копия

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Кафедра «Компьютерные технологии»
Специальность «Вычислительная техника и программное обеспечение»

Допущен к защите
Зав. кафедрой КТ
_____
«____»_____2014 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
пояснительная записка

Тема «Исследование технологических подходов для разработки современных распределённых систем»

Магистрант_____ Сон И.В.
подпись (Ф.И.О.)

Руководитель диссертации_____ Турганбаев Е.С.
подпись (Ф.И.О.)

Рецензент_____ Сатыбалдиев О.С.
подпись (Ф.И.О.)

Алматы, 2014 г.

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Факультет «Информационные технологии»
Специальность «Вычислительная техника и программное обеспечение»
Кафедра «Компьютерные технологии»

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Сон Игорю Владиславовичу
(фамилия, имя, отчество)

Тема диссертации «Исследование различных технологических подходов для разработки современных распределённых систем»
утверждена Ученым советом университета №108 от «16» ноября 2012 г.
Срок сдачи законченной диссертации «___»_____

Цель исследования – создание распределённой системы для работы с отпечатками аудио файлов

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации:
- анализ существующих систем распознавания и снятия отпечатков аудио;
- исследование алгоритма распознавания отпечатков аудио;
- исследование эффективности различных форматов и характеристик аудио;
- исследование алгоритма поиска;
- исследование спектрограмм исходного и искажённого формата.

Перечень графического материала (с точным указанием обязательных чертежей):
- спектрограммы аудио файлов;
- график Ганта.

Рекомендуемая основная литература

- L. Lu, H.J. Zhang, and S.Z. Li. Content-based audio classification and segmentation by using support vector machines. Multimedia Systems, 8(6):482–492, Apr. 2003.
- N. Maddage, C. Xu, M. Kankanhalli, and X. Shao. Content-based music structure analysis with applications to music semantics understanding. In
- Proceedings of the ACM International Conference on Multimedia, pages 112–119. ACM, 2004.

Г Р А Ф И К
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
1. Теоретическая работа		
1.1 Анализ рынка программного обеспечения, предназначенного для обработки отпеатков аудио.	30.11.2012 г.	
1.2 Исследование существующих агоритмов обработки цифровых аудио данных	30.01.2013 г.	
1.3 Постановка задачи разработки	30.02.2013 г.	
1.4 Анализ эффективности различных физических характеристик	30.05.2013 г.	
1.5 Анализ применения характеристик	30.06.2013 г.	
1.6 Анализ и сравнение с готовыми алгоритмами	30.09.2013 г.	
1.7 Подготовка доклада на научную конференцию соискателей и магистрантов	30.12.2013 г.	
2. Экспериментальная работа		
2.1. Создание пробной версии системы снятия отпечатков аудио	30.11.2013 г.	
2.2. Тестирование функционала системы и улучшение на базе тестирования	30.12.2013 г.	
2.3. Анализ эффективности готовой версии системы	30.01.2014 г.	
3. Оформление диссертации	30.04.2014 г.	

Дата выдачи задания_____

Заведующий кафедрой_____(Куралбаев З.К.)
(подпись) (Ф.И.О.)

Руководитель диссертации_____(Турганбаев Е.С.)
(подпись) (Ф.И.О.)

Задание принял к исполнению
магистрант _____(Сон И.В.)
(подпись) (Ф.И.О.)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

ФАКУЛЬТЕТ «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

Допущен к защите:

зав. кафедрой «Компьютерные технологии»

д. ф.-м. н., профессор _____ Куралбаев З.К.

«___»_____2014г.

Магистерская диссертация

Исследование различных технологических подходов для разработки современных распределённых систем

специальность: 6М070400 - Вычислительная техника и программное обеспечение

Магистрант Сон Игорь Владиславович

_____ Сон И.В.

Научный руководитель,

к. ф.-м. н., доцент _____ Турганбаев Е.С.

АЛМАТЫ 2014

АННОТАЦИЯ

В данной диссертационной работе в качестве распределённой системы представлена система идентификации аудио с одной из многочисленных реализаций в виде распознавания звука с присутствием помех. Описываемая в диссертационной работе система имеет распределённую трех-уровневую клиент-серверную архитектуру и используется для снятия и распознавания отпечатков аудио файлов. Результаты исследований позволяют увеличить объём цифровых аудио данных, качество соответствий и уменьшить время поиска.

АНДАТПА

Бұл диссертациялық жұмыста таратылған жүйе ретінде бұрмалануы бар дыбысты айырып тану түрінде іске асырылған аудио сәйкестендіру жүйесі көрсетіледі. Баяндалатын жүйенің негізінде таратылған үш деңгейлі клиент-сервер архитектурасы жатыр. Бұл жүйе аудио файлдардың іздерін шешіп, оларды айырып тану мақсатында пайдаланылады. Зерттеулердің нәтижелері сандық дыбыстық мәліметтердің көлемін арттыруға, сәйкестендіру сапасын жоғарлатуға және іздеу ұзақтығын азайтуға мүмкіндік береді.

ANNOTATION

In this dissertation work as a distributed system is presented an audio system for identifying one of the many implementations in the form of sound recognition with the presence of interference. Described in this dissertation work system has distributed as three-tier client-server architecture and is used to relieve and fingerprint audio files. Research results allowed to increase the volume of digital audio data, the quality matches and decrease the search.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время всё чаще используются сложные системы с распределённой структурой. Распределение обеспечивает отказоустойчивость, надёжность и хорошее быстродействие. В данной работе в качестве распределённой системы представлена система идентификации аудио с одной из многочисленных реализаций в виде распознавания звука с присутствием помех. Описываемая в работе система имеет распределённую трех-уровневую клиент-серверную архитектуру.

В работе часто встречается выражение «отпечаток аудио». Отпечаток аудио - это выражение, которое используется для описания методов обработки аудиоданных.

Есть в настоящий момент несколько известных и устойчивых алгоритмов снятия отпечатков аудио. Новейшие алгоритмы создаются для скорости и, как правило, используют данные, найденные спектральным анализом аудио для создания «отпечатков» аудио.

Поиск по каталогу музыки — это задача, которую можно решать разными путями, как с точки зрения пользователя, так и технологически. К примеру, Яндекс уже довольно давно научился искать и по названиям композиций, и по текстам песен. В этом проекте речь пойдёт о поиске по аудио сигналу, а если конкретно — по записанному с микрофона фрагменту музыкального произведения.

В мире есть всего несколько специализированных компаний, которые профессионально занимаются распознаванием музыкальных треков. Несмотря на то, что предстоит ещё немало сделать, качество распознавания уже сопоставимо с лидерами в этой области. К тому же поиск музыки по аудио фрагменту не самая тривиальная и освещённая тема; надеюсь, что многим будет любопытно узнать подробности.

Снятие отпечатков аудио может быть использовано для идентификации частей аудио файла. Зарубежная полиция использует отпечатки аудио для идентификации краденной музыки на компьютерах. В случае находки музыкальные продюсеры могут использовать аудио отпечатки для поиска злоумышленника. Также как и полиция, всемирно известный сервис YouTube использует систему под названием ContentID, которая в свою очередь использует систему снятия отпечатков аудио, разработанную Audible Magic. ContentID помогает официальным владельцам прав идентифицировать видео на YouTube, которое может использовать их музыку и видео.

Данный проект был применён в ТОО «WithArts» при разработке музыкального проекта «Music Shark». Были использованы ключевые алгоритмы и методы считывания и поиска отпечатков цифровых аудио данных.

1.1 Цели

Этот проект содержит основные аспекты разработки распределённой системы и основывается на реализации алгоритма снятия отпечатков аудио в стиле многих разработанных методов, а также на разработке своей версии, которая принимает во внимание другие аспекты звука, «отпечаток» которого я намереваюсь дополнить и улучшить, что в последствии может улучшить скорость распознавания и качество самого распознавания.

Конечное программное обеспечение будет состоять из сервера, который состоит из Web оболочки и сервиса для отпечатков, и клиентов в виде браузерного клиента и клиента, основанного на платформе Linux.

1.2 Применение отпечатков аудио

Считывание отпечатков имеет множество применений в мире. Например, наиболее популярными в данной сфере мировыми компаниями являются Shazam, SoundHound и с недавнего времени Yandex, благодаря проекту «Яндекс Музыка». Gracenote является компанией, которая также предлагает мобильное программное обеспечение для аудио идентификации, но ещё и имеет программное обеспечение для проигрывателей цифровых данных, как iTunes и Winamp, таким образом, чтобы музыке можно было должным образом дать название, использовать в поиске, группировать и идентифицировать. Gracenote может также использоваться для распознавания речи и используется в настоящее время с коммуникационной системой Ford’s SYNC in-car.

Last.fm, использует снятие отпечатков аудио, чтобы исправить несоответствующие музыкальные метаданные.

Снятие отпечатков аудио может быть использовано для идентификации частей аудио файла. Зарубежная полиция использует отпечатки аудио для идентификации краденной музыки на компьютерах. В случае находки музыкальные продюсеры могут использовать аудио отпечатки для поиска злоумышленника. Также как и полиция, всемирно известный сервис YouTube использует систему под названием ContentID, которая в свою очередь использует систему снятия отпечатков аудио, разработанную Audible Magic. ContentID помогает официальным владельцам прав идентифицировать видео на YouTube, которое может использовать их музыку и видео.

1.3 Современные методы

Современные актуальные методы в основном являются запатентованными с немногими техническими деталями, если разработаны в стране, которая позволяет иметь патенты программного обеспечения. Однако, существует много академической документации по теме идентификации аудио посредством снятия аудио отпечатков, и некоторые компании, как Shazam, также произвели отчеты, дающие краткий обзор алгоритмов, которые они создали.

1.4 Планы по улучшению

Я планирую реализовать элементарный алгоритм снятия отпечатков аудио файла. Как только этот алгоритм заработает, проведу ряд экспериментов с такими характеристиками, как темп или музыкальный ключ, который я надеюсь, сузит поисковое поле и уменьшит ложные положительные соответствия.

2 Аспекты разработки

Чтобы увеличить точность снятия аудио отпечатков есть два принципиальных похода. Первый подход состоит в том, чтобы взять большую выборку или извлечь более "уникальные" характеристики аудио из каждой выборки, но увеличение объема аудио, или желание извлечь больше информации из аудио нормального объёма, могло бы добавить неточность, которая уменьшит точность, и в последствии создаст противоположный намеченному эффект. Считывание большего количества данных с аудио также увеличивает вероятность появления ложных совпадений, так как было бы больше данных для изменения, и в дальнейшем уникальностей, также появляется больше шансов соответствия другого аудио. С большим объемом аудио данных увеличивается длина аудио, которую нужно распознать. Генерируется один небольшой хеш на аудио файл, и поэтому любое изменение составляется обобщённо и уменьшается до такой степени, что незначительные изменения моли бы быть отброшены, если ранее они были бы хешированы и соответствовали запросу.

