**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.

**Пользователи:**

Авторизованный участник курса,

Авторизованный преподаватель курса,

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 sqllite3.
17. **Производительность**
18. Среднее время получения информации о выборке из 10 комментариев пользователей - 1000 ms. Максимальное время ожидания - 5 секунд.
19. Функциональность уведомления пользователя о создании нового комментария или ответа автора курса должна занимать не более 1 минуты с момента отправки комментария пользователем.
20. **Надежность**
21. Критическая неисправность системы должна быть исправлена в течение 12 часов с момента обнаружения неисправности.
22. Неисправность среднего уровня должна быть исправлена в течение 10 рабочих дней.
23. Неисправность низкого уровня должна быть исправлена в течение 20 рабочих дней.
24. Среднее время восстановления не должно превышать (сумма времени простоя за период, деленная на количество инцидентов) 1 час в день.
25. 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.
26. **Устойчивость**
27. Система должна оповещать пользователей о невозможности работы с системой при ошибках соединения с БД или сторонними системами сообщением ”Запрос не может быть выполнен в данный момент, попробуйте позже”.
28. При попытке публикации комментария к курсу, на который пользователь не подписан, система отображает пользователю сообщение: “Станьте участником курса, чтобы оставить комментарий”.
29. При попытке отправки метрик неавторизованным пользователем сервис отображает пользователю сообщение: “not authorized”
30. **Удобство использования**
31. Содержать комментарии к столбцам таблиц и таблицам в базе данных.
32. **Безопасность**
33. Работа с БД осуществляется с помощью пользователя, у которого есть доступ на выполнение SELECT, UPDATE, DELETE операций к таблицам “Комментарии пользователей”, “Уведомления о комментариях для пользователей”.

**Авторизованный пользователь может:**

* **createHistoryRecord** - создать комментарий с обратной связью на курс
* **createAllert** - создать комментарий с обратной связью на курс
* **createError** - создать комментарий с обратной связью на курс
* **createFeedback** - создать комментарий с обратной связью на курс
* **updateFeedback** - обновить комментарий, если пользователь является автором комментария
* **getHistoryRecordByCriticalLevel** является автором комментария или автором курса
* **criticalLevel** является автором комментария или автором курса
* **getHistoryRecordByActionName** - получить комментарии, оставленные собой
* **getHistoryRecordByUserId** - получить незаблокированные комментарии по курсу, если пользователь является подписчиком курса
* **deleteFeedback** - удалить комментарий, если пользователь является автором комментария

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. Комментарии, оставленные пользователями под курсами, должны быть проверены автором курса перед их отображением всем участникам системы.
4. **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 по получению уведомлений о комментариях.
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.