@startuml

title Регистрация пользователя на платформе

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Открывает страницу регистрации

WebUI -> User: Отображает форму регистрации

User -> WebUI: Вводит данные (email, пароль, имя)

User -> WebUI: Отправляет форму регистрации

WebUI -> ServerDB: POST /users

ServerDB -> ServerDB: Проверяет данные на корректность

alt Данные некорректны

ServerDB -> WebUI: 400 Bad Request

WebUI -> User: Отображает ошибку валидации

else Данные корректны

ServerDB -> ServerDB: Создает нового пользователя

ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает сообщение об успешной регистрации

ServerDB -> User: Отправляет письмо подтверждения на email

end

User -> User: Проверяет email и подтверждает регистрацию

User -> WebUI: Переходит по ссылке подтверждения из письма

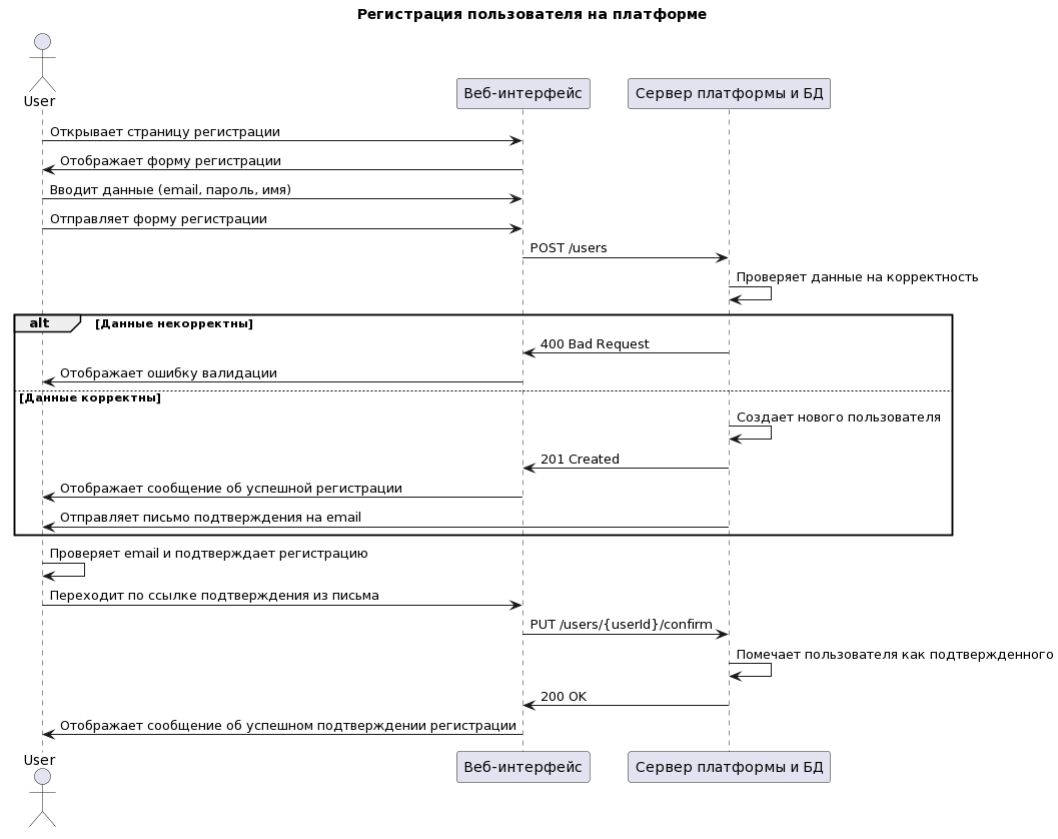
WebUI -> ServerDB: PUT /users/{userId}/confirm

ServerDB -> ServerDB: Помечает пользователя как подтвержденного

ServerDB -> WebUI: 200 OK

WebUI -> User: Отображает сообщение об успешном подтверждении регистрации

@enduml



@startuml

title Авторизация пользователя на платформе через Яндекс ID и OAuth 2.0

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы" as Server

participant "Яндекс ID" as YandexID

User -> WebUI: Переходит на страницу авторизации

WebUI -> User: Перенаправляет на страницу авторизации Яндекс ID

User -> YandexID: Вводит учетные данные Яндекс ID и подтверждает авторизацию

YandexID -> YandexID: Проверяет учетные данные пользователя

YandexID -> Server: Перенаправляет на callback URL с authorization code

Server -> YandexID: Отправляет authorization code

YandexID -> Server: Успешный ответ

Server -> YandexID: Запрашивает access token, используя authorization code

YandexID -> Server: Возвращает access token

Server -> YandexID: Запрашивает информацию о пользователе, используя access token

YandexID -> Server: Возвращает информацию о пользователе

Server -> Server: Создает или обновляет запись пользователя в БД

Server -> Server: Генерирует JWT токен с информацией о пользователе и его ролях

Server -> WebUI: Возвращает JWT токен

WebUI -> User: Сохраняет JWT токен в локальном хранилище (localStorage или cookie)

WebUI -> User: Перенаправляет на главную страницу платформы

loop При каждом запросе к защищенным ресурсам

User -> WebUI: Отправляет запрос к защищенному ресурсу

WebUI -> Server: Отправляет запрос с JWT токеном в заголовке Authorization

Server -> Server: Проверяет и декодирует JWT токен

alt Токен валиден

Server -> Server: Получает информацию о пользователе и его ролях из JWT токена

Server -> Server: Проверяет права доступа пользователя к запрошенному ресурсу

Server -> WebUI: Возвращает запрошенные данные или разрешает выполнение операции

WebUI -> User: Отображает запрошенные данные или результат операции

else Токен невалиден или отсутствует

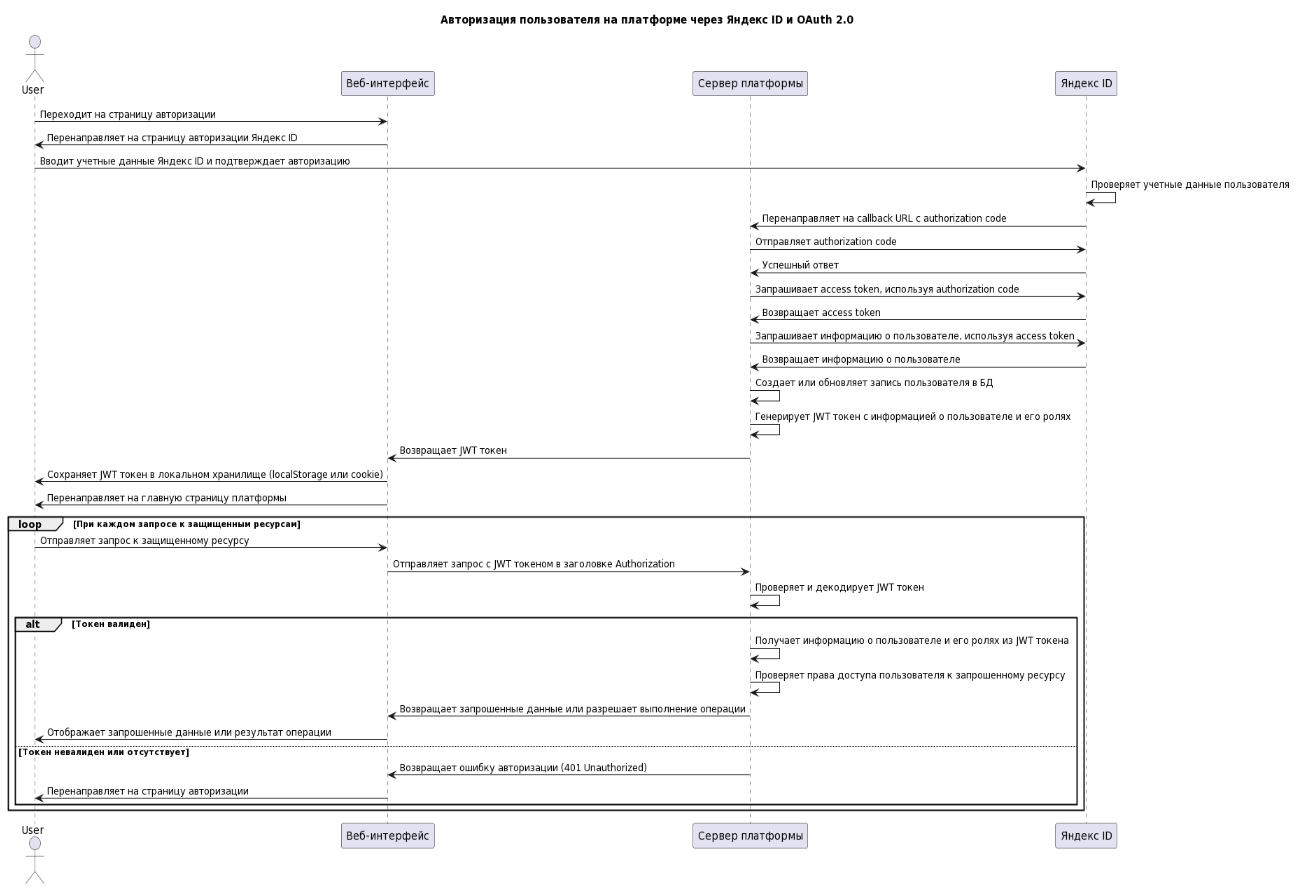
Server -> WebUI: Возвращает ошибку авторизации (401 Unauthorized)

WebUI -> User: Перенаправляет на страницу авторизации

end

end

@enduml



@startuml

title Поиск и фильтрация мероприятий

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Открывает страницу поиска мероприятий

WebUI -> ServerDB: GET /events

ServerDB -> ServerDB: Получает список мероприятий из БД

ServerDB -> WebUI: 200 OK (список мероприятий)

WebUI -> User: Отображает список мероприятий

User -> WebUI: Вводит поисковый запрос

User -> WebUI: Применяет фильтры (город, дата, тема)

WebUI -> ServerDB: GET /events/search?query={searchQuery}

WebUI -> ServerDB: GET /events/filter?city={city}&date={date}&topic={topic}

ServerDB -> ServerDB: Обрабатывает поисковый запрос и фильтры

ServerDB -> ServerDB: Получает отфильтрованный список мероприятий из БД

ServerDB -> WebUI: 200 OK (отфильтрованный список мероприятий)

WebUI -> User: Отображает отфильтрованный список мероприятий

User -> WebUI: Изменяет параметры фильтрации

WebUI -> ServerDB: GET /events/filter?city={updatedCity}&date={updatedDate}&topic={updatedTopic}

ServerDB -> ServerDB: Обрабатывает обновленные параметры фильтрации

ServerDB -> ServerDB: Получает обновленный список мероприятий из БД

ServerDB -> WebUI: 200 OK (обновленный список мероприятий)

WebUI -> User: Отображает обновленный список мероприятий

User -> WebUI: Выбирает мероприятие из списка

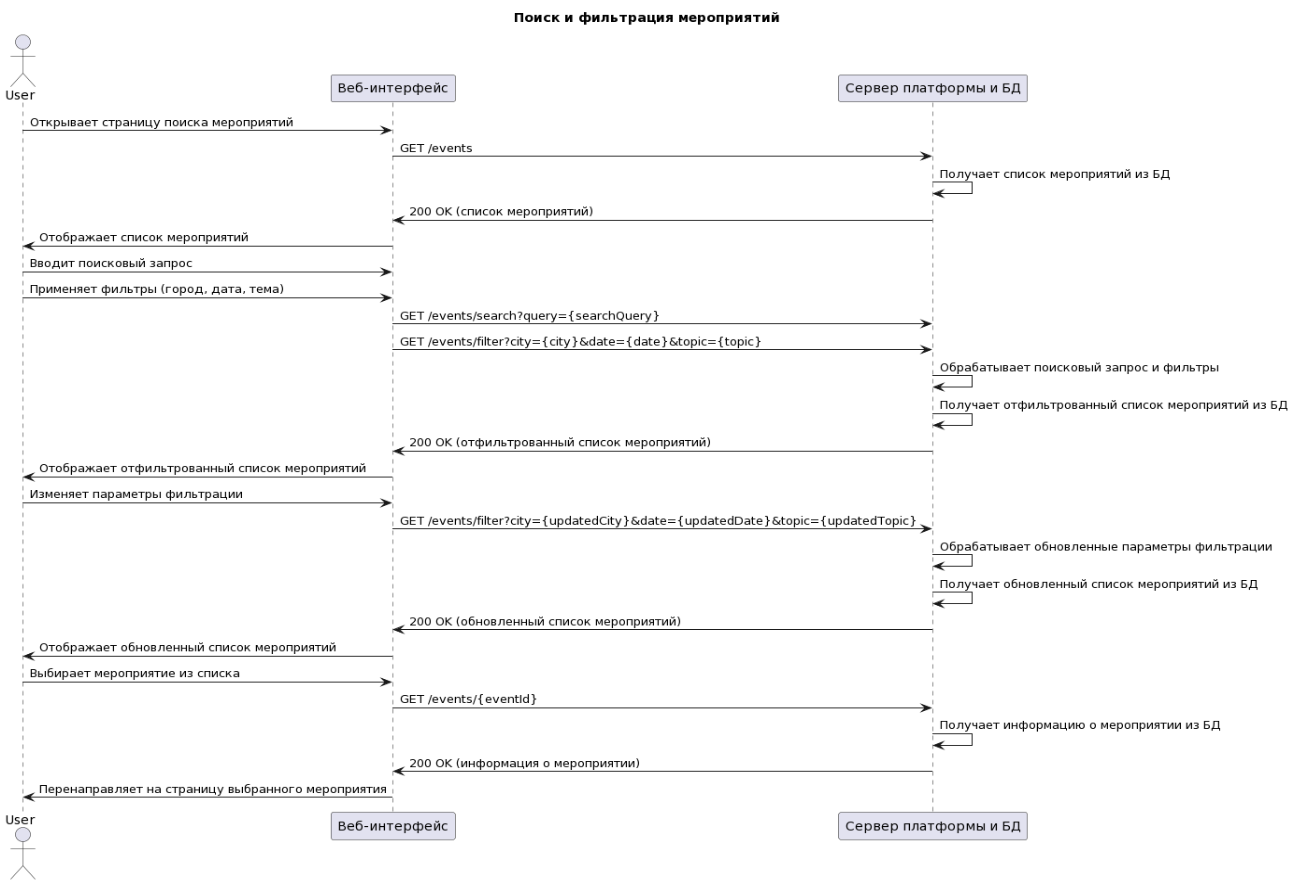
WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> User: Перенаправляет на страницу выбранного мероприятия

