно быть автоматизировано с помощью систем CI/CD.

**Сборка сервиса**

mkdir build

cd build

cmake ..

cmake --build .

**База данных**

В качестве базы данных используется Sqlite3.

База данных содержит таблицу HistoryRecord.

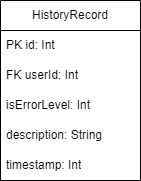


Рисунок 1.Таблица HistoryRecord

Колонки таблицы HistoryRecord:

* id – уникальный идентификатор;
* userId – идентификатор пользователя;
* isErrorLevel – уровень ошибки;
* description – номер курса;
* timestamp – время отправки записи в целочисленном значении.

**Api запросов**

1. POST api/activity/one – создание одной записи активности пользователя historyRecord.

Пример тела запроса:

{

"userId": 1,

"isErrorLevel": 2,

"description": "ошибка открытия каталога",

"timestamp": 1700

}

1. GET api/activity/history\_record\_count – возвращает количество сохраненных активностей пользователя historyRecord.

Пример ответа запроса:

{

1

}

1. GET api/activity/history\_record\_list – возвращает список сохраненных активностей пользователя.

{

{

"id": 1,

"userId": 1,

"isErrorLevel": 2,

"description": "ошибка открытия каталога",

"timestamp": 1717339666

},

{

"id": 2,

"userId": 1,

"isErrorLevel": 0,

"description": "пользователь открыл курс по машинному обучению",

"timestamp": 1717331636

},

{

"id": 3,

"userId": 1,

"isErrorLevel": 3,

"description": "не открылась страница главной",

"timestamp": 1717330610

},

{

"id": 4,

"userId": 1,

"isErrorLevel": 0,

"description": "пользователь зашел в аккаунт",

"timestamp": 1717331230

},

}

1. DELETE api/activity/history\_record\_delete – удаление записи активности

{

"id": 1

}

1. UPDATE api/activity/history\_record\_update – изменение записи об активности

{

"id": 1,

"userId": 1,

"isErrorLevel": 2,

"description": "ошибка открытия каталога",

"timestamp": 1717339666

}

**Структуры**

Структура записи активности:

typedef struct EOP\_history\_record {

long id;

long userId;

long isErrorLevel;

char \*description;

long timestamp;

} EOP\_history\_record;

**Диаграмма классов**

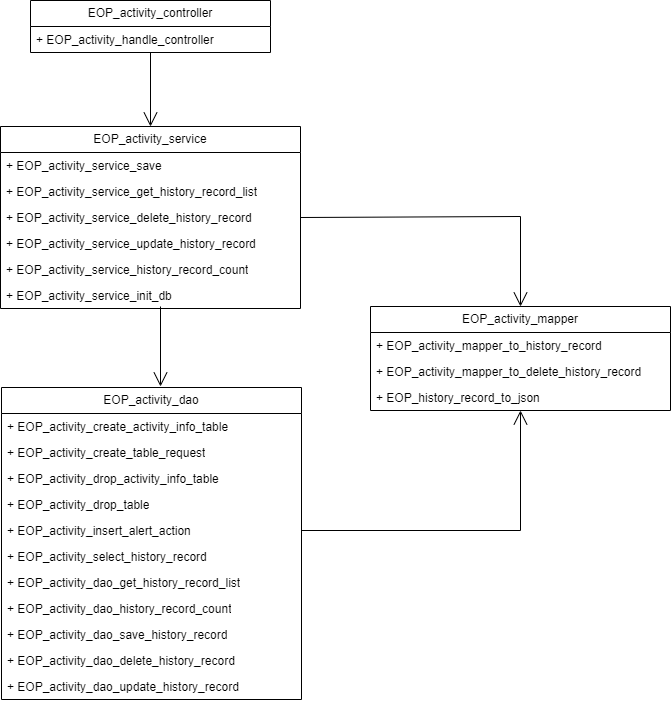


Рисунок 2. – Диаграмма классов сервиса активности

**Упрощенная диаграмма поведения сервиса активности**

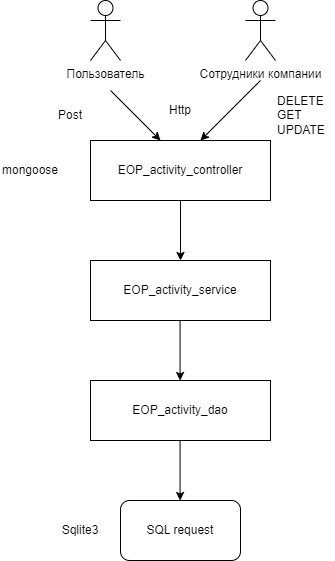


Рисунок 3. – Упрощенная схема поведения