Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по учебному курсу
«Верификация программного обеспечения»

Тестирование программной системы для обработки и проигрывания музыкальных композиций

Выполнил:
студент 4 курса группы 22407
А. Д. Фомин

Петрозаводск — 2022

1. Объект тестирования

Объектом тестирования является программная система для обработки и проигрывания музыкальных композиций, представляющая собой веб-сервис. Зарегистрированный пользователь может загружать аудиофайлы в свою библиотеку с получением анализа его некоторых музыкальных характеристик таких, как темп, тональность, настроение и пр. Пользователь может проигрывать композиции из своей библиотеки, а также составлять из них плейлисты. Также имеется возможность автоматической генерации плейлистов из композиций, схожих по характеристикам, которые можно затем сохранить.

Функциональность объекта тестирования

1. Действия, связанные с аутентификацией пользователя
   1.1. Создание аккаунта на сервисе
   1.2. Авторизация на сервисе
2. Действия, связанные с загрузкой файла
   2.1. Загрузка аудио в пользовательскую библиотеку
   2.2. Анализ и получение музыкальных характеристик
   2.3. Изменение метаданных аудио
3. Действия, связанные с воспроизведением файла
   3.1. Проигрывание выбранного аудиофайла/плейлиста
   3.2. Управление громкостью воспроизводимой композиции
   3.3. Управление позицией воспроизведения
   3.4. Переключение между воспроизводимыми песнями в плейлисте/библиотеке
4. Действия, связанные с управлением композициями
   4.1. Удаление композиции из библиотеки
   4.2. Добваление, композиции в выбранный плейлист
5. Действия, связанные с управлением плейлистами
   5.1. Ручное создание/удаление плейлиста
   5.2. Изменение названия плейлиста
   5.3. Удаление композиций из плейлиста
6. Действия, связанные с автоматической генерацией плейлистов
   6.1. Генерация плейлистов из композиций, схожих по характеристикам
   6.2. Сохранение созданного плейлиста в библиотеку пользователя

Архитектура системы

Схема архитектуры системы находится в приложении А.

Модули и подсистемы:

- Frontend (красный): клиентская часть приложения, выполняемая на компьютере пользователя. Написана на Angular. Состоит из следующих компонентов/сервисов:
  a. Компонент логина
  b. Компонент регистрации
  c. Компонент загрузки + сервис
  d. Компонент бокового меню
     i. Компонент плейлиста
     ii. Сервис генерации плейлистов

e. Компонент плеера + сервис
f. Сервис получения данных композиций

- Backend (фиолетовый): комплекс модулей, исполняемых на сервере
  a. API (зеленый): часть веб-сервиса, отвечающая за обработку запросов пользователя. Написана на NestJS. Состоит из следующих модулей (контроллеров):
     i. Модуль загрузки аудио
     ii. Модуль управления библиотекой
     iii. Модуль управления плейлистами
     iv. Модуль генерации плейлистов
     v. Модуль получения данных о композиции
     vi. Модуль авторизации
  b. Файловая система (ФС): файловое хранилище для загруженных аудиофайлов
  c. База данных (БД): база данных для хранения сведений о пользователях, характеристик композиций, содержания плейлистов
  d. Worker-анализатор (желтый): модуль, непосредственно осуществляющий анализ композиции
  e. ORM-модуль: модуль для связью с базой данных
  f. Queue (очередь): очередь, управляющая очередностью загруженных файлов на обработку воркером

2. Стратегия тестирования

Перед описанием каждого этапа тестирования стоит отметить, что во всех этапах не будет проверяться функциональность worker-модуля, занимающегося анализом композиций. Мероприятия по его тестированию планируется разработать и провести в обозримом будущем. Поэтому в тестах, где должен был быть задействован данный модуль, необходимо произвести замену на фиктивный объект (далее, *mock*), который имитирует работу модуля.

Модульное тестирование

В рамках модульного тестирования будет проверена функциональность модулей, составляющих API backend-части приложения, а именно следующие модули (контроллеры):

- UploadController: контроллер, отвечающий за загрузку аудиофайлов. Содержит внутри себя следующие методы: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File) – загружает файл на сервер, changeMetadata(@Req() request: Request) – изменяет метаданные файла.
- LibraryController: контроллер, отвечающий за управление треками в библиотеке пользователя. Содержит внутри себя следующие методы: deleteTrack(@Req() request: Request) – удаляет трек из библиотеки пользователя, getTracks(@Req() request: Request)– получает список треков из библиотеки/плейлиста пользователя, setTrackPlaylist(@Req() request: Request) – добавляет трек в плейлист, deleteTrackPlaylist(@Req() request: Request)– удаляет трек из плейлиста.
- PlaylistController: контроллер, отвечающий за управление плейлистами пользователя. Содержит внутри себя следующие методы: createPlaylist(@Req() request: Request) – создает плейлист, EditPlaylist(@Req() request: Request) – изменяет данные плейлиста, deletePlaylist(@Req() request: Request) – удаляет плейлист.

- ChannelsController: контроллер, отвечающий за генерацию плейлистов по характеристикам. Содержит внутри себя следующие методы: generateChannels(@Req() request: Request) – генерирует плейлисты по настроению, saveAsPlaylist(@Req() request: Request) – сохраняет сгенерированный плейлист.
- TrackController: контроллер, отвечающий за предоставление данных о композициях. Содержит внутри себя следующие методы: getTrackInfo(@Req() request: Request) – получает информацию о треке, getTrackData(@Req() request: Request) – получает файл трека.
- AuthController: контроллер, отвечающий за аутентификацию пользователей. Содержит внутри себя следующие методы: singup(@Req() request: Request), login(@Req() request: Request) – осуществляет авторизацию на веб-сервисе, logout(@Req() request: Request) – осуществляет деавторизацию на веб-сервисе.

Остальные модули не будут тестироваться в отдельности, так как либо в их основе будут лежать сторонние модули, которые уже были протестированы их разработчиками (модуль очереди, ORM, база данных, файловая система), либо они не будут содержать в себе классов/методов, имплементирующих комплексную логику (модули frontend-части приложения). Также не будет тестироваться worker-модуль.

Модульное тестирование будет проводиться при помощи библиотеки Jest, тем самым, будет автоматизированным. Оценка результатов тестирования будет сформирована с помощью средств, предоставляемой вышеуказанной библиотекой (генерация отчетов по проведению комплекса тестов, формирование статистики).

Интеграционное тестирование

В рамках интеграционного тестирования будет проверено взаимодействие большинства связей, обозначенных ранее на схеме архитектуры приложения. Полный список проверяемых взаимодействий по группам, выбранных инструментов тестирования, а также особых замечаний к тестам представлен в таблице 1.

Группа тестов	Взаимодействующие модули и функции	Средства тест-я	Замечания
Взаимодействие frontend и backend	● Компонент загрузки композиции – UploadController ○ uploadTrackForm – uploadTrack ○ setMeta – changeMetadata ● Сервис получения данных о композициях – TrackController ○ getTrackInfo – getTrackInfo ○ getTrackAudio – getTrackData ● Сервис автоматического создания плейлистов – ChannelsController ○ getMoodMixes – generateChannels ● Компонент плейлиста – LibraryController ○ deleteFromLibrary – deleteTrack ○ getTracks – getTracks ○ setPlaylist – setTrackPlaylist ○ deleteTrackFromPlaylist –	Jest	Из-за сложности задействовать лишь один контроллер при “поднятии” серверной части приложения было принято решение делать mock-объекты, которые имитируют ответы сервера на запросы. Для этого рекомендуется использовать встроенные возможности Jest.

	deleteTrackFromPlaylist ● Компонент плейлиста – PlaylistController ○ deletePlaylist – deletePlaylist ○ editName – editPlaylist ● Компонент плейлиста – ChannelsController ○ saveMix – saveAsPlaylist ● Sidebar-компонент – PlaylistController ○ createPlaylist – createPlaylist ● Sidebar-компонент – AuthController ○ logout – logout ● Компонент регистрации – AuthController ○ signup – signup ● Компонент логина – AuthController ○ login – login		
Проверка выполнения запросов (правильность, записи в БД и файлы в ФС)	Каждый модуль/метод, указанный на этапе модульного тестирования – БД + ФС	Postman, Newman	Для данной группы развертывается вся серверная часть приложения. Postman + newman будут имитировать реальные запросы от пользователей.

Интеракции с ORM-модулем и Queue-модулем тестироваться не будут, так как эти модули являются промежуточными звеньями в цепи взаимодействий ключевых модулей (БД + ФС и API).

Оценка результатов производится с помощью встроенных возможностей средств тестирования (составление отчетов и формирование статистики прохождения тестов).

Аттестационное тестирование

В рамках аттестационного тестирования будут проверяться сценарии использования системы на соответствие заявленным возможностям.

Тестирование будет проводится как автоматизированно, с помощью фреймворка Cypress, так и вручную, с помощью тестировщика. Для проведения последнего вида тестов необходимо развернуть серверную часть приложения и запустить каким-либо из способов клиентскую часть (с помощью IDE, собрать рабочий билд приложения и добавить в ФС бэкенда). Главное, чтобы клиентская часть совершала запросы именно к развернутому бэкенду и предоставляла GUI для взаимодействия.

Оценка результатов данного этапа тестирования будет производится на основе отчетов о тестировании для автоматизированных тестов. Для тестов, выполняемых вручную, оценка производится человеком по критерию соответствия: если фактический результат совпадает с ожидаемым, то тест считается пройденным, в противном случае – проваленным.

Тестирование UI

В рамках тестирования UI будет проведена проверка работы клиентской части приложения и его компонент. Будет проверена корректное отображение лейблов и надписей при создании компонент, заполнении форм и других пользовательских действий. Также будет произведена проверка адаптивности верстки.

Тестирование UI будет осуществляться с помощью Cypress. Окружение для тестов должно быть идентичным окружению, использованном при аттестационном тестировании.

Оценка результатов будет производиться при помощи встроенных возможностей фреймворка (составление отчетов и формирование статистики об ошибках).

Тестирование производительности

В тестировании производительности будет исследоваться время отклика и производительность API серверной части приложения при различных уровнях нагрузки. Тестирование остальных модулей производиться не будет за ненадобностью.

Тесты на данном этапе будут написаны при помощи k6. Окружение тестирования должно состоять из развернутой backend-части.

Оценка результатов будет производиться с помощью встроенных возможностей вышеуказанного инструмента (составление отчета о выполнении тестов и предоставление метрик). По полученным показателям и сравнению их с показателями для среднестатистического веб-сервиса тестировщиками будет сделан вывод о прохождении данного вида тестов.

Условия тестирования

Критерии прохождения тестов

Основным критерием прохождения теста на всех этапах, если в соответствующих разделах не указано явно что-либо иное, будет являться соответствие ожидаемому результату, характер которого указывается перед каждым тестом.

Условия начала, окончания и перехода между этапами тестирования

Каждый этап тестирования начинается с проектирования и написания тестов с момента старта разработки приложения. Запуск тестов для каждого вида происходит поэтапно и только с того момента, как процент пройденных тестов с предыдущей стадии будет равен или превышать 90%. Исключение составляют тестирование UI (может выполняться параллельно с интеграционным тестированием) и тестирование производительности (выполняется только когда кодовая база серверной части готова к прохождению полного комплекса аттестационных тестов). Этапы тестирования никогда полностью не заканчиваются, так как планируется поддержка приложения.

Условия возобновления и приостановки выполнения тестов

При достижении уровня пройденных тестов на каждом из этапов равному 95% или более (за исключением аттестационного – там уровень составляет 100%), тестирование можно приостановить. Тестирование возобновляется при внесении существенных изменений в кодовую базу приложения (добавление нового функционала, исправление багов и прочее).