@enduml



@startuml

title Просмотр информации о мероприятии

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит на страницу мероприятия

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> User: Отображает информацию о мероприятии

User -> WebUI: Просматривает описание мероприятия

User -> WebUI: Просматривает дату, время и место проведения

User -> WebUI: Просматривает список спикеров

User -> WebUI: Просматривает программу мероприятия

alt Мероприятие доступно для регистрации

WebUI -> User: Отображает кнопку "Зарегистрироваться"

else Мероприятие недоступно для регистрации

WebUI -> User: Отображает статус "Регистрация закрыта" или "Мероприятие завершено"

end

User -> WebUI: Нажимает на ссылку "Показать на карте"

WebUI -> ServerDB: GET /events/{eventId}/location

ServerDB -> ServerDB: Получает данные геолокации мероприятия из БД

ServerDB -> WebUI: 200 OK (данные геолокации)

WebUI -> User: Отображает карту с местом проведения мероприятия

User -> WebUI: Нажимает на ссылку "Поделиться мероприятием"

WebUI -> User: Отображает опции для поделиться ссылкой на мероприятие в соцсетях

User -> WebUI: Нажимает на кнопку "Добавить в избранное"

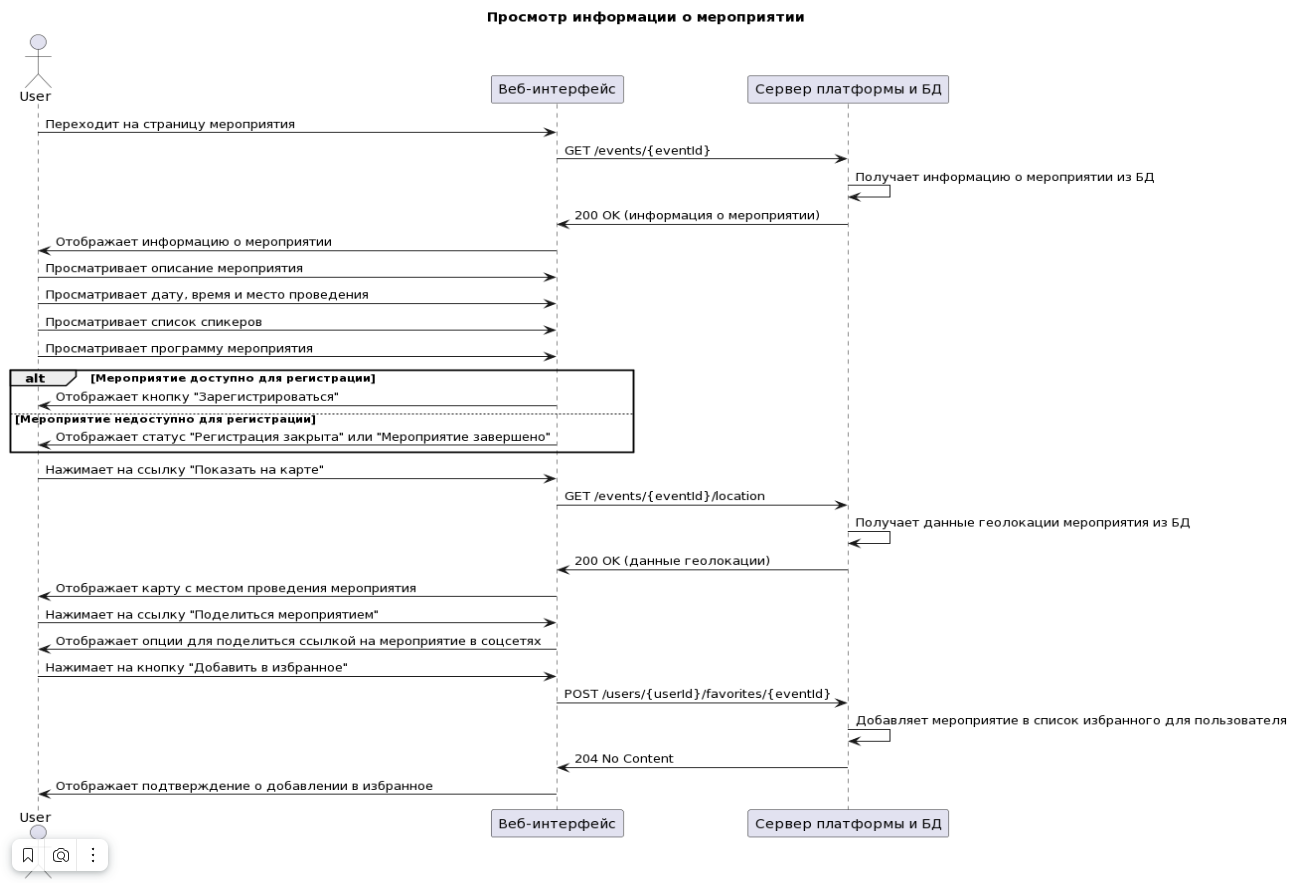
WebUI -> ServerDB: POST /users/{userId}/favorites/{eventId}

ServerDB -> ServerDB: Добавляет мероприятие в список избранного для пользователя

ServerDB -> WebUI: 204 No Content

WebUI -> User: Отображает подтверждение о добавлении в избранное

@enduml



@startuml

title Регистрация на мероприятие

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Нажимает кнопку "Зарегистрироваться" на странице мероприятия

WebUI -> ServerDB: GET /users/{userId}/events/{eventId}

ServerDB -> ServerDB: Проверяет, зарегистрирован ли пользователь на мероприятие

ServerDB -> WebUI: 404 Not Found

alt Пользователь уже зарегистрирован

WebUI -> User: Отображает сообщение "Вы уже зарегистрированы на это мероприятие"

else Пользователь не зарегистрирован

WebUI -> User: Отображает форму регистрации на мероприятие

User -> WebUI: Заполняет форму регистрации

User -> WebUI: Отправляет заполненную форму регистрации

WebUI -> ServerDB: POST /events/{eventId}/register

ServerDB -> ServerDB: Сохраняет данные регистрации пользователя на мероприятие

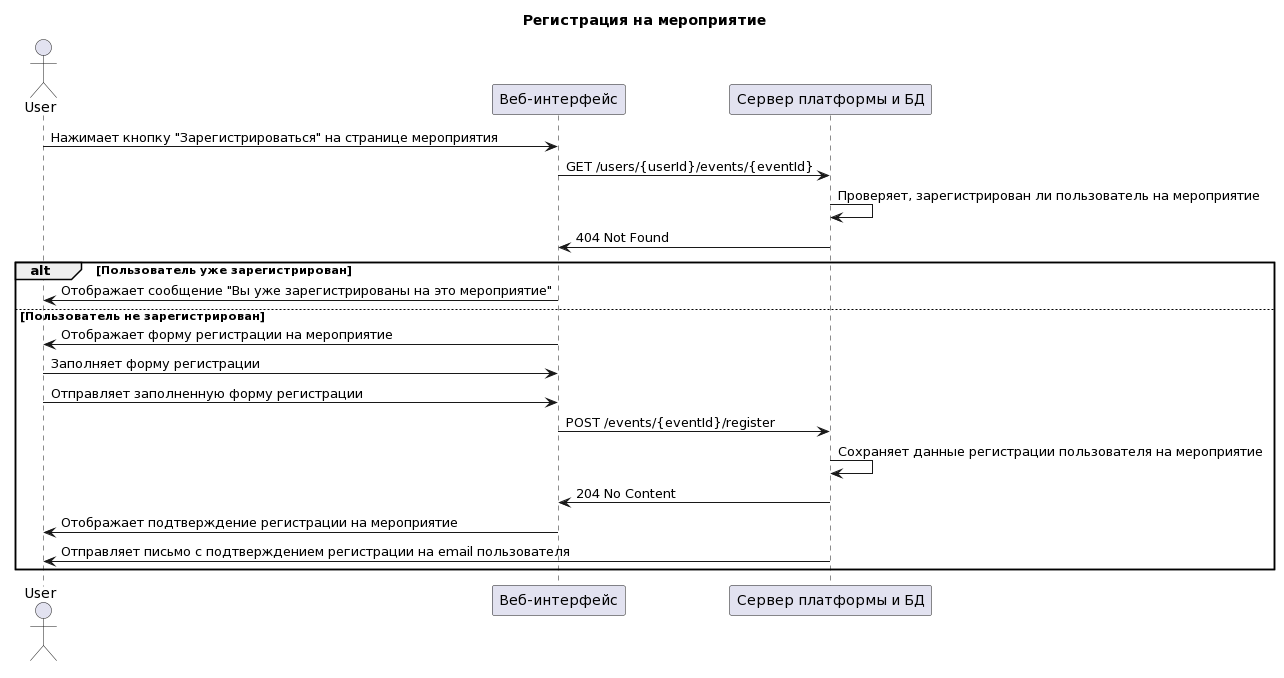
ServerDB -> WebUI: 204 No Content

WebUI -> User: Отображает подтверждение регистрации на мероприятие

ServerDB -> User: Отправляет письмо с подтверждением регистрации на email пользователя

end

@enduml



@startuml

title Оплата участия в мероприятии

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

participant "Платежный шлюз" as PaymentGateway

User -> WebUI: Выбирает тип билета на странице мероприятия

WebUI -> ServerDB: GET /events/{eventId}/tickets/{ticketId}

ServerDB -> ServerDB: Получает информацию о выбранном типе билета из БД

ServerDB -> WebUI: 200 OK (информация о билете)

WebUI -> User: Отображает информацию о выбранном типе билета и кнопку "Оплатить"

User -> WebUI: Нажимает кнопку "Оплатить"

WebUI -> PaymentGateway: Перенаправляет пользователя на страницу оплаты

PaymentGateway -> User: Отображает форму ввода платежных данных

User -> PaymentGateway: Вводит платежные данные и подтверждает оплату

PaymentGateway -> PaymentGateway: Обрабатывает платеж

PaymentGateway -> ServerDB: POST /events/{eventId}/tickets/{ticketId}/payments

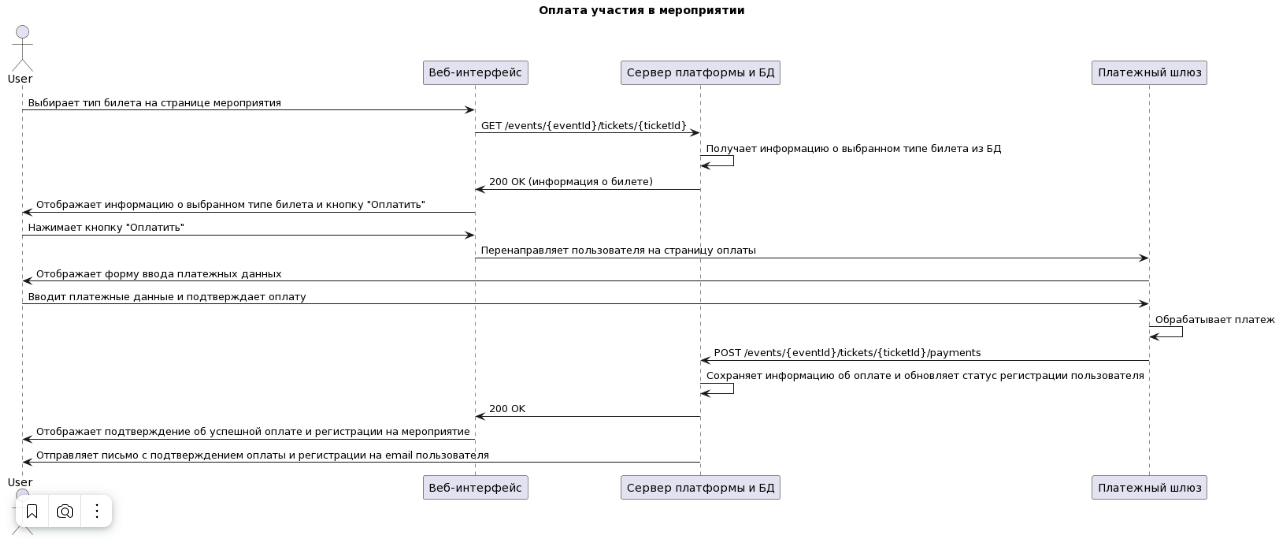
ServerDB -> ServerDB: Сохраняет информацию об оплате и обновляет статус регистрации пользователя

ServerDB -> WebUI: 200 OK

WebUI -> User: Отображает подтверждение об успешной оплате и регистрации на мероприятие

ServerDB -> User: Отправляет письмо с подтверждением оплаты и регистрации на email пользователя

@enduml



@startuml

title Получение уведомлений о мероприятии

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Подписывается на уведомления о мероприятии

WebUI -> ServerDB: POST /users/{userId}/events/{eventId}/notifications

ServerDB -> ServerDB: Сохраняет информацию о подписке пользователя на уведомления

ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает подтверждение о подписке на уведомления

loop Перед началом мероприятия

ServerDB -> ServerDB: Проверяет время до начала мероприятия

alt Время отправки уведомления

ServerDB -> User: Отправляет email-уведомление о предстоящем мероприятии

ServerDB -> User: Отправляет push-уведомление о предстоящем мероприятии

end

end

ServerDB -> User: Отправляет email-уведомление о начале мероприятия

ServerDB -> User: Отправляет push-уведомление о начале мероприятия

loop После завершения мероприятия

ServerDB -> ServerDB: Проверяет статус мероприятия

alt Мероприятие завершено

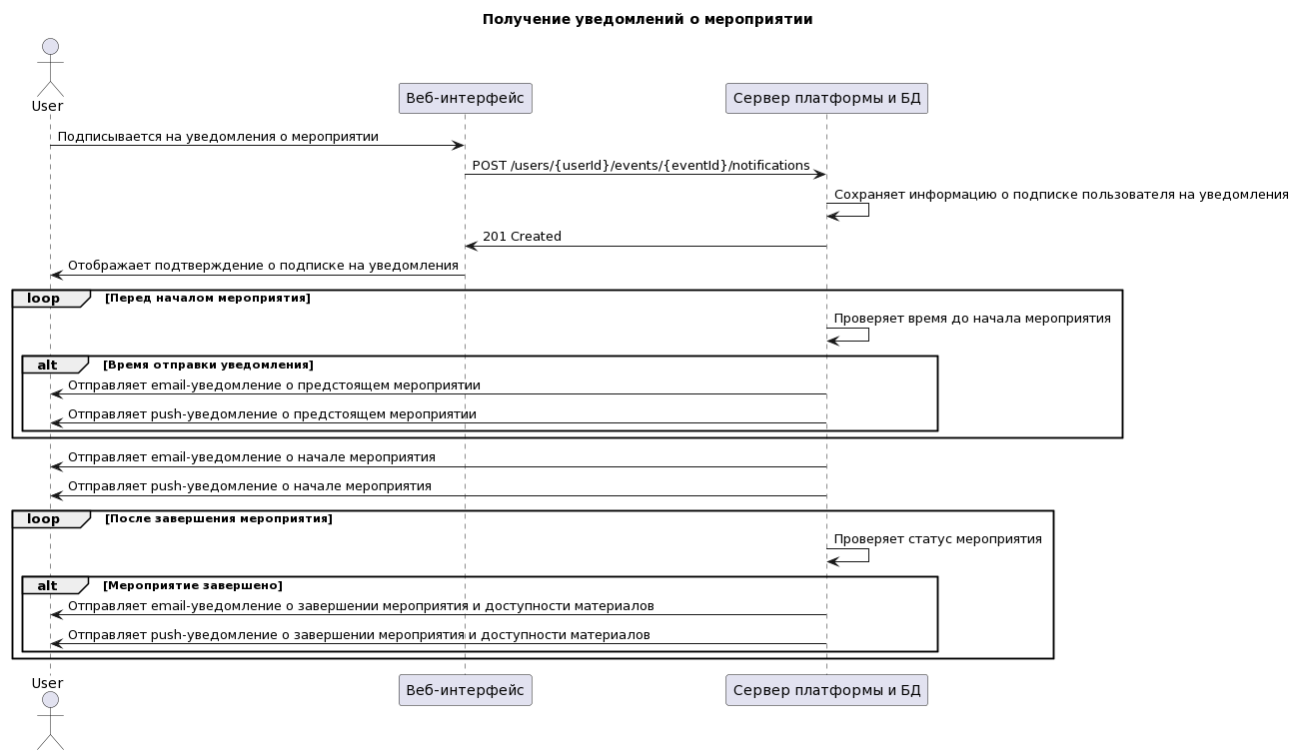
ServerDB -> User: Отправляет email-уведомление о завершении мероприятия и доступности материалов

