федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный университет геодезии и картографии»

(МИИГАиК)

Факультет геоинформатики и информационной безопасности

Специальность 09.03.03 «Прикладная информатика»

Кафедра информатики и геоинформационных технологий

ОТЧЕТ ПО

УЧЕБНОЙ (ОЗНАКОМИТЕЛЬНОЙ) ПРАКТИКЕ

ПО ПРОГРАММИРОВАНИЮ

Исполнитель студент группы 2024-ФГиИБ-ПИ-1б

Никитинская Татьяна Геннадьевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель производственной практики Лебедев Е.Д. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2025

**Дневник практики**

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Выполняемая работа | Отметка о выполнении | Примечание |
| 07.07.2025 – 08.07.2025 | Ознакомление с заданиями практики. Выбор темы проекта (аудиоплеер). Обзор Flet для создания интерфейса. | Выполнено |  |
| 09.07.2025 – 10.07.2025 | Разработка структуры интерфейса. Проектирование вкладок и основных элементов управления. | Выполнено |  |
| 11.07.2025 – 12.07.2025 | Создание класса UIComponents. Реализация базового интерфейса плеера (кнопки, слайдер, заголовок трека). | Выполнено |  |
| 14.07.2025 – 15.07.2025 | Программирование логики интерфейса: слайдер громкости (volume\_change), кнопки управления громкостью (volume\_up, volume\_down). | Выполнено |  |
| 16.07.2025 – 17.07.2025 | Завершение базового интерфейса плеера. Настройка отображения текущего времени и длительности трека. | Выполнено |  |
| 18.07.2025 – 19.07.2025 | Реализация вкладок «В очереди» (update\_queue\_list) и «Статистика» (update\_stats\_list). Настройка их обновления. | Выполнено |  |
| 21.07.2025 – 22.07.2025 | Добавление справки (toggle\_help) и переключателя питания (toggle\_player\_power). Тестирование интерфейса. | Выполнено |  |
| 23.07.2025 – 24.07.2025 | Проведение тестирования интерфейса. Подготовка презентационных материалов. | Выполнено |  |
| 25.07.2025 | Подведение итогов. Написание выводов. Сдача дневника и отчёта. | Выполнено |  |

Оглавление

Введение4

Глава 1 «Описание структуры проекта»5

Глава 2 «Реализация аудиоплеера на Python»9

2.1 Модуль управления воспроизведением (AudioPlayerManager)9

2.2 Модуль графического интерфейса (UIComponents)10

2.3 Правила построения интерфейсов 11

2.4. Логика взаимодействия с интерфейсом12

Глава 3 «Тестирование аудиоплеера на Python»17

* 1. Модульное тестирование17
  2. Нефункциональное тестирование18

Выводы20

Список использованных источников22

**Введение**

Разработка аудиоплеера на языке программирования Python представляет собой увлекательный и практически значимый проект, который объединяет в себе работу с мультимедийными данными, создание интуитивно понятного пользовательского интерфейса и реализацию сложной логики управления воспроизведением. Этот проект был выбран благодаря глубокому интересу к обработке мультимедиа, а также стремлению создать полноценное приложение, которое может быть использовано в реальных сценариях. Аудиоплеер — это не только технически интересная задача, но и возможность применить и углубить навыки программирования, накопленные за год изучения Python, включая работу с библиотеками, структурирование кода, отладку и проектирование интерфейсов.

Цель проекта — создать функциональный аудиоплеер с поддержкой базовых функций, таких как воспроизведение, пауза, переключение треков, регулировка громкости, а также дополнительные возможности, такие как визуализация аудио, управление очередью воспроизведения и статистика прослушиваний. Использование библиотеки Flet для создания графического интерфейса позволяет сделать приложение кроссплатформенным и современным, а интеграция с аудиообработкой предоставляет возможность изучить особенности работы с мультимедийными данными в Python.

**Глава 1 "Описание структуры проекта"**

Аудиоплеер реализован на Python с использованием библиотеки Flet, которая позволяет создавать кроссплатформенные графические интерфейсы.