Вторая идея увеличить точность снятия отпечатков пальцев состояла бы в том, чтобы принять во внимание больше чем только подача и время базируемое аудио восприятие, вычисляя хеш и ища соответствия. Вместо этого можно включить более уникальные факторы к поиску, которые, мы надеемся, сокращают времена поиска, также уменьшая возможности ложных положительных сторон. В то время как у двух частей аудио могли бы быть подобные цифровые отпечатки из-за шумовой интерференции, они могли бы все еще быть различимыми анализом, чтобы найти ключ, музыка была сочинена в или темп. По крайней мере, ограничивая цифровой отпечаток ищет на в пределах расстояния набора от другого фактора, такого как темп, Вы сокращаете список доступных для поиска элементов и уменьшаете времена поиска.

2.1 Обзор

В данной работе будет разработана система снятия отпечатков аудио, затем будет выполнены серии тестов, описанных далее в деталях в разделе тестирования этой работы. Затем, добавляя устойчивые к искажениям характеристики к моему цифровому отпечатку и поисковым функциям. Будут выполнены тесты снова, чтобы видеть, есть ли какие-либо статистически существенные улучшения во времени поиска, в точности или в сокращении ложных соответствий.

В данной работе не будет разработано, или даже реализовано, только устойчивое решение для снятия отпечатков аудио, которое могло бы конкурировать с коммерческими решениями, в настоящий момент доступными, потому что потребуется намного больше времени для исследования, реализации и тестирования системы, такой, как Shazam или Gracenote.

Shazam (в русскоязычном написании Шазам) — это коммерческий кроссплатформенный проект (ранее был доступен только для мобильных устройств, в данный момент существует приложение для Windows 8), осуществляющий поиск информации о песнях. Штаб-квартира располагается в Лондоне. Компания была основана в 1999 году Крисом Бартоном и др. В настоящий момент сервис предоставляет информацию о более чем 11 000 000 треков.

Пользователь Shazam использует микрофон своего мобильного телефона для записи фрагмента музыки, которая играет где-либо. Затем фрагмент сравнивается с центральной базой данных и при успешном сопоставлении выдается информация о треке.

TrackID — сервис, созданный в начале 2006 года калифорнийской компанией Gracenote для телефонов серии Walkman компании Sony Ericsson (кроме ранних моделей серии — W200, W300, W550, W700 и W800), позволяющий узнавать исполнителя и название музыкального произведения по его короткому отрывку, записанному на микрофон телефона. К середине 2007 года сервис TrackID стал стандартным для большинства телефонов Sony Ericsson (кроме некоторых бюджетных моделей), перестав быть эксклюзивной функцией серии Walkman. Также начиная с 2010 года сервис представлен в виде приложения для платформы Android и распространяется компанией Sony бесплатно через Google Play.

2.2 Критерии успеха

Для тестирования программного обеспечения и определения, были ли успехи в достижении цели сокращения ложно-положительных соответствий и увеличения поисковой скорости, я сохраню цифровые отпечатки музыки различных жанров и исполнителей, используя сжатие без потерь, чтобы получить самую соответствующую копию музыки, с которой я могу сравниваться. Ухудшение музыкальных файлов от сжатия (такой, как файл, преобразованный в MP3), где частотный диапазон сокращается, добавляя шум в различных объемах к аудио, чтобы сделать поисковые возможности менее надежными и поэтому нужно проверять целевые соответствия. Используя

ухудшенный файл можно считать отпечатки, используя оба метода и сравнить время, затраченное на поиск и результат. В итоге нужно получить статистически соответствующее улучшение для увеличенного цифрового отпечатка с точки зрения уменьшенного времени поиска и уменьшенных ложных соответствий.

MP3 (более точно, англ. MPEG-1/2/2.5 Layer 3; но не MPEG-3) — это кодек третьего уровня, разработанный командой MPEG, лицензируемый формат файла для хранения аудиоинформации.

MP3 является одним из самых распространённых и популярных форматов цифрового кодирования звуковой информации с потерями. Он широко используется в файлообменных сетях для оценочной передачи музыкальных произведений. Формат может проигрываться практически во всех популярных операционных системах, на большинстве портативных аудиоплееров, а также поддерживается всеми современными моделями музыкальных центров и DVD-плееров.

В формате MP3 используется алгоритм сжатия с потерями, разработанный для существенного уменьшения размера данных, необходимых для воспроизведения записи и обеспечения качества воспроизведения звука, очень близкого к оригинальному (по мнению большинства слушателей), хотя аудиофилы говорят об ощутимом различии. При создании MP3 со средним битрейтом 128 кбит/с в результате получается файл, размер которого примерно равен 1/11 от оригинального файла с CD-Audio. Само по себе несжатое аудио формата CD-Audio имеет битрейт 1411,2 кбит/с. MP3-файлы могут создаваться с высоким или низким битрейтом, который влияет на качество файла-результата. Принцип сжатия заключается в снижении точности некоторых частей звукового потока, что практически неразличимо для слуха на повсеместно распространённой аппаратуре низкой верности воспроизведения звука (например, доминирующее большинство портативных устройств, звуковых карт, музыкальных центров, автомагнитол и прочей не специальной аппаратуры), а также для людей старшего возраста, в связи с естественными необратимыми возрастными изменениями слухового аппарата, однако в большинстве случаев чётко различимы на специальной аппаратуре высокой верности воспроизведения. Данный метод называют кодированием восприятия. При этом на первом этапе строится диаграмма звука в виде последовательности коротких промежутков времени, затем на ней удаляется информация, не различимая человеческим ухом, а оставшаяся информация сохраняется в компактном виде. Данный подход похож на метод сжатия, используемый при сжатии картинок в формат JPEG.

FLAC (англ. Free Lossless Audio Codec) — популярный свободный кодек, предназначенный для сжатия аудиоданных без потерь.

В отличие от аудио-кодеков, обеспечивающих сжатие с потерями (MP3, AAC, WMA, Ogg Vorbis) FLAC, как и любой другой lossless-кодек, не удаляет никакой информации из аудио потока и подходит как для прослушивания музыки на высококачественной звуковоспроизводящей аппаратуре, так и для архивирования аудио коллекции.

Сегодня формат FLAC поддерживается множеством аудио приложений и портативных аудиоплееров, а также имеет большое количество аппаратных реализаций.

2.3 Среда разработки

Чтобы разработать программное обеспечение для этого проекта, я решил использовать Java в качестве языка программирования для Веб интерфейса, детализированные причины, обрисованные в общих чертах в детальном проектировании, разделяют позже в этом отчете. В данной работе используются языки C++ и Python в клиентских приложениях. В качестве базы данных используются Tokyo Tyrant и MySql(клиент - HeidiSQL); среда разработки java – EclipseIDE.

Eclipse (/iklɪ ps/, от англ. затмение) — свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation.

Наиболее известные приложения на основе Eclipse Platform — различные «Eclipse IDE» для разработки ПО на множестве языков (например, наиболее популярный «Java IDE», поддерживавшийся изначально, не полагается на какие-либо закрытые расширения, использует стандартный открытый API для доступа к Eclipse Platform).

Первоначально Eclipse разрабатывалась фирмой IBM как преемник среды разработки IBM Visual Age, в качестве корпоративного стандарта IDE для разработки на разных языках под платформы IBM. По сведениям IBM, проектирование и разработка стоили 40 миллионов долларов. Исходный код был полностью открыт и сделан доступным после того, как Eclipse был передан для дальнейшего развития независимому от IBM сообществу.

В Eclipse 3.0 (2003 год) были выбраны спецификации сервисной платформы OSGi, как архитектура среды исполнения. С версии 3.0 Eclipse перестал быть монолитной IDE, поддерживающей расширения, а сам стал набором расширений. В основе лежат фреймворк OSGi и SWT/JFace, на основе

которых разработан следующий слой —RCP (Rich Client Platform, платформа для разработки полноценных клиентских приложений). RCP служит основой не только для Eclipse, но и для других RCP-приложений, например Azureus и FileArranger. Следующий слой — сам Eclipse, представляющий собой набор расширений RCP — редакторы, панели, перспективы, модуль CVS и модуль Java Development Tools (JDT).

С 2006 года фонд Eclipse координирует ежегодный общий релиз (Simultaneous Release), который происходит в июне. Каждый выпуск включает в себя платформу Eclipse, а также ряд других проектов Eclipse.

Eclipse служит в первую очередь платформой для разработки расширений, чем он и завоевал популярность: любой разработчик может расширить Eclipse своими модулями. Уже существуют Java Development Tools (JDT), C/C++ Development Tools (CDT), разрабатываемые инженерами QNX совместно с IBM, и средства для языков Ada (GNAT bench, Hibachi), COBOL, FORTRAN, PHP и пр. от различных разработчиков. Множество расширений дополняет среду Eclipse менеджерами для работы с базами данных, серверами приложений и др.

Eclipse JDT (Java Development Tools) — наиболее известный модуль, нацеленный на групповую разработку: среда интегрирована с системами управления версиями —CVS, GIT в основной поставке, для других систем (например, Subversion, MS SourceSafe) существуют плагины. Также предлагает поддержку связи между IDE и системой управления задачами (ошибками). В основной поставке включена поддержка трекера ошибок Bugzilla, также имеется множество расширений для поддержки других трекеров (Trac, Jira и др.). В силу бесплатности и высокого качества, Eclipse во многих организациях является корпоративным стандартом для разработки приложений.

Eclipse написана на Java, потому является платформо-независимым продуктом, за исключением библиотеки SWT, которая разрабатывается для всех распространённых платформ (см. ниже). Библиотека SWT используется вместо стандартной для Java библиотеки Swing. Она полностью опирается на нижележащую платформу (операционную систему), что обеспечивает быстроту и натуральный внешний вид пользовательского интерфейса, но иногда вызывает на разных платформах проблемы совместимости и устойчивости приложений.

3 Краткое планирование

Для повышения точности аудио отпечатков есть два основных способа. Первый заключается в том, чтобы извлечь больше «уникальных» аудио характеристик восприятия данных из каждого аудио, но увеличение размера выборки, или попытка извлечь больше информации из обычного размера, могло бы добавить искажение, которое уменьшило бы точность и, следовательно, был бы противоположный ожидаемого эффект. Считывание большего количества данных из образца также увеличивает вероятность ложных соответствий, в то же время каждая крупная выборка будет иметь больше возможностей для вариаций, и поэтому уникальность теряется, и увеличивается шанс на соответствие неверного образца.

Вторая идея для повышения точности отпечатков будет принимать во внимание больше данных, чем просто шаг и время на основе аудио восприятия при генерации хэша и поиске соответствий. Вместо этого можно добавить больше уникальных характеристик к поиску, время которого, как мы надеемся, сократить, сокращая шансы выпадения ложных соответствий.

В этой работе я реализую элементарную систему снятия аудио отпечатков, протестирую через ряд испытаний, описанных более подробно в разделе тестирования этой работы, будет получен базовый набор результатов для сравнения. Тогда, добавив другие факторы к отпечаткам такие, как темп, громкость или ключ к примеру, я могу запустить тесты снова, чтобы увидеть, есть ли статистически значимое улучшение времени поиска, точности или сокращение ложных соответствий.