ServerDB -> User: Отправляет push-уведомление о завершении мероприятия и доступности материалов

end

end

@enduml



@startuml

title Просмотр онлайн-трансляции мероприятия

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

participant "Сервис трансляции" as StreamingService

User -> WebUI: Переходит на страницу онлайн-трансляции мероприятия

WebUI -> ServerDB: GET /users/{userId}/events/{eventId}

ServerDB -> ServerDB: Проверяет, зарегистрирован ли пользователь на мероприятие

ServerDB -> WebUI: 200 OK

alt Пользователь зарегистрирован

WebUI -> User: Отображает плеер онлайн-трансляции

WebUI -> StreamingService: GET /events/{eventId}/stream

StreamingService -> WebUI: 200 OK (поток онлайн-трансляции)

WebUI -> User: Воспроизводит онлайн-трансляцию

else Пользователь не зарегистрирован

WebUI -> User: Отображает сообщение о необходимости регистрации для просмотра трансляции

end

loop Во время трансляции

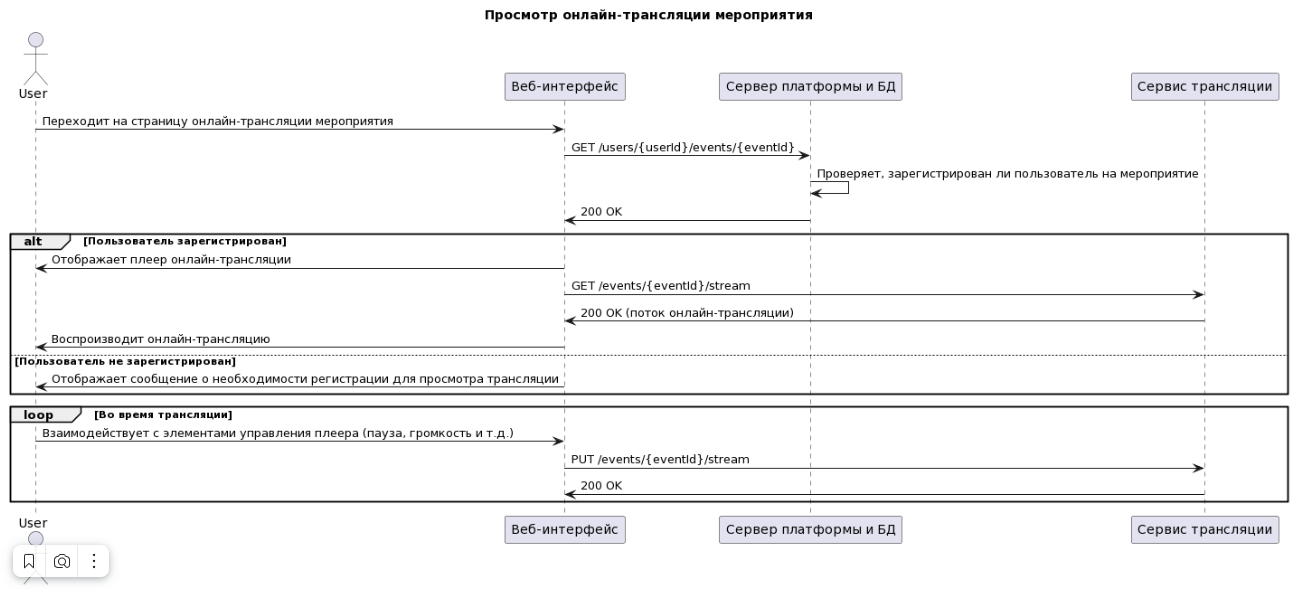
User -> WebUI: Взаимодействует с элементами управления плеера (пауза, громкость и т.д.)

WebUI -> StreamingService: PUT /events/{eventId}/stream

StreamingService -> WebUI: 200 OK

end

@enduml



@startuml

title Взаимодействие с другими участниками мероприятия (Яндекс.Мессенджер, опросы)

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

participant "Яндекс.Мессенджер" as YandexMessenger

User -> WebUI: Переходит на страницу мероприятия

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> User: Отображает страницу мероприятия со ссылкой на чат в Яндекс.Мессенджере

User -> WebUI: Переходит по ссылке на чат мероприятия

WebUI -> YandexMessenger: Перенаправляет пользователя в чат мероприятия

User -> YandexMessenger: Присоединяется к чату мероприятия

User -> YandexMessenger: Отправляет сообщения и взаимодействует с другими участниками

User -> WebUI: Возвращается на страницу мероприятия

WebUI -> ServerDB: GET /events/{eventId}/polls

ServerDB -> ServerDB: Получает список активных опросов для мероприятия из БД

ServerDB -> WebUI: 200 OK (список опросов)

WebUI -> User: Отображает список активных опросов на странице мероприятия

User -> WebUI: Выбирает опрос и переходит на страницу опроса

WebUI -> ServerDB: GET /polls/{pollId}

ServerDB -> ServerDB: Получает информацию об опросе и вариантах ответов из БД

ServerDB -> WebUI: 200 OK (информация об опросе и вариантах ответов)

WebUI -> User: Отображает страницу опроса с вопросом и вариантами ответов

User -> WebUI: Выбирает вариант ответа и отправляет свой ответ

WebUI -> ServerDB: POST /polls/{pollId}/responses

ServerDB -> ServerDB: Сохраняет ответ пользователя в БД

ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает подтверждение о принятии ответа на опрос

loop Периодически (при наличии новых ответов)

WebUI -> ServerDB: GET /polls/{pollId}/results

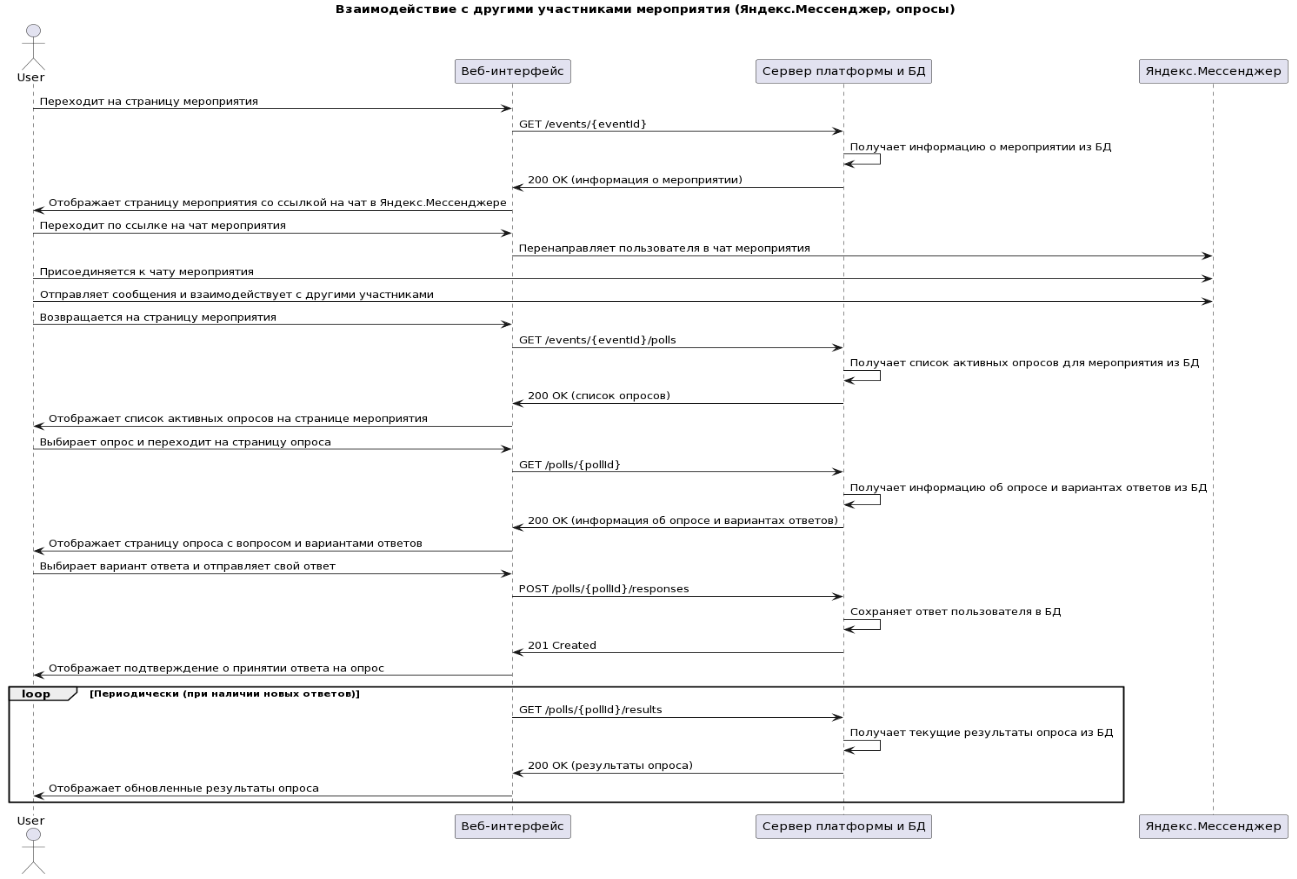
ServerDB -> ServerDB: Получает текущие результаты опроса из БД

ServerDB -> WebUI: 200 OK (результаты опроса)

WebUI -> User: Отображает обновленные результаты опроса

end

@enduml



@startuml

title Получение записи онлайн-трансляции после завершения мероприятия

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

participant "Сервис хранения записей" as StorageService

User -> WebUI: Переходит на страницу мероприятия после его завершения

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> User: Отображает страницу мероприятия с ссылкой на запись трансляции

User -> WebUI: Нажимает на ссылку для просмотра записи

WebUI -> ServerDB: GET /events/{eventId}/recording

ServerDB -> StorageService: GET /recordings/{recordingId}

StorageService -> ServerDB: 200 OK (запись трансляции)

ServerDB -> WebUI: 200 OK (запись трансляции)

WebUI -> User: Воспроизводит запись трансляции

User -> WebUI: Нажимает на кнопку "Скачать запись"

WebUI -> ServerDB: GET /events/{eventId}/recording/download

ServerDB -> StorageService: GET /recordings/{recordingId}/download

StorageService -> ServerDB: 200 OK (файл записи)

ServerDB -> WebUI: 200 OK (файл записи)

WebUI -> User: Скачивает файл записи трансляции на устройство пользователя

@enduml



@startuml

title Создание мероприятия организатором

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Organizer -> WebUI: Переходит на страницу создания мероприятия

WebUI -> Organizer: Отображает форму создания мероприятия

Organizer -> WebUI: Заполняет форму с информацией о мероприятии

Organizer -> WebUI: Загружает изображение для мероприятия

Organizer -> WebUI: Отправляет заполненную форму

WebUI -> ServerDB: POST /events

ServerDB -> ServerDB: Проверяет и сохраняет данные мероприятия

ServerDB -> ServerDB: Генерирует уникальный идентификатор мероприятия

ServerDB -> WebUI: 201 Created (информация о созданном мероприятии)

WebUI -> Organizer: Отображает подтверждение о создании мероприятия

Organizer -> WebUI: Переходит на страницу созданного мероприятия

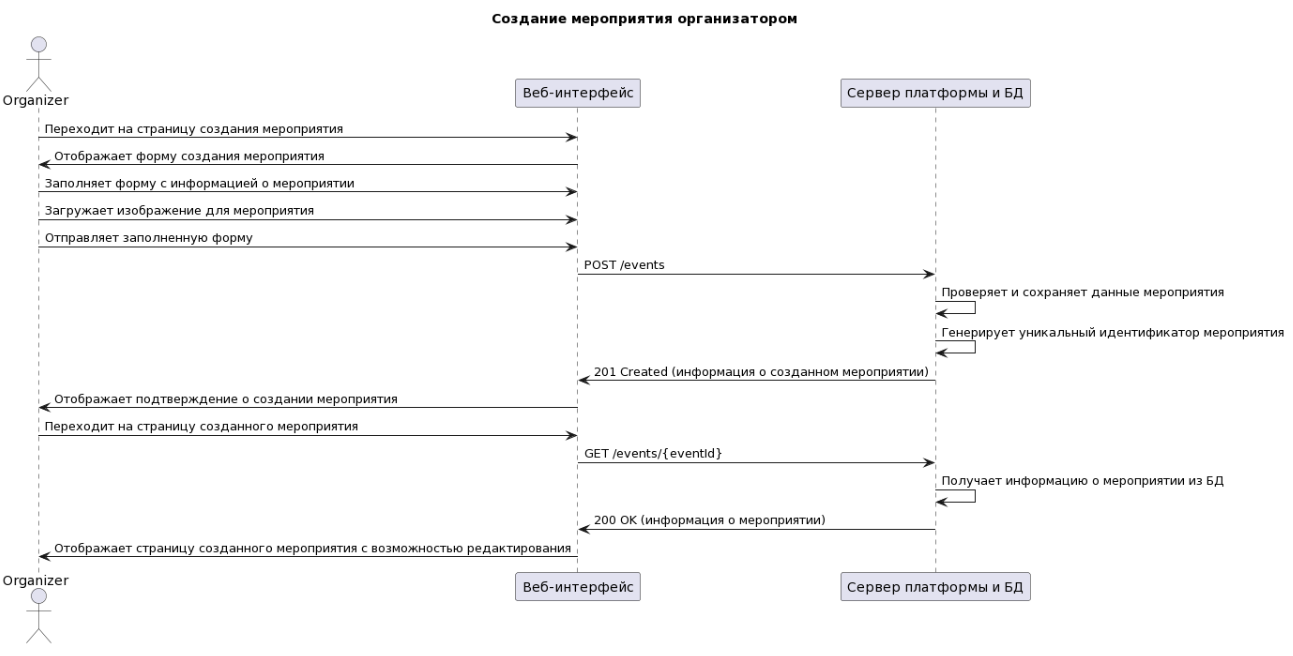
WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Organizer: Отображает страницу созданного мероприятия с возможностью редактирования