3. План тестов

Системные требования

Не предъявляются

Описание модульных тестов

No. теста	B-01
Цель теста	Проверка работы функции обработки запроса на загрузку файла при соответствии всем ограничениям (описаны ниже)
Тип теста	Позитивный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым) file – объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ● Размер файла: не более 50 Мб ● Длина композиции: не менее 30 с. и не более 15 мин. ● Формат файла: wav, aif, mp3, ogg или flac
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	string: “Audio created on server”

Таблица 2 – Описание теста B-01

No. теста	B-02
Цель теста	Проверка работы функции обработки запроса на загрузку файла при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (должно быть пустым или отсутствовать) file – объект аудиофайла в формате multipart/form-data
Косвенные входные	–

данные	
Ожидаемый результат	UnauthorizedException

Таблица 3 – Описание теста B-02

No. теста	B-03
Цель теста	Проверка работы функции обработки запроса на загрузку файла при неправильном заголовке авторизации в запросе
Тип теста	Негативный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) file – объект аудиофайла в формате multipart/form-data
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	UnauthorizedException

Таблица 4 – Описание теста B-03

No. теста	B-04
Цель теста	Проверка работы функции обработки запроса на загрузку файла при отсутствии файла в теле запроса
Тип теста	Негативный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) file – объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами (должен быть пустым или отсутствовать)
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	BadRequestException

Таблица 5 – Описание теста B-04

No. теста	B-05
Цель теста	Проверка работы функции обработки запроса на загрузку файла при загрузке файла в неподдерживаемом формате
Тип теста	Негативный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым) file – объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ● Формат файла: любой, кроме wav, aif, mp3, ogg или flac
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 6 – Описание теста B-05

No. теста	B-06
Цель теста	Проверка работы функции обработки запроса на загрузку файла при загрузке файла длительностью менее 30 с.
Тип теста	Негативный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым) file – объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ● Формат файла: wav, aif, mp3, ogg или flac ● Длина композиции: менее 30 с.
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 7 – Описание теста B-06

No. теста	B-07

Цель теста	Проверка работы функции обработки запроса на загрузку файла при загрузке файла длительностью более 15 мин.
Тип теста	Негативный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) file – объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ● Формат файла: wav, aif, mp3, ogg или flac ● Длина композиции: более 15 мин.
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 8 – Описание теста B-07

No. теста	B-08
Цель теста	Проверка работы функции обработки запроса на загрузку файла при загрузке файла размером более 50 Мб.
Тип теста	Негативный
Объект тестирования	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) file – объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ● Формат файла: wav, aif, mp3, ogg или flac ● Длина композиции: не менее 30 с. и не более 15 мин. ● Размер файла: более 50 Мб
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 9 – Описание теста B-08

No. теста	B-09

Цель теста	Проверка работы функции обработки запроса на изменение метаданных файла
Тип теста	Позитивный
Объект тестирования	UploadContoller: changeMetadata(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор аудиофайла на сервере ● body: JSON-строка формата { title: string, artist: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	string: “Audio {id} metadata changed”

Таблица 10 – Описание теста В-09

No. теста	В-10
Цель теста	Проверка работы функции обработки запроса на изменение метаданных файла при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	UploadContoller: changeMetadata(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● params[‘trackId’]: идентификатор аудиофайла на сервере ● body: JSON-строка формата { title: string, artist: string }
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 11 – Описание теста В-10

No. теста	В-11
Цель теста	Проверка работы функции обработки запроса на изменение метаданных файла при отсутствии параметра trackId в запросе
Тип теста	Негативный

Объект тестирования	UploadContoller: changeMetadata(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя ● params['trackId']: идентификатор аудиофайла на сервере (должен быть пустым или отсутствовать) ● body: JSON-строка формата { title: string, artist: string }
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 12 – Описание теста B-11

No. теста	B-12
Цель теста	Проверка работы функции обработки запроса на изменение метаданных файла при неправильном заголовке авторизации в запросе (т. е. композиция не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	UploadContoller: changeMetadata(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым) ● params['trackId']: идентификатор аудиофайла на сервере ● body: JSON-строка формата { title: string, artist: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	ForbiddenException

Таблица 13 – Описание теста B-12

No. теста	B-13
Цель теста	Проверка работы функции обработки запроса на изменение метаданных файла при параметре trackId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	UploadContoller: changeMetadata(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями:

	● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор аудиофайла на сервере ● body: JSON-строка формата { title: string, artist: string }
Косвенные входные данные	Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 14 – Описание теста B-13

No. теста	B-14
Цель теста	Проверка работы функции обработки запроса на изменение метаданных файла при теле запроса не соответствующем формату (см. ниже)
Тип теста	Негативный
Объект тестирования	UploadContoller: changeMetadata(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор аудиофайла на сервере ● body: **не** JSON-строка формата { title: string, artist: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 15 – Описание теста B-14

No. теста	B-15
Цель теста	Проверка работы функции обработки запроса на удаление композиции из библиотеки
Тип теста	Позитивный
Объект тестирования	LibraryContoller: deleteTrack(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор аудиофайла на сервере

Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	string: “Audio {idAudio} deleted from {idUser}’s library”

Таблица 16 – Описание теста B-15

No. теста	B-16
Цель теста	Проверка работы функции обработки запроса на удаление композиции из библиотеки при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrack(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● params[‘trackId’]: идентификатор аудиофайла на сервере
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 17 – Описание теста B-16

No. теста	B-17
Цель теста	Проверка работы функции обработки запроса на удаление композиции из библиотеки при отсутствии параметра trackId.
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrack(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор аудиофайла на сервере (должен быть пустым или отсутствовать)
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 18 – Описание теста B-17

No. теста	B-18
Цель теста	Проверка работы функции обработки запроса на удаление композиции из библиотеки при неправильном заголовке авторизации в запросе (т. е. композиция не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrack(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор аудиофайла на сервере
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	ForbiddenException

Таблица 19 – Описание теста B-18

No. теста	B-19
Цель теста	Проверка работы функции обработки запроса на удаление композиции из библиотеки при параметре trackId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrack(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор аудиофайла на сервере
Косвенные входные данные	Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 20 – Описание теста B-19

No. теста	B-20
Цель теста	Проверка работы функции обработки запроса на получение списка композиций из библиотеки.

Тип теста	Позитивный
Объект тестирования	LibraryContoller: getTracks(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым)
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, непосредственно получающий список ID композиций, должен возвращать [1, 2, 3, 4]
Ожидаемый результат	array: [1, 2, 3, 4]

Таблица 21 – Описание теста B-20

No. теста	B-21
Цель теста	Проверка работы функции обработки запроса на получение списка композиций из плейлиста
Тип теста	Позитивный
Объект тестирования	LibraryContoller: getTracks(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат. Метод, непосредственно получающий список ID композиций, должен возвращать [1, 2, 3, 4]
Ожидаемый результат	array: [1, 2, 3, 4]

Таблица 22 – Описание теста B-21

No. теста	B-22
Цель теста	Проверка работы функции обработки запроса на получение списка композиций при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	LibraryContoller: getTracks(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть

	пустым или отсутствовать) ● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 23 – Описание теста B-22

No. теста	B-23
Цель теста	Проверка работы функции обработки запроса на удаление композиции из библиотеки при неправильном заголовке авторизации в запросе (т. е. плейлист не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	LibraryContoller: getTracks(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции,. должен возвращать отрицательный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	ForbiddenException

Таблица 24 – Описание теста B-23

No. теста	B-24
Цель теста	Проверка работы функции обработки запроса на получение списка композиций при параметре playlistId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	LibraryContoller: getTracks(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат.

Ожидаемый результат	NotFoundException

Таблица 25 – Описание теста B-24

No. теста	B-25
Цель теста	Проверка работы функции обработки запроса на изменение принадлежности композиции к плейлисту
Тип теста	Позитивный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	string: “Track {trackId} has been added to playlist {playlistId}”

Таблица 26 – Описание теста B-25

No. теста	B-26
Цель теста	Проверка работы функции обработки запроса на изменение принадлежности композиции к плейлисту при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 27 – Описание теста B-26

No. теста	B-27
Цель теста	Проверка работы функции обработки запроса на изменение принадлежности композиции к плейлисту при отсутствии параметра trackId в запросе
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции (должен быть пустым или отсутствовать)
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 28 – Описание теста B-27

No. теста	B-28
Цель теста	Проверка работы функции обработки запроса на изменение принадлежности композиции к плейлисту при отсутствии параметра playlistId в запросе
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя ● params[‘playlistId’]: идентификатор плейлиста (должен быть пустым или отсутствовать) ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 29 – Описание теста B-28

No. теста	B-29
Цель теста	Проверка работы функции обработки запроса на изменение

	принадлежности композиции к плейлисту при неправильном заголовке авторизации в запросе (т. е. плейлист не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	ForbiddenException

Таблица 30 – Описание теста B-29

No. теста	B-30
Цель теста	Проверка работы функции обработки запроса на изменение принадлежности композиции к плейлисту при неправильном заголовке авторизации в запросе (т. е. трек не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	ForbiddenException

Таблица 31 – Описание теста B-30

No. теста	B-31
Цель теста	Проверка работы функции обработки запроса на изменение принадлежности композиции к плейлисту при параметре playlistId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 32 – Описание теста B-31

No. теста	B-32
Цель теста	Проверка работы функции обработки запроса на изменение принадлежности композиции к плейлисту при параметре trackId, уже существующем в плейлисте с идентификатором playlistId.
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	ConflictException

Таблица 33 – Описание теста B-32

No. теста	B-33
Цель теста	Проверка работы функции обработки запроса на изменение

	принадлежности композиции к плейлисту при параметре trackId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 34 – Описание теста В-33

No. теста	В-34
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста
Тип теста	Позитивный
Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	string: “Track {trackId} has been deleted from playlist {playlistId}”

Таблица 35 – Описание теста В-34

No. теста	В-35
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста при отсутствии заголовка авторизации в запросе
Тип теста	Негативный

Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 36 – Описание теста B-35

No. теста	B-36
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста при отсутствии параметра trackId в запросе
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции (должен быть пустым или отсутствовать)
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 37 – Описание теста B-36

No. теста	B-37
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста при отсутствии параметра playlistId в запросе
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя ● params[‘playlistId’]: идентификатор плейлиста (должен быть пустым или отсутствовать) ● params[‘trackId’]: идентификатор композиции

Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 38 – Описание теста B-37

No. теста	B-38
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста при неправильном заголовке авторизации в запросе (т. е. плейлист не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	ForbiddenException

Таблица 39 – Описание теста B-38

No. теста	B-39
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста при неправильном заголовке авторизации в запросе (т. е. трек не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность

	playlistId внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	ForbiddenException

Таблица 40 – Описание теста B-39

No. теста	B-40
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста при параметре playlistId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 41 – Описание теста B-40

No. теста	B-41
Цель теста	Проверка работы функции обработки запроса на удаление композиции из плейлиста при параметре trackId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● params[‘trackId’]: идентификатор композиции
Косвенные входные данные	Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый	NotFoundException

результат	

Таблица 42 – Описание теста В-41

No. теста	В-42
Цель теста	Проверка работы функции обработки запроса на создание плейлиста
Тип теста	Позитивный
Объект тестирования	PlaylistController: createPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● body: JSON-объект формата { name: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	string: “Playlist {playlistId} created on server”

Таблица 43 – Описание теста В-42

No. теста	В-43
Цель теста	Проверка работы функции обработки запроса на создание плейлиста при отсутствии заголовка авторизации
Тип теста	Негативный
Объект тестирования	PlaylistController: createPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● body: JSON-объект формата { name: string }
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 44 – Описание теста В-43

No. теста	В-44
Цель теста	Проверка работы функции обработки запроса на создание плейлиста при неправильном заголовке авторизации (несуществующий пользователь/токен сессии)

Тип теста	Негативный
Объект тестирования	PlaylistController: createPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● body: JSON-объект формата { name: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	UnauthorizedException

Таблица 45 – Описание теста B-44

No. теста	B-45
Цель теста	Проверка работы функции обработки запроса на создание плейлиста при отсутствии тела запроса
Тип теста	Негативный
Объект тестирования	PlaylistController: createPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● body: JSON-объект формата { name: string } (должен быть пустым или отсутствовать)
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	BadRequestException

Таблица 46 – Описание теста B-45

No. теста	B-46
Цель теста	Проверка работы функции обработки запроса на создание плейлиста при неправильном теле запроса
Тип теста	Негативный
Объект тестирования	PlaylistController: createPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● body: **не** JSON-объект формата { name: string }

Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 47 – Описание теста B-46

No. теста	B-47
Цель теста	Проверка работы функции обработки запроса на изменение плейлиста
Тип теста	Позитивный
Объект тестирования	PlaylistController: editPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым) ● params['playlistId']: идентификатор плейлиста ● body: JSON-объект формата { name: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	string: “Playlist {playlistId} changed”

Таблица 48 – Описание теста B-47

No. теста	B-48
Цель теста	Проверка работы функции обработки запроса на изменение плейлиста при отсутствии заголовка авторизации
Тип теста	Негативный
Объект тестирования	PlaylistController: editPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (должен быть пустым или отсутствовать) ● params['playlistId']: идентификатор плейлиста ● body: JSON-объект формата { name: string }
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 49 – Описание теста B-48

No. теста	B-49
Цель теста	Проверка работы функции обработки запроса на изменение плейлиста при неправильном заголовке авторизации (т. е. Плейлист не принадлежит пользователю)
Тип теста	Негативный
Объект тестирования	PlaylistController: editPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● params[‘playlistId’]: идентификатор плейлиста ● body: JSON-объект формата { name: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnauthorizedException

Таблица 50 – Описание теста B-49

No. теста	B-50
Цель теста	Проверка работы функции обработки запроса на изменение плейлиста при отсутствии тела запроса
Тип теста	Негативный
Объект тестирования	PlaylistController: editPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● body: JSON-объект формата { name: string } (должен быть пустым или отсутствовать)
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	BadRequestException

Таблица 51 – Описание теста B-50

No. теста	B-51
Цель теста	Проверка работы функции обработки запроса на изменение плейлиста при отсутствии параметра запроса playlistId
Тип теста	Негативный
Объект тестирования	PlaylistController: editPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым) ● params['playlistId']: идентификатор плейлиста (должен быть пустым или отсутствовать) ● body: JSON-объект формата { name: string }
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 52 – Описание теста B-51

No. теста	B-52
Цель теста	Проверка работы функции обработки запроса на изменение плейлиста при параметре playlistId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	PlaylistController: editPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым) ● params['playlistId']: идентификатор плейлиста ● body: JSON-объект формата { name: string }
Косвенные входные данные	Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 53 – Описание теста B-52

No. теста	B-53
Цель теста	Проверка работы функции обработки запроса на изменение плейлиста при неправильном теле запроса
Тип теста	Негативный

Объект тестирования	PlaylistController: editPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста ● body: **не** JSON-объект формата { name: string }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 54 – Описание теста B-53

No. теста	B-54
Цель теста	Проверка работы функции обработки запроса на удаление плейлиста
Тип теста	Позитивный
Объект тестирования	PlaylistController: deletePlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	string: “Playlist {playlistId} changed”

Таблица 55 – Описание теста B-54

No. теста	B-55
Цель теста	Проверка работы функции обработки запроса на удаление плейлиста при отсутствии заголовка авторизации
Тип теста	Негативный
Объект тестирования	PlaylistController: deletePlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать)

	● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 56 – Описание теста B-55

No. теста	B-56
Цель теста	Проверка работы функции обработки запроса на удаление плейлиста при неправильном заголовке авторизации (т. е. Плейлист не принадлежит пользователю)
Тип теста	Негативный
Объект тестирования	PlaylistController: deletePlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать) ● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность playlistId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnauthorizedException

Таблица 57 – Описание теста B-56

No. теста	B-57
Цель теста	Проверка работы функции обработки запроса на удаление плейлиста при отсутствии параметра запроса playlistId
Тип теста	Негативный
Объект тестирования	PlaylistController: deletePlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста (должен быть пустым или отсутствовать)
Косвенные входные данные	–

Ожидаемый результат	BadRequestException

Таблица 58 – Описание теста В-57

No. теста	В-58
Цель теста	Проверка работы функции обработки запроса на удаление плейлиста при параметре playlistId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	PlaylistController: deletePlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘playlistId’]: идентификатор плейлиста
Косвенные входные данные	Метод, проверяющий валидность playlistId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 59 – Описание теста В-58

No. теста	В-59
Цель теста	Проверка работы функции обработки запроса на генерацию плейлистов
Тип теста	Позитивный
Объект тестирования	ChannelsController: generateChannels(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым)
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, возвращающий список треков с характеристиками из базы данных, должен давать результат описанный в приложении В.
Ожидаемый результат	JSON-string: “{ playlists: [{ title: “Happy Mix”, tracks: [0, 1, 5, 7, 10, 14] }] }”

Таблица 60 – Описание теста В-59

No. теста	В-60

Цель теста	Проверка работы функции обработки запроса на генерацию плейлистов при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	ChannelsController: generateChannels(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (должно быть пустым или отсутствовать)
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 61 – Описание теста B-60

No. теста	B-61
Цель теста	Проверка работы функции обработки запроса на генерацию плейлистов при неправильном заголовке авторизации в запросе
Тип теста	Негативный
Объект тестирования	ChannelsController: generateChannels(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (должен быть не пустым)
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	UnauthorizedException

Таблица 62 – Описание теста B-61

No. теста	B-62
Цель теста	Проверка работы функции обработки запроса на генерацию плейлистов при малом количестве треков
Тип теста	Позитивный
Объект тестирования	ChannelsController: generateChannels(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (не должен быть пустым)

Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, возвращающий список треков с характеристиками из базы данных, должен давать результат описанный в приложении В (массив состоит только из первых двух строк).
Ожидаемый результат	JSON-string: “{ playlists: [ ] }”

Таблица 63 – Описание теста В-62

No. теста	В-63
Цель теста	Проверка работы функции обработки запроса на сохранение сгенерированных плейлистов
Тип теста	Позитивный
Объект тестирования	ChannelsController: saveAsPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● body: JSON-объект формата { title: string, tracks: int[] }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, возвращающий список треков с характеристиками из базы данных, должен давать результат описанный в приложении В.
Ожидаемый результат	string: “Playlist {playlistId} saved”

Таблица 64 – Описание теста В-63

No. теста	В-64
Цель теста	Проверка работы функции обработки запроса на сохранение сгенерированных плейлистов при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	ChannelsController: saveAsPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должно быть пустым или отсутствовать) ● body: JSON-объект формата { title: string, tracks: int[] }
Косвенные входные данные	–

Ожидаемый результат	UnauthorizedException

Таблица 65 – Описание теста B-64

No. теста	B-65
Цель теста	Проверка работы функции обработки запроса на сохранение сгенерированных плейлистов при неправильном заголовке авторизации в запросе
Тип теста	Негативный
Объект тестирования	ChannelsController: saveAsPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● body: JSON-объект формата { title: string, tracks: int[] }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	UnauthorizedException

Таблица 66 – Описание теста B-65

No. теста	B-66
Цель теста	Проверка работы функции обработки запроса на сохранение сгенерированных плейлистов при отсутствии тела запроса
Тип теста	Негативный
Объект тестирования	ChannelsController: saveAsPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● body: JSON-объект формата { title: string, tracks: int[] } (должен быть пустым или отсутствовать)
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	BadRequestException

Таблица 67 – Описание теста B-66

No. теста	B-67

Цель теста	Проверка работы функции обработки запроса на сохранение сгенерированных плейлистов при неправильном формате тела запроса
Тип теста	Негативный
Объект тестирования	ChannelsController: saveAsPlaylist(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● body: **не** JSON-объект формата { title: string, tracks: int[] }
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnprocessableEntityException

Таблица 68 – Описание теста B-67

No. теста	B-68
Цель теста	Проверка работы функции обработки запроса на получение информации о треке
Тип теста	Позитивный
Объект тестирования	TracksController: getTrackInfo(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор трека
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат. Метод, возвращающий трек из базы данных, должен давать результат описанный в пункте ниже.
Ожидаемый результат	JSON-string: “{ id: 0, title: “Title 0”, artist: “Artist 0”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }”

Таблица 69 – Описание теста B-68

No. теста	B-69
Цель теста	Проверка работы функции обработки запроса на получение информации о треке при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	TracksController: getTrackInfo(@Req() request: Request)

Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должно быть пустым или отсутствовать) ● params[‘trackId’]: идентификатор трека
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 70 – Описание теста B-69

No. теста	B-70
Цель теста	Проверка работы функции обработки запроса на получение информации о треке при неправильном заголовке авторизации в запросе (т. е. трек не находится в библиотеке пользователя)
Тип теста	Негативный
Объект тестирования	TracksController: getTrackInfo(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● params[‘trackId’]: идентификатор трека
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnauthorizedException

Таблица 71 – Описание теста B-70

No. теста	B-71
Цель теста	Проверка работы функции обработки запроса на получение информации о треке при отсутствии параметра trackId
Тип теста	Негативный
Объект тестирования	TracksController: getTrackInfo(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● params[‘trackId’]: идентификатор трека (должен быть пустым или отсутствовать)
Косвенные входные	Метод, проверяющий валидность trackId внутри функции, должен

данные	возвращать отрицательный результат.
Ожидаемый результат	BadRequestException

Таблица 72 – Описание теста B-71

No. теста	B-73
Цель теста	Проверка работы функции обработки запроса на получение информации о треке при параметре trackId, не указывающим ни на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	TracksController: getTrackInfo(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● params[‘trackId’]: идентификатор трека
Косвенные входные данные	Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 75 – Описание теста B-74

No. теста	B-75
Цель теста	Проверка работы функции обработки запроса на получение файла трека
Тип теста	Позитивный
Объект тестирования	TracksController: getTrackData(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (не должен быть пустым) ● params[‘trackId’]: идентификатор трека
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать положительный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат. Метод, возвращающий трек из базы данных, должен давать результат описанный в пункте ниже.
Ожидаемый результат	JSON-string: { id: 0, title: “Title 0”, data: “test” }”

Таблица 76 – Описание теста B-75

No. теста	B-76
Цель теста	Проверка работы функции обработки запроса на получение файла трека при отсутствии заголовка авторизации в запросе
Тип теста	Негативный
Объект тестирования	TracksController: getTrackData(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (должно быть пустым или отсутствовать) ● params['trackId']: идентификатор трека
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 77 – Описание теста B-76

No. теста	B-77
Цель теста	Проверка работы функции обработки запроса на получение файла трека при неправильном заголовке авторизации в запросе (т. е. трек не находится в библиотеке пользователя)
Тип теста	Негативный
Объект тестирования	TracksController: getTrackData(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers['Authorization']: токен сессии пользователя (должен быть не пустым) ● params['trackId']: идентификатор трека
Косвенные входные данные	Метод, проверяющий валидность токена внутри функции, должен возвращать отрицательный результат. Метод, проверяющий валидность trackId внутри функции, должен возвращать положительный результат.
Ожидаемый результат	UnauthorizedException

Таблица 78 – Описание теста B-77

No. теста	B-78
Цель теста	Проверка работы функции обработки запроса на получение файла трека при отсутствии параметра trackId

Тип теста	Негативный
Объект тестирования	TracksController: getTrackData(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● params[‘trackId’]: идентификатор трека (должен быть пустым или отсутствовать)
Косвенные входные данные	Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	BadRequestException

Таблица 79 – Описание теста B-78

No. теста	B-79
Цель теста	Проверка работы функции обработки запроса на получение файла трека при параметре trackId, не указывающим ни на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	TracksController: getTrackData(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым) ● params[‘trackId’]: идентификатор трека
Косвенные входные данные	Метод, проверяющий валидность trackId внутри функции, должен возвращать отрицательный результат.
Ожидаемый результат	NotFoundException

Таблица 80 – Описание теста B-79

No. теста	B-80
Цель теста	Проверка работы функции обработки запроса на регистрацию
Тип теста	Позитивный
Объект тестирования	AuthController: signup(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: JSON-объект формата { username: string (непустой), password: string (непустой, не менее 8 символов) }

Косвенные входные данные	Метод, проверяющий уникальность username в БД внутри функции, возвращает положительный результат.
Ожидаемый результат	JSON-string: { accessToken: {accessToken} }

Таблица 81 – Описание теста B-80

No. теста	B-81
Цель теста	Проверка работы функции обработки запроса на регистрацию при наличии пользователя с идентичным именем пользователя в БД
Тип теста	Негативный
Объект тестирования	AuthController: signup(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: JSON-объект формата { username: string (непустой), password: string (непустой, не менее 8 символов) }
Косвенные входные данные	Метод, проверяющий уникальность username в БД внутри функции, возвращает отрицательный результат.
Ожидаемый результат	ConflictException

Таблица 82 – Описание теста B-81

No. теста	B-82
Цель теста	Проверка работы функции обработки запроса на регистрацию при отсутствии тела запроса
Тип теста	Негативный
Объект тестирования	AuthController: signup(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: JSON-объект формата { username: string (непустой), password: string (непустой, не менее 8 символов) } (отсутствует или является пустым)
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 83 – Описание теста B-82

No. теста	B-83

Цель теста	Проверка работы функции обработки запроса на регистрацию при несоответствии формата тела запроса
Тип теста	Негативный
Объект тестирования	AuthController: signup(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: **не** JSON-объект формата { username: string (непустой), password: string (непустой, не менее 8 символов) }
Косвенные входные данные	Метод, проверяющий уникальность username в БД внутри функции, возвращает положительный результат (если username непустой и присутствует).
Ожидаемый результат	UnprocessableEntityException

Таблица 84 – Описание теста В-83

No. теста	В-84
Цель теста	Проверка работы функции обработки запроса на логин
Тип теста	Позитивный
Объект тестирования	AuthController: login(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: JSON-объект формата { username: string (непустой), password: string (непустой) }
Косвенные входные данные	Метод, проверяющий уникальность правильность ввода полей, сравнивая со значениями из БД, внутри функции, возвращает положительный результат.
Ожидаемый результат	JSON-string: { accessToken: {accessToken} }

Таблица 85 – Описание теста В-84

No. теста	В-85
Цель теста	Проверка работы функции обработки запроса на логин при вводе данных пользователя, не находящихся в БД
Тип теста	Позитивный
Объект тестирования	AuthController: login(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: JSON-объект формата { username: string (непустой), password: string (непустой) }

Косвенные входные данные	Метод, проверяющий уникальность правильность ввода полей, сравнивая со значениями из БД, внутри функции, возвращает отрицательный результат.
Ожидаемый результат	UnauthorizedException

Таблица 86 – Описание теста B-85

No. теста	B-86
Цель теста	Проверка работы функции обработки запроса на логин при отсутствии тела запроса
Тип теста	Негативный
Объект тестирования	AuthController: login(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: JSON-объект формата { username: string (непустой), password: string (непустой) } (отсутствует или является пустым)
Косвенные входные данные	–
Ожидаемый результат	BadRequestException

Таблица 87 – Описание теста B-86

No. теста	B-87
Цель теста	Проверка работы функции обработки запроса на логин при несоответствии формата тела запроса
Тип теста	Негативный
Объект тестирования	AuthController: login(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● body: **не** JSON-объект формата { username: string (непустой), password: string (непустой) }
Косвенные входные данные	–
Ожидаемый результат	UnprocessableEntityException

Таблица 88 – Описание теста B-87

No. теста	B-88

Цель теста	Проверка работы функции обработки запроса на выход с сайта
Тип теста	Позитивный
Объект тестирования	AuthController: logout(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым)
Косвенные входные данные	Метод, проверяющий уникальность правильность ввода полей, сравнивая со значениями из БД, внутри функции, возвращает положительный результат.
Ожидаемый результат	string: “Logged out successfully”

Таблица 89 – Описание теста B-88

No. теста	B-89
Цель теста	Проверка работы функции обработки запроса на логин при отсутствии токена авторизации
Тип теста	Негативный
Объект тестирования	AuthController: logout(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть пустым или отсутствовать)
Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 90 – Описание теста B-89

No. теста	B-90
Цель теста	Проверка работы функции обработки запроса на логин при неправильном токене авторизации (несуществующий пользователь/сессия)
Тип теста	Негативный
Объект тестирования	AuthController: logout(@Req() request: Request)
Входные данные	request – объект типа Request со следующими полями: ● headers[‘Authorization’]: токен сессии пользователя (должен быть не пустым)

Косвенные входные данные	–
Ожидаемый результат	UnauthorizedException

Таблица 91 – Описание теста B-90

Описание интеграционных тестов

No. теста	I-01
Цель теста	Проверка взаимодействия БД, ФС и обработки запроса на загрузку файла при соответствии всем ограничениям (описаны ниже)
Тип теста	Позитивный
Объект тестирования	Связь UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) ● Тело запроса содержит объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ○ Размер файла: не более 50 Мб ○ Длина композиции: не менее 30 с. и не более 15 мин. ○ Формат файла: wav, aif, mp3, ogg или flac
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “201”, message: “Audio created on server” } В таблице tblTracks должна появиться новая запись id | title | artist | bpm | tone | happiness | danceability | energy 16 | {Хэш-название аудио} | {Название исполнителя} | null | null | null | null | null В ФС должен появиться новый файл {Хэш-название аудио}.