4 Проектирование

Требуется выполнить некоторые тесты работы и сравнительные тесты, чтобы проверить, что реализация была верной в данный период времени, и методы разработки на правильность.

Ранее мною не разрабатывалось программное обеспечение для необработанного аудио, т.к. это сложный и низкий уровень. Было принято решение, что первые несколько сравнительных тестов должны быть сделаны, чтобы узнать, как Java соответствует поставленным целям, т.е. как язык обрабатывает аудиоданные и как этим можно было бы управлять этим.

Поиск цифровых аудио данных тоже заслуживает отдельного внимания, т.к. быстродействие стоит тут на первом месте. Для этой цели хорошо подходит известная разработанная на Java распределённая платформа Apache Solr. Одними из главных преимуществ данной технологии являются полнотекстовый поиск, динамическая кластеризация, репликация. Нагрузка на веб-сервера также будет распределена. Детальная архитектура распределённой системы представлена в приложении С.

4.1 Чтение звукового файла

Цифровой аудио формат — формат представления звуковых данных, используемый при цифровой звукозаписи, а также для дальнейшего хранения записанного материала на компьютере и других электронных носителях информации, так называемых звуковых носителях.

Аудиофайл (файл, содержащий звукозапись) — компьютерный файл, состоящий из информации об амплитуде и частоте звука, сохранённой для дальнейшего воспроизведения на компьютере или проигрывателе.

Существуют различные понятия звукового формата.

Формат представления звуковых данных в цифровом виде зависит от способа квантования аналогово-цифровым преобразователем (АЦП). В аудиотехнике в настоящее время наиболее распространены два вида квантования :

- импульсно-кодовая модуляция
- сигма-дельта-модуляция

Зачастую разрядность квантования и частоту дискретизации указывают для различных звуковых устройств записи и воспроизведения как формат представления цифрового звука (24 бита/192 кГц; 16 бит/48 кГц).

Формат файла определяет структуру и особенности представления звуковых данных при хранении на запоминающем устройстве ПК. Для устранения избыточности аудио данных используются аудиокодеки, при помощи которых производится сжатие аудиоданных. Выделяют три группы звуковых форматов файлов:

- аудиоформаты без сжатия, такие как WAV, AIFF
- аудиоформаты со сжатием без потерь (APE, FLAC)
- аудиоформаты, с применением сжатия с потерями (mp3, ogg)

Особняком стоят модульные музыкальные форматы файлов. Созданные синтетически или из семплов заранее записанных живых инструментов, они, в основном, служат для создания современной электронной музыки (MOD). Также сюда можно отнести формат MIDI, который не является звукозаписью, но при этом с помощьюсеквенсора позволяет записывать и воспроизводить музыку, используя определенный набор команд в текстовом виде.

Форматы носителей цифрового звука применяют как для массового распространения звуковых записей (CD, SACD), так и в профессиональной звукозаписи (DAT, минидиск).

Для систем пространственного звучания также можно выделить форматы звука, в основном являющиеся звуковым многоканальным сопровождением к кинофильмам. Такие системы имеют целые семейства форматов от двух крупных конкурирующих компаний Digital Theater Systems Inc. — DTS и Dolby Laboratories Inc. — Dolby Digital.

Также форматом называют количество каналов в системах многоканального звука (5.1; 7.1). Изначально такая система была разработана для кинотеатров, но впоследствии была расширена для систем домашнего кинотеатра.

Поэтому первый сравнительный тест, который я сделал, должен был читать аудиоданные с микрофона. Можно было утверждать, что чтение аудио с файла будет лучшим тестом, я стремился понять, как Java обрабатывает аудио, таким образом, я пошел с самым простым методом, проверкой. Это не заняло у меня много времени, был спрограммирован простой класс Java для установки и чтения из микрофона, всё это требовало минимального поиска и чтения

соответствующей документации языка Java. Чтобы протестировать тот этот класс на правильность работы, я записал 10 секунд с аудио файла, по умолчанию в устройстве в необработанных форматах это 44 100 аудио в секунду. Получившиеся 441 000 аудио файлы я записал в стандарт, скопировал их вручную и вставил необработанные аудиоданные в Microsoft Excel, используя функциональность линейной диаграммы, чтобы сгенерировать волновую диаграмму.

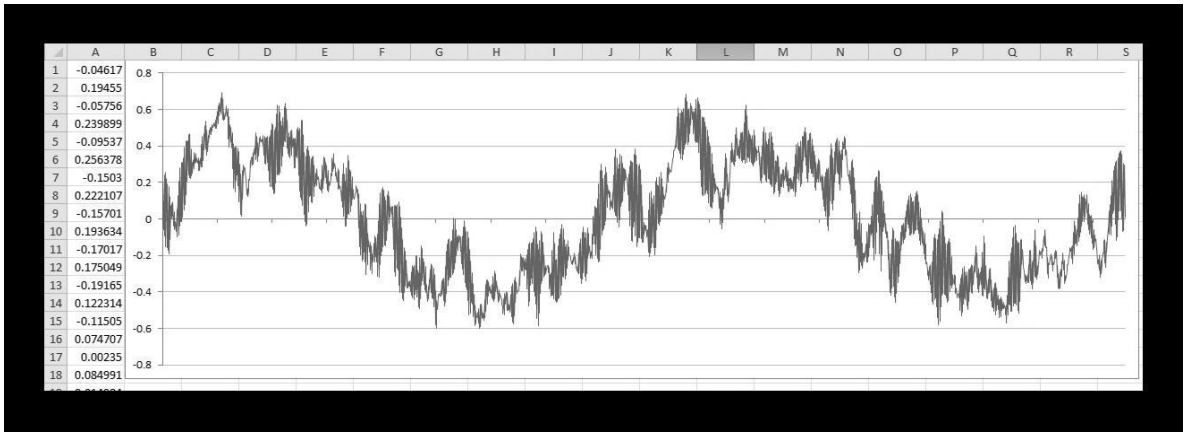


Рисунок 1 – Линейная диаграмма вывода, созданная, используя Microsoft Excel

Согласно графику, показанному на рисунке 1, я понял, что мой код для чтения необработанных аудиоданных работал правильно.

4.2 Спектрограммы

Спектрограмма (сонограмма) — изображение, показывающее зависимость спектральной плотности мощности сигнала от времени. Спектрограммы применяются для идентификации речи, анализа звуков животных, в различных областях музыки, радио- и гидролокации, обработке речи, сейсмологии и в других областях.

Наиболее распространенным представлением спектрограммы является двумерная диаграмма: на горизонтальной оси представлено время, по вертикальной оси — частота; третье измерение с указанием амплитуды на определенной частоте в конкретный момент времени представлено интенсивностью или цветом каждой точки изображения.

Есть много вариантов представления: иногда вертикальная и горизонтальная оси включены так, что время бежит вверх и вниз, иногда амплитуда представлена вершинами в трёхмерном пространстве, а не цветом или интенсивностью. Частота и амплитуда осей может быть линейными или

логарифмическими, в зависимости от того, с какой целью используется график. Аудио обычно может быть представлено с логарифмической осью амплитуды (зачастую, в децибелах или дБ), и частота будет линейной, чтобы подчеркнуть гармонические отношения, или логарифмической, чтобы подчеркнуть музыкальные, тональные отношения.

Спектрограмма обычно создаются одним из двух способов: аппроксимируется, как набор фильтров, полученных из серии полосовых фильтров (это был единственный способ до появления современных методов цифровой обработки сигналов), или рассчитывается по сигналу времени, используя оконное преобразование Фурье. Эти два способа фактически образуют разные квадратичные частотно-временные распределения, но эквивалентны при некоторых условиях.

Метод полосовых фильтров обычно используется в аналоговой обработке для разделения входного сигнала на частотные диапазоны.

Создание спектрограммы с помощью оконного преобразования Фурье обычно выполняется методами цифровой обработки. Производится цифровая выборка данных во временной области. Сигнал разбивается на части, которые, как правило, перекрываются, и затем производится преобразование Фурье, чтобы рассчитать величину частотного спектра для каждой части. Каждая часть соответствует вертикальной линии на изображении — значение амплитуды в зависимости от частоты в каждый момент времени. Спектры или временные графики располагаются рядом на изображении или трёхмерной диаграмме.

Как только я увидел, что считывание аудиоданных с микрофона производится корректно, так сразу же появилась идея для следующего эксперимента это обработка сигнала в виде спектрограммы. Считанные данные с микрофона не полностью полезны в текущем формате, временном интервале, храня только амплитуду сигнала по времени. Вместо этого для разработки системы, мне нужно было узнать, какие частоты присутствуют в сигнале, что обрабатывается. Это важно для снятия отпечатков аудио и в общем очень важно для тестирования работы.

В то время как я мог записать свои собственные объекты и методы, чтобы обработать данные временного интервала преобразования в данные домена частоты, используя наиболее распространенный метод, преобразование Фурье, я решил, что это будет намного более простым, более надежным и более эффективным использованием моего времени, вместо этого использовать существующий класс. Лучшее решение, которое я счел записанным для Java, было от университетов Принстона, основанное на анализе данных.

Как только данные передали через функцию преобразования Фурье, это могло было бы отрисовано также как и волновая форма. Только вместо того, чтобы показать амплитуду волны в течение долгого времени, были показаны величины различных частот для блоков данных временного интервала.

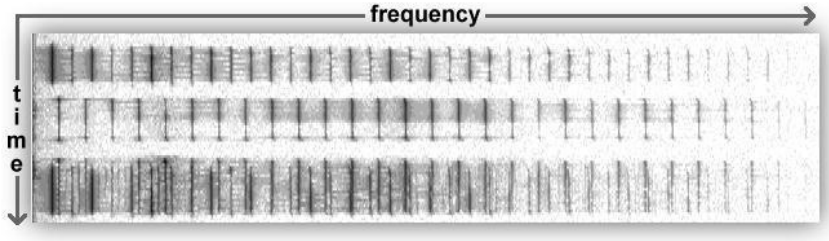


Рисунок 2 – Спектрограмма тонов гармоники

Пример, представленный на рисунке 2 выше (зеркально отражено на 90 ° по часовой стрелке для отображения пространства), показывает, в линейную нелогарифмическую, масштабируемую спектрограмму.

Эта спектрограмма была первой идентификацией, которая у меня появилась, с которой мой код мог работать как без ошибок, и это было схожим со спектрограммами, представленными в статье Эйвери Ли-Чуна, которая была использована мной в качестве научной литературы.

Чтобы продолжить работать со спектрограммами дальше, я сначала попробовал послушать немного реальной музыки, у которой есть отличительные тоны, этой композицией была «Linkin Park – In the end»из альбома «HybridTheory», чтобы увидеть, можно ли использовать фактическое аудио как распознаваемое изображение.