@enduml



@startuml

title Редактирование информации о мероприятии организатором

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Organizer -> WebUI: Переходит на страницу редактирования мероприятия

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Organizer: Отображает форму редактирования мероприятия с предзаполненными данными

Organizer -> WebUI: Изменяет информацию о мероприятии

Organizer -> WebUI: Загружает новое изображение для мероприятия (опционально)

Organizer -> WebUI: Отправляет обновленную форму

WebUI -> ServerDB: PUT /events/{eventId}

ServerDB -> ServerDB: Проверяет и сохраняет обновленные данные мероприятия

ServerDB -> WebUI: 200 OK (обновленная информация о мероприятии)

WebUI -> Organizer: Отображает подтверждение о сохранении изменений

Organizer -> WebUI: Переходит на страницу мероприятия

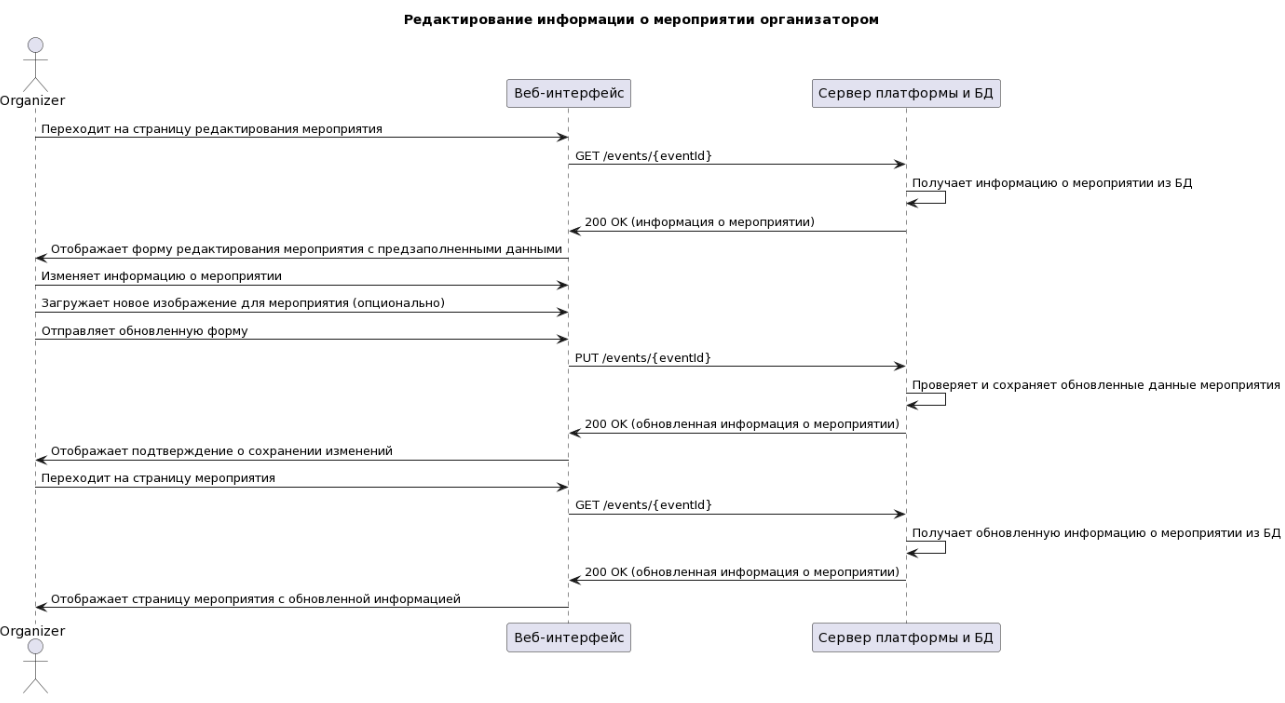
WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает обновленную информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (обновленная информация о мероприятии)

WebUI -> Organizer: Отображает страницу мероприятия с обновленной информацией

@enduml



@startuml

title Публикация мероприятия на платформе

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Organizer -> WebUI: Переходит на страницу созданного мероприятия

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Organizer: Отображает страницу мероприятия с кнопкой "Опубликовать"

Organizer -> WebUI: Нажимает на кнопку "Опубликовать"

WebUI -> ServerDB: PUT /events/{eventId}/publish

ServerDB -> ServerDB: Проверяет, что все обязательные поля мероприятия заполнены

alt Все обязательные поля заполнены

ServerDB -> ServerDB: Изменяет статус мероприятия на "Опубликовано"

ServerDB -> WebUI: 200 OK (обновленная информация о мероприятии)

WebUI -> Organizer: Отображает подтверждение о публикации мероприятия

else Не все обязательные поля заполнены

ServerDB -> WebUI: 400 Bad Request (список незаполненных полей)

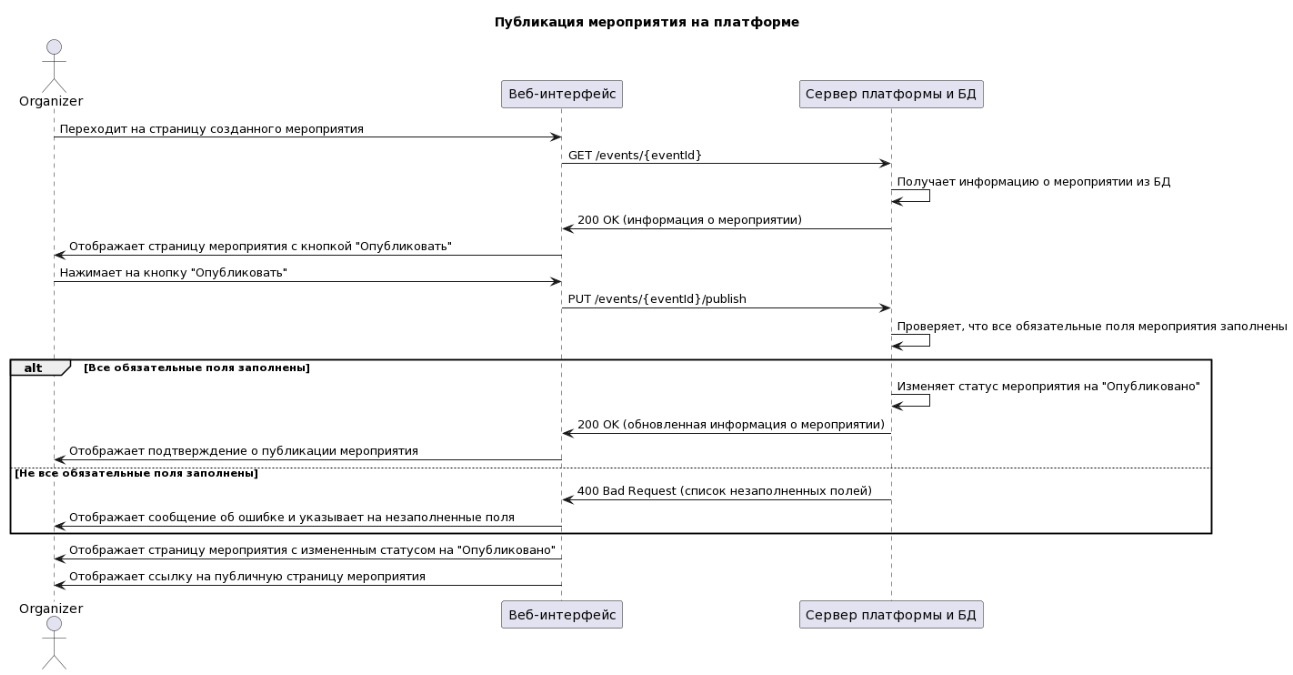
WebUI -> Organizer: Отображает сообщение об ошибке и указывает на незаполненные поля

end

WebUI -> Organizer: Отображает страницу мероприятия с измененным статусом на "Опубликовано"

WebUI -> Organizer: Отображает ссылку на публичную страницу мероприятия

@enduml



@startuml

title Модерация мероприятий администратором платформы

actor Administrator

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Administrator -> WebUI: Переходит на страницу модерации мероприятий

WebUI -> ServerDB: GET /admin/events?status=pending

ServerDB -> ServerDB: Получает список мероприятий, ожидающих модерации, из БД

ServerDB -> WebUI: 200 OK (список мероприятий)

WebUI -> Administrator: Отображает список мероприятий, ожидающих модерации

loop Для каждого мероприятия, ожидающего модерации

Administrator -> WebUI: Выбирает мероприятие для модерации

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о выбранном мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Administrator: Отображает информацию о выбранном мероприятии

Administrator -> WebUI: Проверяет информацию о мероприятии

Administrator -> WebUI: Принимает решение о публикации или отклонении мероприятия

alt Решение - публикация

WebUI -> ServerDB: PUT /admin/events/{eventId}/approve

ServerDB -> ServerDB: Изменяет статус мероприятия на "Опубликовано"

else Решение - отклонение

WebUI -> ServerDB: PUT /admin/events/{eventId}/reject

ServerDB -> ServerDB: Изменяет статус мероприятия на "Отклонено"

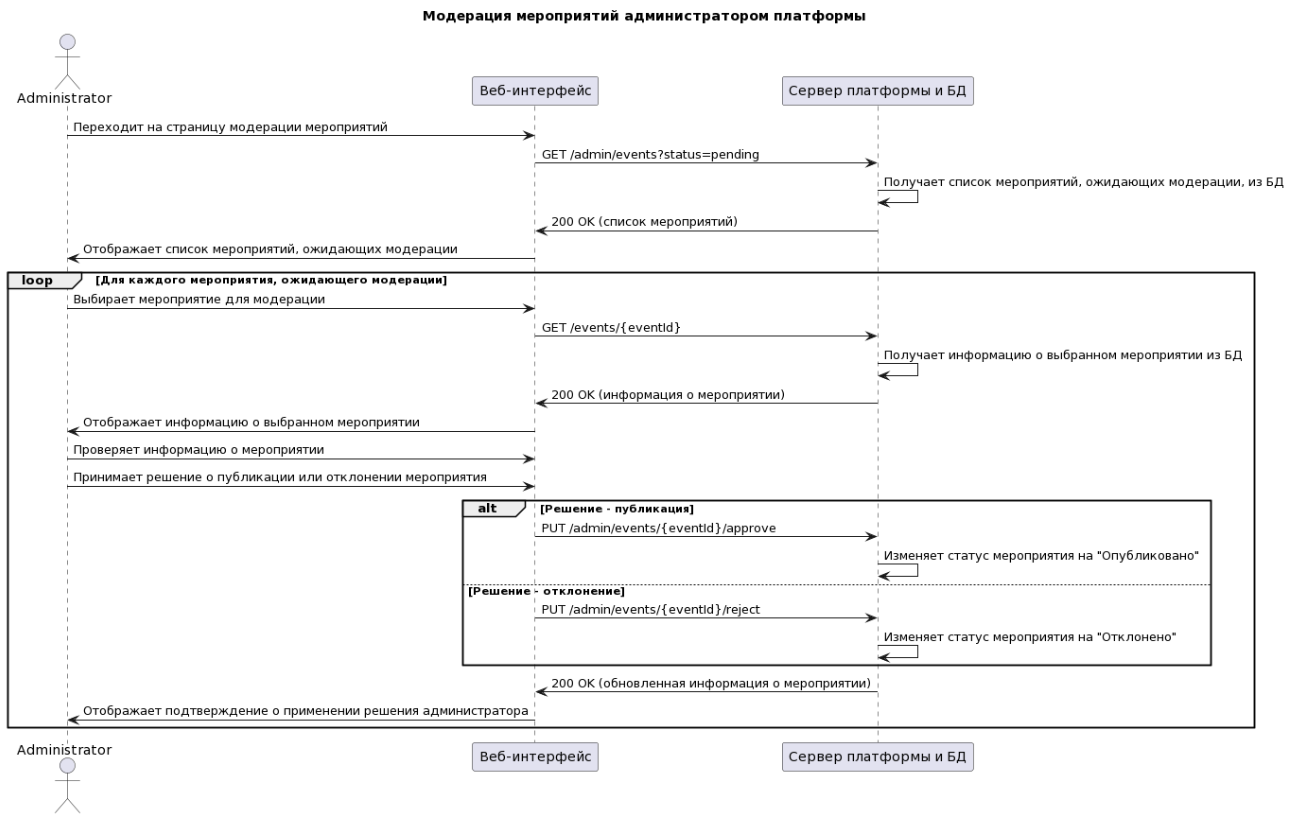
end

ServerDB -> WebUI: 200 OK (обновленная информация о мероприятии)

WebUI -> Administrator: Отображает подтверждение о применении решения администратора

end

@enduml



@startuml

title Просмотр списка зарегистрированных участников организатором

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Organizer -> WebUI: Переходит на страницу мероприятия в разделе "Мои мероприятия"

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Organizer: Отображает страницу мероприятия с кнопкой "Список участников"

Organizer -> WebUI: Нажимает на кнопку "Список участников"

WebUI -> ServerDB: GET /events/{eventId}/participants

ServerDB -> ServerDB: Получает список зарегистрированных участников мероприятия из БД

ServerDB -> WebUI: 200 OK (список участников)

WebUI -> Organizer: Отображает список зарегистрированных участников мероприятия

Organizer -> WebUI: Применяет фильтры и сортировку к списку участников (опционально)

WebUI -> ServerDB: GET /events/{eventId}/participants?filter={filter}&sort={sort}