Таблица 92 – Описание теста I-01

No. теста	I-02
Цель теста	Проверка взаимодействия БД и обработки запроса на загрузку файла при отсутствии файла в теле запроса
Тип теста	Негативный

Объект тестирования	Связь UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) ● Тело запроса пустое
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “400”, message: “Bad request” } Таблицы в БД остаются без изменений

Таблица 93 – Описание теста I-02

No. теста	I-03
Цель теста	Проверка взаимодействия БД и обработки запроса на загрузку файла при загрузке файла размером более 50 Мб.
Тип теста	Негативный
Объект тестирования	Связь UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) ● Тело запроса содержит объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ○ Размер файла: более 50 Мб ○ Длина композиции: не менее 30 с. и не более 15 мин. ○ Формат файла: wav, aif, mp3, ogg или flac
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “422”, message: “Unprocessable entity” } Таблицы в БД остаются без изменений

Таблица 94 – Описание теста I-03

No. теста	I-04
Цель теста	Проверка взаимодействия БД и обработки запроса на изменение метаданных файла
Тип теста	Позитивный
Объект тестирования	Связь UploadContoller: changeMetadata(@Req() request: Request) и БД

<table>
<tr><td>Входные данные</td><td>request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами:
Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Параметр ‘trackId’ содержит идентификатор аудиофайла на сервере: 0
Тело запроса составляет JSON-строку формата { title: “New Title”, artist: “New Artist” }</td></tr>
<tr><td>Косвенные входные данные</td><td>В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.</td></tr>
<tr><td>Ожидаемый результат</td><td>Response: { statusCode: “200”, message: “Audio 0 metadata changed” }
В таблице tblTracks должна появиться измененная запись
<table><tr><td>id</td><td>title</td><td>artist</td><td>bpm</td><td>tone</td><td>happiness</td><td>danceability</td><td>energy</td></tr><tr><td>0</td><td>New Title</td><td>New Artist</td><td>120</td><td>Cmajor</td><td>9</td><td>9</td><td>6</td></tr></table></td></tr>
</table>

Таблица 95 – Описание теста I-04

No. теста	I-05
Цель теста	Проверка взаимодействия БД и обработки запроса на изменение метаданных файла при неправильном заголовке авторизации в запросе (т. е. композиция не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	Связь UploadContoller: changeMetadata(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (выбирается случайно) ● Параметр ‘trackId’ содержит идентификатор аудиофайла на сервере: 0 ● Тело запроса составляет JSON-строку формата { title: “New Title”, artist: “New Artist” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “403”, message: “Forbidden” } Таблицы в БД остаются без изменений

Таблица 96 – Описание теста I-05

No. теста	I-06
Цель теста	Проверка взаимодействия БД и обработки запроса на изменение

	метаданных файла при параметре trackId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	Связь UploadContoller: changeMetadata(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘trackId’ содержит идентификатор аудиофайла на сервере: 22 ● Тело запроса составляет JSON-строку формата { title: “New Title”, artist: “New Artist” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “404”, message: “Not found” } Таблицы в БД остаются без изменений

Таблица 97 – Описание теста I-06

No. теста	I-07
Цель теста	Проверка взаимодействия БД и обработки запроса на изменение метаданных файла при теле запроса не соответствующем формату (см. ниже)
Тип теста	Негативный
Объект тестирования	Связь UploadContoller: changeMetadata(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘trackId’ содержит идентификатор аудиофайла на сервере: 0 ● Тело запроса составляет JSON-строку формата { artist: “New Artist”, new-title: “New Title” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “422”, message: “Unprocessable entity” } Таблицы в БД остаются без изменений

Таблица 98 – Описание теста I-07

No. теста	I-08

Цель теста	Проверка взаимодействия БД и обработки запроса на удаление композиции из библиотеки
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: deleteTrack(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘trackId’ содержит идентификатор аудиофайла на сервере: 0
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “200”, message: “Audio 0 deleted from 0’s library” } В таблице tblTracks должна удалиться запись с id = 0

Таблица 99 – Описание теста I-08

No. теста	I-09
Цель теста	Проверка взаимодействия БД и обработки запроса на получение списка композиций из библиотеки.
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: getTracks(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “200”, body: [0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] }

Таблица 100 – Описание теста I-09

No. теста	I-10
Цель теста	Проверка взаимодействия БД и обработки запроса на получение списка композиций из плейлиста
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: getTracks(@Req() request: Request) и БД

Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “200”, body: [0, 1, 2, 3] }

Таблица 101 – Описание теста I-10

No. теста	I-11
Цель теста	Проверка взаимодействия БД и обработки запроса на удаление композиции из библиотеки при неправильном заголовке авторизации в запросе (т. е. плейлист не находится в библиотеке данного пользователя)
Тип теста	Негативный
Объект тестирования	Связь LibraryContoller: getTracks(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 3
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “403”, message: “Forbidden” }

Таблица 102 – Описание теста I-11

No. теста	I-12
Цель теста	Проверка взаимодействия БД и обработки запроса на получение списка композиций при параметре playlistId, не указывающем на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	Связь LibraryContoller: getTracks(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами:

	● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 4
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “404”, message: “Not found” }

Таблица 103 – Описание теста I-12

No. теста	I-13
Цель теста	Проверка взаимодействия БД и обработки запроса на изменение принадлежности композиции к плейлисту
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: setTrackPlaylist(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1 ● Параметр ‘trackId’ содержит идентификатор трека на сервере: 10
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “200”, message: “Track 10 has been added to playlist 1” } В таблице tblPlaylistsTracks появилась новая запись: id | idPlaylist | idTrack 22 | 1 | 10

Таблица 104 – Описание теста I-13

No. теста	I-14
Цель теста	Проверка взаимодействия БД и обработки запроса на изменение принадлежности композиции к плейлисту при параметре trackId, уже существующем в плейлисте с идентификатором playlistId.
Тип теста	Негативный

Объект тестирования	Связь LibraryContoller: setTrackPlaylist(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1 ● Параметр ‘trackId’ содержит идентификатор трека на сервере: 1
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “409”, message: “Conflict” } Таблицы БД остаются без изменений

Таблица 105 – Описание теста I-14

No. теста	I-15
Цель теста	Проверка взаимодействия БД и обработки запроса на удаление композиции из плейлиста
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1 ● Параметр ‘trackId’ содержит идентификатор трека на сервере: 0
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “200”, message: “Track 0 has been deleted from playlist 1” } Из таблицы tblPlaylistTracks должна исчезнуть запись с id = 16

Таблица 106 – Описание теста I-15

No. теста	I-16
Цель теста	Проверка взаимодействия БД и обработки запроса на создание плейлиста
Тип теста	Позитивный
Объект тестирования	Связь PlaylistController: createPlaylist(@Req() request: Request) и БД

<table>
<tr><td>Входные данные</td><td>request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами:
Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Тело запроса: JSON-объект формата { name: “new mix” }</td></tr>
<tr><td>Косвенные входные данные</td><td>В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.</td></tr>
<tr><td>Ожидаемый результат</td><td>Response: { statusCode: “201”, message: “Playlist 4 created on server” }
В таблице tblPlaylists появилась новая запись:<table><tr><td>id</td><td>idUser</td><td>name</td></tr><tr><td>4</td><td>0</td><td>new mix</td></tr></table></td></tr>
</table>

Таблица 107 – Описание теста I-16

<table>
<tr><td>No. теста</td><td>I-17</td></tr>
<tr><td>Цель теста</td><td>Проверка взаимодействия БД и обработки запроса на изменение плейлиста</td></tr>
<tr><td>Тип теста</td><td>Позитивный</td></tr>
<tr><td>Объект тестирования</td><td>Связь PlaylistController: editPlaylist(@Req() request: Request) и БД</td></tr>
<tr><td>Входные данные</td><td>request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами:
Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1
Тело запроса: JSON-объект формата { name: “new mix v2” }</td></tr>
<tr><td>Косвенные входные данные</td><td>В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.</td></tr>
<tr><td>Ожидаемый результат</td><td>Response: { statusCode: “201”, message: “Playlist 1 changed” }
В таблице tblPlaylists появилась измененная запись:<table><tr><td>id</td><td>idUser</td><td>name</td></tr><tr><td>1</td><td>0</td><td>new mix v2</td></tr></table></td></tr>
</table>

Таблица 108 – Описание теста I-17

No. теста	I-18

Цель теста	Проверка взаимодействия БД и обработки запроса на удаление плейлиста
Тип теста	Позитивный
Объект тестирования	Связь PlaylistController: deletePlaylist(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 3
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: { statusCode: “200”, message: “Playlist 3 deleted” } В таблице tblPlaylists исчезла запись с id = 3. Также исчезли записи в tblPlaylistsTracks с id = 20 и 21.

Таблица 109 – Описание теста I-18

No. теста	I-19
Цель теста	Проверка взаимодействия БД и обработки запроса на генерацию плейлистов
Тип теста	Позитивный
Объект тестирования	Связь ChannelsController: generateChannels(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “200”, body: { playlists: [{ title: “Happy Mix”, tracks: [0, 1, 5, 7, 10, 14] }] } }”

Таблица 110 – Описание теста I-19

No. теста	I-20
Цель теста	Проверка взаимодействия БД и обработки запроса на генерацию плейлистов при отсутствии треков
Тип теста	Позитивный

Объект тестирования	Связь ChannelsController: generateChannels(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С, за исключением того, что в tblTracks, tblPlaylists и tblPlaylistsTracks нет записей.
Ожидаемый результат	Response: “{ statusCode: “200”, body: { playlists: [ ] }”

Таблица 111 – Описание теста I-20

No. теста	I-21
Цель теста	Проверка взаимодействия БД и обработки запроса на генерацию плейлистов при малом количестве треков
Тип теста	Позитивный
Объект тестирования	Связь ChannelsController: generateChannels(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С, за исключением того, что в tblTracks, tblPlaylists и tblPlaylistsTracks существуют только их первые по счету записи.
Ожидаемый результат	Response: “{ statusCode: “200”, body: { playlists: [ ] }”

Таблица 112 – Описание теста I-21

No. теста	I-22
Цель теста	Проверка взаимодействия БД и обработки запроса на сохранение сгенерированных плейлистов
Тип теста	Позитивный
Объект тестирования	Связь ChannelsController: saveAsPlaylist(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами:

<table>
<tr><td></td><td>

Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста)
Тело запроса: JSON-объект формата { title: “Happy Mix”, tracks: [0, 1, 5, 7, 10, 14] }

</td></tr>
<tr><td>Косвенные входные данные</td><td>В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.</td></tr>
<tr><td>Ожидаемый результат</td><td>Response: “{ statusCode: “201”, message: “Playlist 4 saved” }
В таблице tblPlaylists появилась новая запись:
<table>
<tr><td>id</td><td>idUser</td><td>name</td></tr>
<tr><td>4</td><td>0</td><td>Happy Mix</td></tr>
</table>
В таблице tblPlaylistsTracks появились новые записи:
<table>
<tr><td>id</td><td>idPlaylist</td><td>idTrack</td></tr>
<tr><td>22</td><td>4</td><td>0</td></tr>
<tr><td>23</td><td>4</td><td>1</td></tr>
<tr><td>24</td><td>4</td><td>5</td></tr>
<tr><td>25</td><td>4</td><td>7</td></tr>
<tr><td>26</td><td>4</td><td>10</td></tr>
<tr><td>27</td><td>4</td><td>14</td></tr>
</table>
</td></tr>
</table>

Таблица 113 – Описание теста I-22

No. теста	I-23
Цель теста	Проверка взаимодействия БД и обработки запроса на сохранение сгенерированных плейлистов при неправильном формате тела запроса
Тип теста	Негативный
Объект тестирования	Связь ChannelsController: saveAsPlaylist(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Тело запроса: JSON-объект формата { title: “Happy Mix”, tracks: [“hello”, “good morning”] }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.

Ожидаемый результат	Response: “{ statusCode: “422”, message: “Unprocessable entity” }” Таблицы в БД остаются без изменений.

