МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
СӘТБАЕВ УНИВЕРСИТЕТІ
Институт кибернетики и информационных технологий
Кафедра "Программная инженерия"

Айтанова Ақерке Айдосқызы

Музыкальный плейлист основанный на эмоциях

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

Специальность 5В070400 – Вычислительная техника и программное обеспечение

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий
Кафедра "Программная инженерия"
5В070400 – Вычислительная техника и программное обеспечение

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
Доктор Ph. D.
_____М.Тұрдалыұлы
"06" Июня 2021 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
На тему: "Музыкальный плейлист основанный на эмоциях"
по специальности 5В070400 – Вычислительная техника и программное обеспечение

Выполнила Айтанова А. А.

Научный руководитель
Магистр технических наук,
Сениор-лектор
_____А.Куникеев
"06" Июня 2021 г.

Алматы 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

5В070400 – Вычислительная техника и программное обеспечение

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
Доктор Ph. D.
_____ М.Тұрдалыұлы
"06" июня 2021 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Айтановой Акерке Айдоскызы*

Тема: Музыкальный плейлист основанный на эмоциях

Исходные данные к дипломному проекту: *Описание необходимых функций проекта.*

Перечень подлежащих разработке в дипломном проекте вопросов:
а) реализация музыкального мобильного приложения с распознаванием лица;

б) проектирование и разработка пользовательского интрефейса;

в) разработка, отладка и тестирование мобильного приложения;

Перечень графического материала (с точным указанием обязательных чертежей): *представлены 14 слайдов презентации.*

ГРАФИК
подготовки дипломного
проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю и консультантам	Примечание
1. Анализ предметной области, разработка технического задания	21.03.2021	Выполнено
2. Сбор данных для моделей машинного обучения	21.02.2021	Выполнено
3. Разработка моделей	10.03.2021	Выполнено
4. Разработка бэкенд-сервера	30.04.2021	Выполнено
5. Разработка мобильного приложения	15.05.2021	Выполнено
6. Написание пояснительной записки к дипломному проекту	28.05.2021	Выполнено

Подписи
консультантов и нормоконтролера на законченный дипломный
проектс указанием относящихся к ним разделов проекта

Наименования разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Программное обеспечение	Рамазан Акерке, ассистент	31.05.2021	
Нормоконтролер	Марғұлан Қабылжан, лектор	31.05.2021	

Научный руководитель _____ Куникеев А.

Задание принял к исполнению обучающийся _____ Айтанова А.

Дата "06" июня 2021г.

АННОТАЦИЯ

Данный дипломный проект посвящен разработке инновационного мобильного приложения для ОС Android. Emotion-based music player умеет распознавать эмоции пользователя, при помощи которых создается соответствующий список музыкальных композиций.

Приложение направлено на ускорение и автоматизирование процесса подбора музыкальных плейлистов при помощи рекомендательной системы на основе распознавания эмоций, отражающихся на лице пользователя. Пользователь после аутентификации в мобильном приложении, наводит камеру на уровень лица, после чего модель распознавания эмоций считывает текущее настроение пользователя и выдает готовый плейлист с правильно подобранными музыкальными композициями.

В этой работе рассматривается решение автоматизации подбора музыкального плейлиста пользователя. Для решения был создан проект, состоящий из трех частей: мобильное приложение, построенное при помощи Android Studio и языка программирования Kotlin, модель распознавания эмоций, реализованная на языке программирования Python, и бэкенд сервер при помощи фреймворка Flask на языке программирования Python.

АНДАТПА

Бұл дипломдық жоба Android ОЖ арналған инновациялық мобильді қосымшасын әзірлеуге арналған. Emotion-based music player пайдаланушының эмоцияларын тани отыра, олардың көмегімен музыкалық композициялардың тиісті тізімін жасақтайды.

Қосымша пайдаланушының бетінде көрінетін эмоцияларды тануға негізделген ұсыныс жүйесін қолдана отырып, музыкалық ойнату тізімдерін құрастыру процесін жеделдетуге және автоматтандыруға бағытталған. Пайдаланушы мобильді қосымшада аутентификациядан кейін камераны бет деңгейіне бағыттайды, содан кейін эмоцияны тану моделі пайдаланушының қазіргі көңіл-күйін оқиды және сәйкес таңдалған музыкалық композициялармен дайын ойнату тізімін береді.

Бұл жұмыста біз пайдаланушының музыкалық ойнату тізімін таңдауды автоматтандыру шешімін қарастырамыз. Берілген мәселені шешу үшін үш бөліктен тұратын жоба құрылды: Android Studio және Kotlin бағдарламалау тілінің көмегімен жасалған мобильді қосымша, Python бағдарламалау тілінде іске асырылған эмоцияны тану моделі және Python бағдарламалау тілінде Flask жақтауының көмегімен бэкенд сервер.

ANNOTATION

This graduation project is dedicated to the development of mobile application for the Android OS. Emotion-based music player is able to recognize the emotions of the user, through which a corresponding list of musical compositions is created.

The application is aimed at speeding up and automating the process of selecting music playlists by recommendation system based on the recognition of emotions reflected on the user's face. After authentication in the mobile app, the user points the camera at the face level, after which the emotion recognition model detects the current mood of the user and outputs a ready-made playlist with correctly selected musical compositions.

In this article, we consider a solution for automating the selection of a user's music playlist. For the solution, we created a project consisting of three parts: a mobile application created by using Android Studio and the Kotlin programming language, an emotion recognition model implemented in the Python programming language and a backend server using the Flask framework in the Python programming language.

СОДЕРЖАНИЕ

Введение

Недавние исследования подтверждают, что люди отзываются и реагируют на музыку, и что музыка оказывает сильное влияние на мозговую деятельность человека. Среднестатистический казахстанец слушает до четырех часов музыки каждый день. Люди склонны слушать музыку в зависимости от настроения и интересов. Этот проект направлен на создание приложения, предлагающего песни для пользователей в зависимости от настроения, распознавая выражения лица.

Выражение лица - это форма невербального общения. Компьютерное зрение - это междисциплинарная область, которая помогает передать машинам понимание цифровых изображений или видео на высоком уровне. В этой системе компьютерного зрения компоненты используются для определения эмоций пользователя через мимику. Как только эмоция распознается, система предлагает плейлист для этой эмоции, позволяющий пользователю сэкономить время при выборе и воспроизведении песни вручную. Музыкальный проигрыватель на основе эмоций также отслеживает такие данные пользователя, как количество воспроизведений каждой песни, сортировка песен по категориям и уровню интереса, а также реорганизует каждый раз плейлист. Система также уведомляет пользователя о песнях, которые никогда не воспроизводятся, поэтому их можно удалить или изменить.

1 Исследовательский раздел

1.1 Цель разработки

Необходимо разработать:

1) Мобильное музыкальное приложение с возможностью:
   - аутентификации через Spotify, Google и т.д.;
   - автоматического создания музыкального плейлиста;
   - отправки снимка на бэкенд сервер;
   - добавления или создания снимка (изображение лица);
2) Бэкенд сервер с возможностью:
   - отправки и получения данных;
   - автоматического создания музыкального плейлиста;
3) ИИ для распознавания лица (эмоций), который включает в себя:
   - модель CNN;
   - Haar cascade для обнаружения лица;
4) Модель для класстеризации музыкальных композиций:
   - реализация K-means алгоритма;
   - реализация подключени к Spotify;

Автоматический выбор песен и их организация на основе настроения пользователя дает пользователю лучший опыт использования приложения. Этого можно добиться, если система реагирует на эмоции пользователя, экономя время, которое было бы потрачено на ввод информации вручную.