**AudioPlayerManager**: отвечает за логику воспроизведения и управление плейлистом. Основные функции:

* + Управление воспроизведением (воспроизведение, пауза, переключение треков).
  + Проверка доступности аудиофайлов по URL с использованием библиотеки requests.
  + Обработка событий плеера (изменение состояния, позиции трека, загрузка трека).
  + Форматирование времени воспроизведения и управление громкостью.

**UIComponents**: Отвечает за создание графического интерфейса. Основные элементы:

* + Отображение названия трека, времени воспроизведения и длительности.
  + Элементы управления: кнопки воспроизведения/паузы, переключения треков, регулировки громкости.
  + Ползунок для навигации по треку и визуализации прогресса воспроизведения.
  + Дополнительные функции: переключатель питания и панель справки.

Плейлист состоит из предопределенных URL-адресов аудиофайлов, загружаемых из интернета. Интерфейс выполнен в темной теме с использованием визуальных элементов (ползунки, кнопки, текстовые поля) для удобного взаимодействия с пользователем.

Разработка, выполненная мной, сосредоточена на программной части:

* Реализация функционала аудиоплеера с использованием Flet для создания интуитивно понятного интерфейса.
* Обеспечение стабильной работы воспроизведения и обработки сетевых аудиофайлов.
* Создание модульной структуры кода, позволяющей легко расширять функционал в будущем.

**Структура проекта**

* **Программная часть**:
  + audio\_player.py: основной файл с логикой плеера и интерфейсом.
  + Используемые библиотеки: flet, flet\_audio, requests.

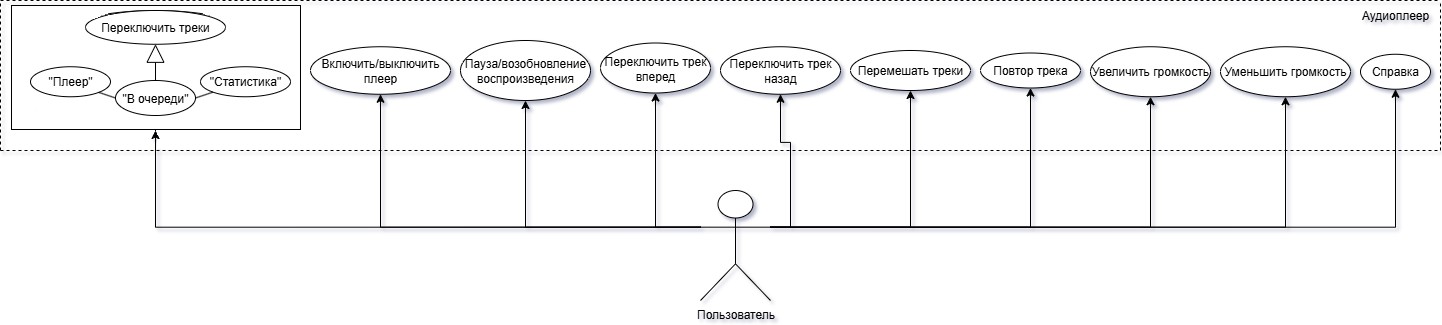


Рис. 1 «Диаграмма вариантов использования»