ServerDB -> ServerDB: Применяет фильтры и сортировку к списку участников

ServerDB -> WebUI: 200 OK (отфильтрованный и отсортированный список участников)

WebUI -> Organizer: Отображает отфильтрованный и отсортированный список участников

Organizer -> WebUI: Выбирает участника из списка для просмотра подробной информации (опционально)

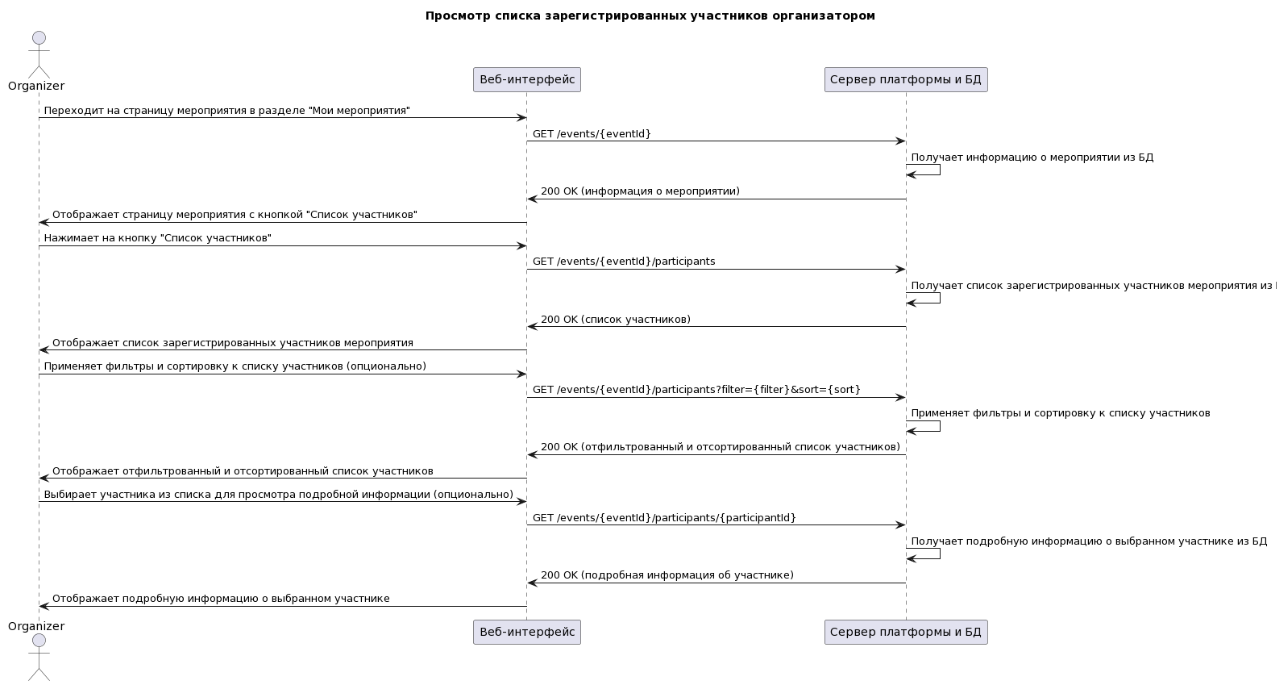
WebUI -> ServerDB: GET /events/{eventId}/participants/{participantId}

ServerDB -> ServerDB: Получает подробную информацию о выбранном участнике из БД

ServerDB -> WebUI: 200 OK (подробная информация об участнике)

WebUI -> Organizer: Отображает подробную информацию о выбранном участнике

@enduml



@startuml

title Взаимодействие организатора с участниками мероприятия

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Organizer -> WebUI: Переходит на страницу мероприятия в разделе "Мои мероприятия"

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Organizer: Отображает страницу мероприятия с кнопкой "Сообщения участникам"

Organizer -> WebUI: Нажимает на кнопку "Сообщения участникам"

WebUI -> ServerDB: GET /events/{eventId}/participants

ServerDB -> ServerDB: Получает список зарегистрированных участников мероприятия из БД

ServerDB -> WebUI: 200 OK (список участников)

WebUI -> Organizer: Отображает форму отправки сообщения участникам

Organizer -> WebUI: Вводит текст сообщения и выбирает получателей (всем участникам или выборочно)

Organizer -> WebUI: Отправляет сообщение

WebUI -> ServerDB: POST /events/{eventId}/messages

ServerDB -> ServerDB: Сохраняет сообщение и связывает его с выбранными участниками

ServerDB -> WebUI: 201 Created (информация об отправленном сообщении)

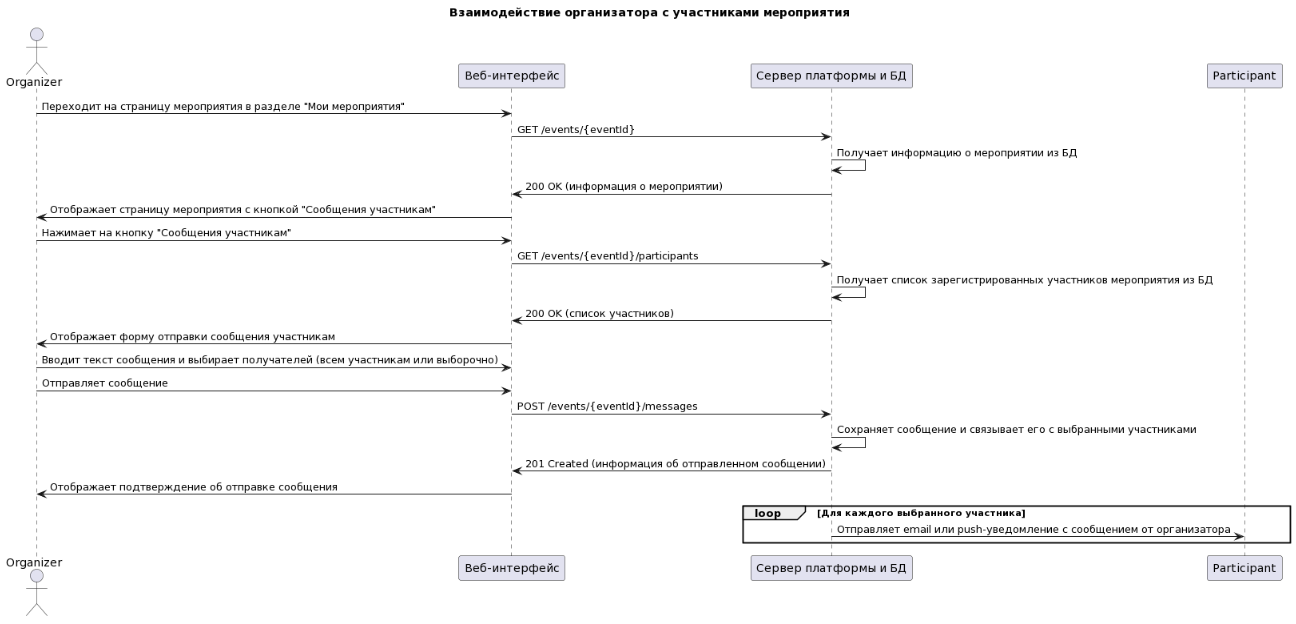
WebUI -> Organizer: Отображает подтверждение об отправке сообщения

loop Для каждого выбранного участника

ServerDB -> Participant: Отправляет email или push-уведомление с сообщением от организатора

end

@enduml



@startuml

title Проведение онлайн-трансляции мероприятия организатором

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

participant "Сервис трансляции" as StreamingService

Organizer -> WebUI: Переходит на страницу мероприятия в разделе "Мои мероприятия"

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Organizer: Отображает страницу мероприятия с кнопкой "Начать трансляцию"

Organizer -> WebUI: Нажимает на кнопку "Начать трансляцию"

WebUI -> StreamingService: POST /streams

StreamingService -> StreamingService: Создает новую трансляцию и генерирует ссылку на нее

StreamingService -> WebUI: 201 Created (информация о созданной трансляции)

WebUI -> ServerDB: PUT /events/{eventId}

ServerDB -> ServerDB: Обновляет информацию о мероприятии с ссылкой на трансляцию

ServerDB -> WebUI: 200 OK (обновленная информация о мероприятии)

WebUI -> Organizer: Отображает интерфейс управления трансляцией с ссылкой на трансляцию

Organizer -> WebUI: Настраивает параметры трансляции (качество, метаданные и т.д.)

WebUI -> StreamingService: PUT /streams/{streamId}

StreamingService -> StreamingService: Применяет настройки к текущей трансляции

StreamingService -> WebUI: 200 OK

WebUI -> Organizer: Отображает подтверждение о применении настроек трансляции

Organizer -> WebUI: Начинает трансляцию

WebUI -> StreamingService: PUT /streams/{streamId}/start

StreamingService -> StreamingService: Начинает принимать и передавать видео и аудио данные

StreamingService -> WebUI: 200 OK

WebUI -> Organizer: Отображает индикатор активной трансляции

loop Во время трансляции

Organizer -> WebUI: Взаимодействует с интерфейсом управления трансляцией

WebUI -> StreamingService: PUT /streams/{streamId}

StreamingService -> StreamingService: Обрабатывает команды управления трансляцией

StreamingService -> WebUI: 200 OK

WebUI -> Organizer: Отображает результаты выполнения команд

end

Organizer -> WebUI: Завершает трансляцию

WebUI -> StreamingService: PUT /streams/{streamId}/stop

StreamingService -> StreamingService: Останавливает прием и передачу видео и аудио данных

StreamingService -> WebUI: 200 OK

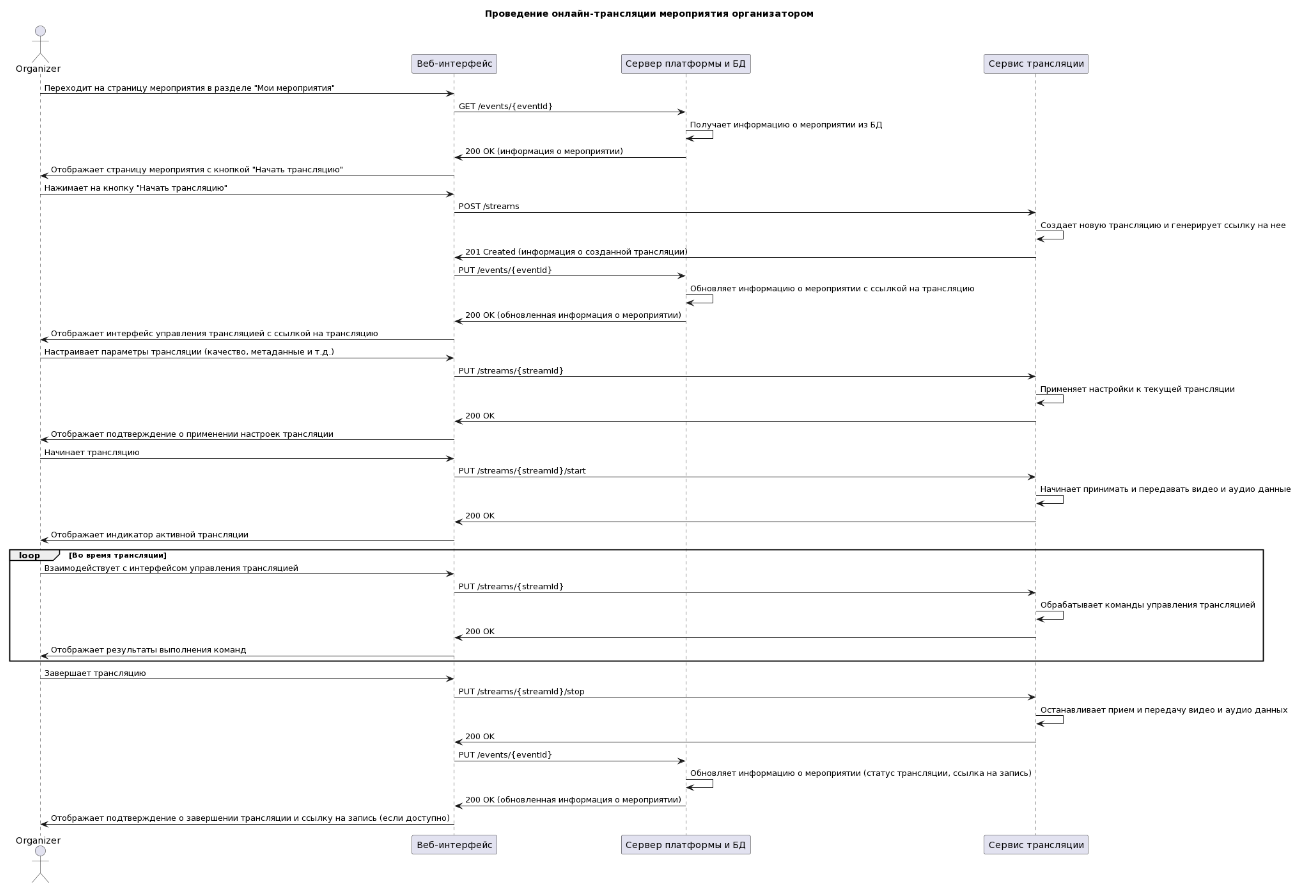
WebUI -> ServerDB: PUT /events/{eventId}

ServerDB -> ServerDB: Обновляет информацию о мероприятии (статус трансляции, ссылка на запись)

ServerDB -> WebUI: 200 OK (обновленная информация о мероприятии)

WebUI -> Organizer: Отображает подтверждение о завершении трансляции и ссылку на запись (если доступно)

@enduml



@startuml

title Получение обратной связи и аналитики по завершении мероприятия организатором

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Organizer -> WebUI: Переходит на страницу завершенного мероприятия в разделе "Мои мероприятия"

WebUI -> ServerDB: GET /events/{eventId}?include=analytics,feedback

ServerDB -> ServerDB: Получает информацию о мероприятии и связанные с ним данные (отзывы, оценки, статистику) из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии, отзывы, оценки, статистика)

WebUI -> Organizer: Отображает страницу мероприятия с разделом "Аналитика и обратная связь"

Organizer -> WebUI: Просматривает общую статистику по мероприятию (количество участников, средняя оценка и т.д.)

Organizer -> WebUI: Просматривает список отзывов и оценок от участников

Organizer -> WebUI: Фильтрует и сортирует отзывы и оценки по различным критериям

Organizer -> WebUI: Просматривает аналитические графики и диаграммы (динамика регистраций, распределение оценок и т.д.)