В этом приложении используется обученная модель распознавания лица на языке программирования Python. Система включает алгоритм CNN, который считывает изображения и делит эмоции на семь классов. Алгоритм рекомендательной системы класстеризует песни на классы эмоций. После чего, согласно полученной эмоции от алгоритма CNN, рекомендационная система создает плейлист.

1.2 Термины и сокращения

В таблице 1 сформулированы все термины и сокращения, которые используются в предметной области разрабатываемого проекта, а также специфические термины, связанные с программной реализацией проекта и используемыми технологиями при разработке.

Таблица 1 - Термины, сокращения, и их определения

Сокращение или термин	Определение
Android	Операционная система Google с открытым исходным кодом для мобильных устройств на базе Linux.
Фреймворк	Каркас, который состоит из множества различных библиотек, которые облегчают разработку программного продукта или сайта
Spotify	Интернет-сервис потокового аудио, позволяющий легально прослушивать музыкальные композиции, аудиокниги и подкасты, не скачивая их на устройство.
API	(сокр. от англ. Application programming interface) Интерфейс прикладного программирования —это вычислительный интерфейс к программному компоненту или системе, который определяет, как другие компоненты или системы могут его использовать.——
Git	Это бесплатная система управления версиями с открытым исходным кодом, предназначенная для быстройи эффективной работы с небольшимии очень крупными проектами.
CNN	Свёрточная нейронная сеть (англ. convolutional neural network, CNN) — специальная архитектура искусственных нейронных сетей, нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения (англ. deep learning).
UI	(сокр. от англ. User interface) Пользовательский интерфейс.
JVM	Виртуальная машина Java
ARM	(от англ. Advanced RISC Machine — усовершенствованная RISC-машина) — система команд и семейство описаний и готовых топологий 32-битных и 64-

	битных микропроцессорных/ микроконтроллерных ядер, разрабатываемых компанией ARM Limited.
UML	Единый язык моделирования

1.3 Предметная область

1.3.1 Android

Android - это операционная система Google с открытым исходным кодом для мобильных устройств на базе Linux. По данным января 2021 года ОС Android является лидирующей мобильной операционной системой во всем мире, контролируя рынок мобильных ОС с долей 71,93%. Google Android и Apple iOS совместно владеют более 99% мирового рынка.

Операционная система Android поддерживает большое количество приложений на смартфонах, которые особенно удобны для пользователей. Технические средства, поддерживающее программное обеспечение Android, основаны на платформе архитектуры ARM. Благодаря тому, что, Android ОС – ОС с открытым исходным кодом, она бесплатна и ее может использовать каждый.

Android - это операционная система, представляющая собой стек программных компонентов, который разделен на пять разделов и четыре основных уровня:

- Ядро Linux;
- Библиотеки;
- Среда выполнения Android;
- Фреймворк приложения;
- Приложения;

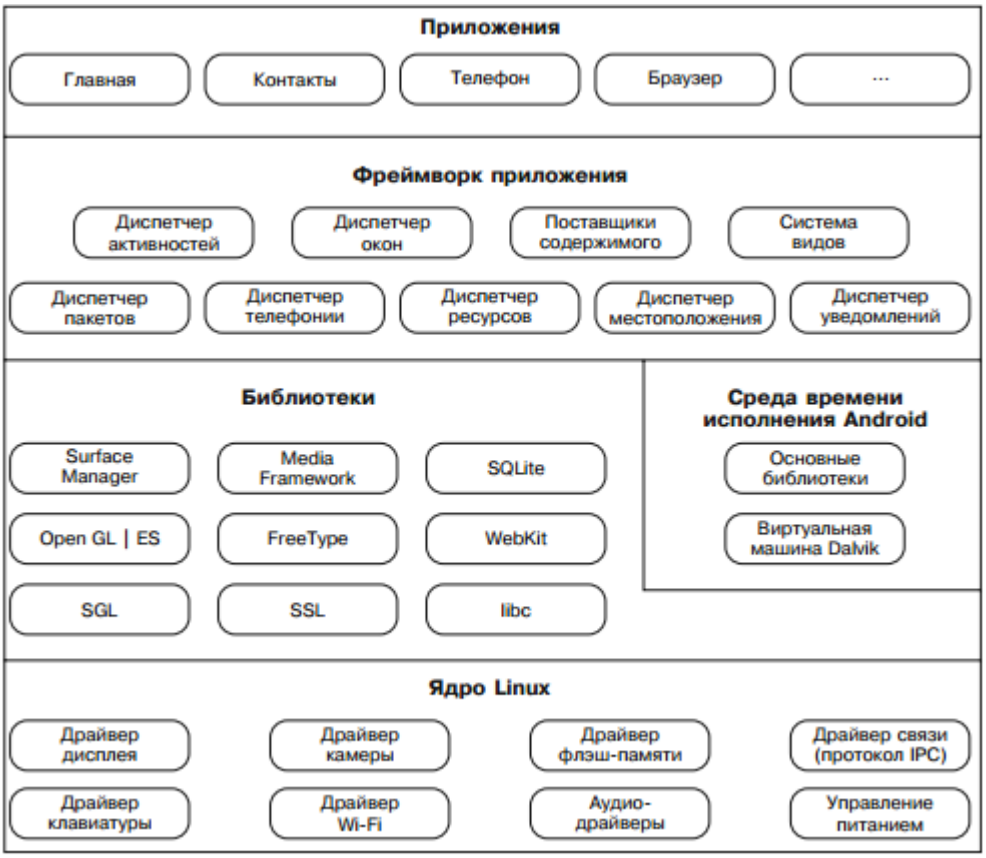


Рисунок 1.3.2 – Архитектура ОС Android

1.3.2 Искусственный интеллект

Искусственный интеллект - это способность машины (цифрового компьютера или управляемого компьютером робота) демонстрировать человеческие способности, такие как мышление, обучение, планирование и творчество. ИИ позволяет техническим системам воспринимать окружающую среду, справляться с тем, что они воспринимают, решать проблемы и действовать для достижения конкретной цели. Компьютер получает данные - уже подготовленные или собранные с помощью своих собственных датчиков, таких как камера, - обрабатывает их и реагирует.

Системы искусственного интеллекта способны в определенной степени адаптировать свое поведение, анализируя последствия предыдущих действий и работая автономно.

С 1940-х годах, когда только разработали цифровой компьютер было доказано, что компьютеры могут быть запрограммированы для выполнения очень сложных задач - например, обнаружения доказательств математических теорем или игры в шахматы - с большим мастерством. Однако, несмотря на продолжающийся прогресс в скорости обработки компьютерных данных и емкости памяти, еще нет программ, которые могли бы соответствовать

человеческой гибкости в более широких областях или в задачах, требующих больших повседневных знаний. С другой стороны, некоторые программы достигли уровня производительности человеческих экспертов и профессионалов при выполнении определенных конкретных задач, так что искусственный интеллект в этом ограниченном смысле можно найти в таких разнообразных приложениях, как медицинская диагностика, компьютерные поисковые системы и распознавание голоса или почерка.