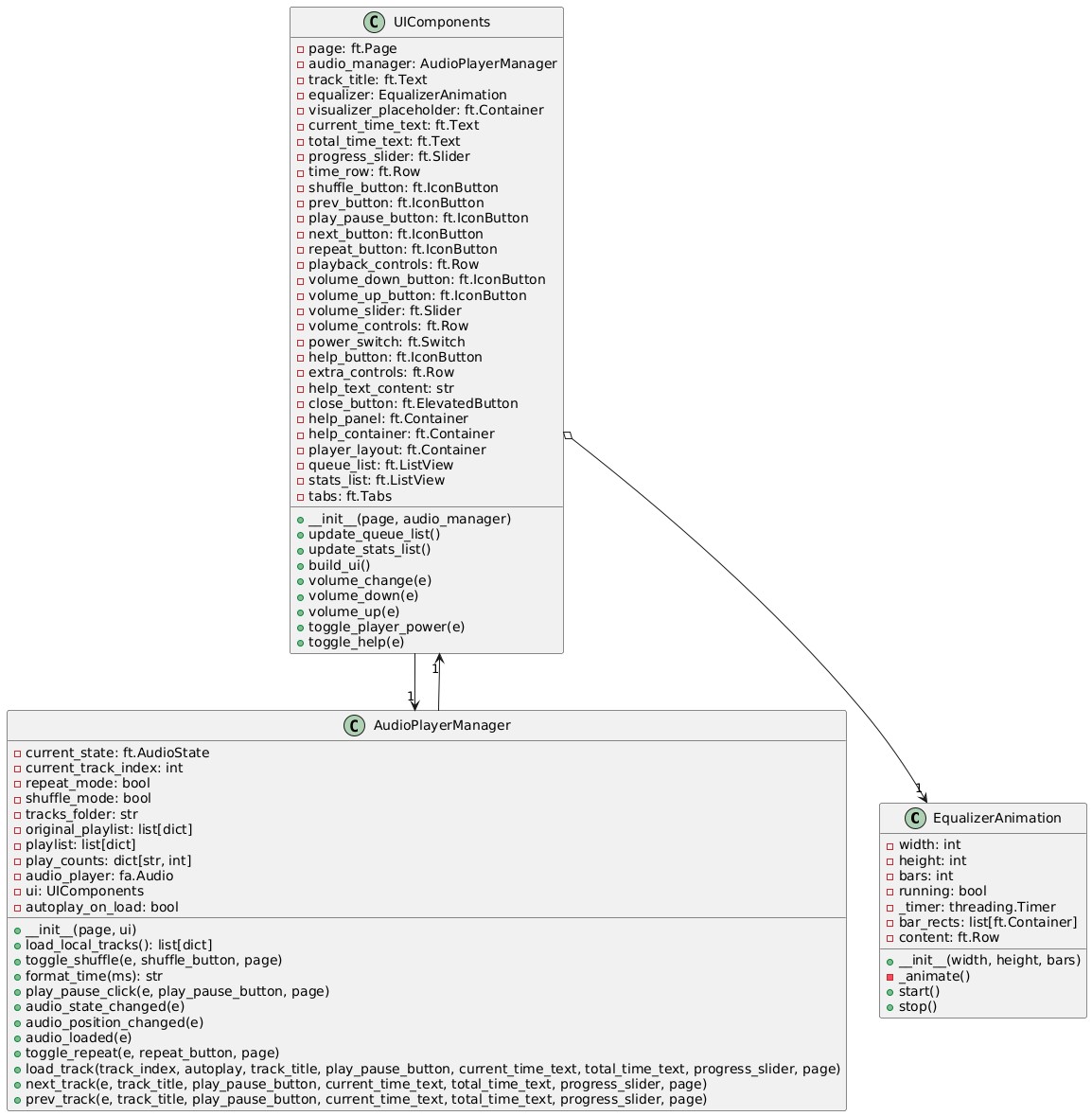


Рис. 2 «Диаграмма классов»

**Глава 2 «Тестирование аудиоплеера на Python»**

**2.1. Модуль управления воспроизведением (AudioPlayerManager)**

Отвечает за логику воспроизведения аудио и управление плейлистом. Он реализован в виде класса AudioPlayerManager в файле audio\_player.py.

**Основные функции**

* **Управление воспроизведением**: Методы play\_pause\_click, next\_track, prev\_track обеспечивают запуск, паузу и переключение треков. Воспроизведение осуществляется через объект ft.Audio, который поддерживает базовые операции (пуск, пауза, установка позиции).
* **Проверка доступности треков**: Метод check\_url использует библиотеку requests для проверки доступности аудиофайлов по URL, предотвращая ошибки при загрузке.
* **Обработка событий**: Методы audio\_state\_changed, audio\_position\_changed и audio\_loaded обрабатывают изменения состояния плеера, позиции воспроизведения и загрузку трека, обновляя интерфейс соответственно.
* **Форматирование времени**: Метод format\_time преобразует миллисекунды в читаемый формат для отображения текущего времени и длительности трека.