WebUI -> ServerDB: GET /events/{eventId}/analytics

ServerDB -> ServerDB: Получает необходимые данные для построения графиков и диаграмм из БД

ServerDB -> WebUI: 200 OK (данные для аналитики)

WebUI -> Organizer: Отображает графики и диаграммы с аналитикой мероприятия

Organizer -> WebUI: Экспортирует отчет с обратной связью и аналитикой в формате PDF или CSV

WebUI -> ServerDB: GET /events/{eventId}/report

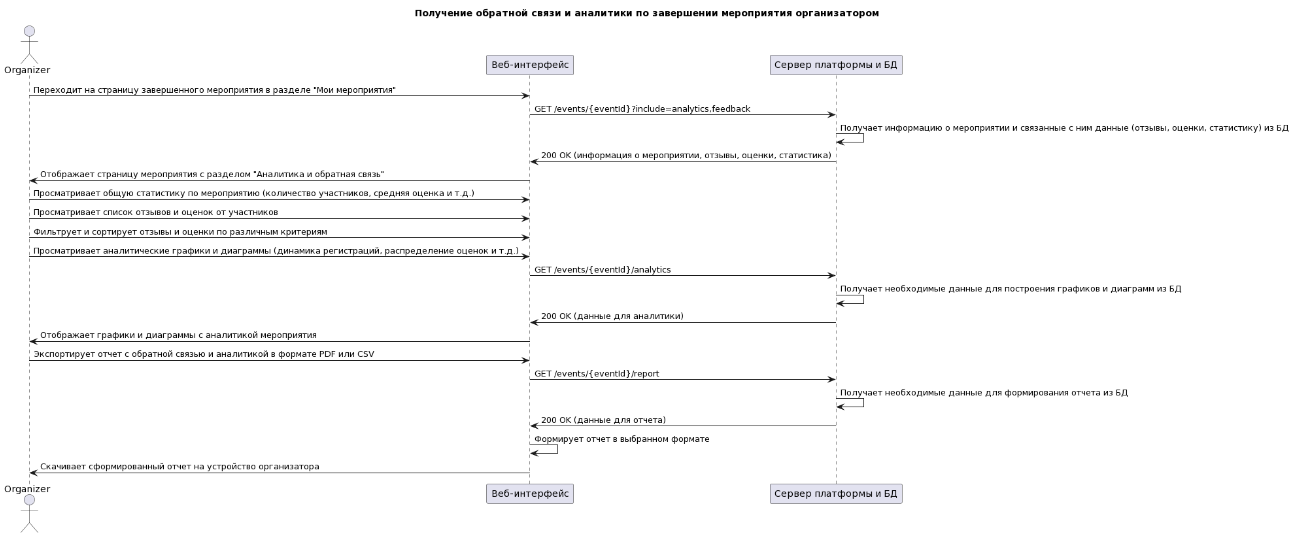
ServerDB -> ServerDB: Получает необходимые данные для формирования отчета из БД

ServerDB -> WebUI: 200 OK (данные для отчета)

WebUI -> WebUI: Формирует отчет в выбранном формате

WebUI -> Organizer: Скачивает сформированный отчет на устройство организатора

@enduml



@startuml

title Экспорт списка участников мероприятия организатором

actor Organizer

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

Organizer -> WebUI: Переходит на страницу мероприятия в разделе "Мои мероприятия"

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> Organizer: Отображает страницу мероприятия с кнопкой "Экспорт списка участников"

Organizer -> WebUI: Нажимает на кнопку "Экспорт списка участников"

WebUI -> ServerDB: GET /events/{eventId}/participants

ServerDB -> ServerDB: Получает список зарегистрированных участников мероприятия из БД

ServerDB -> WebUI: 200 OK (список участников)

WebUI -> Organizer: Отображает диалоговое окно с опциями экспорта (формат файла, поля для включения)

Organizer -> WebUI: Выбирает желаемый формат файла (CSV, Excel, JSON) и поля для включения в экспорт

Organizer -> WebUI: Подтверждает экспорт

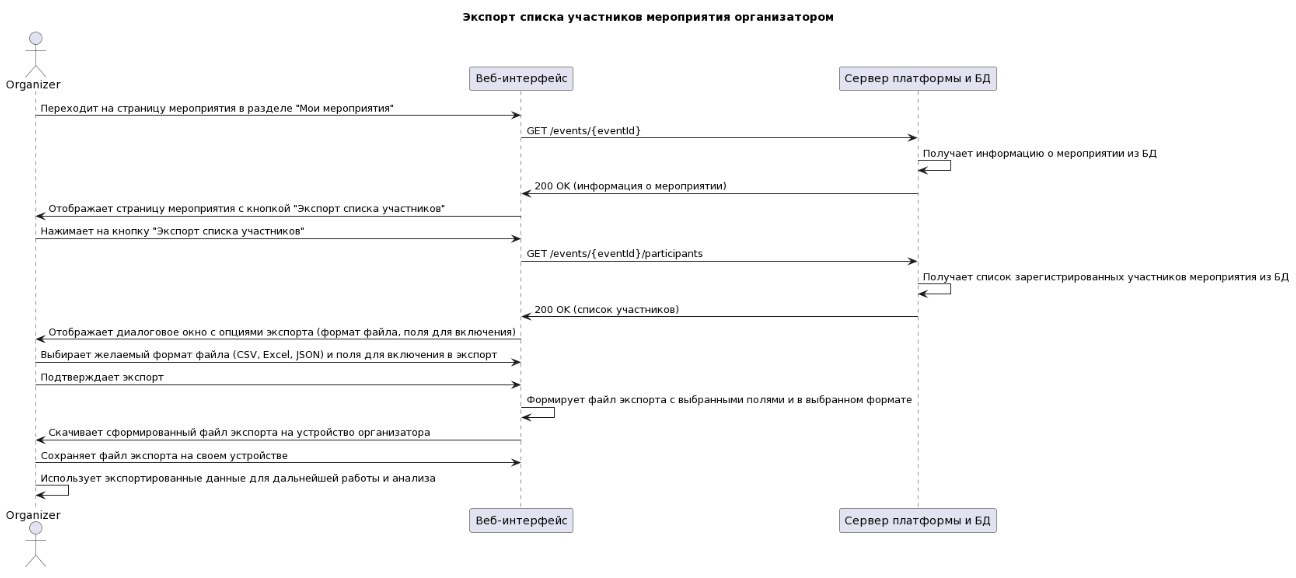
WebUI -> WebUI: Формирует файл экспорта с выбранными полями и в выбранном формате

WebUI -> Organizer: Скачивает сформированный файл экспорта на устройство организатора

Organizer -> WebUI: Сохраняет файл экспорта на своем устройстве

Organizer -> Organizer: Использует экспортированные данные для дальнейшей работы и анализа

@enduml



@startuml

title Управление настройками профиля и уведомлений пользователем

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит в раздел "Настройки профиля"

WebUI -> ServerDB: GET /users/{userId}/profile

ServerDB -> ServerDB: Получает текущие настройки профиля пользователя из БД

ServerDB -> WebUI: 200 OK (настройки профиля пользователя)

WebUI -> User: Отображает форму редактирования настроек профиля с текущими значениями

User -> WebUI: Изменяет настройки профиля (имя, фамилия, фото, описание и т.д.)

User -> WebUI: Сохраняет изменения

WebUI -> ServerDB: PUT /users/{userId}/profile

ServerDB -> ServerDB: Сохраняет обновленные настройки профиля пользователя в БД

ServerDB -> WebUI: 200 OK

WebUI -> User: Отображает подтверждение об успешном сохранении изменений

User -> WebUI: Переходит в раздел "Настройки уведомлений"

WebUI -> ServerDB: GET /users/{userId}/notifications/settings

ServerDB -> ServerDB: Получает текущие настройки уведомлений пользователя из БД

ServerDB -> WebUI: 200 OK (настройки уведомлений пользователя)

WebUI -> User: Отображает форму редактирования настроек уведомлений с текущими значениями

User -> WebUI: Изменяет настройки уведомлений (email, push, темы уведомлений и т.д.)

User -> WebUI: Сохраняет изменения

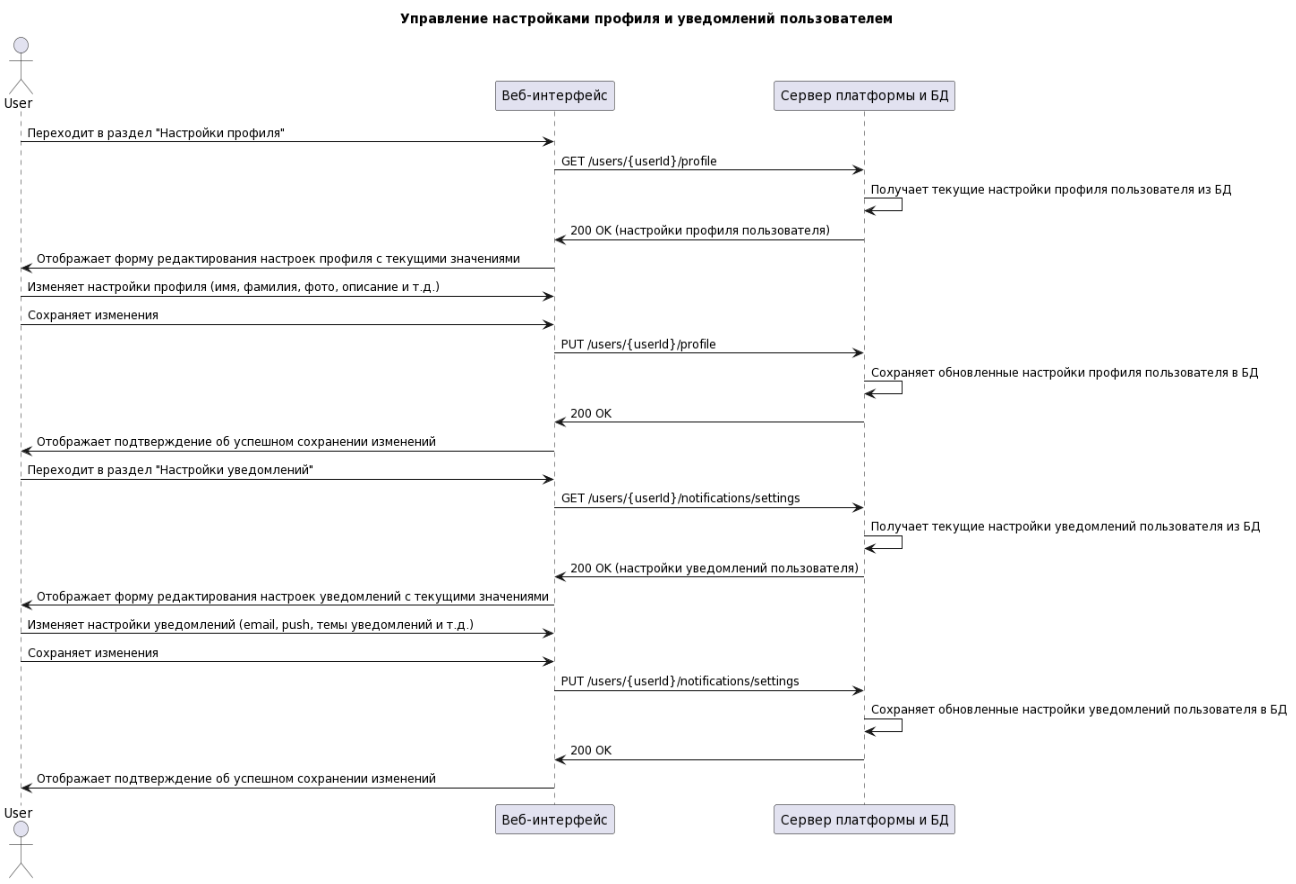
WebUI -> ServerDB: PUT /users/{userId}/notifications/settings

ServerDB -> ServerDB: Сохраняет обновленные настройки уведомлений пользователя в БД

ServerDB -> WebUI: 200 OK

WebUI -> User: Отображает подтверждение об успешном сохранении изменений

@enduml



@startuml

title Редактирование профиля пользователя

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит в раздел "Мой профиль"

WebUI -> ServerDB: GET /users/{userId}

ServerDB -> ServerDB: Получает информацию о профиле пользователя из БД

ServerDB -> WebUI: 200 OK (информация о профиле пользователя)

WebUI -> User: Отображает страницу профиля пользователя с текущей информацией и кнопкой "Редактировать"

User -> WebUI: Нажимает на кнопку "Редактировать"

WebUI -> User: Отображает форму редактирования профиля с текущими значениями полей

User -> WebUI: Изменяет информацию в полях формы (имя, фамилия, фото, описание, ссылки на соцсети и т.д.)

User -> WebUI: Сохраняет изменения

WebUI -> ServerDB: PUT /users/{userId}

ServerDB -> ServerDB: Сохраняет обновленную информацию о профиле пользователя в БД

ServerDB -> WebUI: 200 OK

WebUI -> User: Отображает подтверждение об успешном сохранении изменений

User -> WebUI: Просматривает обновленную информацию на странице своего профиля

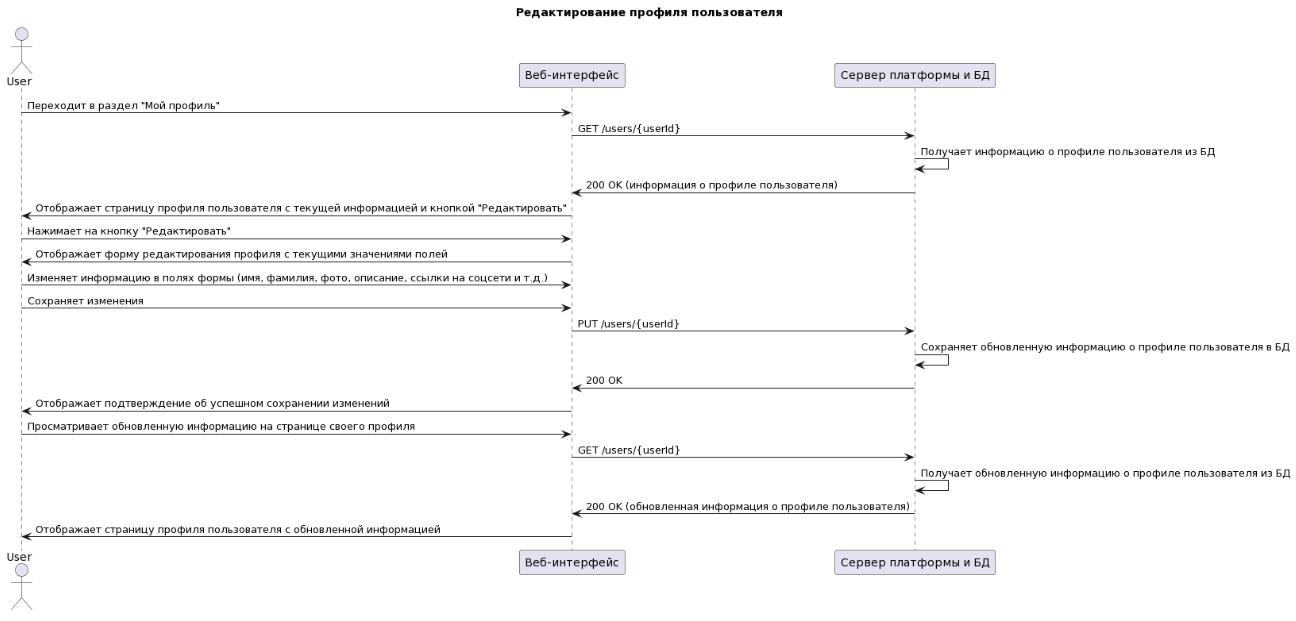
WebUI -> ServerDB: GET /users/{userId}

ServerDB -> ServerDB: Получает обновленную информацию о профиле пользователя из БД