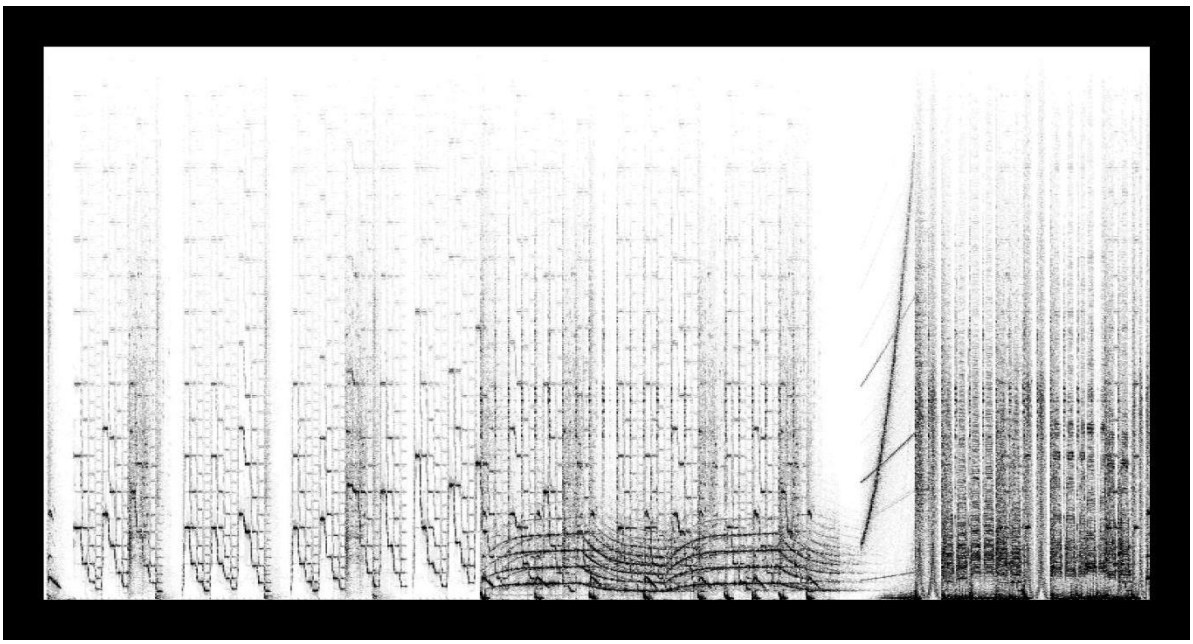


Рисунок 3 – «Linkin Park – In the end»Спектрограмма

Как можно видеть в рисунке 3 выше, у этой дорожки есть несколько, иногда накладываемых, частот, которые изменяют шаг и пик интенсивно время от времени. Это дало мне понять, что решение для снятия отпечатков аудио, посредством чего я вычисляю ключевые пункты от спектрограммы, действительно были бы возможны, поскольку это явно заметно, как музыка составляется из частот и как они изменяются в течение долгого времени.

Чтобы протестировать точность спектрограммы, которая была построена, я решил прокрутить немного музыки, где автор целеустремленно исполнял частоты, показанные в спектрограмме в виде изображений. Это аспект стенографии и в действительности не в рамках этого проекта, но это заверило бы меня, что корректные величины для частот были представлены на спектрограмме.

Эйфекс Твин, электронный музыкант и композитор, выпустил альбом, который называется «Window Licker», на втором треке, которого нарисовано изображение спектрограммы. И трек от «Venetian Shares» под названием «Look» из альбома под названием «Songs about my cats», на котором показаны изображения кошек музыканта.

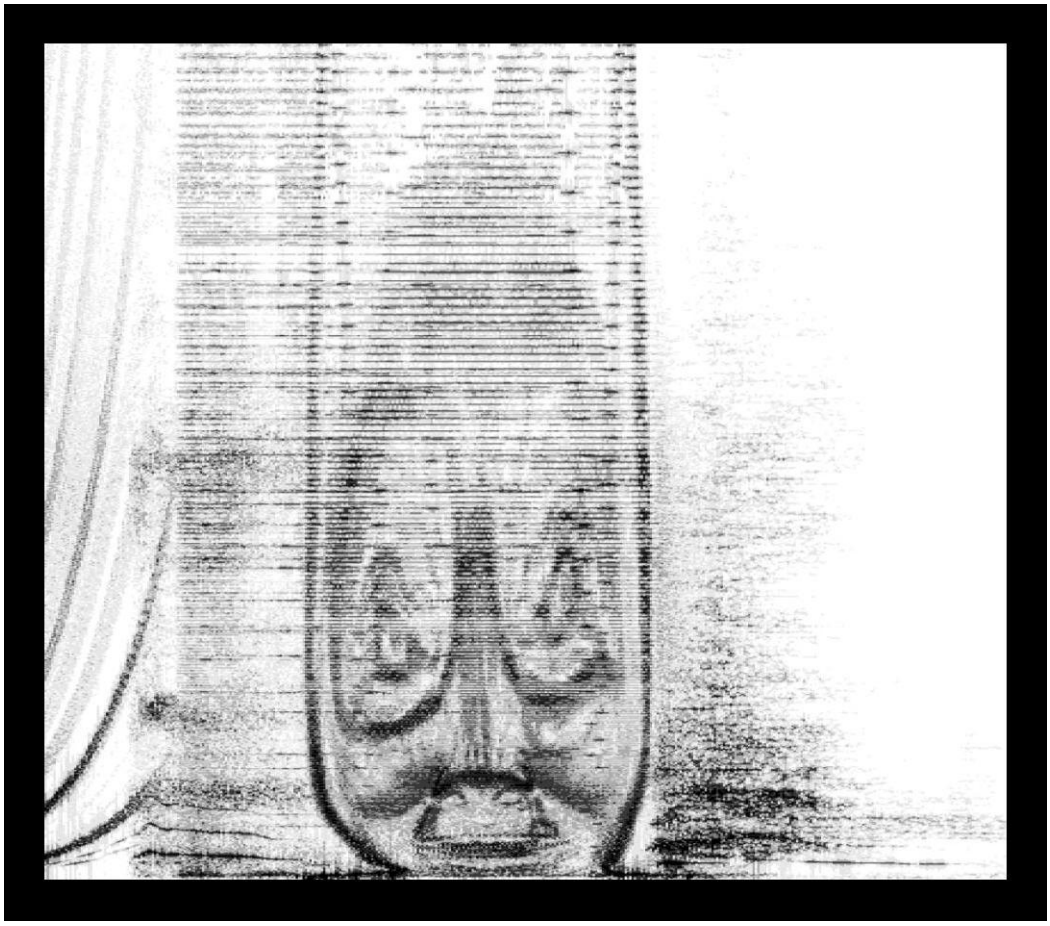


Рисунок 4 – Спектрограмма Эйфекс Твина

Выше представлено лицо Эйфекс Твина на рисунке 4, и оно кажется искаженным. Искажение происходит из-за моей спектрограммы, которая применяет алгоритм, использующий линейную шкалу, и программное обеспечение Эйфекс Твина, используемое, чтобы закодировать изображение его лица в аудио, используя логарифмический масштаб частоты.

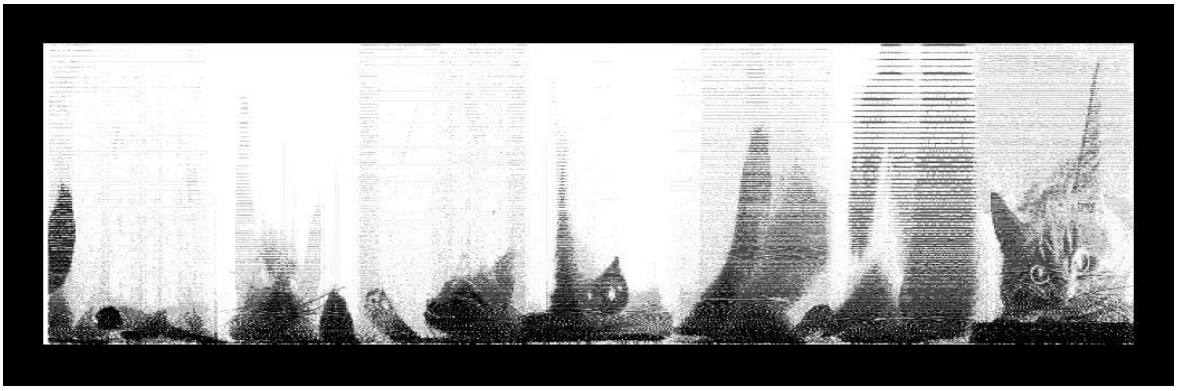


Рисунок 5 - Спектрограмма трека «Venetians Shares»

Есть некоторое небольшое искажение на рисунке 5, выше, на музыкальном треке под названием «Venetians Shares», но это намного менее

примечательно. Кошки легко видимы в первых двух частях, и последних показаны части очень хорошо. Средние две части менее понятны для разбора из-за моего линейного масштаба частоты.

На первый взгляд кажется, что это работа преобразования Фурье и кода рендеринга спектрограммы, но как я и ожидал, существуют некоторые свойственные дефекты. Из-за гармоник в аудио можно получить отражения, показываемые на верхних частях спектрограммы. Однако отражения звука - только половина яркой величины, таким образом, следует пересмотреть алгоритмы обнаружения функции. Кроме того, из-за mp3 формата, большинство музыки не расширяется в верхнюю половину человеческого спектра слушания, и таким образом может быть проигнорировано установлением верхнего предела частоты, находя характеристические точки.

Поскольку функция преобразования Фурье разработана для использования синусоидальных волн, а объёмные волны могут быть определены, используя синусоидальные волны с рядами Фурье, если представить спектрограмму объёмной волны, которая развертывает от 50-25000Hz гармоники синусоидальных волн, которые могут представить объёмную волну, очевидно, как замечено на рисунке 6 изображения спектрограммы.

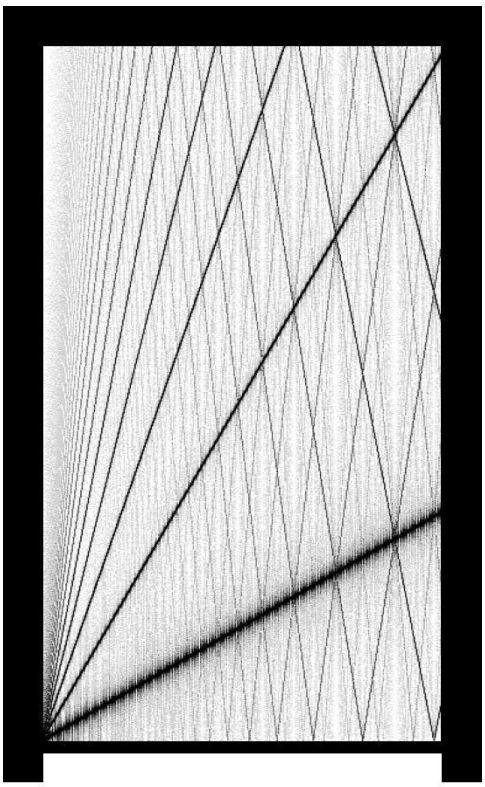


Рисунок 6 - Развертка Прямоугольной волны. Спектрограмма

4.3 Считывание аудио

Чтение аудиоданных с микрофона было относительно простой задачей, но это не достаточно для использования конечного программного обеспечения, которое я хотел разработать. Вместо этого я хотел генерировать различные версии файла с переменными уровнями ухудшения, чтобы использовать в качестве моих данных тестирования. В то время как я, возможно, проигрывал эти файлы через динамики, с микрофон добавлялись дополнительные, не поддающиеся контролю, искажения в тестовые данные. У меня появилась хорошая идея создать код, который мог бы вместо этого считать аудиоданные, напрямую из аудио файлов.