**Реализация**

Класс инициализирует плейлист с предопределенными URL-адресами аудиофайлов и создает объект ft.Audio с начальными настройками (громкость 0.5, автозапуск выключен). Методы взаимодействуют с интерфейсом через объект UIComponents, передавая данные для обновления текстовых полей, ползунков и кнопок.

**2.2. Модуль графического интерфейса (UIComponents)**

Отвечает за создание и управление графическим интерфейсом аудиоплеера. Он реализован в виде класса UIComponents в файле audio\_player.py.

**Основные функции**

* **Создание интерфейса**: Метод build\_ui инициализирует все элементы интерфейса, включая текстовые поля, кнопки, ползунки и контейнеры.
* **Обработка действий пользователя**: Методы volume\_change, seek\_position\_change, toggle\_player\_power, toggle\_help, volume\_down, volume\_up реагируют на действия пользователя, такие как изменение громкости, перемотка трека, включение/выключение плеера и вызов справки.
* **Визуализация состояния**: Элементы интерфейса (например, track\_title, progress\_slider, current\_time\_text) обновляются в реальном времени в зависимости от состояния плеера.

**Реализация**

Класс UIComponents создает фиксированное окно (340x480 пикселей) с темной темой. Основные элементы:

* **Текстовые поля**: track\_title для названия трека, current\_time\_text и total\_time\_text для времени воспроизведения.
* **Кнопки управления**: play\_pause\_button, prev\_button, next\_button для воспроизведения и переключения треков.
* **Ползунки**: progress\_slider для навигации по треку и volume\_slider для регулировки громкости.
* **Дополнительные элементы**: power\_switch для включения/выключения плеера и help\_button для отображения справки.
* **Контейнеры**: visualizer\_placeholder (заглушка для визуализации), help\_container для панели справки.

Интерфейс организован в виде вертикальной колонки (ft.Column) с горизонтальными рядами (ft.Row) для компактного размещения элементов.

**2.3. Правила построения интерфейсов**

1. **Простота и интуитивность**:
   * Интерфейс минималистичен, содержит только необходимые элементы управления (воспроизведение, переключение треков, громкость).
   * Иконки для кнопок (play\_circle\_filled\_rounded, skip\_next\_rounded) выбраны стандартные и узнаваемые, что упрощает взаимодействие.
   * Текстовые поля (track\_title, current\_time\_text) предоставляют четкую информацию о состоянии плеера.
2. **Консистентность стиля**:
   * Использована темная тема (ft.ThemeMode.DARK) для снижения нагрузки на глаза и современного вида.
   * Цветовая схема основана на оттенках фиолетового (ft.Colors.PURPLE\_ACCENT\_100/400) для кнопок и ползунков, обеспечивая визуальное единство.
   * Все элементы имеют закругленные углы (border\_radius) и отступы (padding, margin) для эстетичного вида.
3. **Доступность**:
   * Переключатель питания (power\_switch) отключает элементы управления, предотвращая нежелательные действия, и меняет их цвет на серый для визуальной индикации.
   * Панель справки (help\_container) предоставляет краткие инструкции по управлению плеером, доступные по клику на кнопку справки.
4. **Модульность и масштабируемость**:
   * Интерфейс организован в виде отдельных компонентов (ft.Text, ft.Slider, ft.IconButton), что упрощает модификацию и добавление новых функций.
   * Контейнеры (ft.Container, ft.Column, ft.Row) используются для структурирования элементов, обеспечивая гибкость компоновки.