Таблица 114 – Описание теста I-23

No. теста	I-24
Цель теста	Проверка взаимодействия БД и обработки запроса на получение информации о треке
Тип теста	Позитивный
Объект тестирования	Связь TracksController: getTrackInfo(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘trackId’ содержит идентификатор трека: 0
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “200”, body: “{ id: 0, title: “Title 0”, artist: “Artist 0”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }” }”

Таблица 115 – Описание теста I-24

No. теста	I-25
Цель теста	Проверка взаимодействия БД и обработки запроса на получение информации о треке при неправильном заголовке авторизации в запросе (т. е. трек не находится в библиотеке пользователя)
Тип теста	Негативный
Объект тестирования	Связь TracksController: getTrackInfo(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘trackId’ содержит идентификатор трека: 16
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “401”, message: “Unauthorized” }”

Таблица 116 – Описание теста I-25

No. теста	I-26
Цель теста	Проверка взаимодействия БД и обработки запроса на получение информации о треке при параметре trackId, не указывающим ни на какую-либо запись в БД
Тип теста	Негативный
Объект тестирования	Связь TracksController: getTrackInfo(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘trackId’ содержит идентификатор трека: 22
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “404”, message: “Not found” }”

Таблица 117 – Описание теста I-26

No. теста	I-27
Цель теста	Проверка взаимодействия БД и обработки запроса на получение файла трека
Тип теста	Позитивный
Объект тестирования	Связь TracksController: getTrackData(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста) ● Параметр ‘trackId’ содержит идентификатор трека: 0
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “200”, body: { id: 0, title: “Title 0”, data: [содержимое файла из ФС] } }”

Таблица 118 – Описание теста I-27

No. теста	I-28
Цель теста	Проверка взаимодействия БД и обработки запроса на регистрацию
Тип теста	Позитивный

Объект тестирования	Связь AuthController: signup(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Тело запроса: JSON-объект формата { username: “User 2”, password: “hellohello” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “200”, body: { accessToken: {accessToken (генерируется случайно)} } }”

Таблица 119 – Описание теста I-28

No. теста	I-29
Цель теста	Проверка взаимодействия БД и обработки запроса на регистрацию при наличии пользователя с идентичным именем пользователя в БД
Тип теста	Негативный
Объект тестирования	Связь AuthController: signup(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Тело запроса: JSON-объект формата { username: “User 0”, password: “hellohello” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “409”, message: “Conflict” }”

Таблица 120 – Описание теста I-29

No. теста	I-30
Цель теста	Проверка взаимодействия БД и обработки запроса на регистрацию при несоответствии формата тела запроса
Тип теста	Негативный
Объект тестирования	Связь AuthController: signup(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Тело запроса: JSON-объект формата { username: “User 2”, password: “hell” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.

Ожидаемый результат	Response: “{ statusCode: “422”, message: “Unprocessable entity” }”

Таблица 121 – Описание теста I-30

No. теста	I-31
Цель теста	Проверка взаимодействия БД и обработки запроса на логин
Тип теста	Позитивный
Объект тестирования	Связь AuthController: login(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Тело запроса: JSON-объект формата { username: “User 0”, password: “hellohello” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “200”, body: { accessToken: {accessToken (генерируется случайно)} } }”

Таблица 122 – Описание теста I-31

No. теста	I-32
Цель теста	Проверка взаимодействия БД и обработки запроса на логин при вводе данных пользователя, не находящихся в БД
Тип теста	Позитивный
Объект тестирования	Связь AuthController: login(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Тело запроса: JSON-объект формата { username: “User 2”, password: “hellohello” }
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “401”, message: “Unauthorized” }”

Таблица 123 – Описание теста I-32

No. теста	I-33
Цель теста	Проверка взаимодействия БД и обработки запроса на выход с сайта

Тип теста	Позитивный
Объект тестирования	Связь AuthController: logout(@Req() request: Request) и БД
Входные данные	request – запрос, посылаемый с помощью Postman/Newman со следующими свойствами: ● Заголовок ‘Authorization’ содержит токен сессии пользователя (не должен быть пустым) (получается во время теста
Косвенные входные данные	В БД содержатся таблицы, структура и начальное состояние которых описано в приложении С.
Ожидаемый результат	Response: “{ statusCode: “200”, body: “Logged out successfully” }”

Таблица 124 – Описание теста I-33

No. теста	I-34
Цель теста	Проверка взаимодействия клиента и обработки запроса на загрузку файла при соответствии всем ограничениям (описаны ниже)
Тип теста	Позитивный
Объект тестирования	Связь UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File) и UploadComponent: UploadFormData()
Входные данные	Поле формы файла содержит объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ● Размер файла: не более 50 Мб ● Длина композиции: не менее 30 с. и не более 15 мин. ● Формат файла: wav, aif, mp3, ogg или flac
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 201.
Ожидаемый результат	Функция в UploadComponent выполняется без ошибок

Таблица 125 – Описание теста I-34

No. теста	I-35
Цель теста	Проверка взаимодействия клиента и обработки запроса на загрузку файла при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File) и UploadComponent: UploadFormData()

Входные данные	Поле формы файла содержит объект аудиофайла в формате multipart/form-data, обладающий следующими свойствами: ● Размер файла: не более 50 Мб ● Длина композиции: не менее 30 с. и не более 15 мин. ● Формат файла: wav, aif, mp3, ogg или flac
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 201.
Ожидаемый результат	Функция в UploadComponent обрабатывает исключение (вызывается processError)

Таблица 126 – Описание теста I-35

No. теста	I-36
Цель теста	Проверка взаимодействия клиента и обработки запроса на изменение метаданных файла
Тип теста	Позитивный
Объект тестирования	Связь UploadContoller: changeMetadata(@Req() request: Request) и UploadComponent: setMeta(trackId: int)
Входные данные	● Параметр ‘trackId’ содержит идентификатор аудиофайла на сервере: 0 ● Поля формы в компоненте выглядят следующим образом: { title: “New Title”, artist: “New Artist” }
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в UploadComponent завершает выполнение без исключений

Таблица 127 – Описание теста I-36

No. теста	I-37
Цель теста	Проверка взаимодействия клиента и обработки запроса на изменение метаданных файла при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь UploadContoller: changeMetadata(@Req() request: Request) и UploadComponent: setMeta(trackId: int)
Входные данные	● Параметр ‘trackId’ содержит идентификатор аудиофайла на сервере: 0 ● Поля формы в компоненте выглядят следующим образом: { title: “New Title”, artist: “New Artist” }
Косвенные входные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.

данные	
Ожидаемый результат	Функция в UploadComponent обрабатывает исключение (вызывается processError)

Таблица 128 – Описание теста I-37

No. теста	I-38
Цель теста	Проверка взаимодействия клиента и обработки запроса на удаление композиции из библиотеки
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: deleteTrack(@Req() request: Request) и SongListItemComponent: deleteFromLibrary()
Входные данные	Параметр ‘trackId’ внутри компонента содержит идентификатор аудиофайла на сервере: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в SongListItemComponent завершает выполнение без исключений

Таблица 129 – Описание теста I-38

No. теста	I-39
Цель теста	Проверка взаимодействия клиента и обработки запроса на удаление композиции из библиотеки при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь LibraryContoller: deleteTrack(@Req() request: Request) и SongListItemComponent: deleteFromLibrary()
Входные данные	● Параметр ‘trackId’ внутри компонента содержит идентификатор аудиофайла на сервере: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в SongListItemComponent обрабатывает исключение (вызывается processError)

Таблица 130 – Описание теста I-39

No. теста	I-40
Цель теста	Проверка взаимодействия клиента и обработки запроса на получение

	списка композиций из плейлиста
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: getTracks(@Req() request: Request) и PlaylistComponent: getTracks()
Входные данные	● Параметр ‘playlistId’ внутри компонента содержит идентификатор плейлиста на сервере: 1
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200 и телом ответа = [0, 1, 2, 3].
Ожидаемый результат	Функция PlaylistComponent возвращает [0, 1, 2, 3]

Таблица 131 – Описание теста I-40

No. теста	I-41
Цель теста	Проверка взаимодействия клиента и обработки запроса на получение списка композиций из плейлиста при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь LibraryContoller: getTracks(@Req() request: Request) и PlaylistComponent: getTracks()
Входные данные	● Параметр ‘playlistId’ внутри компонента содержит идентификатор плейлиста на сервере: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в PlaylistComponent обрабатывает исключение (вызывается processError)

Таблица 132 – Описание теста I-41

No. теста	I-42
Цель теста	Проверка взаимодействия клиента и обработки запроса на изменение принадлежности композиции к плейлисту
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: setTrackPlaylist(@Req() request: Request) и SongListItemComponent: setPlaylist(playlistId: int)
Входные данные	● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1 ● Параметр ‘trackId’ внутри компонента содержит идентификатор плейлиста на сервере: 0

Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в SongListItemComponent завершает выполнение без исключений

Таблица 133 – Описание теста I-42

No. теста	I-43
Цель теста	Проверка взаимодействия клиента и обработки запроса на изменение принадлежности композиции к плейлисту при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь LibraryContoller: setTrackPlaylist(@Req() request: Request) и SongListItemComponent: setPlaylist(playlistId: int)
Входные данные	● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1 ● Параметр ‘trackId’ внутри компонента содержит идентификатор плейлиста на сервере: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в SongListItemComponent обрабатывает исключение (вызывается processError)

Таблица 134 – Описание теста I-43

No. теста	I-44
Цель теста	Проверка взаимодействия клиента и обработки запроса на удаление композиции из плейлиста
Тип теста	Позитивный
Объект тестирования	Связь LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request) и SongListItemComponent: deleteTrackFromPlaylist(playlistId: int)
Входные данные	● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1 ● Параметр ‘trackId’ внутри компонента содержит идентификатор плейлиста на сервере: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в SongListItemComponent завершает выполнение без исключений

Таблица 135 – Описание теста I-44

No. теста	I-45
Цель теста	Проверка взаимодействия клиента и обработки запроса на удаление композиции из плейлиста при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request) и SongListItemComponent: deleteTrackFromPlaylist(playlistId: int)
Входные данные	● Параметр ‘playlistId’ содержит идентификатор плейлиста на сервере: 1 ● Параметр ‘trackId’ внутри компонента содержит идентификатор плейлиста на сервере: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в SongListItemComponent обрабатывает исключение (вызывается processError)

Таблица 136 – Описание теста I-45

No. теста	I-46
Цель теста	Проверка взаимодействия клиента и обработки запроса на создание плейлиста
Тип теста	Позитивный
Объект тестирования	Связь PlaylistController: createPlaylist(@Req() request: Request) и SidebarComponent: createPlaylist(name: string)
Входные данные	● Параметр ‘name’: “new mix”
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 201.
Ожидаемый результат	Функция в SidebarComponent завершает выполнение без исключений

Таблица 137 – Описание теста I-46

No. теста	I-47
Цель теста	Проверка взаимодействия клиента и обработки запроса на создание плейлиста при проблемах на стороне сервера
Тип теста	Негативный

Объект тестирования	Связь PlaylistController: createPlaylist(@Req() request: Request) и SidebarComponent: createPlaylist(name: string)
Входные данные	● Параметр ‘name’: “new mix”
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 201.
Ожидаемый результат	Функция в SidebarComponent обрабатывает исключение (вызывается processError)

Таблица 138 – Описание теста I-47

No. теста	I-48
Цель теста	Проверка взаимодействия клиента и обработки запроса на изменение плейлиста
Тип теста	Позитивный
Объект тестирования	Связь PlaylistController: editPlaylist(@Req() request: Request) и PlaylistComponent: editPlaylist(name: string)
Входные данные	● Параметр ‘name’: “new mix v2” ● Параметр ‘playlistId’ внутри компонента содержит идентификатор плейлиста на сервере: 1
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 201.
Ожидаемый результат	Функция в PlaylistComponent завершает выполнение без исключений

Таблица 139 – Описание теста I-48

No. теста	I-49
Цель теста	Проверка взаимодействия клиента и обработки запроса на изменение плейлиста при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь PlaylistController: editPlaylist(@Req() request: Request) и PlaylistComponent: editPlaylist(name: string)
Входные данные	● Параметр ‘name’: “new mix v2” ● Параметр ‘playlistId’ внутри компонента содержит идентификатор плейлиста на сервере: 1
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 201.
Ожидаемый	Функция в PlaylistComponent обрабатывает исключение (вызывается

результат	processError)

Таблица 140 – Описание теста I-49

No. теста	I-50
Цель теста	Проверка взаимодействия клиента и обработки запроса на удаление плейлиста
Тип теста	Позитивный
Объект тестирования	Связь PlaylistController: deletePlaylist(@Req() request: Request) и PlaylistComponent: deletePlaylist()
Входные данные	● Параметр ‘playlistId’ внутри компонента содержит идентификатор плейлиста на сервере: 1
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в PlaylistComponent завершает выполнение без исключений

Таблица 141 – Описание теста I-50

No. теста	I-51
Цель теста	Проверка взаимодействия клиента и обработки запроса на удаление плейлиста при проблемах на стороне сервера
Тип теста	Позитивный
Объект тестирования	Связь PlaylistController: deletePlaylist(@Req() request: Request) и PlaylistComponent: deletePlaylist()
Входные данные	● Параметр ‘playlistId’ внутри компонента содержит идентификатор плейлиста на сервере: 1
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в PlaylistComponent обрабатывает исключение (вызывается processError)

Таблица 142 – Описание теста I-51

No. теста	I-52
Цель теста	Проверка взаимодействия клиента и обработки запроса на генерацию плейлистов
Тип теста	Позитивный

Объект тестирования	Связь ChannelsController: generateChannels(@Req() request: Request) и ChannelsService: getMoodMixes()
Входные данные	–
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200 и телом ответа = { playlists: [{ title: “Happy Mix”, tracks: [0, 1, 5, 7, 10, 14] }] }.
Ожидаемый результат	Функция в ChannelsService возвращает JSON-объект: { playlists: [{ title: “Happy Mix”, tracks: [0, 1, 5, 7, 10, 14] }] }

Таблица 143 – Описание теста I-52

No. теста	I-53
Цель теста	Проверка взаимодействия клиента и обработки запроса на генерацию плейлистов при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь ChannelsController: generateChannels(@Req() request: Request) и ChannelsService: getMoodMixes()
Входные данные	–
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в ChannelsService обрабатывает исключение (вызывается processError)

Таблица 144 – Описание теста I-53

No. теста	I-54
Цель теста	Проверка взаимодействия клиента и обработки запроса на сохранение сгенерированных плейлистов
Тип теста	Позитивный
Объект тестирования	Связь ChannelsController: generateChannels(@Req() request: Request) и ChannelsService: saveAsPlaylist(name: string, trackIds: int[])
Входные данные	● Параметр ‘name’: “Happy Mix” ● Параметр ‘trackIds’: [0, 1, 5, 7, 10, 14]
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в ChannelsService завершает выполнение без исключений