Первоначально я намеревался поддерживать различные аудио форматы файла, поскольку тогда я бы мог в дальнейшем легко сравнить цифровые отпечатки, сгенерированные с аудио файла, сохраненного в форматах "с потерями", как MP3, где сжатие уничтожает часть исходного сигнала, и форматы без потерь, как FLAC, которые хранят целый сигнал. Java делает эту задачу довольно трудной. Вместо этого я акцентировал внимание на API звука java, который мог, предположительно обработать файлы в необработанном волновом формате. Однако, после нескольких дней неопознаваемых форм

волны и ошибок я решил пойти другим путём. Краткий поиск альтернативных способов для считывания волновых файлов привел к классу, разработанному доктором Эндрю Гринстедом. Класс работал, хотя в большой степени был изменён и сокращён в моей реализации, так как много методов, таких как запись, не использовались. Я также изменил класс, чтобы расширить интерфейс аудио класса для чтения так, что следует добавить поддержку для других аудио форматов, для этого нужно просто расширить остальную часть программного обеспечения.

4.4 Графический интерфейс

Интерфейс пользователя, он же пользовательский интерфейс (UI — англ. user interface) — разновидность интерфейсов, в котором одна сторона представлена человеком (пользователем), другая — машиной/устройством. Представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными, чаще всего сложными, машинами, устройствами и аппаратурой.

Весьма часто термин применяется по отношению к компьютерным программам, однако под ним может подразумеваться набор средств, методов и правил взаимодействия любой системы, управляемой человеком.

Несколько широко распространённых примеров:

- меню на экране телевизора + пульт дистанционного управления;
- дисплей электронного аппарата (автомагнитолы, часов) + набор кнопок и переключателей для настройки;
- приборная панель (автомобиля, самолёта) + рычаги управления.

Интерфейс двунаправленный (интерактивный) — когда устройство, получив команды от пользователя и исполнив их, выдаёт информацию пользователю наличествующими у неё средствами — визуальными, звуковыми, тактильными и т. п. (приняв которую, пользователь выдаёт устройству последующие команды предоставленными в его распоряжение средствами: кнопки, переключатели, регуляторы, сенсоры, голосом, и т. д.).

Поскольку интерфейс есть совокупность, то есть он состоит из элементов, которые, сами по себе, также могут состоять из элементов (так, экран дисплея может содержать в себе другие окна, которые, в свою очередь, могут содержать панели, кнопки и прочие интерфейсные элементы).

Особое и отдельное внимание в интерфейсе пользователя традиционно уделяется его эффективности и удобству

пользования (юзабельности). Понятный, удобный, дружественный — его основные характеристики.

Под совокупностью средств и методов интерфейса пользователя подразумеваются:

Средства:

- вывода информации из устройства к пользователю — весь доступный диапазон воздействий на организм человека (зрительных, слуховых, тактильных, обонятельных и т.д.) — экраны (дисплеи, проекторы) и лампочки, динамики, зуммеры и сирены, вибромоторы и т.д. и т.п.
- ввода информации/команд пользователем в устройство — множество всевозможных устройств для контроля состояния человека — кнопки, переключатели, потенциометры, датчики положения и движения, сервоприводы, жесты лицом и руками, даже съём мозговой активности пользователя.

По наличию тех или иных средств ввода, интерфейсы разделяются на типы — жестовый, голосовой, брэйн, и т.д., возможны смешанные варианты. Средства эти должны быть необходимыми и достаточными, быть удобными и практичными, расположенными и скомпонованными разумно и понятно, соответствовать физиологии человека, не должны приводить к негативным последствиям для организма пользователя (всё это входит в понятие эргономики).

Методы:

- набор правил, заложенных разработчиком устройства, согласно которым совокупность действий пользователя должна привести к необходимой реакции устройства и выполнения требуемой задачи — т. н. логический интерфейс
  Правила эти должны быть достаточно ясны для понимания, естественны и легки для запоминания (всё это входит в понятие юзабилити)

Увеличение в устройстве (при равной функциональности) средств ввода-вывода даёт упрощение построения методов управления и упрощение правил пользования, но зато приводит к сложности восприятия информации пользователем — интерфейс становится *перегруженным*. И наоборот — уменьшение средств отображения и контроля приводит к усложнению правил управления — каждый элемент несёт на себе слишком много функций. Потому проектировщики интерфейсов стараются принять компромиссное решение между этими двумя крайностями в каждом отдельном случае.

Безопасность одним из основных направлений исследований в области обеспечения безопасности пользовательских интерфейсов, и, в частности, визуальных интерфейсов пользователя, является разработка моделей информационной безопасности при условии комплексного учета информационных, функциональных, психофизиологических и экологических аспектов безопасности. Это связано, прежде всего, с включением информационного фактора в состав факторов среды систем человек-компьютер и информационным характером почти всех происходящих в области распространения ИП процессов. Наименее разработанным областям проблематики защиты информации в системе человек-компьютер (СЧК) соответствуют такие угрозы, как:

- искажение воспринимаемой пользователем информации за счет ее зашумления источниками среды на рабочем месте пользователя;
- потеря или искажение воспринимаемой пользователем информации из-за физической, семантической или синтаксической несогласованности ее представления пользователю;
- искажение представлений пользователя о реальном состоянии объекта управления за счет скрытых информационных воздействий и неадекватное принятие им решений в процессе решения задач в рамках СЧК.

Мое предыдущее использование Java состояло в том, что нужно было разработать систему сервлета, у которой не было никакого пользовательского интерфейса кроме интерфейса веб-подключения. Я также хотел использовать IDE Eclipse, который не идет ни с какими встроенными, стандартными средствами разработки GUI. В то время как я мог записать код для генерированияGUI вручную, как разработчик GUI Java я решил, что будет легче иметь мгновенную визуальную обратную связь и некоторый визуальный инструментарий, чтобы легко разработать и управлять интерфейсом. Поиск в Интернете выдал Jigloo как лучший плагинEclipse для SWING, java API должен обеспечить графические интерфейсы пользователя для программ java.

4.5 Базы данных

База данных — представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ).

Многие специалисты указывают на распространённую ошибку, состоящую в некорректном использовании термина «база данных» вместо термина «система управления базами данных», и указывают на необходимость различения этих понятий.

В литературе предлагается множество определений понятия «база данных», отражающих скорее субъективное мнение тех или иных авторов, однако общепризнанная единая формулировка отсутствует.

Определения из международных стандартов:

- База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.
- База данных — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, причём такое собрание данных, которое поддерживает одну или более областей применения.

Определения из авторитетных монографий:

- База данных — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.
- База данных — некоторый набор перманентных (постоянно хранимых) данных, используемых прикладными программными системами какого-либо предприятия.
- База данных — совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

В определениях наиболее часто (явно или неявно) присутствуют следующие отличительные признаки:

1. БД хранится и обрабатывается в вычислительной системе.
   Таким образом, любые внекомпьютерные хранилища информации (архивы, библиотеки, картотеки и т. п.) базами данных не являются.
2. Данные в БД логически структурированы (систематизированы) с целью обеспечения возможности их эффективного поиска и обработки в вычислительной системе.
   Структурированность подразумевает явное выделение составных частей (элементов), связей между ними, а также типизацию элементов и связей,

при которой с типом элемента (связи) соотносится определённая семантика и допустимые операции.

3. БД включает схему, или метаданные, описывающие логическую структуру БД в формальном виде (в соответствии с некоторой метамоделью).
«Постоянные данные в среде базы данных включают в себя схему и базу данных. Схема включает в себя описания содержания, структуры и ограничений целостности, используемые для создания и поддержки базы данных. База данных включает в себя набор постоянных данных, определённых с помощью схемы. Система управления данными использует определения данных в схеме для обеспечения доступа и управления доступом к данным в базе данных».

Из перечисленных признаков только первый является строгим, а другие допускают различные трактовки и различные степени оценки. Можно лишь установить некоторую степень соответствия требованиям к БД.

В такой ситуации не последнюю роль играет общепринятая практика. В соответствии с ней, например, не называют базами данных файловые архивы, Интернет-порталы или электронные таблицы, несмотря на то, что они в некоторой степени обладают признаками БД. Принято считать, что эта степень в большинстве случаев недостаточна (хотя могут быть исключения).

Мой единственный предыдущий опыт с Java и базами данных был между Java и Oracle, в то время как я работал над одним проектом. Oracle был допустимой системой базы данных, я мог использовать его для хранения цифровых отпечатков, но я решил, что издержки, требуемые для системы базы данных Oracle, были слишком большими. Для разработки данного проекта я остановился наMySQL и Tokyo Tyrant, соображения детального проектирования представлены в главе детального проектирования позже в этой работе.

У Java есть драйвер подключения к MySQL, JDBC, и к счастью кто-то реализовал перекрестный API java платформы для Tokyo Tyrant, который я смог использовать.

Чтобы протестировать библиотеку, java и производительность MySQL, я записал тестовый класс в java, чтобы создать базу данных с одной таблицей в, вставить строки со словами, затем сделать выборку поддерживаемых строк. В этот обработанный первый раз и было относительно просто выполнить выборку с драйвером подключения. Основанный на этом тесте я нашел библиотеку удовлетворенной мои потребности, и я мог продолжить с разработкой.

4.6 Считывание отпечатков

Для проверки того, что я обладаю всем необходимым для разработки алгоритма в java, который генерирует цифровой отпечаток, я решил провести несколько тестов на выборку, и смог найти характерные точки в аудио и увидеть, как легко можно было сравнить их с базой данных других точек.

Чтобы сделать этот эксперимент, я скопировал класс рендеринга спектрограммы и изменил это, чтобы вывести на экран точки характеристик для более позднего сравнения. Чтобы идентифицировать характеристические точки, я начал с простой соединенной высокой системой считывания, где данные домена частоты разделяются на группы, или полосы, частот. Каждая полоса циклично ищет частоту с самой высокой величиной. Эту наибольшую частоту величины считают устойчивой и она становится моей характеристической точкой.

Упрощение более простого метода, чем метод в устойчивой системе Shazam, где он проверяет всё вокруг характеристической точки, чтобы устранить шум, затрудняющий распознавание, воплотило производительный метод считывания, работавший хорошо, и получил примерно подобные результаты.

Чтобы протестировать, были ли характеристические точки сопоставимы после искажения, я сохранял два образа спектрограмм, показывая характеристические точки. Сообщения, обрабатывающие эти изображения, могли быть объединены, используя инструмент редактирования изображения, чтобы удалить все точки кроме характерных, и наложить их таким способом, чтобы проверить точность. Пример дается в рисунке 7, где красные пиксели - исходный файл, синий ухудшенный файл, и зеленый то, где они накладываются точно. В идеальном виде это должны быть только зеленые пиксели, но можно видеть, как появляется шум, где характерные точки находятся в полосах, где не было основной функции прежде.