**2.4. Логика взаимодействия с интерфейсом**

* **AudioPlayerManager → UIComponents**: Менеджер передает данные о состоянии плеера (позиция трека, название, длительность) для обновления текстовых полей и ползунков. Например, метод audio\_position\_changed обновляет progress\_slider и current\_time\_text.
* **UIComponents → AudioPlayerManager**: Действия пользователя (нажатие кнопок, изменение ползунков) вызывают методы менеджера. Например, клик по play\_pause\_button вызывает play\_pause\_click, который управляет воспроизведением.

**Реализация взаимодействия**

* Инициализация: В функции main создается экземпляр AudioPlayerManager, затем UIComponents, после чего менеджер получает ссылку на интерфейс для обновлений.
* Обновление интерфейса: Все изменения (состояние плеера, позиция трека, громкость) сопровождаются вызовом page.update() для синхронизации интерфейса.
* Обработка ошибок: Если URL трека недоступен, track\_title отображает сообщение об ошибке, а воспроизведение блокируется.

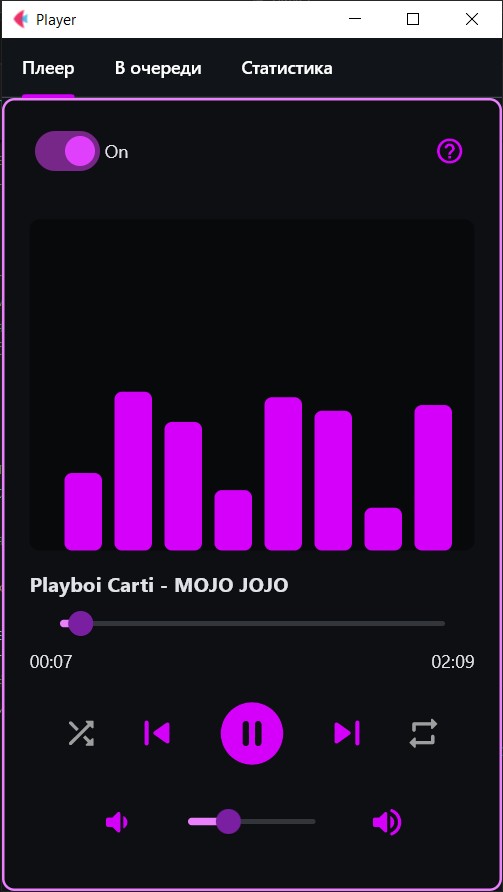


Рис. 3 «Интерфейс плеера»

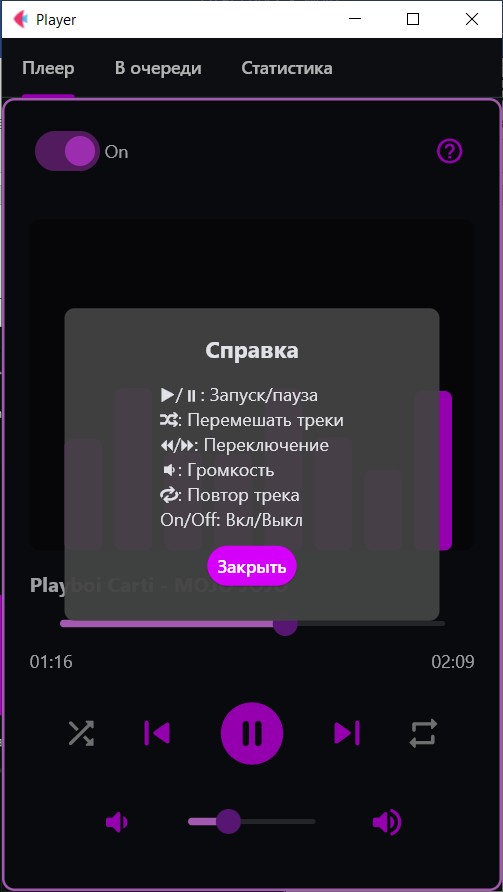


Рис. 4 «Справка»