1.3.3 Computer vision

Computer vision (CV) является большой сферой компьютерных наук, которая основывается на воссоздании непростого механизма работы человеческого зрения и делает компьютерам возможным определять и обрабатывать объекты на изображениях и видео так же, как это делает человеческий мозг.

Концепция компьютерного зрения основана на обучении компьютеров обработке изображения на уровне пикселей и его пониманию. Технически машины пытаются извлекать визуальную информацию, обрабатывать ее и интерпретировать результаты с помощью специальных программных алгоритмов. Вот несколько общих задач, для которых могут использоваться системы компьютерного зрения:

Классификация объектов. Система анализирует визуальный контент и классифицирует объект на фото/видео по определенной категории. Например, система может найти собаку среди всех объектов на изображении.

Идентификация объекта. Система анализирует визуальный контент и идентифицирует конкретный объект на фото/видео. Например, система может найти конкретную собаку среди собак на изображении.

Отслеживание объектов. Система обрабатывает видео, находит объект (или объекты), соответствующие критериям поиска, и отслеживает его движение.

1.3.4 Распознавание лица

Технология распознавания лиц используется для сопоставления фотографий лиц людей с их личностями. Эта технология интегрирована в основные продукты, которые мы используем каждый день. Например, Facebook использует компьютерное зрение для идентификации людей на фотографиях.

Распознавание лиц является важнейшей технологией для биометрической

аутентификации. Многие мобильные устройства, доступные сегодня на рынке, позволяют пользователям разблокировать устройства, показывая свои лица. Для распознавания лиц используется фронтальная камера; мобильные устройства обрабатывают это изображение и на основе анализа могут определить, авторизован ли человек, который держит устройство, на этом устройстве. Великолепие этой технологии в том, что она работает очень быстро. Системы лицевых технологий могут быть разными, но в целом они работают следующим образом:

Распознавание лиц: камера обнаруживает и находит изображение лица в одиночестве или в толпе. На изображении может быть изображен человек, смотрящий прямо перед собой или в профиль.

Анализ лица: затем снимается и анализируется изображение лица. Большинство технологий распознавания лиц полагаются на 2D-изображения, а не на 3D-изображения, потому что они могут более удобно сопоставить 2D-изображение с общедоступными фотографиями или фотографиями в базе данных. Программа считывает геометрию лица пользователя. Ключевые факторы включают расстояние между глазами, глубину глазниц, расстояние от лба до подбородка, форму скул и контур губ, ушей и подбородка. Цель состоит в том, чтобы определить черты лица, по которым лицо будет отличаться.

Преобразование снимка в данные: процесс захвата лица преобразует аналоговую информацию (лицо) в набор цифровых данных на основе черт лица человека. Иначе говоря, анализ лица преобразуется в математическую формулу и цифровой код называется «отпечатком лица». Никому не секрет, что отпечатки пальцев у каждого человека уникальны, точно так же можно сказать, что у каждого индивидуума есть свой собственный отпечаток лица.

2 Технологический раздел

2.1 Среда разработки

Интегрированная среда разработки (IDE) является программной платформой, которая предоставляет программистам целый набор инструментов для разработки. Среда была создана для работы с конкретными платформами приложений и исключения преград, относящихся к жизненному циклу разработки ПО. Интегрированные среды разработки используются для создания программного обеспечения, так как они намного ускоряют процесс разработки, предоставляя один инструмент со всеми функциями и устраняя необходимость в интеграции. IDE предназначены для написания кода на определенную платформу и имеют интегрированные функции, которые знают, как работает платформа и как использовать функции платформы путем компиляции кода, отладки кода или интеллектуального автоматического завершения кода.

2.2 Android Studio

Android Studio - это официальная интегрированная среда разработки (IDE) для разработки приложений Android, основанная на IntelliJ IDEA, интегрированной среде разработки Java для программного обеспечения, и включает в себя инструменты для редактирования кода и разработки.

2.2.1 Kotlin

Kotlin - это универсальный, бесплатный, статически типизированный «прагматичный» язык программирования с открытым исходным кодом, изначально разработанный для JVM и Android, который сочетает в себе функции объектно-ориентированного и функционального программирования. Он ориентирован на совместимость, безопасность, ясность и поддержку инструментов.

Kotlin возник в JetBrains, компании, стоящей за IntelliJ IDEA, в 2010 году и работает с открытым исходным кодом с 2012 года. В настоящее время команда Kotlin насчитывает более 90 постоянных членов из JetBrains, а проект Kotlin на GitHub насчитывает более 300 участников. JetBrains использует Kotlin во многих своих продуктах, включая флагманскую IntelliJ IDEA.

2.3 Flask Framework

Flask - это фреймворк Python, основанный на Werkzeug, Jinja2 и вдохновленный фреймворком Sinatra Ruby, доступный по лицензии BSD. Он был разработан в росоо Армином Ронахером. Хотя Flask довольно молод по сравнению с большинством фреймворков Python, он имеет большие перспективы и уже завоевал популярность среди разработчиков Python.

2.3.1 Python

Python – высокоуровневый, интерпретируемый и динамически типизированный язык программирования, который обеспечивает легкую отладку и ускоряет разработку прототипов приложений. Python был разработан голландским программистом Гвидо Ван Россумом, который был провозглашен «Доброжелательным диктатором на всю жизнь», пока не оставил этот пост в 2018 году.

Идея разработки языка основана на принципе DRY (Don’t Repeat Yourself) и удобочитаемости. Библиотека Python настолько огромна, что вы можете найти в ней большинство необходимых функций (инструменты веб-сервисов, Интернет, строковые операции и т. д.). На случай, если понадобится что-то еще, есть индекс пакетов Python с более чем 200 тысячами пакетов, которые можно легко импортировать с помощью диспетчера пакетов Python.

2.4 Convolutional Neural Networks

Сверточная нейронная сеть (ConvNet / CNN) - это алгоритм глубокого обучения, который может получать входное изображение, распределять значимость (обучаемые веса и смещения) различным аспектам / объектам на изображении и иметь возможность отличать один от другого. Вместо предварительной обработки данных для получения таких функций, как текстуры и формы, CNN принимает в качестве входных данных только необработанные пиксельные данные изображения и «учится» извлекать эти функции и, в конечном итоге, делает вывод, какой объект они составляют.

Для начала CNN получает входную карту характеристик: трехмерную матрицу, в которой размер первых двух измерений соответствует длине и ширине изображений в пикселях. Размер третьего измерения - 3 (соответствует трем каналам цветного изображения: красному, зеленому и синему). CNN состоит из набора модулей, каждый из которых выполняет три операции.

1. Свертка

Свертка извлекает фрагменты входной карты функций и применяет к ним фильтры для вычисления новых функций, создавая выходную карту функций или свернутую функцию (которая может иметь другой размер и глубину, чем входная карта функций). Свертки определяются двумя параметрами:

- Размер извлекаемых фрагментов (обычно 3x3 или 5x5 пикселей).
- Глубина выходной карты функций, которая соответствует количеству применяемых фильтров.