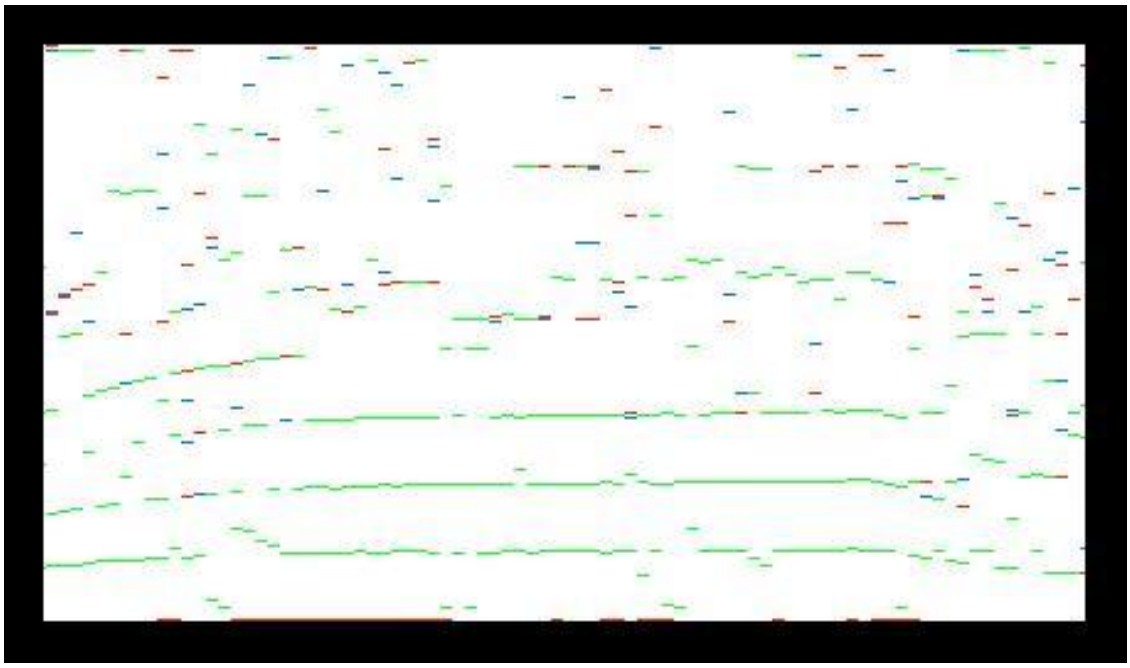


Рисунок 7 – Сравнение характерных точек

4.7 Другие факторы

Так же как метод для считывая отпечатков аудио, я также нуждался в дополнительном методе снятия отпечатков для различия другие факторов от аудио. Используя темп, ключ и громкость вне контекста этой работы так вместо того, чтобы разработать это непосредственно я использовал известный, надежный сторонний API от EchoNest, к которому я мог получить доступ через библиотеку java.

В первый раз я услышал об EchoNest, когда посетил видео одной конференции в Лондоне, названной лондонским музыкальным днем взлома, где разработчики собираются для разработки маленьких проектов, которые имеют отношение к музыке. Было опубликовано описание конференции и демонстрация некоторых из функций их API, которые я записал для данного проекта.

После этого, я продолжил тестирования своей системы, чтобы проверить дополнительные собранные мной данные на полезность и точность.

Результаты проверки заняли долгое время, и, наконец, я получил информацию о музыкальном размере, темпе, ключе, громкости как показано в рисунке 8 ниже.

```
send-err-> http://developer.echonest.com/api/v4/track/profile?md5=d47cc300523097582c1e83fdc0600e18&api_key=VZK0SNBGSEUD84U8S
recv-err-> {"response":{"status":{"message":"The Identifier specified does not exist: d47cc300523097582c1e83fdc0600e18","code":5,"ve
Waiting
Waiting
Waiting
Waiting
 ====== TRVUQZP12F971DE09C =======
Title   : WONDERBOY
audio   : null
foreign : null
Analysis: https://echonest-analysis.s3.amazonaws.com:443/TR/TRVUQZP12F971DE09C/3/full.json?Signature=IDkUd7FRf3A0nTz%2FhCtdnENYGDA%3
Artist  : Tenacious D
MD5     : null
Duration: 30.07111
Key     : 7
Loudness: -7.66
Mode    : 1
Preview : null
Release : null
Status  : COMPLETE
Tempo   : 100.189
Time Sig: 4
```

Рисунок 8 - Анализ Вложенного множества Эха

Я использовал дополнительные метаданные для улучшения моего алгоритма поиска соответствий.

5 Детальное проектирование

5.1 Технологии разработки

Чтобы разработать все программное обеспечение для этого проекта, я решил использовать язык программирования Java вместе с IDE Eclipse.

Когда я выбирал язык разработки для этого проекта, я начал с составления списка языков, с которыми я уже работал и оценил их согласно моему знакомству с ними, пригодностью для цифровой обработки сигналов и доступности сторонних библиотек и общей поддержки. Были выделены 5 языков:

1. Python
2. Java
3. C++
4. Ruby
5. C

Из этого списка я игнорировал Ruby потому что, не было достаточного кол-ва библиотек для облегчения процесса обработки аудиоданных. В то время как это можно было скомпилировать на нижний уровень в собственный код, это не полностью надежно и не является действительно решением проблемы. Кроме того, много библиотек, имеющих дело с Ruby и аудиоданными, пишется в C, так использование Ruby было бы главным образом бессмысленно кроме

части высокоуровневой поддержки, которую предлагает язык.

Ruby (англ. ruby — рубин, произносится ['ru:bɪ] —) — динамический, рефлективный, интерпретируемыйвысокоуровневый язык программирования для быстрого и удобного объектно-ориентированного программирования. Язык обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, сборщиком мусора и многими другими возможностями. Ruby близок по особенностям синтаксиса к языкам Perl и Eiffel, по объектно-ориентированному подходу — к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и Клу.

Кроссплатформенная реализация интерпретатора языка является полностью свободной[5].

C++ имеет в наличии большую поддержку и библиотеки, и может быть применён в разработке кросс-платформенной системы.

C++ — компилируемый статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так инизкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Python - один из моих любимых языков, и действительно имеет поддержку обработки аудиоданных.

Python (МФА: [□pʌ ɪ θ(ə)n]; в русском языке распространено название) — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное,функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Питоне организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализации интерпретаторов для JVM (с возможностью компиляции), MSIL (с возможностью компиляции), LLVM и других. Проект PyPy предлагает реализацию Питона на самом Питоне, что уменьшает затраты на изменения языка и постановку экспериментов над новыми возможностями.

Python — активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют стандарт ANSI, ISO или другие официальные стандарты, их роль выполняет CPython.

Казалось, что язык Смог быть лучшим выбором из-за низкоуровневой поддержки, которая поможет создать оптимизированную обработку

аудиоданных. Однако, часть обработки более высокого уровня, которая будет сделана, такие как рендеринг изображения и графические интерфейсы пользователя, не настолько просто сделать, используя C. На данный момент я уже знаком с языкомC довольно хорошо, но, к сожалению, есть временные ограничения на разработку проекта, поэтому, этот язык не походил на лучший для реализации данной задачи.

Java имеет в наличии большую поддержку и библиотеки, такую же отличную скорость какC, дает легкий доступ к низкоуровневым конструкциям где необходимо обрабатывать высокоуровневый код такой, как рендеринг изображения и графические интерфейсы пользователя, с непринужденностью.Я имел опыт работы с языком Java и мне очень нравится этот язык разработки, поэтому после недолгих раздумываний, я решил использовать в основном Java.

Java — объектно-ориентированный язык программирования, разработанный компанией SunMicrosystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры. Дата официального выпуска — 23 мая 1995 года.

Изначально язык назывался Oak («дуб») разрабатывался Джеймсом Гослингом для программирования бытовых электронных устройств. Впоследствии он был переименован в Java и стал использоваться для написания клиентских приложений и серверного программного обеспечения. Назван в честь марки кофе Java, которая, в свою очередь, получила наименование одноимённого острова (Ява), поэтому на официальной эмблеме языка изображена чашка с парящим кофе. Существует и другая версия происхождения названия языка, связанная с аллюзией на кофе-машину как пример бытового устройства, для программирования которого изначально язык создавался.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия

программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят то, что исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java. В последнее время был внесен ряд усовершенствований, которые несколько увеличили скорость выполнения программ на Java:

- применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде,
- широкое использование платформенно-ориентированного кода (native-код) в стандартных библиотеках,
- аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами фирмы ARM).

- По данным сайта shootout.alioth.debian.org, для семи разных задач время выполнения на Java составляет в среднем в полтора-два раза больше, чем для C/C++, в некоторых случаях Java быстрее, а в отдельных случаях в 7 раз медленнее. С другой стороны, для большинства из них потребление памяти Java-машиной было в 10-30 раз больше, чем программой на C/C++. Также примечательно исследование, проведённое компанией Google, согласно которому отмечается

  примерах на Java в сравнении с аналогичными программами на C++.

- Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине[16]. Эти идеи нашли также выражение в спецификации общеязыковой инфраструктуры CLI, заложенной в основу платформы .NET компанией Microsoft.

Как только я решил использовать Java, я начал задумываться об IDE для разработки моего проекта. Ранее в качестве среды разработки я использовал NetBeans, IntelliJIDEA, Eclipse. Пока я думал о новомIDE, и пошел на низкий уровень с использованием текстового редактора и командной строки, чтобы скомпилировать и отладить мой код, в итоге решил использоватьEclipse, поскольку я нахожу, что у этогоIDE есть все инструменты, в которых я нуждался для проекта, например, контейнеры сервлетов.

Говоря о базах данных, мой проект также требует необычную БД. Те что приходили на ум, были MySQL, Oracle, PostgreSQL или SQLite. MySQL и PostgreSQL и имеют подобные функции и являются оба подходящими.Oracle мог бы быть бесплатным для этого использования, но требует довольно сложной установки. В то время как у Oracle есть большинство функций, у него ничего нет для требований данного проекта. У SQLite есть достаточно многие функции использования в этой системе. Между MySQL, PostgreSQL и SQLite, MySQL выделился как лучший выбор из-за эффективности, доступности библиотеки для java и документации, также была выбрана Tokyo Tyrant для хранения отпечатков, поскольку эта система не требует сложной установки сервера/клиента, емкости для больших объемов соединений и запросов, большого набора данных или высокого параллелизма. Oracle также удовлетворил бы заданию, но он потребовал бы намного большей вычислительной мощности.

MySQL (МФА: [maɪ □ɛ skju□□ɛ l]) — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой SunMicrosystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

26 февраля 2008 года SunMicrosystems приобрела MySQLAB за $1 млрд, 27 января 2010 года Oracle приобрела SunMicrosystems за $7,4 млрд и включила MySQL в свою линейку СУБД.