Таблица 145 – Описание теста I-54

No. теста	I-55
Цель теста	Проверка взаимодействия клиента и обработки запроса на сохранение сгенерированных плейлистов при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь ChannelsController: generateChannels(@Req() request: Request) и ChannelsService: saveAsPlaylist(name: string, trackIds: int[])
Входные данные	● Параметр ‘name’: “Happy Mix” ● Параметр ‘trackIds’: [0, 1, 5, 7, 10, 14]
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в ChannelsService обрабатывает исключение (вызывается processError)

Таблица 146 – Описание теста I-55

No. теста	I-56
Цель теста	Проверка взаимодействия клиента и обработки запроса на получение информации о треке
Тип теста	Позитивный
Объект тестирования	Связь TracksController: getTrackInfo(@Req() request: Request) и TracksService: getTrackInfo(trackId: int)
Входные данные	● Параметр ‘trackId’ содержит идентификатор трека: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200 и телом ответа “{ id: 0, title: “Title 0”, artist: “Artist 0”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }”.
Ожидаемый результат	Функция в TracksService возвращает { id: 0, title: “Title 0”, artist: “Artist 0”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }

Таблица 147 – Описание теста I-56

No. теста	I-57
Цель теста	Проверка взаимодействия клиента и обработки запроса на получение информации о треке при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь TracksController: getTrackInfo(@Req() request: Request) и TracksService: getTrackInfo(trackId: int)
Входные данные	● Параметр ‘trackId’ содержит идентификатор трека: 0

Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в TracksService обрабатывает исключение (вызывается processError)

Таблица 148 – Описание теста I-57

No. теста	I-58
Цель теста	Проверка взаимодействия клиента и обработки запроса на получение файла трека
Тип теста	Позитивный
Объект тестирования	Связь TracksController: getTrackData(@Req() request: Request) и TracksService: getTrackAudio(trackId: int)
Входные данные	● Параметр ‘trackId’ содержит идентификатор трека: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200 и телом ответа = { id: 0, title: “Title 0”, data: [‘test’] } }.
Ожидаемый результат	Функция в TracksService возвращает { id: 0, title: “Title 0”, data: [‘test’] }

Таблица 149 – Описание теста I-58

No. теста	I-59
Цель теста	Проверка взаимодействия клиента и обработки запроса на получение файла трека при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь TracksController: getTrackData(@Req() request: Request) и TracksService: getTrackAudio(trackId: int)
Входные данные	● Параметр ‘trackId’ содержит идентификатор трека: 0
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в TracksService обрабатывает исключение (вызывается processError)

Таблица 150 – Описание теста I-59

No. теста	I-60
Цель теста	Проверка взаимодействия клиента и обработки запроса на регистрацию

Тип теста	Позитивный
Объект тестирования	Связь AuthController: signup(@Req() request: Request) и SignupComponent: signup()
Входные данные	● Поля формы компонента содержат следующую информацию: { username: “User 2”, password: “hellohello” }
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в SignupComponent завершает выполнение без исключений

Таблица 151 – Описание теста I-60

No. теста	I-61
Цель теста	Проверка взаимодействия клиента и обработки запроса на регистрацию при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь AuthController: signup(@Req() request: Request) и SignupComponent: signup()
Входные данные	● Поля формы компонента содержат следующую информацию: { username: “User 2”, password: “hellohello” }
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в SignupComponent обрабатывает исключение (вызывается processError)

Таблица 152 – Описание теста I-61

No. теста	I-62
Цель теста	Проверка взаимодействия клиента и обработки запроса на логин
Тип теста	Позитивный
Объект тестирования	Связь AuthController: login(@Req() request: Request) и LoginComponent: login()
Входные данные	● Поля формы компонента содержат следующую информацию: { username: “User 0”, password: “hellohello” }
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый	Функция в LoginComponent завершает выполнение без исключений

результат	

Таблица 153 – Описание теста I-62

No. теста	I-63
Цель теста	Проверка взаимодействия клиента и обработки запроса на логин при проблемах на стороне сервера
Тип теста	Негативный
Объект тестирования	Связь AuthController: login(@Req() request: Request) и LoginComponent: login()
Входные данные	• Поля формы компонента содержат следующую информацию: { username: “User 0”, password: “hellohello” }
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в LoginComponent обрабатывает исключение (вызывается processError)

Таблица 154 – Описание теста I-63

No. теста	I-64
Цель теста	Проверка взаимодействия клиента и обработки запроса на выход с сайта
Тип теста	Позитивный
Объект тестирования	Связь AuthController: logout(@Req() request: Request) и SidebarComponent: logout()
Входные данные	–
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния 200.
Ожидаемый результат	Функция в SidebarComponent завершает выполнение без исключений

Таблица 155 – Описание теста I-64

No. теста	I-65
Цель теста	Проверка взаимодействия клиента и обработки запроса на выход с сайта при проблемах на стороне сервера
Тип теста	Позитивный

Объект тестирования	Связь AuthController: logout(@Req() request: Request) и SidebarComponent: logout()
Входные данные	–
Косвенные входные данные	Заглушка бэкэнда возвращает ответ с кодом состояния **не** 200.
Ожидаемый результат	Функция в SidebarComponent обрабатывает исключение (вызывается processError)

Таблица 156 – Описание теста I-65

Описание аттестационных тестов

No. теста	A-01
Цель теста	Выполнение сценария визита на сайт
Тип теста	Позитивный (автоматизированный)
Объект тестирования	Интерфейс логина, страницы аутентификации/главного экрана
Входные данные	Исполняемый сценарий: пользователь заходит на сайт.
Косвенные входные данные	–
Ожидаемый результат	Пользователь заходит на сайт и видит модальный диалог для входа на сервис, неактивные компоненты плеера и боковой панели

Таблица 157 – Описание теста A-01

No. теста	A-02
Цель теста	Выполнение сценария аутентификации на сервере с проверкой элементов интерфейса
Тип теста	Позитивный (автоматизированный)
Объект тестирования	Интерфейс логина, страницы аутентификации/главного экрана после аутентификации на сервисе
Входные данные	Исполняемый сценарий: пользователь заходит на сайт и попадает на страницу входа на сайт. Найдя поля для ввода вводит данные для авторизации на сайте.
Косвенные входные данные	Вводимые данные о пользователе есть в БД.
Ожидаемый результат	Логин на сервис проходит успешно. Пользователь перенаправляется на главный экран приложения. В компоненте боковой панели отображается имя пользователя. В cookies браузера присутствует

	acess_token.

Таблица 158 – Описание теста А-02

No. теста	А-03
Цель теста	Выполнение сценария загрузки композиции на сервер с проверкой элементов интерфейса
Тип теста	Позитивный (ручной)
Объект тестирования	Интерфейс загрузки
Входные данные	Исполняемый сценарий: авторизованный пользователь нажимает на кнопку загрузки файла в компоненте боковой панели. Пользователь затем выбирает файл, соответствующий ограничениям (размер файла: не более 50 Мб, длина композиции: не менее 30 с. и не более 15 мин., формат файла: wav, aif, mp3, ogg или flac). При появлении диалога об изменении метаданных файла, пользователь ничего не изменяет и подтверждает загрузку
Косвенные входные данные	–
Ожидаемый результат	Загрузка проходит успешно. Композиция с неизменёнными данными отображается в библиотеке пользователя в разделе “В обработке”. Данные о характеристиках отсутствуют, вместо них – индикация того, что композиция находится в очереди на обработку.

Таблица 159 – Описание теста А-03

No. теста	А-04
Цель теста	Выполнение сценария проигрывания обработанной композиции с проверкой элементов интерфейса
Тип теста	Позитивный (ручной)
Объект тестирования	Интерфейс плеера
Входные данные	Исполняемый сценарий: авторизованный пользователь нажимает на кнопку проигрывания обработанной композиции из своей библиотеки. Затем пользователь кликает на кнопку воспроизведения/паузы дважды с задержкой в примерно 2 с. между нажатиями. После этого пользователь двигает ползунок громкости воспроизведения.
Косвенные входные данные	В БД есть данные о выбранной композиции, а также ссылка на файл аудио в ФС
Ожидаемый результат	В компоненте плеера отображается информация о композиции. Начинает проигрываться аудиофайл. После первого нажатия на кнопку паузы воспроизведение композиции приостанавливается, после повторного – возобновляется. При движении ползунка громкости

	изменяется громкость воспроизведения

Таблица 160 – Описание теста А-04

No. теста	А-05
Цель теста	Выполнение сценария создания плейлистов и добавления в него композиций с проверкой элементов интерфейса
Тип теста	Позитивный (автоматизированный)
Объект тестирования	Интерфейс плейлистов
Входные данные	Исполняемый сценарий: авторизованный пользователь нажимает на кнопку создания плейлиста, при появлении диалога вводит для него желаемое название. Затем добавляет несколько (2-3) композиций в созданный плейлист с помощью кнопки контекстного меню у компоненты SongListItem. После этого пользователь переходит к созданному плейлисту, кликая на появившуюся боковой панели опцию
Косвенные входные данные	В БД есть данные о выбранных композициях
Ожидаемый результат	В компоненте боковой панели, в разделе плейлисты отображается вкладка с созданным плейлистом с выбранным для плейлиста названием. При переходе к компоненте плейлиста, список композиций в нем должен соответствовать выбору треков ранее

Таблица 161 – Описание теста А-05

No. теста	А-06
Цель теста	Выполнение сценария использования инструмента генерации плейлистов с проверкой элементов интерфейса
Тип теста	Позитивный (ручной)
Объект тестирования	Интерфейс генерации плейлистов, интерфейс плейлистов
Входные данные	Исполняемый сценарий: авторизованный пользователь нажимает на кнопку автоматической генерации плейлистов из боковой панели
Косвенные входные данные	В БД есть количество записей о композициях достаточное для создания нескольких (2-3) плейлистов.
Ожидаемый результат	После некоторого ожидания в боковой панели появляется список сгенерированных плейлистов с возможностью перейти на страницу каждого плейлиста отдельно

Таблица 162 – Описание теста А-06

Описание тестирования UI

No. теста	U-01
Цель теста	Проверка интерфейса формы регистрации при попытке отправить форму с незаполненными полями
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма регистрации (SignupComponent)
Входные данные	В форму не введены данные в полях “Пароль”, “Подтверждение пароля”. Была произведена попытка отправить данные формы
Косвенные входные данные	–
Ожидаемый результат	Необходимые поля для заполнения подсвечиваются красным и отмечаются звездочкой. Также под каждым необходимым полем для заполнения отображается лейбл “Это поле необходимо заполнить”

Таблица 163 – Описание теста U-01

No. теста	U-02
Цель теста	Проверка интерфейса формы регистрации при попытке отправить форму с некорректным паролем
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма регистрации (SignupComponent)
Входные данные	В форме в поле “Пароль” был введен текст длиной менее 8 символов. Была произведена попытка отправить данные формы
Косвенные входные данные	–
Ожидаемый результат	Поле для заполнения подсвечивается красным. Также под полем отображается лейбл “Длина пароля должна составлять не менее 8 символов”

Таблица 164 – Описание теста U-02

No. теста	U-03
Цель теста	Проверка интерфейса формы регистрации при попытке отправить форму с несовпадающим полем для проверки пароля
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма регистрации (SignupComponent)

Входные данные	В форме в поле “Пароль” был введен текст длиной не менее 8 символов, в поле “Подтверждение пароля” был введен текст, отличный от введенного ранее. Была произведена попытка отправить данные формы
Косвенные входные данные	–
Ожидаемый результат	Поле “Подтверждение пароля” подсвечивается красным. Также под полем отображается лейбл “Пароли должны совпадать”

Таблица 165 – Описание теста U-03

No. теста	U-04
Цель теста	Проверка интерфейса формы регистрации при попытке отправить форму с несовпадающим полем для проверки пароля
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма регистрации (SignupComponent)
Входные данные	Все поля формы заполнены (пароль не менее 8 символов, поля проверки совпадают), имя пользователя выбрано таким образом, что запись о нём существует в БД. Была произведена попытка отправить данные формы
Косвенные входные данные	В БД есть данные о пользователе с выбранным никнеймом
Ожидаемый результат	Поле “Имя пользователя” подсвечивается красным. Также под полем отображается лейбл “Это имя пользователя уже занято”

Таблица 166 – Описание теста U-04

No. теста	U-05
Цель теста	Проверка интерфейса формы входа на сайт при попытке отправить данные, не относящиеся к какому-либо пользователю в БД
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма входа на сайт (LoginComponent)
Входные данные	Все поля формы заполнены (или не заполнены). Данные подобраны таким образом, что в БД нет соответствующих им записей. Была произведена попытка отправить данные формы
Косвенные входные данные	В БД нет данных о пользователе с выбранным никнеймом/паролем
Ожидаемый результат	Логин на сервис проваливается. Под формой появляется лейбл “Не удалось войти на сайт”

Таблица 167 – Описание теста U-05

No. теста	U-06
Цель теста	Проверка интерфейса загрузки файла
Тип теста	Позитивный (автоматизированный)
Объект тестирования	Форма загрузки файла (UploadComponent)
Входные данные	Поле формы для файла не заполнено (файл не выбран)
Косвенные входные данные	–
Ожидаемый результат	Кнопка для отправки файла на сервер неактивна

Таблица 168 – Описание теста U-06

No. теста	U-07
Цель теста	Проверка интерфейса загрузки файла при попытке отправить файл неподдерживаемого формата
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма загрузки файла (UploadComponent)
Входные данные	Поле формы для файла заполнено (файл выбран). Формат файла не является ни одним из следующих значений: wav, aif, mp3, ogg или flac. Была произведена попытка отправить данные формы
Косвенные входные данные	–
Ожидаемый результат	Файл не отправляется. Под полем для выбора файла отображается лейбл “Неподдерживаемый формат”

Таблица 169 – Описание теста U-07

No. теста	U-08
Цель теста	Проверка интерфейса загрузки файла при попытке отправить файл неподдерживаемой длительности
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма загрузки файла (UploadComponent)
Входные данные	Поле формы для файла заполнено (файл выбран). Формат файла является одним из следующих значений: wav, aif, mp3, ogg или flac. Длина композиции: менее 30 с. или более 15 мин. Была произведена

	попытка отправить данные формы
Косвенные входные данные	–
Ожидаемый результат	Файл не отправляется. Под полем для выбора файла отображается лейбл “Длина композиции должна быть не менее 30 с. и не более 15 мин.”