Во время обучения CNN «изучает» оптимальные значения для матриц фильтров, которые позволяют извлекать значимые особенности (текстуры, края, формы) из входной карты признаков. По мере того, как количество фильтров (глубина выходной карты признаков), применяемых к входу, увеличивается, увеличивается и количество функций, которые CNN может извлечь. Однако компромисс заключается в том, что фильтры составляют большую часть ресурсов, расходуемых CNN, поэтому время обучения также увеличивается по мере добавления дополнительных фильтров. Кроме того, каждый фильтр, добавленный в сеть, обеспечивает меньшую инкрементную ценность, чем предыдущий, поэтому инженеры стремятся создавать сети, в которых используется минимальное количество фильтров, необходимых для извлечения функций, необходимых для точной классификации изображений.

2. ReLU

После каждой операции свертки CNN применяет преобразование Rectified Linear Unit (ReLU) к свернутому элементу, чтобы внести нелинейность в модель. Функция ReLU, возвращает x для всех значений $x > 0$ и возвращает 0 для всех значений $x \leq 0$.

3. Объединение

После ReLU идет этап объединения, на котором CNN выполняет субдискретизацию свернутой функции (для экономии времени обработки), уменьшая количество измерений карты функций, сохраняя при этом наиболее важную информацию о функциях. Общий алгоритм, используемый для этого процесса, называется максимальным пулом.

Максимальный пул работает аналогично свертке. Мы перемещаемся по карте функций и извлекаем фрагменты заданного размера. Для каждой плитки максимальное значение выводится на новую карту функций, а все остальные значения отбрасываются. Максимальные операции объединения принимают два параметра:

- Размер фильтра максимального объединения (обычно 2x2 пикселя)
- Шаг: расстояние в пикселях, разделяющее каждую извлеченную плитку.

В отличие от свертки, когда фильтры перемещаются по карте признаков пиксель за пикселем, при максимальном объединении шаг определяет места, где извлекается каждый фрагмент. Для фильтра 2x2 шаг 2 указывает, что операция максимального объединения будет извлекать все неперекрывающиеся фрагменты 2x2 из карты функций.

4. Полностью связанные слои

В конце сверточной нейронной сети находится один или несколько полностью связанных слоев (когда два слоя «полностью соединены», каждый узел первого уровня подключается к каждому узлу второго уровня). Их задача - выполнить классификацию на основе признаков, извлеченных свертками. Как правило, последний полностью связанный слой содержит функцию активации softmax, которая выводит значение вероятности от 0 до 1 для каждой из классификационных меток, которую модель пытается предсказать.

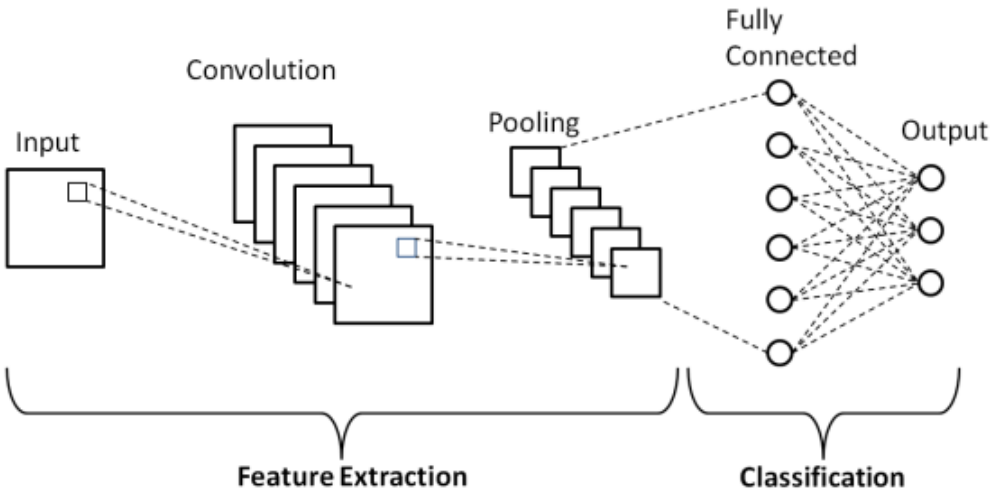


Рисунок 2.4 – Архитектура CNN

2.5 Spotify

Spotify - это служба цифровой музыки, подкастов и видео, которая дает вам доступ к миллионам песен и другому контенту от создателей со всего мира. Основные функции, такие как воспроизведение музыки, совершенно бесплатны, но вы также можете перейти на Spotify Premium.

2.5.1 Spotify Web API

Веб-API - это интерфейс прикладного программирования, который может связывать вместе несколько типов программного обеспечения. В этом случае он позволяет веб-разработчикам использовать обширные музыкальные данные Spotify. Spotify имеет огромную музыкальную базу, состоящую из более чем 50 миллионов песен; их общедоступный API позволяет вызывать данные об исполнителе, альбоме, песне, плейлисте или родственном исполнителе. Он

предоставляет разработчикам SDK, известный как Software Development Kit. Это набор инструментов для разработки программного обеспечения, доступных в одном устанавливаемом пакете.

3 Проектная часть

3.1 Архитектура системы взаимодействия

Разрабатываемое приложение предназначено для открытого пользования и является совокупностью трех систем, которые работают на клиент-серверной архитектуере. Основной частью проекта является мобильное приложение для платформы Android, написанное на языке Kotlin в среде разработки Android Studio. Мобильное приложение отправляет изображение backend серверу, написанному на языке Python с использование фреймворка Flask. Далее изображение обрабатывается при помощи модели машинного обучения CNN, которая была написана на Python. Полученный ответ в виде строки с обнаруженным настроением передается обратно в мобильное приложение и формируется соответственный список музыкальных композиций. Музыка, в свою очередь, тоже обрабатывается ИИ, при помощи K-Means Clustering алгоритма.