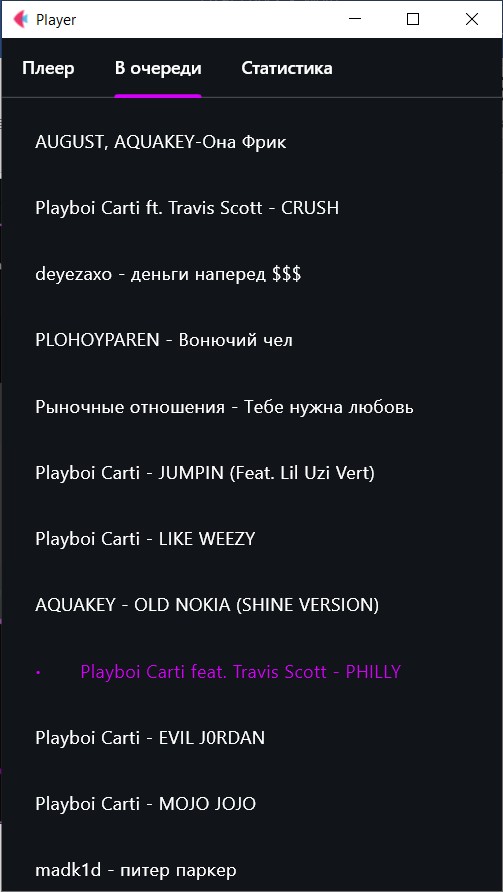


Рис. 5 «Вкладка «В очереди»»

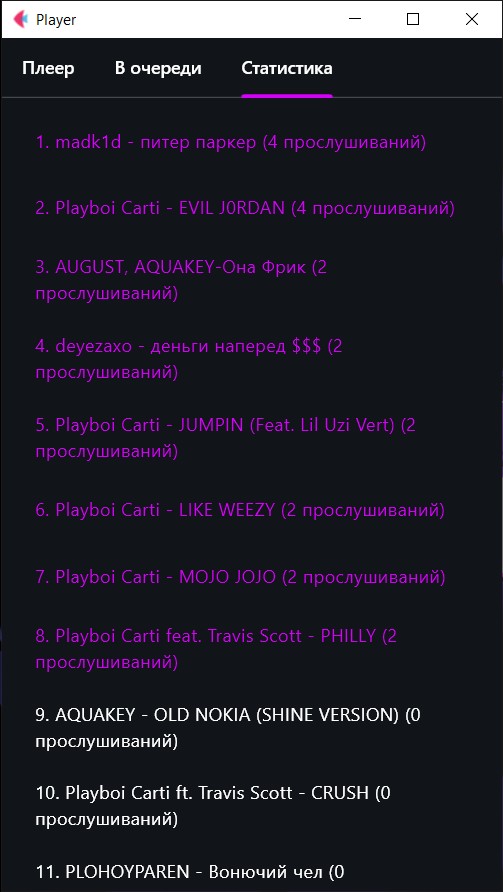


Рис. 6 «Вкладка «Статистика»»

**Глава 3 «Тестирование аудиоплеера на Python»**

Тестировался плеер, чтобы он работал без ошибок и был удобным. Проверялось два типа тестов: модульные (для отдельных функций) и нефункциональные (для скорости).

**3.1 Модульное тестирование**

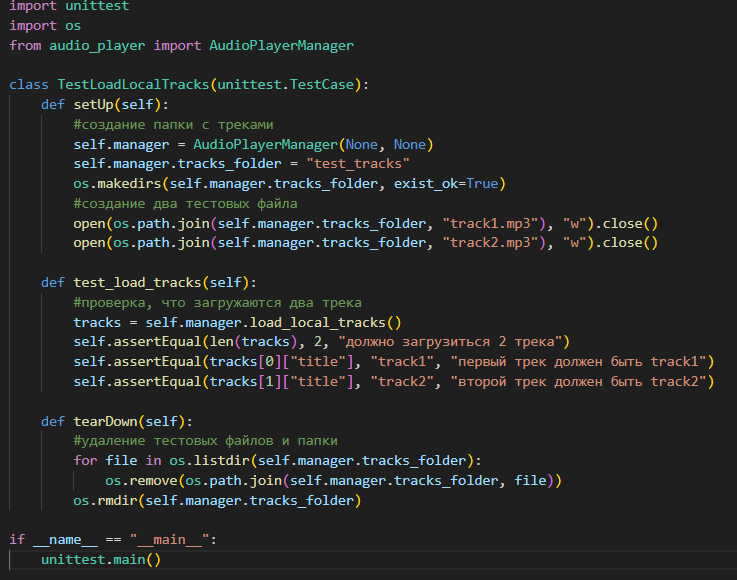
Тестировался класс AudioPlayerManager, который управляет треками. Использоваласб библиотека unittest. Проверялось две функции:

* load\_local\_tracks – загружает треки из папки.
* toggle\_shuffle – перемешивает плейлист, но оставляет текущий трек на месте.

Тест 1: Загрузка треков

Проверялась функция load\_local\_tracks находит все mp3-файлы в папке. Создалось два тестовых файла (track1.mp3, track2.mp3) и проверила, что плеер их видит.

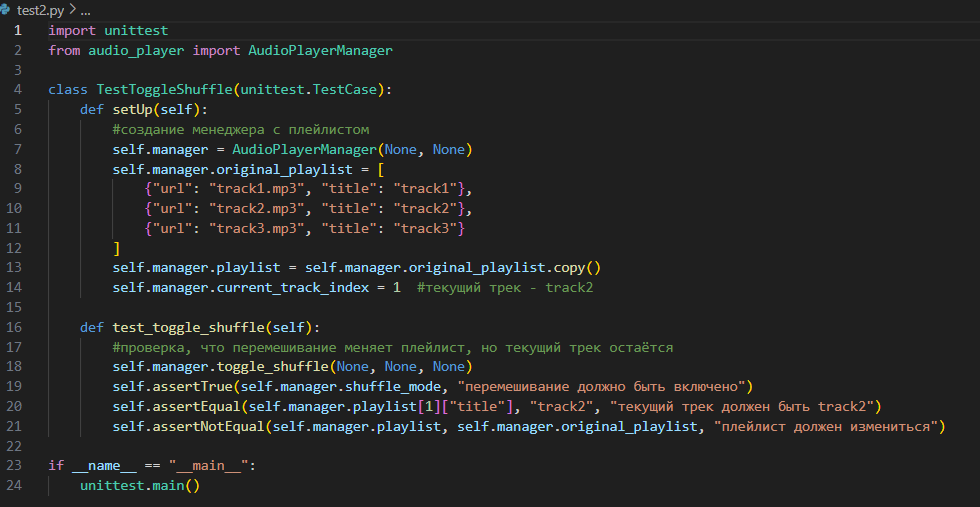
Листинг 1. Модульный тест №1



Тест 2: Перемешивание плейлиста

Проверялось toggle\_shuffle меняет порядок треков, но текущий трек остаётся на своей позиции. Сделала плейлист из трёх треков и проверила, что после включения перемешивания он меняется, но текущий трек не сдвигается.

Листинг 2. Модульный тест №2



* 1. **Нефункциональное тестирование**

Смотрела как быстро переключаются треки и не зависает ли анимация эквалайзера, если долго слушать музыку.

Тест 3: Скорость переключения треков

Переключалось 20 треков подряд и смотрела, чтобы каждое переключение занимало меньше 0.5 секунды.

Тест 4: Анимация эквалайзера

Включила треки на 10 минут и смотрела, чтобы анимация эквалайзера не тормозила.