ServerDB -> WebUI: 200 OK (обновленная информация о профиле пользователя)

WebUI -> User: Отображает страницу профиля пользователя с обновленной информацией

@enduml



@startuml

title Добавление мероприятия в избранное

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит на страницу интересующего мероприятия

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает информацию о мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о мероприятии)

WebUI -> User: Отображает страницу мероприятия с кнопкой "Добавить в избранное"

User -> WebUI: Нажимает на кнопку "Добавить в избранное"

WebUI -> ServerDB: POST /users/{userId}/favorites/{eventId}

ServerDB -> ServerDB: Добавляет связь между пользователем и мероприятием в БД

ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает подтверждение о добавлении мероприятия в избранное

User -> WebUI: Переходит в раздел "Избранные мероприятия" в личном кабинете

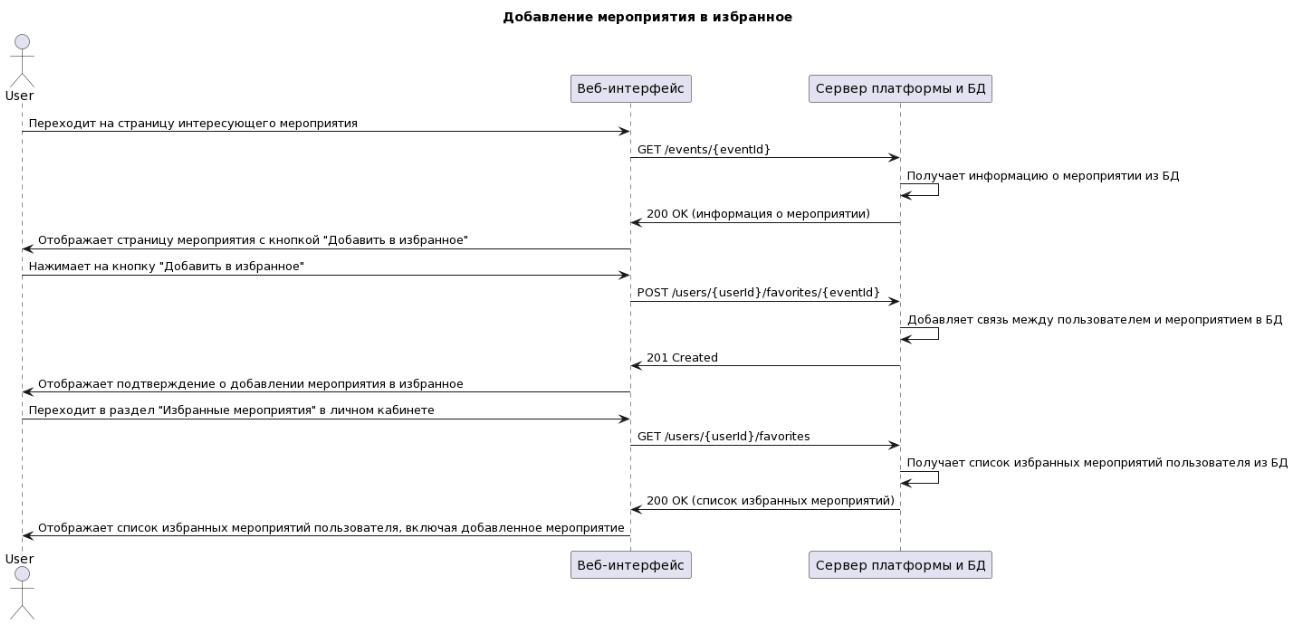
WebUI -> ServerDB: GET /users/{userId}/favorites

ServerDB -> ServerDB: Получает список избранных мероприятий пользователя из БД

ServerDB -> WebUI: 200 OK (список избранных мероприятий)

WebUI -> User: Отображает список избранных мероприятий пользователя, включая добавленное мероприятие

@enduml



@startuml

title Поиск и просмотр прошедших мероприятий

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит на страницу поиска мероприятий

WebUI -> User: Отображает форму поиска мероприятий с фильтрами и опцией "Прошедшие мероприятия"

User -> WebUI: Выбирает опцию "Прошедшие мероприятия"

User -> WebUI: Применяет дополнительные фильтры (тематика, место проведения, дата)

User -> WebUI: Нажимает на кнопку "Найти"

WebUI -> ServerDB: GET /events?type=past&topic={topic}&location={location}&date={date}

ServerDB -> ServerDB: Выполняет поиск прошедших мероприятий в БД по указанным фильтрам

ServerDB -> WebUI: 200 OK (список прошедших мероприятий)

WebUI -> User: Отображает список найденных прошедших мероприятий

User -> WebUI: Выбирает интересующее прошедшее мероприятие из списка

WebUI -> ServerDB: GET /events/{eventId}

ServerDB -> ServerDB: Получает подробную информацию о выбранном прошедшем мероприятии из БД

ServerDB -> WebUI: 200 OK (информация о прошедшем мероприятии)

WebUI -> User: Отображает страницу с подробной информацией о выбранном прошедшем мероприятии

User -> WebUI: Просматривает информацию о прошедшем мероприятии (описание, дата проведения, место проведения, спикеры, материалы)

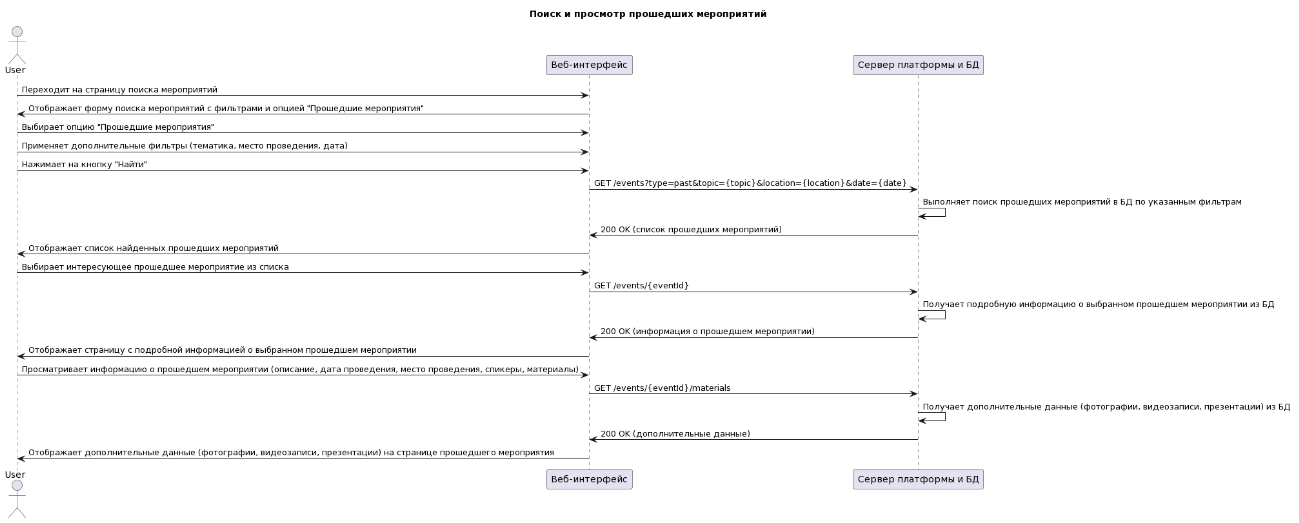
WebUI -> ServerDB: GET /events/{eventId}/materials

ServerDB -> ServerDB: Получает дополнительные данные (фотографии, видеозаписи, презентации) из БД

ServerDB -> WebUI: 200 OK (дополнительные данные)

WebUI -> User: Отображает дополнительные данные (фотографии, видеозаписи, презентации) на странице прошедшего мероприятия

@enduml



@startuml

title Взаимодействие с другими пользователями на платформе (личные сообщения)

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит в раздел "Сообщения" в личном кабинете

WebUI -> ServerDB: GET /users/{userId}/messages

ServerDB -> ServerDB: Получает список диалогов пользователя из БД

ServerDB -> WebUI: 200 OK (список диалогов)

WebUI -> User: Отображает список диалогов пользователя

User -> WebUI: Выбирает диалог с другим пользователем или нажимает "Новое сообщение"

alt Выбран существующий диалог

WebUI -> ServerDB: GET /users/{userId}/messages/{dialogId}

ServerDB -> ServerDB: Получает историю сообщений выбранного диалога из БД

ServerDB -> WebUI: 200 OK (история сообщений)

WebUI -> User: Отображает историю сообщений выбранного диалога

else Новое сообщение

WebUI -> User: Отображает форму создания нового сообщения

User -> WebUI: Вводит имя пользователя или выбирает из списка контактов

User -> WebUI: Вводит текст сообщения

User -> WebUI: Нажимает "Отправить"

WebUI -> ServerDB: POST /users/{userId}/messages

ServerDB -> ServerDB: Сохраняет новое сообщение и создает новый диалог (если необходимо) в БД

ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает отправленное сообщение в новом диалоге

end

loop Обмен сообщениями

User -> WebUI: Вводит текст сообщения

User -> WebUI: Нажимает "Отправить"

WebUI -> ServerDB: POST /users/{userId}/messages/{dialogId}

ServerDB -> ServerDB: Сохраняет новое сообщение в БД

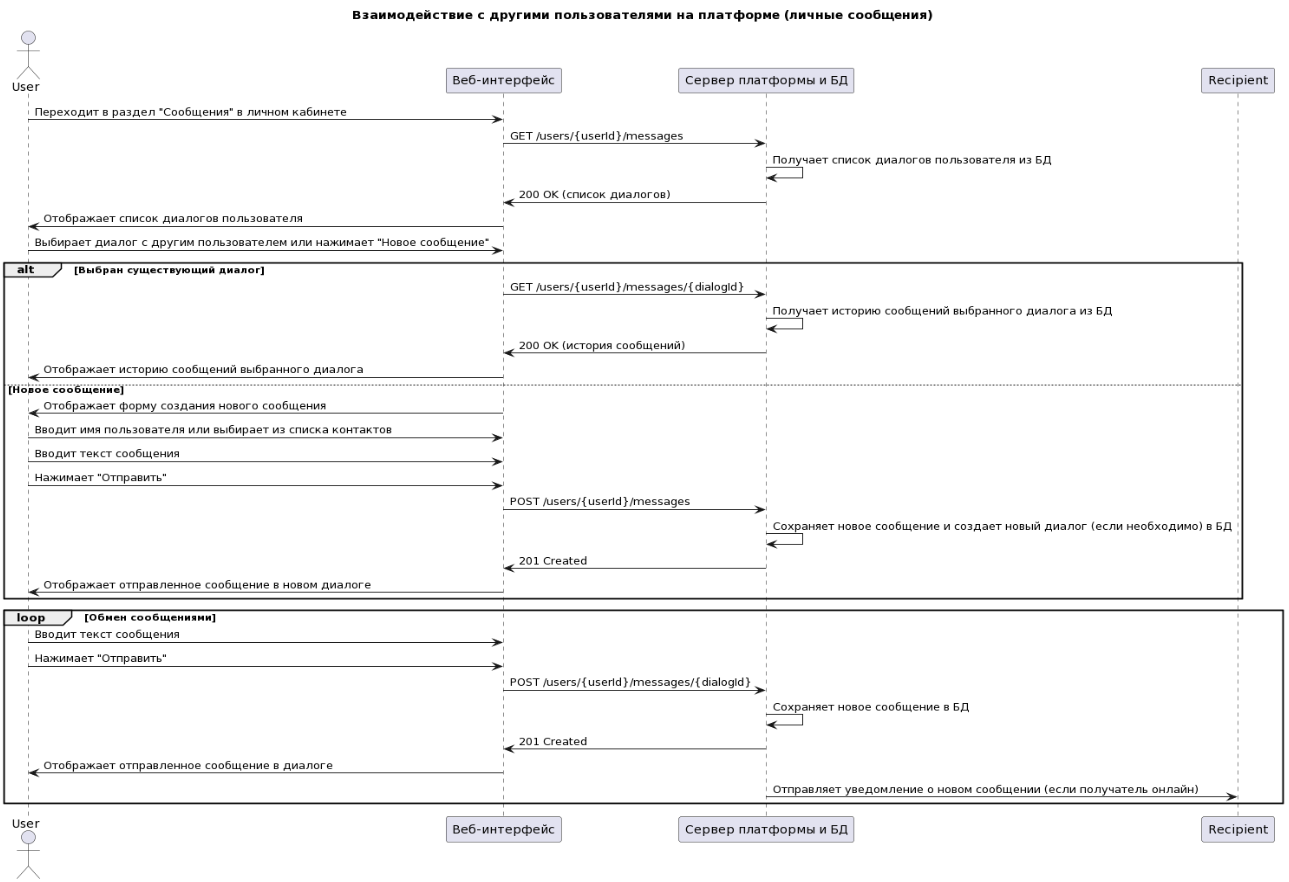
ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает отправленное сообщение в диалоге

ServerDB -> Recipient: Отправляет уведомление о новом сообщении (если получатель онлайн)

end

@enduml



@startuml

title Взаимодействие с другими пользователями на платформе (добавление в друзья)

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит на страницу профиля другого пользователя

WebUI -> ServerDB: GET /users/{userId}

ServerDB -> ServerDB: Получает информацию о профиле пользователя из БД

ServerDB -> WebUI: 200 OK (информация о профиле пользователя)

WebUI -> User: Отображает страницу профиля пользователя с кнопкой "Добавить в друзья"

User -> WebUI: Нажимает на кнопку "Добавить в друзья"

WebUI -> ServerDB: POST /users/{userId}/friends/{friendId}

ServerDB -> ServerDB: Создает запись о запросе дружбы в БД

ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает подтверждение об отправке запроса дружбы