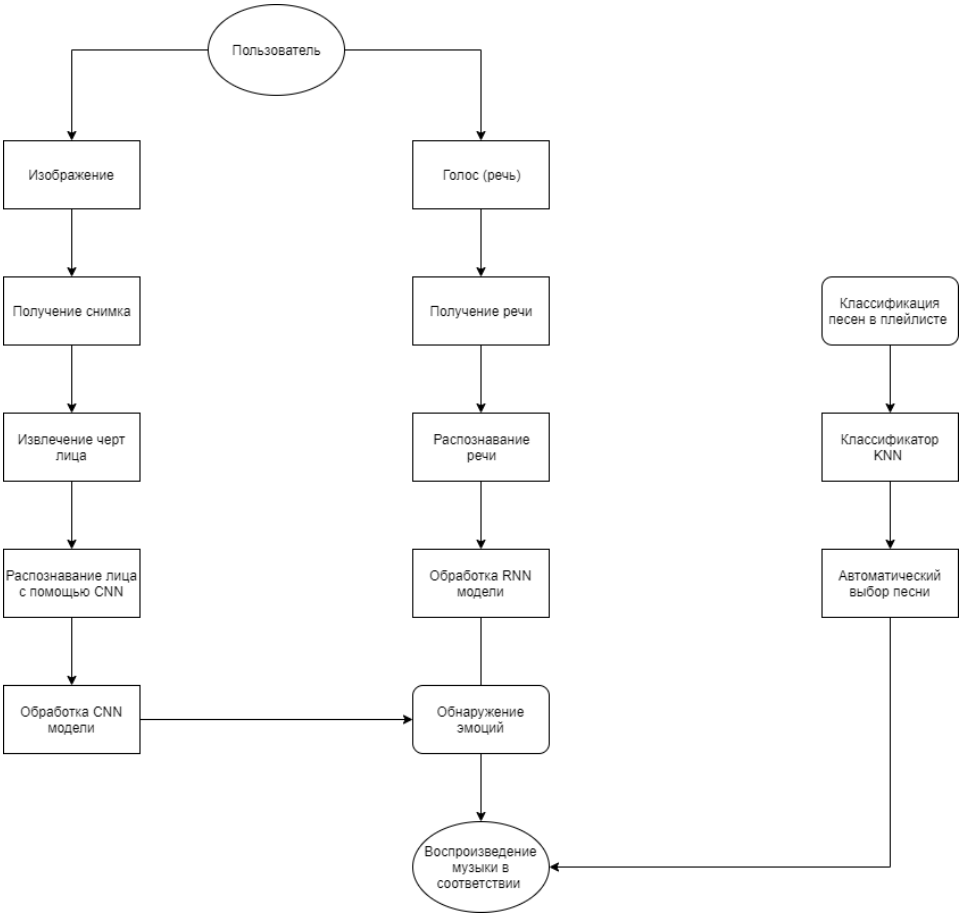


Рисунок 3.1 - Архитектура работы приложения

3.2 Описание диаграммы деятельности

Диаграмма UML показывает унифицированное визуальное представление системы UML, позволяющее разработчикам или владельцам бизнеса понимать, анализировать и выполнять структуру и поведение своей системы.

Общая структура программного обеспечения является основой каждой системы и оказывает значительное влияние на качество программного обеспечения и его ремонтопригодность. Любая новая функция или изменение необходимо интегрировать в эту структуру, не мешая другим существующим частям системы и не нарушая их. Это помогает поддерживать хорошо организованную общую структуру системы, особенно в гибких процессах разработки программного обеспечения.

Для создания диаграммы была использована бесплатная веб-платформа lucidchart.com, созданная компанией Lucid Software Inc.

Работа приложения начинается с аутентификации пользователя через Spotify или Google. Благодаря Spotify API у приложения есть доступ к обширным данным как музыкальные композиции, пользовательские данные и артисты.

После того как пользователь аутентифицировался, у него создается Личный кабинет (Профиль пользователя), где можно управлять своим аккаунтом и просматривать статистику своих прослушиваний.

Так же в приложении есть Главная страница, где пользователь может формировать плейлисты и прослушивать их. Страница представляет собой список музыкальных композиций.

Самой главной страницей и идеей всего приложения является Камера. Нажав на нее, пользователь может сделать снимок и зафиксировать настроение в режиме лайв или выбрать фото из галереи смартфона с соответствующей эмоцией. Далее снимок отправляется на Flask-сервер.

Flask-сервер в свою очередь ждет запрос от мобильного приложения. Как только он получает изображение/снимок, сохраняет его в главной директории как файл. После чего CNN модель обрабатывает изображение: обнаруживает лицо с помощью алгоритма HAAR cascade и применяет 4-ч слойную нейронную сеть, в следствии чего возвращается ответ в виде строки настроения, определенное на изображении. Полученный результат передается в модель K-mean algorithm, которая является кластеризатором песен.

K-mean algorithm получает настроение и генерирует соответственный плейлист, основываясь на 5-ти признаках таких как танцевальность, громкость, наличие речи, акустичность и живость музыкальных композиций.

Далее формируется JSON-файл, состоящий из списка музыкальных композиций, названия плейлиста и прочее. После чего передатся обратно в мобильное приложение и выводится на Главной странице.

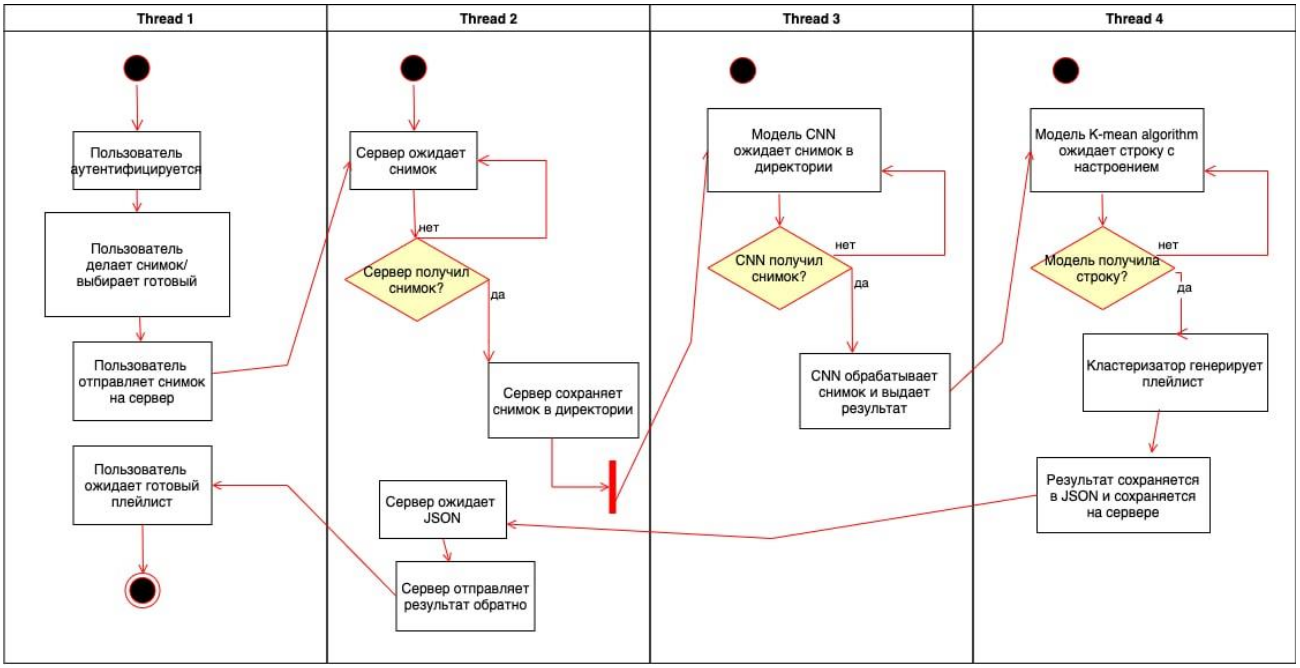


Рисунок 3.2.1 – Диаграмма деятельности

3.3 Разработка пользовательского интерфейса

Разработка пользовательского интерфейса является ключевым фактором в разработке программного обеспечения. Пользовательский интерфейс - это процесс повышения удовлетворенности пользователей за счет повышения удобства использования и доступности продукта, веб-страницы или приложения.