Сообществом разработчиков MySQL созданы различные ответвления кода, такие как Drizzle (англ.), OurDelta, PerconaServer, и MariaDB. Все эти ответвления уже существовали на момент поглощения компании Sun корпорацией Oracle.

5.2 UML

UML (англ. UnifiedModelingLanguage — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем.UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (англ. generalization), агрегация (англ. aggregation) и поведение) и больше сконцентрироваться на проектировании и архитектуре.

Диаграмма классов (Classdiagram) — статическая структурная диаграмма, описывающая структуру системы, демонстрирующая классы системы, их атрибуты, методы и зависимости между классами.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

- концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;

- точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;
- точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

Диаграмма обзора взаимодействия (Interactionoverviewdiagram) — разновидность диаграммы деятельности, включающая фрагменты диаграммы последовательности и конструкции потока управления.

Этот тип диаграмм включает в себя диаграммы Sequencediagram (диаграммы последовательностей действий) и Collaborationdiagram (диаграммы сотрудничества). Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

Общепринятая методика моделирования программного обеспечения во время стадии проектирования должна иметь схемы UML. Для моей системы есть относительно простая схема варианта использования, поскольку программное обеспечение легко разделяется и не соединяется ни с чем внешним помимо пользователя. На рисунке9 изображена схема варианта использования, показывающая краткий обзор схемы системной функциональности.

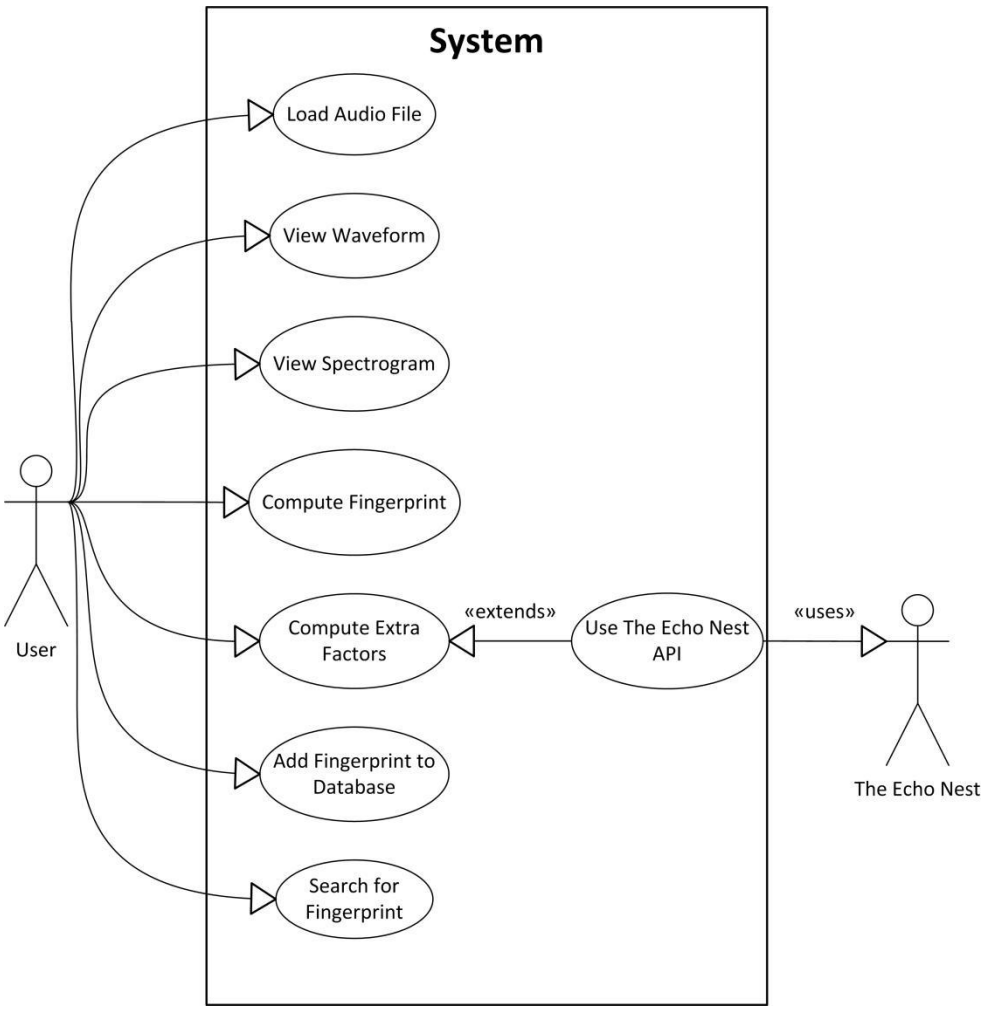


Рисунок 9 – UML – Вариант использования

У UML также есть тип схемы, помогающий разработчикам понять, как процессы взаимодействуют друг с другом, схемой последовательности. Я разработал три из них, чтобы они помогали мне совершенствовать последовательность операций, которые происходят, когда пользователь считывает отпечатки с файла, ищет его в базе данных или сохраняет его в базе данных.

Рисунок 10 на следующей странице отображает схему последовательности, обрисовывающую в общих чертах процесс снятия и для нормальных и для увеличенных цифровых отпечатков.

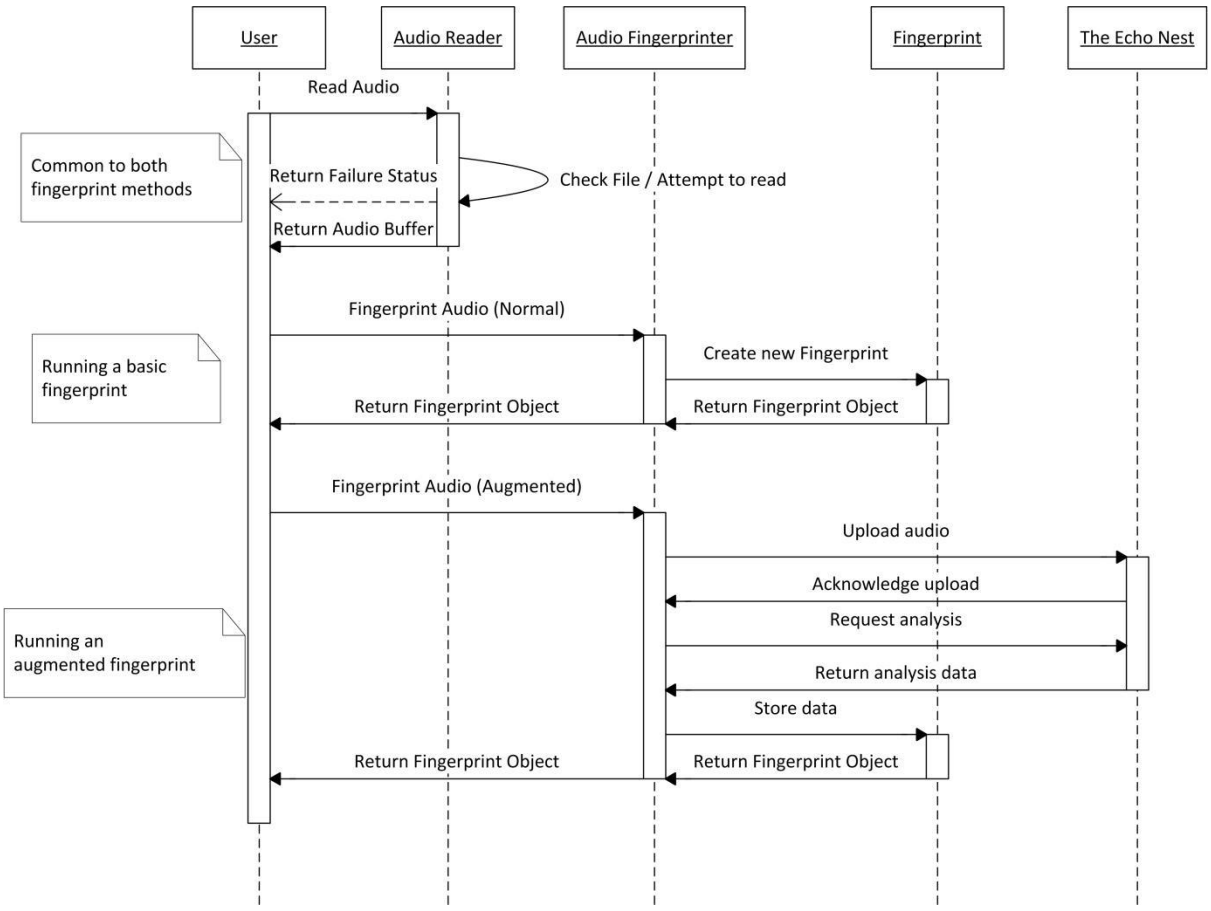


Рисунок 10 - UML - Снятие отпечатков пальцев

Следующая схема, рисунок 11, обрисовывает в общих чертах последовательность операций для сохранения цифрового отпечатка. Это характерно и для нормальных и для увеличенных цифровых отпечатков, поскольку дополнительные данные для увеличенного цифрового отпечатка хранятся наряду с регулярной информацией об аудио.

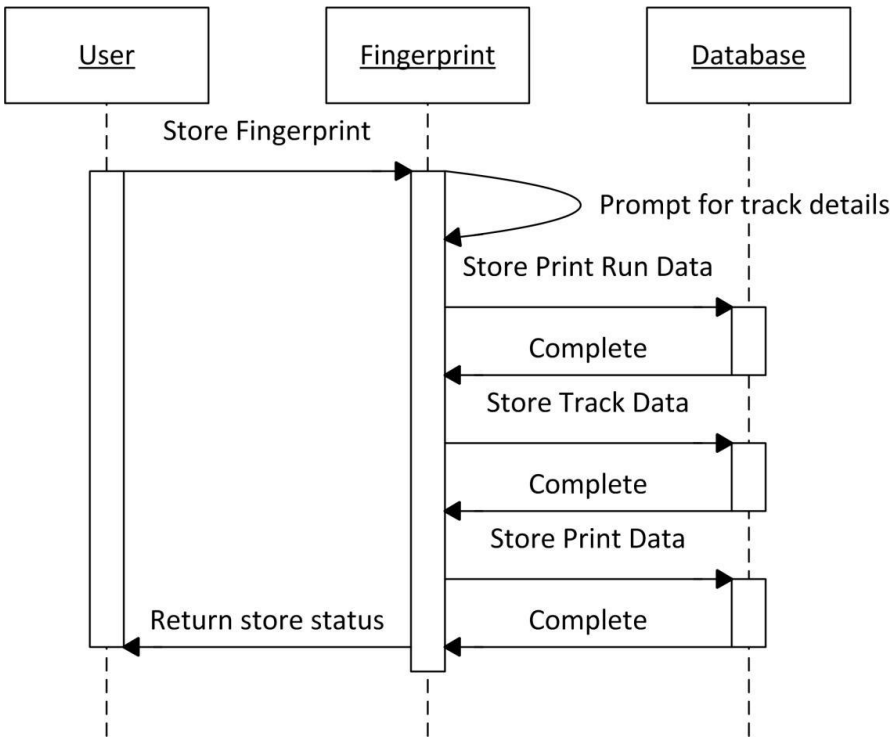


Рисунок 11 - UML - Цифровой отпечаток хранилища

Следующая схема последовательности, рисунок 12, показывает краткий обзор поиска соответствий, предполагая, что объект цифрового отпечатка уже создавался, выполняя процесс цифрового отпечатка.