Таблица 2: «Результаты тестирования»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Вид тестирования** | **Описание теста** | **Входные данные** | **Ожидаемый результат** | **Фактический результат** | **Статус** | **Кто проводил** |
| 1 | Модульное тестирование | Функция load\_local\_tracks | Папка с файлами: track1.mp3, track2.mp3 | Плеер загружает два трека | Плеер загрузил два трека | Пройдено | Ласица А.А. |
| 2 | Модульное тестирование | Функция toggle\_shuffle | Плейлист из трёх треков, включено перемешивание | Треки перемешаны, текущий трек на месте | Треки перемешаны, текущий трек на месте | Пройдено | Никитинская Т.Г. |
| 3 | Нефункциональное тестирование | Скорость переключения треков | Переключение 20 треков | Переключение за < 0.5 сек | Скорость очень маленкая и незаметная | Пройдено | Ласица А.А. |
| 4 | Нефункциональное тестирование | Анимация эквалайзера | Воспроизведение 10 минут | Анимация не зависает | Анимация не зависает | Пройдено | Никитинская Т.Г. |

Тесты прошли успешно. Плеер работает нормально, ничего не ломается.

**Выводы**

Разработка аудиоплеера на языке программирования Python с использованием библиотек Flet и flet\_audio стала важным этапом в освоении технологий создания мультимедийных приложений. Этот проект позволил не только реализовать функциональное приложение для воспроизведения музыки, но и углубить знания в области программирования, проектирования пользовательских интерфейсов и обработки мультимедийных данных. Созданный аудиоплеер сочетает в себе современный и интуитивно понятный интерфейс с широким набором функций, таких как воспроизведение, пауза, переключение треков, регулировка громкости, поддержка режимов случайного воспроизведения и повтора, а также визуализация аудио в виде анимированного эквалайзера. Эти возможности делают приложение удобным и привлекательным для пользователей, а его модульная структура открывает перспективы для дальнейшего расширения функционала.

В ходе работы над проектом были успешно применены навыки, полученные за год изучения Python, включая объектно-ориентированное программирование, работу с библиотеками, управление потоками и обработку событий. Использование библиотеки Flet позволило создать кроссплатформенный графический интерфейс, который отличается эстетичным дизайном. Реализация таких компонентов, как EqualizerAnimation для визуализации, AudioPlayerManager для управления логикой воспроизведения и UIComponents для построения интерфейса, демонстрирует глубокое понимание принципов модульного программирования и разделения ответственности в коде. В заключение, разработка аудиоплеера стала не только практически значимым проектом, но и ценным опытом, который укрепил мои навыки программирования, проектирования интерфейсов и работы с мультимедиа. Полученные знания и умения, включая работу с библиотеками Flet и flet\_audio, создают прочную основу для дальнейших разработок в области мультимедийных приложений. Аудиоплеер стал важным шагом в моем профессиональном развитии, открывая новые возможности для исследования технологий и создания инновационных решений в области программирования.

**Список использованных источников**

1. Документация библиотеки Flet [Электронный ресурс] URL:<https://flet.dev/docs/> (дата обращения 24.07.2025)
2. Simple Music Player in Python - Flet Tutorial [Электронный ресурс] URL: <https://rutube.ru/video/cd0679923ab63e90103c0909d39752ab/>(дата обращения 24.07.2025)
3. Библиотека Python Flet для UI дизайна приложений [Электронный ресурс] URL:<https://itproger.com/course/python-gui/2>(дата обращения 24.07.2025)
4. Гайд по Flet: приступая к работе. Создание приложений на Python [Электронный ресурс] URL:

<https://www.ixbt.com/live/sw/gayd-po-flet-pristua.html>

(дата обращения 24.07.2025)

1. Колесинский, М. В. Flet: разработка кроссплатформенных приложений на Python / М. В. Колесинский // Цифровые, компьютерные и информационные технологии в науке и образовании: Сборник статей II Межрегиональной научно-практической конференции с международным участием, Брянск, 14–15 ноября 2024 года. – Брянск: Брянский государственный университет им. акад. И.Г. Петровского, 2025. – С. 306–308.
2. Куликов, Д. А. Программирование на Python с использованием библиотеки Flet / Д. А. Куликов, О. В. Осипов // Информационные технологии. Радиоэлектроника. Телекоммуникации (ITRT- 2024): Материалы XI Международной заочной научно-технической конференции, Тольятти, 29 ноября 2024 года. – Тольятти: Издательскополиграфический центр ПВГУС, 2025. – С. 26–31.