UI дизайн приложения был разрботан с помощью сервиса Figma.com, который является инструментом для дизайна с возможностью командной работы в режиме реального времени.

При первичном открытии приложения выводится приветственная начальная страница.

Рисунок 3.3.1 – Начальная страница

После нажатия на кнопку «Начать» пользователь попадает на страницу аутентификации.

Рисунок 3.3.2 – Страница аутентификации

У пользователя есть два варианта аутентификации: через Spotify или Google. Аутентификация происходит через API.

После удачной аутентификации пользователю открывается Главная страница. Так же, пользователь может легко переходить по страницам благодаря меню навигации расположенное в нижней части экрана.

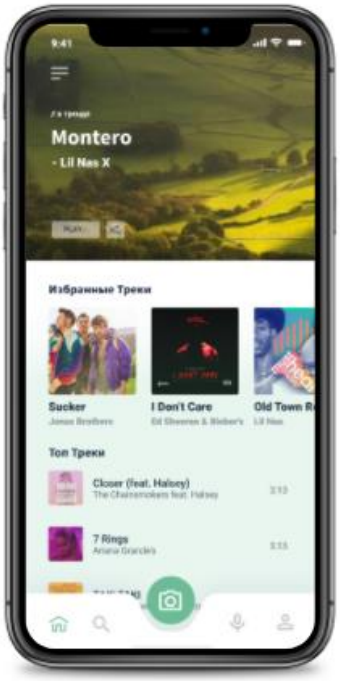


Рисунок 3.3.3 – Главная страница

С помощью меню навигации пользователь может перейти в Личный кабинет, нажав иконку профиля. На странице профиля есть статистика настроения, которая формируется в зависимости от прослушанных плейлистов.

Так же личная информация как Фамилия и Имя, а также фото-аватар.

Рисунок 3.3.4 – Профиль пользователя

Нажав на главную зеленую кнопку в меню навигации «Камера» пользователю открывается страница Камера, где он может сделать снимок или выбрать последнее фото из галереи.

Рисунок 3.3.5 – Страница Камера

Рисунок 3.3.6 – Выбор фото из галереи

Далее нажав на кнопку «Выбрать» изображение отправляется на backend сервер на обработку. После обработки изображения, возвращается ответ в таком виде:

Рисунок 3.3.7 – Ответ от сервера

После нажатия кнопки «Готово» формируется плейлист с соответствующим настроением.

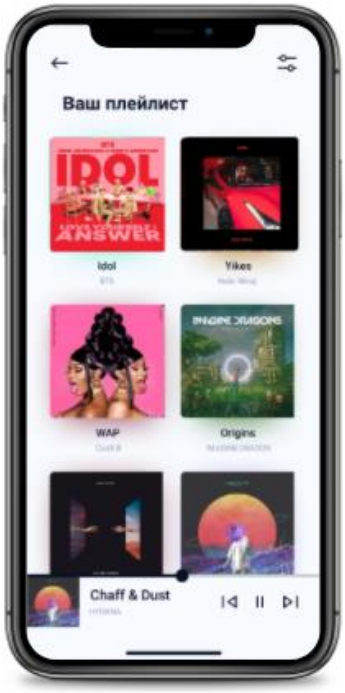


Рисунок 3.3.8 – Плейлист

Если нажать на иконку музыки попадаем на страницу песни:

Рисунок 3.3.8 – Страница песни

4 Экспериментальный раздел

4.1 Реализация аутентификации в мобильном приложении

Существует несколько способов регистрации пользователей в программном обеспечении, однако самой продвинутой и защищенной является аутентификация. Аутентификация – это идентификация пользователя с помощью учетных данных.

Для аутентификации в приложении у пользователя есть выбор между Spotify API и Google API. Благодаря чему приложение может получать и управлять данными через интернет, используя HTTP протокол.

Чтобы использовать Spotify API нужно зарегестрироваться в developer.spotify.com и зарегестрировать свое приложение для того, чтобы получить ClientID, после чего записываем URI перенаправления (RedirectURI).

Сначала мы открываем AuthenticationRequest с нашим ClientID, типом ответа, который нам нужен (в данном случае токен аутентификации) и RedirectURI.

Затем мы устанавливаем наши запрашиваемые данные, то есть разные разрешения, которые нам нужно запросить у пользователя, например, разрешение на чтение его личной информации. Запрошенные области будут отображаться пользователю, и он должен предоставить их приложению.

После того как пользователь удачно аутентифицируется в приложении, приложение получает токен, который мы сохраняем в нашем постоянном хранилище с помощью SharedPreferences.

Дальше при помощи Get запросов получаем пользовательские данные как JSON файл, записав в запрос наш полученный ранее токен.

4.2 Реализация CNN

Как упоминалось во второй части пояснительной записки (технологический раздел), CNN - алгоритм глубокого обучения, который может получать входное изображение, распределять значимость различным объектам на изображении и иметь возможность отличать один от другого.

Для того, чтобы определить эмоцию на лице человека, распределяем эмоции на классы. Получилось 7 больших классов таких как злость, счастье, удивленность, грусть, отвращение, напуганность и нейтральность.

Сначала с помощью Haar cascade модели обнаруживаем лицо на изображении, затем генерируем патчи с частями лица. Далее из патчей извлекаем признаки и пускаем обученную модель CNN.

Для обучение модели были использованы 4х слойная нейронная сеть и 35 тысяч тренировачных данных, собранные из платформы Kaggle.

После обработки изображения модель формирует ответ в виде названия одного из 7-ми классов, описанных выше.

4.3 Реализация связи моделей машинного обучения с бэкенд-сервером

Для того, чтобы наша модель машинного обучения была доступна для конечных пользователей мы развертываем модель при помощи Flask сервера.

Во-первых, записываем коды моделей машинного обучения (CNN и K-means algorithm) в файл с расширением .py в нашем бэкенд проекте.

Сериализируем модель преобразовав объект в поток символов с помощью библиотеки pickle.

Далее создаем API, который получает и отправляет данные между моделями и мобильным приложением. Учитывая то, что наш Flask server умеет получать изоброжения от мобильного приложения, создаем POST запрос для отправки изображения в модель CNN. И GET запрос для получения результата обработки изображения в виде строки настроения.

Далее полученный ответ при помощи POST запроса отправляется в K-means algorithm для получения плейлита с соответствующим настроением. Ответ возвращается в формате JSON.

ЗАКЛЮЧЕНИЕ

В рамках данного дипломного проекта поставленная цель была выполнена успешно. В результате реализовано клиент-серверное мобильное приложение на Android, которое, при помощи машинного обучения, определяет эмоцию пользователя и создает плейлист под настроение. Данный проект весомо сокращает время подбора музыкальных композиций и создает идеальный плейлист для текущего настроения.

В процессе подготовки к реализации проекта была подробно изучена предметная область распознавания лиц, эмоций посредством нейронных сетей. Было выявлено, что наличие модели распознавания эмоций значительно улучшает работу продуктов, особенно, если это напрямую связано с пользователями.

В процессе разработки и проектирования архитектуры проекта была проведена детальная работа над подбором оптимальных технологий для общения между частями проекта и выявлены наиболее удачные и быстрые.