Таблица 170 – Описание теста U-08

No. теста	U-09
Цель теста	Проверка интерфейса загрузки файла при попытке отправить файл большого размера
Тип теста	Негативный (автоматизированный)
Объект тестирования	Форма загрузки файла (UploadComponent)
Входные данные	Поле формы для файла заполнено (файл выбран). Формат файла является одним из следующих значений: wav, aif, mp3, ogg или flac. Файл проходит ограничение по длительности из предыдущего теста. Размер файла более 50 Мб. Была произведена попытка отправить данные формы
Косвенные входные данные	–
Ожидаемый результат	Файл не отправляется. Под полем для выбора файла отображается лейбл “Размер файла не должен превышать 50 Мб”

Таблица 171 – Описание теста U-09

No. теста	U-10
Цель теста	Проверка интерфейса компоненты элемента композиции в плейлисте
Тип теста	Позитивный (автоматизированный)
Объект тестирования	Компонент SongListItem (Child of PlaylistComponent)
Входные данные	Внутреннее поле ‘trackId’ компонента равно 0 при создании
Косвенные входные данные	В БД есть данные о запрашиваемой композиции { id: 0, title: “Title 0”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }
Ожидаемый результат	● Текст в лейбле названия композиции = “Title 0” ● Текст в лейбле названия исполнителя = “A” ● Значение поля характеристики bpm = 120 ● Значение поля характеристики tone = C Major ● Значение happiness обозначается иконкой для высокого уровня

	● Значение danceability обозначается иконкой для высокого уровня ● Значение energy обозначается иконкой для среднего уровня

Таблица 172 – Описание теста U-10

Описание тестирования производительности

No. теста	L-01
Цель теста	Проверка производительности сервера (исполнение запроса данных о комопзиции)
Тип теста	Позитивный (автоматизированный)
Объект тестирования	Backend API
Входные данные	Количество виртуальных пользователей = 100
Косвенные входные данные	В БД есть данные о запрашиваемой композиции { id: 0, title: “Title 0”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }
Ожидаемый результат	Данные о производительности сервера в форме отчета

Таблица 173 – Описание теста L-01

No. теста	L-02
Цель теста	Проверка производительности сервера (исполнение запроса данных о комопзиции)
Тип теста	Позитивный (автоматизированный)
Объект тестирования	Backend API
Входные данные	Количество виртуальных пользователей = 1000
Косвенные входные данные	В БД есть данные о запрашиваемой композиции { id: 0, title: “Title 0”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }
Ожидаемый результат	Данные о производительности сервера в форме отчета

Таблица 173 – Описание теста L-01

No. теста	L-03
Цель теста	Проверка производительности сервера (исполнение запроса данных о комопзиции)

Тип теста	Позитивный (автоматизированный)
Объект тестирования	Backend API
Входные данные	Количество виртуальных пользователей = 10000
Косвенные входные данные	В БД есть данные о запрашиваемой композиции { id: 0, title: “Title 0”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }
Ожидаемый результат	Данные о производительности сервера в форме отчета

Таблица 173 – Описание теста L-03

Пример реализации тестов

```
beforeAll(async () => {
  app = await Test.createTestingModule({
    controllers: [LibraryController],
    providers: [LibraryService],
  }).compile();

  libraryService = app.get<LibraryService>(LibraryService);
  libraryController = app.get<LibraryController>(LibraryController);
});

Run | Debug
describe('getAudioInfo', () => {
  Run | Debug
  it('should return "Welcome to api!"', async () => {
    const result = ['1', '2', '3', '4'];
    jest.spyOn(libraryService, 'getTracks').mockImplementation((id) => result);

    expect(await libraryController.getTracks(1)).toBe(result);
  });
});
```

Рисунок 1 – Пример для теста B-24

```
beforeEach(async () => {
  TestBed.configureTestingModule({
    imports: [HttpClientTestingModule],
    providers: [ TracksService ]
  })
  .compileComponents();

  httpTestingController = TestBed.inject(HttpTestingController);
  service = TestBed.inject(TracksService);
});

Run | Debug
describe('getTrackInfo', () => {
  Run | Debug
  it('should return info object', () => {
    const result = {
      id: 0, title: "Title 0", artist: "Artist 0",
      bpm: 120, tone: "Cmajor",
      happiness: 9, danceability: 9, energy: 6
    };

    service.getTrackInfo(0)
      .subscribe(data => {
        expect(data).toEqual(result)
      });

    const req = httpTestingController.expectOne(getTrackInfoUrl + '?trackId=0');
    expect(req.request.method).toEqual('GET');

    req.flush(result);
  });
});
```

Рисунок 2 – Пример для теста I-56

Оценка покрытия тестирования

Расчет тестового покрытия относительно исполняемого кода программного обеспечения проводится по формуле

$$T_{cov} = \frac{L_{tc}}{L_{code}} \times 100\%,$$

где:

- T_{cov} – тестовое покрытие

- L_{tc} – количество строк кода, покрытых тестами
- L_{code} – количество строк кода (общее)

Также для оценки покрытия используются программные средства инструментов автоматического тестирования (Jest).

К сожалению, на данный момент, в текущем этапе разработки приложения, произвести оценку покрытия кода тестами не представляется возможным.

4. Журнал тестирования

Модульное тестирование

Дата	Тестировщик / наблюдатель	Тест	Количество запусков	Итоговый результат	Обнаруженные ошибки
13.12.2022	А. Д. Фомин	B-01	1	Не пройдено	Отчет E1
–	–	B-02	–	Не протестировано	–
–	–	B-03	–	Не протестировано	–
–	–	B-04	–	Не протестировано	–
–	–	B-05	–	Не протестировано	–
–	–	B-06	–	Не протестировано	–
–	–	B-07	–	Не протестировано	–
–	–	B-08	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-09	1	Не пройдено	Отчет E2
–	–	B-10	–	Не протестировано	–
–	–	B-11	–	Не протестировано	–
–	–	B-12	–	Не протестировано	–
–	–	B-13	–	Не протестировано	–
–	–	B-14	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-15	1	Не пройдено	Отчет E3
–	–	B-16	–	Не протестировано	–
–	–	B-17	–	Не протестировано	–
–	–	B-18	–	Не протестировано	–

–	–	B-19	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-20	1	Не пройдено	Отчет E4
–	–	B-21	–	Не протестировано	–
–	–	B-22	–	Не протестировано	–
–	–	B-23	–	Не протестировано	–
–	–	B-24	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-25	1	Не пройдено	Отчет E5
–	–	B-26	–	Не протестировано	–
–	–	B-27	–	Не протестировано	–
–	–	B-28	–	Не протестировано	–
–	–	B-29	–	Не протестировано	–
–	–	B-30	–	Не протестировано	–
–	–	B-31	–	Не протестировано	–
–	–	B-32	–	Не протестировано	–
–	–	B-33	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-34	1	Не пройдено	Отчет E6
–	–	B-35	–	Не протестировано	–
–	–	B-36	–	Не протестировано	–
–	–	B-37	–	Не протестировано	–
–	–	B-38	–	Не протестировано	–
–	–	B-39	–	Не протестировано	–
–	–	B-40	–	Не протестировано	–
–	–	B-41	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-42	1	Не пройдено	Отчет E7
–	–	B-43	–	Не протестировано	–
–	–	B-44	–	Не протестировано	–
–	–	B-45	–	Не протестировано	–
–	–	B-46	–	Не протестировано	–

13.12.2022	А. Д. Фомин	B-47	1	Не пройдено	Отчет E8
–	–	B-48	–	Не протестировано	–
–	–	B-49	–	Не протестировано	–
–	–	B-50	–	Не протестировано	–
–	–	B-51	–	Не протестировано	–
–	–	B-52	–	Не протестировано	–
–	–	B-53	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-54	1	Не пройдено	Отчет E9
–	–	B-55	–	Не протестировано	–
–	–	B-56	–	Не протестировано	–
–	–	B-57	–	Не протестировано	–
–	–	B-58	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-59	1	Не пройдено	Отчет E10
–	–	B-60	–	Не протестировано	–
–	–	B-61	–	Не протестировано	–
–	–	B-62	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-63	1	Не пройдено	Отчет E11
–	–	B-64	–	Не протестировано	–
–	–	B-65	–	Не протестировано	–
–	–	B-66	–	Не протестировано	–
–	–	B-67	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-68	1	Не пройдено	Отчет E12
–	–	B-69	–	Не протестировано	–
–	–	B-70	–	Не протестировано	–
–	–	B-71	–	Не протестировано	–
–	–	B-72	–	Не протестировано	–
–	–	B-73	–	Не протестировано	–
–	–	B-74	–	Не протестировано	–

13.12.2022	А. Д. Фомин	B-75	1	Не пройдено	Отчет E13
–	–	B-76	–	Не протестировано	–
–	–	B-77	–	Не протестировано	–
–	–	B-78	–	Не протестировано	–
–	–	B-79	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-80	1	Не пройдено	Отчет E14
–	–	B-81	–	Не протестировано	–
–	–	B-82	–	Не протестировано	–
–	–	B-83	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-84	1	Не пройдено	Отчет E15
–	–	B-85	–	Не протестировано	–
–	–	B-86	–	Не протестировано	–
–	–	B-87	–	Не протестировано	–
13.12.2022	А. Д. Фомин	B-88	1	Не пройдено	Отчет E16
–	–	B-89	–	Не протестировано	–
–	–	B-90	–	Не протестировано	–

Таблица 174 – Журнал блочного тестирования

Интеграционное тестирование

Дата	Тестировщик / наблюдатель	Тест	Количество запусков	Итоговый результат	Обнаруженные ошибки
–	–	I-01	–	Не протестировано	–
–	–	I-02	–	Не протестировано	–
–	–	I-03	–	Не протестировано	–
–	–	I-04	–	Не протестировано	–
–	–	I-05	–	Не протестировано	–
–	–	I-06	–	Не протестировано	–
–	–	I-07	–	Не протестировано	–
–	–	I-08	–	Не протестировано	–

–	–	I-09	–	Не протестировано	–
–	–	I-10	–	Не протестировано	–
–	–	I-11	–	Не протестировано	–
–	–	I-12	–	Не протестировано	–
–	–	I-13	–	Не протестировано	–
–	–	I-14	–	Не протестировано	–
–	–	I-15	–	Не протестировано	–
–	–	I-16	–	Не протестировано	–
–	–	I-17	–	Не протестировано	–
–	–	I-18	–	Не протестировано	–
–	–	I-19	–	Не протестировано	–
–	–	I-20	–	Не протестировано	–
–	–	I-21	–	Не протестировано	–
–	–	I-22	–	Не протестировано	–
–	–	I-23	–	Не протестировано	–
–	–	I-24	–	Не протестировано	–
–	–	I-25	–	Не протестировано	–
–	–	I-26	–	Не протестировано	–
–	–	I-27	–	Не протестировано	–
–	–	I-28	–	Не протестировано	–
–	–	I-29	–	Не протестировано	–
–	–	I-30	–	Не протестировано	–
–	–	I-31	–	Не протестировано	–
–	–	I-32	–	Не протестировано	–
–	–	I-33	–	Не протестировано	–
–	–	I-34	–	Не протестировано	–
–	–	I-35	–	Не протестировано	–
–	–	I-36	–	Не протестировано	–

–	–	I-37	–	Не протестировано	–
–	–	I-38	–	Не протестировано	–
–	–	I-39	–	Не протестировано	–
–	–	I-40	–	Не протестировано	–
–	–	I-41	–	Не протестировано	–
–	–	I-42	–	Не протестировано	–
–	–	I-43	–	Не протестировано	–
–	–	I-44	–	Не протестировано	–
–	–	I-45	–	Не протестировано	–
–	–	I-46	–	Не протестировано	–
–	–	I-47	–	Не протестировано	–
–	–	I-48	–	Не протестировано	–
–	–	I-49	–	Не протестировано	–
–	–	I-50	–	Не протестировано	–
–	–	I-51	–	Не протестировано	–
–	–	I-52	–	Не протестировано	–
–	–	I-53	–	Не протестировано	–
–	–	I-54	–	Не протестировано	–
–	–	I-55	–	Не протестировано	–
–	–	I-56	–	Не протестировано	–
–	–	I-57	–	Не протестировано	–
–	–	I-58	–	Не протестировано	–
–	–	I-59	–	Не протестировано	–
–	–	I-60	–	Не протестировано	–
–	–	I-61	–	Не протестировано	–
–	–	I-62	–	Не протестировано	–
–	–	I-63	–	Не протестировано	–
–	–	I-64	–	Не протестировано	–