ServerDB -> Recipient: Отправляет уведомление о новом запросе дружбы

Recipient -> WebUI: Переходит в раздел "Запросы дружбы" в личном кабинете

WebUI -> ServerDB: GET /users/{userId}/friends/requests

ServerDB -> ServerDB: Получает список запросов дружбы пользователя из БД

ServerDB -> WebUI: 200 OK (список запросов дружбы)

WebUI -> Recipient: Отображает список запросов дружбы пользователя

Recipient -> WebUI: Принимает или отклоняет запрос дружбы

WebUI -> ServerDB: PUT /users/{userId}/friends/requests/{requestId}

ServerDB -> ServerDB: Обновляет статус запроса дружбы в БД

alt Запрос принят

ServerDB -> ServerDB: Создает связь "друзья" между пользователями в БД

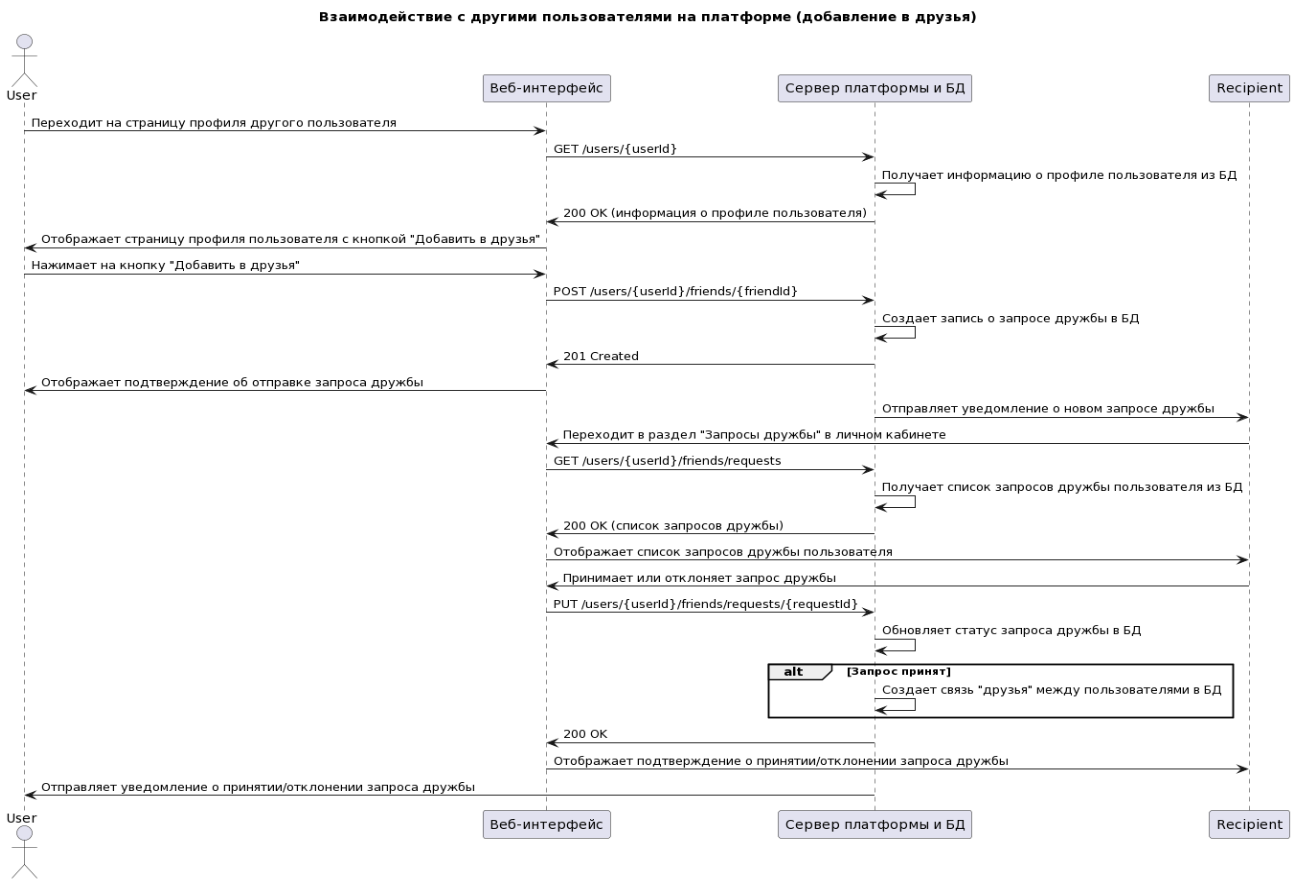
end

ServerDB -> WebUI: 200 OK

WebUI -> Recipient: Отображает подтверждение о принятии/отклонении запроса дружбы

ServerDB -> User: Отправляет уведомление о принятии/отклонении запроса дружбы

@enduml



@startuml

title Интеграция с сервисами Яндекса (Яндекс.Кассы)

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

participant "Яндекс.Кассы" as YandexCheckout

User -> WebUI: Выбирает билет и нажимает "Оплатить"

WebUI -> ServerDB: POST /orders

ServerDB -> ServerDB: Создает новый заказ в БД

ServerDB -> WebUI: 201 Created (информация о заказе)

WebUI -> YandexCheckout: POST /payments

YandexCheckout -> YandexCheckout: Создает новую платежную сессию

YandexCheckout -> WebUI: 201 Created (информация о платежной сессии)

WebUI -> User: Перенаправляет пользователя на страницу оплаты Яндекс.Кассы

User -> YandexCheckout: Вводит платежные данные и подтверждает оплату

YandexCheckout -> YandexCheckout: Обрабатывает платеж

YandexCheckout -> ServerDB: POST /payments/callback

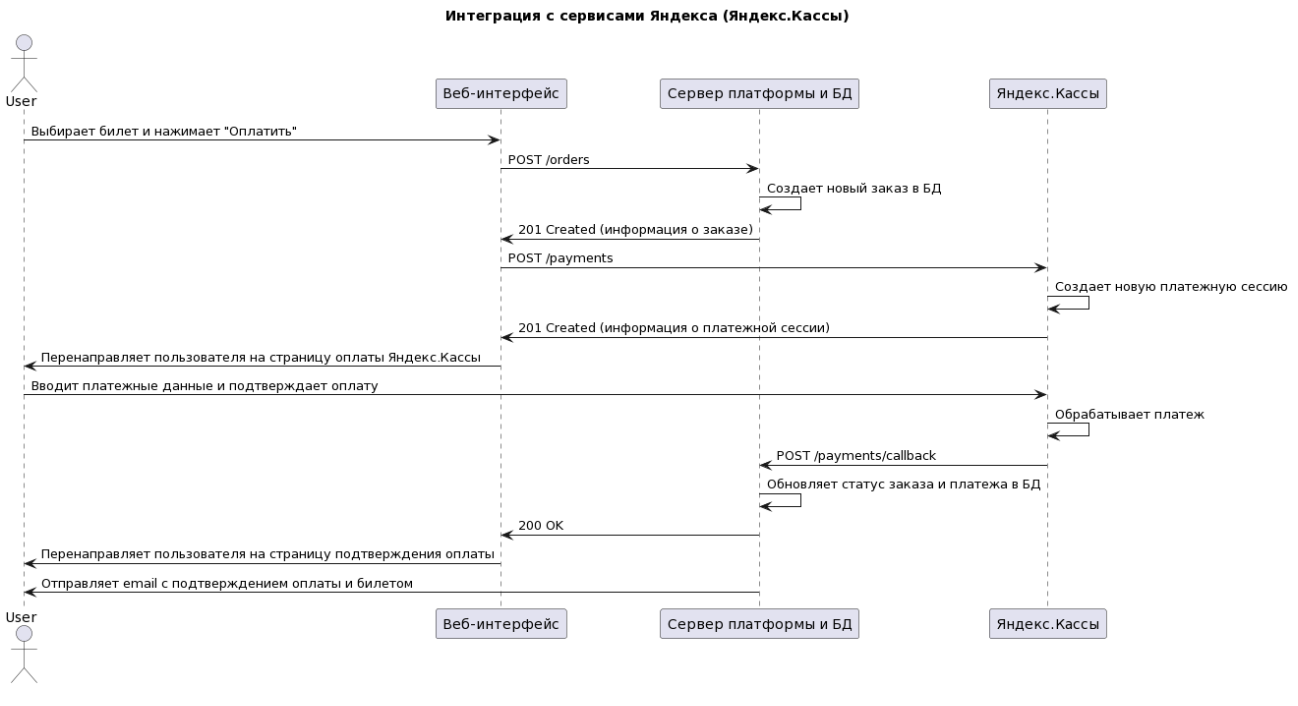
ServerDB -> ServerDB: Обновляет статус заказа и платежа в БД

ServerDB -> WebUI: 200 OK

WebUI -> User: Перенаправляет пользователя на страницу подтверждения оплаты

ServerDB -> User: Отправляет email с подтверждением оплаты и билетом

@enduml



@startuml

title Подписка/отписка от рассылок и уведомлений платформы

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит в раздел "Настройки уведомлений" в личном кабинете

WebUI -> ServerDB: GET /users/{userId}/notification\_settings

ServerDB -> ServerDB: Получает текущие настройки уведомлений пользователя из БД

ServerDB -> WebUI: 200 OK (настройки уведомлений)

WebUI -> User: Отображает текущие настройки уведомлений пользователя

User -> WebUI: Изменяет настройки уведомлений (подписка/отписка от рассылок, типы уведомлений)

WebUI -> ServerDB: PUT /users/{userId}/notification\_settings

ServerDB -> ServerDB: Сохраняет обновленные настройки уведомлений пользователя в БД

ServerDB -> WebUI: 200 OK

WebUI -> User: Отображает подтверждение об успешном обновлении настроек уведомлений

alt Подписка на рассылку

ServerDB -> User: Добавляет пользователя в список получателей рассылки

else Отписка от рассылки

ServerDB -> User: Удаляет пользователя из списка получателей рассылки

end

loop При появлении новых уведомлений

ServerDB -> ServerDB: Проверяет настройки уведомлений пользователя

alt Пользователь подписан на данный тип уведомлений

ServerDB -> User: Отправляет уведомление по электронной почте или через push-уведомление

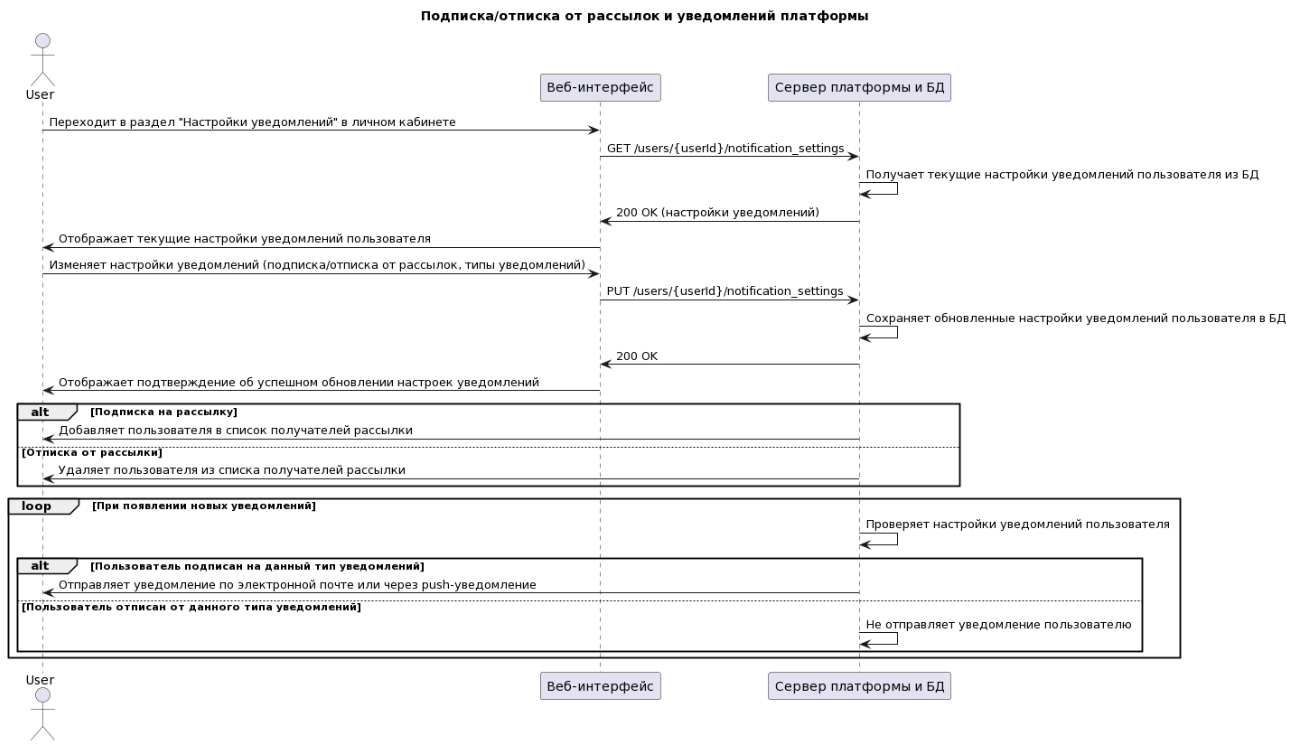
else Пользователь отписан от данного типа уведомлений

ServerDB -> ServerDB: Не отправляет уведомление пользователю

end

end

@enduml



@startuml

title Возможность оставить отзыв и оценку после посещения мероприятия

actor User

participant "Веб-интерфейс" as WebUI

participant "Сервер платформы и БД" as ServerDB

User -> WebUI: Переходит на страницу прошедшего мероприятия, которое посетил

WebUI -> ServerDB: GET /events/{eventId}/attendance?userId={userId}

ServerDB -> ServerDB: Проверяет, что пользователь действительно посетил мероприятие

ServerDB -> WebUI: 200 OK

WebUI -> User: Отображает форму для оставления отзыва и оценки мероприятия

User -> WebUI: Заполняет форму отзыва (текст, оценка)

User -> WebUI: Отправляет отзыв

WebUI -> ServerDB: POST /events/{eventId}/reviews

ServerDB -> ServerDB: Сохраняет отзыв и оценку пользователя в БД

ServerDB -> WebUI: 201 Created

WebUI -> User: Отображает подтверждение о публикации отзыва

User -> WebUI: Переходит на страницу мероприятия

WebUI -> ServerDB: GET /events/{eventId}/reviews

ServerDB -> ServerDB: Получает список отзывов и средний рейтинг мероприятия из БД

ServerDB -> WebUI: 200 OK (список отзывов и средний рейтинг)

WebUI -> User: Отображает список отзывов и средний рейтинг мероприятия на странице

@enduml