Мобильное приложение “Музыкальный плейлист основанный на эмоциях” представляет собой законченный продукт, который имеет удобный пользовательский интерфейс

и оптимальную и эффективную производительность. Приложение поможет легко и быстро сгенерировать плейлист лишь по фотографии, где отражена текущая эмоция.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Flask’s documentation // Электронная версия на сайте https://flask.palletsprojects.com/en/2.0.x/
2. Гундфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение // Электронная версия на сайте https://liters.ru
3. Адриан Р. Deep learning for Computer Vision with Python // Электронная версия на сайте https://liters.ru
4. Джурда П., The Book of Why // Электронная версия на сайте https://liters.ru
5. Клепман М., Высоконагруженные приложения // Электронная версия на сайте https://liters.ru
6. Макконнелл С., Совершенный код // Электронная версия на сайте https:// all-ebooks.com
7. Гриффитс Д. Head First. Программирование для Android // Электронная версия на сайте https://all-ebooks.com
8. Cascade classifier // Электронная версия на сайте https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
9. How do CNN works? // Электронная версия на сайте https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/
10. Kotlin and Android // Электронная версия на сайте https://developer.android.com/kotlin/first
11. Understanding K-means Clustering // Электронная версия на сайте https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

Приложение А
(обязательное)

Техническое задание

А1.1 Техническое задание на разработку мобильного приложения с распознаванием эмоций

Настоящее техническое задание распространяется на разработку мобильного приложения с распознаванием эмоций. Предполагается что использовать данную систему будут пользователи смартфонов с Android. Мобильное приложение позволит значительно сократить процесс подбора музыкальных композиций.

А1.1.1 Основания для разработки

Редактор разрабатывается на основании устного распоряжения Заместителя Председателя Правления АО «Институт цифровой техники и технологий» по разработке программных продуктов.

А1.1.2 Назначение

Разрабатываемое мобильное приложение предназначено для генерации плейлистов, основанных на эмоциях, которые распознает модель машинного обучения.

А1.1.3 Требования к функциональным характеристикам

Мобильное приложение должно обеспечить возможность выполнения следующих функций:

- распознавание эмоций;
- аутентификация пользователя;
- вывод данных при помощи Spotify API;
- поиск по музыкальным композициям;

Продолжение приложения А

А1.1.4 Требования к надежности

Обеспечить конфиденциальность данных пользователя.

Приложение Б

(обязательное)

Текст программы

/*Реализация аутентификации*/

```
public class MainActivity extends AppCompatActivity {

    public static final String CLIENT_ID = "089d841ccc194c10a77afad9e1c11d54";
    public static final String REDIRECT_URI = "spotify-sdk://auth";
    public static final int AUTH_TOKEN_REQUEST_CODE = 0x10;
    public static final int AUTH_CODE_REQUEST_CODE = 0x11;

    private final OkHttpClient mOkHttpClient = new OkHttpClient();
    private String mAccessToken;
    private String mAccessCode;
    private Call mCall;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().setTitle(String.format(
                Locale.US, "Spotify Auth Sample %s", BuildConfig.LIB_VERSION_NAME));
    }

    @Override
    protected void onDestroy() {
        cancelCall();
        super.onDestroy();
    }

    public void onGetUserProfileClicked(View view) {
        if (mAccessToken == null) {
            final Snackbar snackbar = Snackbar.make(findViewById(R.id.activity_main),
R.string.warning_need_token, Snackbar.LENGTH_SHORT);
            snackbar.getView().setBackgroundColor(ContextCompat.getColor(this,
R.color.colorAccent));
            snackbar.show();
            return;
        }

        final Request request = new Request.Builder()
                .url("https://api.spotify.com/v1/me")
                .addHeader("Authorization", "Bearer " + mAccessToken)
                .build();
```

```
        cancelCall();
        mCall = mOkHttpClient.newCall(request);

        mCall.enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                setResponse("Failed to fetch data: " + e);
            }

            @Override
            public void onResponse(Call call, Response response) throws IOException {
                try {
                    final JSONObject jsonObject = new JSONObject(response.body().string());
                    setResponse(jsonObject.toString(3));
                } catch (JSONException e) {
                    setResponse("Failed to parse data: " + e);
                }
            }
        });
    }

    public void onRequestCodeClicked(View view) {
        final AuthorizationRequest request =
getAuthenticationRequest(AuthorizationResponse.Type.CODE);
        AuthorizationClient.openLoginActivity(this, AUTH_CODE_REQUEST_CODE, request);
    }

    public void onRequestTokenClicked(View view) {
        final AuthorizationRequest request =
getAuthenticationRequest(AuthorizationResponse.Type.TOKEN);
        AuthorizationClient.openLoginActivity(this, AUTH_TOKEN_REQUEST_CODE, request);
    }

    public void onClearCredentialsClicked(View view) {
        AuthorizationClient.clearCookies(this);
    }

    private AuthorizationRequest getAuthenticationRequest(AuthorizationResponse.Type type) {
        return new AuthorizationRequest.Builder(CLIENT_ID, type, getRedirectUri().toString())
                .setShowDialog(false)
                .setScopes(new String[]{"user-read-email"})
                .setCampaign("your-campaign-token")
                .build();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
        super.onActivityResult(requestCode, resultCode, data);
        final AuthorizationResponse response = AuthorizationClient.getResponse(resultCode, data);

        if (AUTH_TOKEN_REQUEST_CODE == requestCode) {
            mAccessToken = response.getAccessToken();
            updateTokenView();
        } else if (AUTH_CODE_REQUEST_CODE == requestCode) {
            mAccessCode = response.getCode();
            updateCodeView();
        }
    }

    private void setResponse(final String text) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                final TextView responseView = findViewById(R.id.response_text_view);
                responseView.setText(text);
            }
        });
    }

    private void updateTokenView() {
        final TextView tokenView = findViewById(R.id.token_text_view);
        tokenView.setText(getString(R.string.token, mAccessToken));
    }

    private void updateCodeView() {
        final TextView codeView = findViewById(R.id.code_text_view);
        codeView.setText(getString(R.string.code, mAccessCode));
    }

    private void cancelCall() {
        if (mCall != null) {
            mCall.cancel();
        }
    }

    private Uri getRedirectUri() {
        return Uri.parse(REDIRECT_URI);
    }
}
```

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт кибернетики и информационных технологий

Айтанова Акерке
5В070400 – Вычислительная техника и программное обеспечение

ОТЗЫВ НАУЧНОГО РУКОВОДИТЕЛЯ

к дипломному проекту

Тема: «Музыкальный плейлист основанный на эмоциях».

Работа самостоятельна и выполнена на высоком уровне. Используются современные технологии и программные платформы.

Работа состоит из 4-х разделов, включая введение, заключение и приложение.

В первом разделе работы представлено общее описание. Проведен анализ и описание предметной области, также раздел раскрывает цель разработки динамического редактора.

Во втором разделе представлены технологии для создания динамического редактора. Проведено сравнение используемых технологий и выбраны лучшие для данного проекта.

В третьем разделе описана проектная часть дипломного проекта. Представлена архитектура системы взаимодействия. Имеется диаграмма деятельности, которая визуально представляет последовательность действий. Также в разделе представлено проектирование пользовательского интерфейса. Наглядно продемонстрированы и описаны все окна редактора.