–	–	I-65	–	Не протестировано	–

Таблица 175 – Журнал интеграционного тестирования

Аттестационное тестирование

Дата	Тестировщик / наблюдатель	Тест	Количество запусков	Итоговый результат	Обнаруженные ошибки
13.12.2022	А. Д. Фомин	A-01	1	Пройдено	–
–	–	A-02	–	Не протестировано	–
–	–	A-03	–	Не протестировано	–
–	–	A-04	–	Не протестировано	–
–	–	A-05	–	Не протестировано	–
–	–	A-06	–	Не протестировано	–

Таблица 176 – Журнал аттестационного тестирования

Тестирование UI

Дата	Тестировщик / наблюдатель	Тест	Количество запусков	Итоговый результат	Обнаруженные ошибки
–	–	U-01	–	Не протестировано	–
–	–	U-02	–	Не протестировано	–
–	–	U-03	–	Не протестировано	–
–	–	U-04	–	Не протестировано	–
–	–	U-05	–	Не протестировано	–
–	–	U-06	–	Не протестировано	–
–	–	U-07	–	Не протестировано	–
–	–	U-08	–	Не протестировано	–
–	–	U-09	–	Не протестировано	–
–	–	U-10	–	Не протестировано	–

Таблица 177 – Журнал тестирования UI

Тестирование производительности

Дата	Тестировщик / наблюдатель	Тест	Количество запусков	Итоговый результат	Обнаруженные ошибки
–	–	L-01	–	Не протестировано	–
–	–	L-02	–	Не протестировано	–
–	–	L-03	–	Не протестировано	–

Таблица 178 – Журнал тестирования производительности

5. Журнал ошибок

No. отчета	E1
No. теста	B-01
Дата составления	13.12.2022
Модуль	UploadContoller: uploadTrack(@Req() request: Request, @UploadedFile() file: Express.Multer.File)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Audio created on server”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 179 – Отчет об ошибке E1

No. отчета	E2
No. теста	B-09
Дата составления	13.12.2022
Модуль	UploadContoller: changeMetadata(@Req() request: Request))
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Audio {id} metadata changed”

Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 180 – Отчет об ошибке E2

No. отчета	E3
No. теста	B-15
Дата составления	13.12.2022
Модуль	LibraryContoller: deleteTrack(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Audio {idAudio} deleted from {idUser}’s library”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 181 – Отчет об ошибке E3

No. отчета	E4
No. теста	B-20
Дата составления	13.12.2022
Модуль	LibraryContoller: getTracks(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	array: [1, 2, 3, 4]
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 182 – Отчет об ошибке E4

No. отчета	E5
No. теста	B-25
Дата составления	13.12.2022
Модуль	LibraryContoller: setTrackPlaylist(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Track {trackId} has been added to playlist {playlistId}”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 183 – Отчет об ошибке E5

No. отчета	E6
No. теста	B-34
Дата составления	13.12.2022
Модуль	LibraryContoller: deleteTrackFromPlaylist(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Track {trackId} has been deleted from playlist {playlistId}”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 184 – Отчет об ошибке E6

No. отчета	E7
No. теста	B-42
Дата составления	13.12.2022

Модуль	PlaylistController: createPlaylist(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Playlist {playlistId} created on server”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 185 – Отчет об ошибке E7

No. отчета	E8
No. теста	B-47
Дата составления	13.12.2022
Модуль	PlaylistController: editPlaylist(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Playlist {playlistId} changed”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 186 – Отчет об ошибке E8

No. отчета	E9
No. теста	B-54
Дата составления	13.12.2022
Модуль	PlaylistController: deletePlaylist(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Playlist {playlistId} changed”

Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 187 – Отчет об ошибке E9

No. отчета	E10
No. теста	B-59
Дата составления	13.12.2022
Модуль	ChannelsController: generateChannels(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	JSON-string: “{ playlists: [{ title: “Happy Mix”, tracks: [0, 1, 5, 7, 10, 14] }] }”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 188 – Отчет об ошибке E10

No. отчета	E11
No. теста	B-63
Дата составления	13.12.2022
Модуль	ChannelsController: saveAsPlaylist(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Playlist {playlistId} saved”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 189 – Отчет об ошибке E11

No. отчета	E12
No. теста	B-68
Дата составления	13.12.2022
Модуль	TracksController: getTrackInfo(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	JSON-string: “{ id: 0, title: “Title 0”, artist: “Artist 0”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 }”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 190 – Отчет об ошибке E12

No. отчета	E13
No. теста	B-75
Дата составления	13.12.2022
Модуль	TracksController: getTrackData(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	JSON-string: { id: 0, title: “Title 0”, data: “test” }”
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 191 – Отчет об ошибке E13

No. отчета	E14
No. теста	B-80
Дата составления	13.12.2022

Модуль	AuthController: signup(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	JSON-string: { accessToken: {accessToken} }
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 192 – Отчет об ошибке E14

No. отчета	E15
No. теста	B-84
Дата составления	13.12.2022
Модуль	AuthController: login(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	JSON-string: { accessToken: {accessToken} }
Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 193 – Отчет об ошибке E15

No. отчета	E16
No. теста	B-88
Дата составления	13.12.2022
Модуль	AuthController: logout(@Req() request: Request)
Тип ошибки	Срабатывание исключения
Ожидаемый результат	string: “Logged out successfully”

Фактическое значение	NotImplementedException
Предлагаемое исправление	Изменить имплементацию тестируемого метода
Состояние ошибки	Не исправлено

Таблица 194 – Отчет об ошибке E16

6. Результаты

Было проведено тестирование приложения, хоть и в малом объеме, но тем не менее, оно уже помогло выявить несколько ошибок в реализации веб-сервиса, которые были задокументированы и требуют рассмотрения в дальнейшем. С точки зрения полноты, на данный момент уровень тестирования приложения является неудовлетворительным, но у полученных итогов есть и положительные аспекты (найдены ошибки для исправления).

На нынешнем этапе, по результатам тестирования можно сказать, что система **не** является работоспособной. Её разработка будет продолжена в обозримом будущем, и немного позднее система снова будет участвовать в процессе тестирования.

Приложение

Приложение А — Схема архитектуры системы

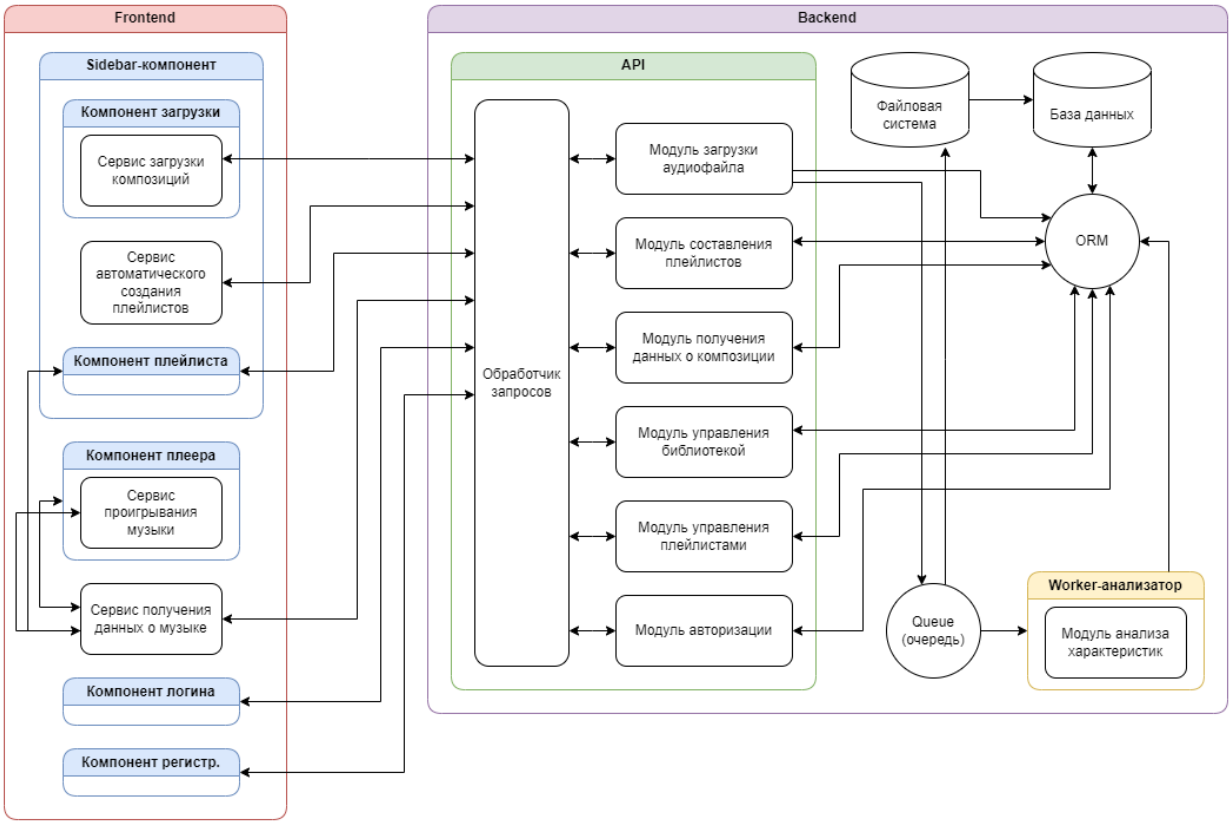

Рисунок 3 – Схема архитектуры системы

Приложение В — Данные, возвращаемые заглушками

```
{
  tracks: [
    { id: 0, title: “Title 0”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 },
    { id: 1, title: “Title 1”, artist: “A”, bpm: 128, tone: “Aminor”, happiness: 8, danceability: 9, energy: 7 },
    { id: 2, title: “Title 2”, artist: “A”, bpm: 98, tone: “Gbmajor”, happiness: 7, danceability: 9, energy: 6 },
    { id: 3, title: “Title 3”, artist: “A”, bpm: 160, tone: “Cmajor”, happiness: 5, danceability: 8, energy: 8 },
    { id: 4, title: “Title 4”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 2, danceability: 1, energy: 6 },
    { id: 5, title: “Title 5”, artist: “A”, bpm: 145, tone: “Aminor”, happiness: 9, danceability: 5, energy: 9 },
    { id: 6, title: “Title 6”, artist: “A”, bpm: 180, tone: “Gbmajor”, happiness: 9, danceability: 9, energy: 6 },
    { id: 7, title: “Title 7”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 },
    { id: 8, title: “Title 8”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 10, danceability: 9, energy: 10 },
    { id: 9, title: “Title 9”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 1, danceability: 9, energy: 6 },
    { id: 10, title: “Title 10”, artist: “A”, bpm: 120, tone: “Aminor”, happiness: 9, danceability: 9, energy: 4 },
    { id: 11, title: “Title 11”, artist: “A”, bpm: 60, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 },
    { id: 12, title: “Title 12”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 3, danceability: 9, energy: 6 },
    { id: 13, title: “Title 13”, artist: “A”, bpm: 130, tone: “Aminor”, happiness: 4, danceability: 9, energy: 5 },
```

```
    { id: 14, title: “Title 14”, artist: “A”, bpm: 120, tone: “Cmajor”, happiness: 9, danceability: 9, energy: 6 },
    { id: 15, title: “Title 15”, artist: “A”, bpm: 150, tone: “Gbmajor”, happiness: 9, danceability: 9, energy: 6 }
  ]
}
```

Листинг 1 – Данные возвращаемые заглушкой для generateChannels()

Приложение С — Начальное состояние БД для групп тестов

tblTracks							
id	**title**	**artist**	**bpm**	**tone**	**happiness**	**danceability**	**energy**
0	Title 0	Artist 0	120	Cmajor	9	9	6
1	Title 1	Artist 0	128	Aminor	8	9	7
2	Title 2	Artist 0	98	Gbmajor	7	9	6
3	Title 3	Artist 0	160	Cmajor	5	8	8
4	Title 4	Artist 0	120	Cmajor	2	1	6
5	Title 5	Artist 0	145	Gbmajor	9	5	9
6	Title 6	Artist 0	180	Cmajor	9	9	6
7	Title 7	Artist 0	120	Cmajor	9	9	6
8	Title 8	Artist 0	120	Cmajor	10	9	10
9	Title 9	Artist 0	120	Cmajor	1	9	6
10	Title 10	Artist 0	120	Aminor	9	9	4
11	Title 11	Artist 0	60	Cmajor	9	9	6
12	Title 12	Artist 0	120	Cmajor	3	9	6
13	Title 13	Artist 0	130	Aminor	4	9	5
14	Title 14	Artist 0	120	Cmajor	9	9	6
15	Title 15	Artist 0	150	Cmajor	9	9	6

Таблица 195 – Таблица tblTracks

tblTrackLocations

id	idTracks	flieLocation
0	0	path/to/file

Таблица 196 – Таблица tblTrackLocations

tblUsers			
id	**username**	**password**	**libraryId**
0	User 0	Хэш от “hellohello”	0
1	User 1	Хэш от “hellohello”	2

Таблица 197 – Таблица tblUsers

tblPlaylists			
id	**userId**	**name**	**isLibrary**
0	0	null	true
1	0	My mix	false
2	1	null	true
3	1	My mix	false

Таблица 198 – Таблица tblPlaylists

tblPlaylistsTracks		
id	**playlistId**	**trackId**
0	0	0
1	0	1
2	0	2
3	0	3
4	1	4
5	0	5
6	0	6
7	0	7
8	0	8

9	0	9
10	0	10
11	0	11
12	0	12
13	0	13
14	0	14
15	1	15
16	2	0
17	2	1
18	2	2
19	2	3
20	3	4
21	3	15

Таблица 199 – Таблица tblPlaylistsTracks