В четвертом разделе был описан функционал, и то, как данные функции были реализованы.

Дипломный проект выполнен с учетом всех требований, предъявляемых к дипломному проекту по специальности 5В070400 – «Вычислительная техника и программное обеспечение», студентка Айтанова А. рекомендована к защите дипломного проекта и заслуживает присвоения академической степени «бакалавра» по специальности 5В070400 – «Вычислительная техника и программное обеспечение».

Научный руководитель: Магистр технических наук, Сениор-лектор
Куникеев Айдын

Метаданные

Название
Айтанова_диплом.docx

Автор
Айтанова Акерке

Научный руководитель
Айдын Куникеев

Подразделение
ИКиИТ

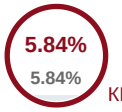
Список возможных попыток манипуляций с текстом

В этом разделе вы найдете информацию, касающуюся манипуляций в тексте, с целью изменить результаты проверки. Для того, кто оценивает работу на бумажном носителе или в электронном формате, манипуляции могут быть невидимы (может быть также целенаправленное вписывание ошибок). Следует оценить, являются ли изменения преднамеренными или нет.

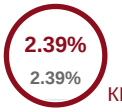
Замена букв	ß	0
Интервалы	A→	19
Микропробелы	·	22
Белые знаки	ß	0
Парафразы (SmartMarks)	a	19

Объем найденных подобий

Обратите внимание!Высокие значения коэффициентов не означают плагиат. Отчет должен быть проанализирован экспертом.



КП1



КП2

КЦ

25
Длина фразы для коэффициента подобия 2

4779
Количество слов

38710
Количество символов

Подобия по списку источников

Просмотрите список и проанализируйте, в особенности, те фрагменты, которые превышают КП №2 (выделенные жирным шрифтом). Используйте ссылку «Обозначить фрагмент» и обратите внимание на то, являются ли выделенные фрагменты повторяющимися короткими фразами, разбросанными в документе (совпадающие сходства), многочисленными короткими фразами расположенные рядом друг с другом (парафразирование) или обширными фрагментами без указания источника ("криптоцитаты").

10 самых длинных фраз

Цвет текста

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ И АДРЕС ИСТОЧНИКА URL (НАЗВАНИЕ БАЗЫ)	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://www.kaspersky.ru/resource-center/definitions/what-is-facial-recognition	45	0.94 %
2	**КОРПОРАТИВНАЯ_СИСТЕМА_РЕГИСТРАЦИИ_ПОСЕТИТЕЛЕЙ_ПОПОВ_К.docx** Попов Константин Евгеньевич **5/18/2017** Satbayev University (ИКиИТ)	41	0.86 %
3	**Штарк ДП.docx** Владислав Штарк **1/28/2020** Turan University (TU) (Б - Вычислительная техника и программное обеспечение)	28	0.59 %
4	https://satsis.info/news/15177-eksperty-smogli-obmanut-kameru-videonablyudeniya-s-pomoschyu-abstraktnoy-kartinki.html	22	0.46 %

5	https://www.kaspersky.ru/resource-center/definitions/what-is-facial-recognition	20	0.42 %
6	https://ru.wikipedia.org/wiki/ARM_architecture	20	0.42 %
7	https://www.kaspersky.ru/resource-center/definitions/what-is-facial-recognition	18	0.38 %
8	**Разработка конструкции безбалансирного гидроприводного станка качалки с подачей 35 м3/сут** Казиев Рустем Ерланович **5/10/2018** Satbayev University (Г_М_И)	15	0.31 %
9	**Разработка конструкции безбалансирного гидроприводного станка качалки с подачей 35 м3/сут** Казиев Рустем Ерланович **5/10/2018** Satbayev University (Г_М_И)	15	0.31 %
10	https://ru.wikipedia.org/wiki/ARM_architecture	9	0.19 %

из базы данных RefBooks (0.00 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

из домашней базы данных (2.30 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**Разработка конструкции безбалансирного гидроприводного станка качалки с подачей 35 м3/сут** Казиев Рустем Ерланович **5/10/2018** Satbayev University (Г_М_И)	61 (7)	1.28 %
2	**КОРПОРАТИВНАЯ_СИСТЕМА_РЕГИСТРАЦИИ_ПОСЕТИТЕЛЕЙ_ПОПОВ_К.docx** Попов Константин Евгеньевич **5/18/2017** Satbayev University (ИКиИТ)	41 (1)	0.86 %
3	**Вялых А Методы и устройства компенсации реактивной мощности при электроснабжении нелинейных и резкопеременных нагрузок в СЭС предприятия.doc** Вялых А.С. **5/21/2018** Satbayev University (И_И_В_Т)	8 (1)	0.17 %

из программы обмена базами данных (0.59 %)

ПОРЯДКОВЫЙ НОМЕР	НАЗВАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	**Штарк ДП.docx** Владислав Штарк **1/28/2020** Turan University (TU) (Б - Вычислительная техника и программное обеспечение)	28 (1)	0.59 %

из интернета (2.95 %)

ПОРЯДКОВЫЙ НОМЕР	ИСТОЧНИК URL	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)	
1	https://www.kaspersky.ru/resource-center/definitions/what-is-facial-recognition	90 (4)	1.88 %
2	https://ru.wikipedia.org/wiki/ARM_architecture	29 (2)	0.61 %
3	https://satsis.info/news/15177-eksperty-smogli-obmanut-kameru-videonablyudeniya-s-pomoschyu-abstraktnoy-kartinki.html	22 (1)	0.46 %

Список принятых фрагментов (нет принятых фрагментов)

ПОРЯДКОВЫЙ НОМЕР	СОДЕРЖАНИЕ	КОЛИЧЕСТВО ИДЕНТИЧНЫХ СЛОВ (ФРАГМЕНТОВ)

Протокол анализа Отчета подобия Научным руководителем

Заявляю, что я ознакомился(-ась) с Полным отчетом подобия, который был сгенерирован Системой выявления и предотвращения плагиата в отношении работы:

Автор: Айтанова Акерке

Название: Айтанова_диплом.docx

Координатор:Айдын Куникеев

Коэффициент подобия 1:5.84

Коэффициент подобия 2:2.39

Замена букв:0

Интервалы:19

Микропробелы: 22

Белые знаки: 0

После анализа Отчета подобия констатирую следующее:

☐ обнаруженные в работе заимствования являются добросовестными и не обладают признаками плагиата. В связи с чем, признаю работу самостоятельной и допускаю ее к защите;

☐ обнаруженные в работе заимствования не обладают признаками плагиата, но их чрезмерное количество вызывает сомнения в отношении ценности работы по существу и отсутствием самостоятельности ее автора. В связи с чем, работа должна быть вновь отредактирована с целью ограничения заимствований;

☐ обнаруженные в работе заимствования являются недобросовестными и обладают признаками плагиата, или в ней содержатся преднамеренные искажения текста, указывающие на попытки сокрытия недобросовестных заимствований. В связи с чем, не допускаю работу к защите.

Обоснование:

.....

27.05.2021 .....

Дата *Подпись Научного руководителя*