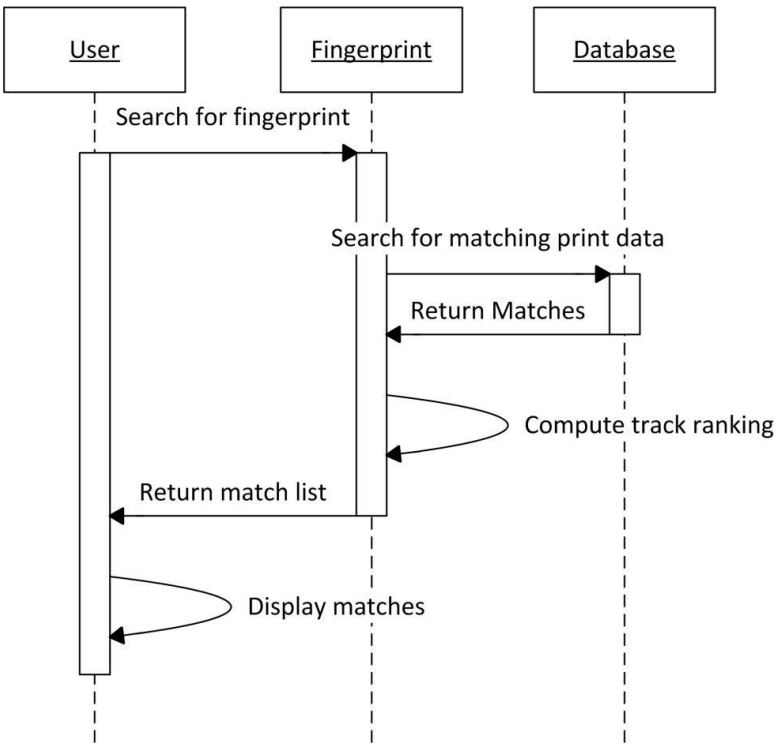


Рисунок 12 - Поиск цифровых отпечатков

Я также сделал рисунок 13, диаграммы классов, обрисовывая в общих чертах классы, методы и элементы каждого класса и как они взаимодействуют друг с другом. Отношения вызываются классом WindowMain, который создает экземпляры(объекты) других классов, но основной фокус вокруг интерфейса цифрового отпечатка, который реализуют классы цифрового отпечатка, fingerprinter классы генерируются и хранятся в главном окне.

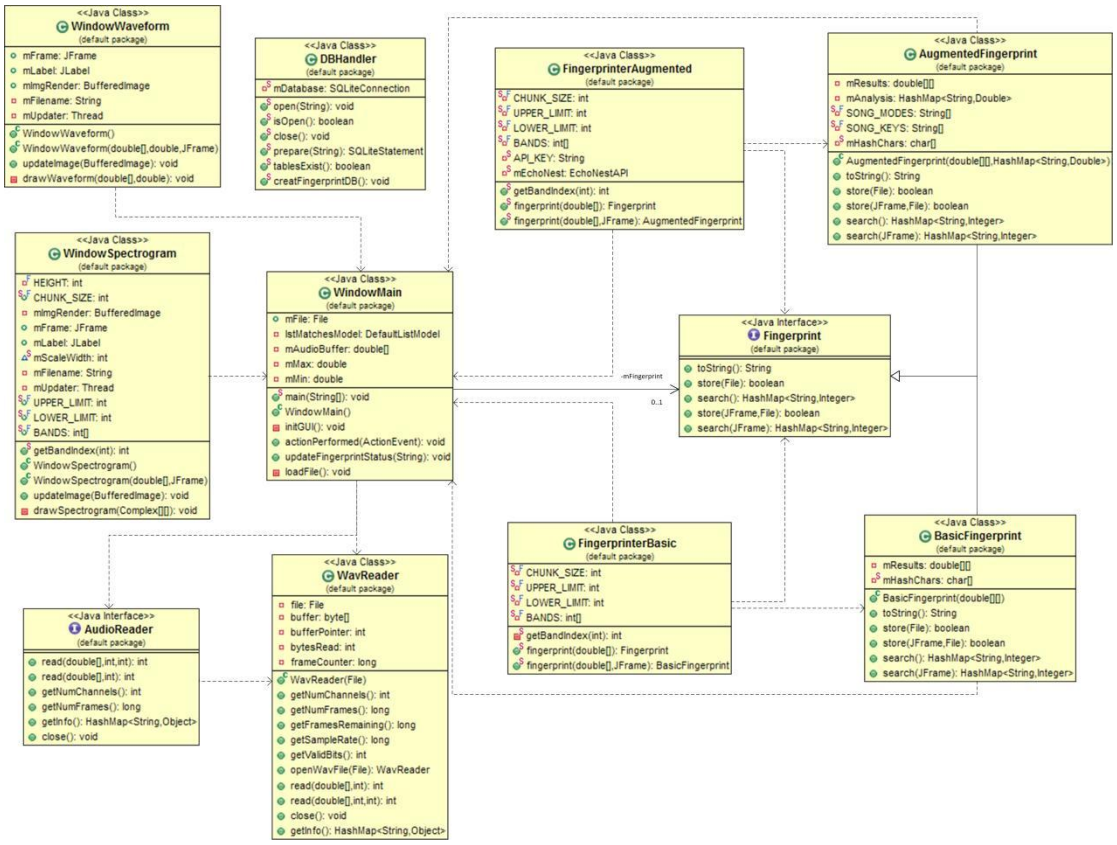


Рисунок 13 - Диаграмма классов

5.3 Юзабилити

Удобство использования, Юзабилити (англ. usability — дословно «возможность использования», «способность быть использованным», «полезность») — понятие в микроэргономике, эргономическая характеристика степени удобности предмета для применения пользователями при достижении определённых целей в некотором контексте. Термин имеет связь с понятием «эргономичность», но в отличие от последнего меньше ассоциируется с технической эстетикой, с внешним видом и более привязан к утилитарности используемого объекта.

Юзабилити — это научно-прикладная дисциплина, занимающаяся повышением эффективности, продуктивности и удобства пользования инструментами деятельности. От эргономики юзабилити отличает заинтересованность в эффективности работы пользователя (потребителя), а не человеко-машинной системы в целом.

Часть программного обеспечения, которое я планирую разработать,

является интерфейсом, через который люди могут видеть цифровой отпечаток и результаты, а также иметь опции для того, чтобы просмотреть волновые формы и спектрограммы. Интерфейс не должен показывать слишком много данных, но в то же время должен быть удобным и понятным.

В юзабилити есть 8 "золотых правил" проектирования интерфейса:

1. Содержательность
2. Быстрый доступ сочетаниями клавиш
3. Обратная связь
4. Диалоговые окна проекта с возможностью закрытия
5. Обработка ошибок
6. Легкое реверсирование действий
7. «Хлебные крошки»
8. Быстродействие (быстрая загрузка)

Из-за немногих средств управления несложно реализовать быстрый доступ сочетаниями клавиш. Можно добавить сочетание клавиш такое, как «Ctrl+O» для того, чтобы загрузить файл, это бы облегчило и ускорило процесс заливки для частых пользователей системы.

Информативная обратная связь будет выведена на экран везде, где необходимо. Когда делаешь такие задачи, как рендеринг спектрограммы или добавление данных цифрового отпечатка к базе данных, может потребоваться больше времени, чем пользователь может ожидать. Нужно вывести на экран текущий статус и сообщить пользователю базового процесса через панель заголовка окна или метку на форме, таким образом пользователь будет иметь информацию о статусе выполняемого действия.

Диалоговые окна должны уведомлять о закрытие или предупреждать пользователя через окно предупреждений, когда действие завершено или обновился интерфейс. В случае визуальных окон представления аудио, волновой формы и рисунка спектрограммы, я могу обновить представление.

Обработки ошибок будут сделаны повсюду. Использования javas try/catch структуру, я могу зафиксировать любые ошибки, которые действительно возникают.

Поскольку единственной вещью, которая будет изменена пользователем, является база данных, когда цифровой отпечаток сохранен в ней, нет никаких функций, которые могли бы реверсировать событие.

Местоположение в управлении остается исключительно в

пользовательских руках в этой системе. Убирается только контроль времени, когда интерфейс занят, используя всю вычислительную мощность, чтобы вычислить операции с аудиоданными. В остальных случаях у пользователя есть полный контроль.

Вся информация, которая могла бы быть запрошена пользователем для этой системы, будет выведена на экран на главном окне. Тем самым элементы такие, как расположение файла, с которого в настоящий момент берут отпечатки, выводятся на экран.

Прототипный пользовательский дизайн интерфейса для главного окна на рис. 14 и 15. Я решил, что волновая форма, и окна спектрограммы не нуждаются ни в каком проекте с точки зрения юзабилити, поскольку не было никаких средств управления, только отображение.

Рисунок 14 - HCI - Главное окно

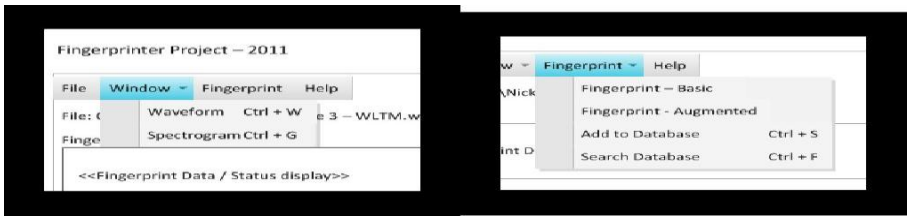


Рисунок 15 - HCI - Меню

5.4 Проектирование базы данных

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области» (например, «концептуальная модель базы данных» и «концептуальная модель предметной области»), поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности.

Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

Чаще всего концептуальная модель базы данных включает в себя:

- описание информационных объектов или понятий предметной области и связей между ними.
- описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Физическое проектирование — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя

ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т.п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т.д.

Модель «сущность-связь» (англ. “Entity-Relationship model”), или ER-модель, предложенная П. Ченом в 1976 г., является наиболее известным представителем класса семантических (концептуальных, инфологических) моделей предметной области. ER-модель обычно представляется в графической форме, с использованием оригинальной нотации П. Чена, называемой ER-диаграмма, либо с использованием других графических нотаций (Crow'sFoot, Information Engineering и др.).

Основные преимущества ER-моделей:

- наглядность;
- модели позволяют проектировать базы данных с большим количеством объектов и атрибутов;
- ER-модели реализованы во многих системах автоматизированного проектирования баз данных (например, ERWin).

Основные элементы ER-моделей:

- объекты (сущности);
- атрибуты объектов;
- связи между объектами.

Сущность — объект предметной области, имеющий атрибуты.

Связь между сущностями характеризуется:

- типом связи (1:1, 1:N, N:M);
- классом принадлежности. Класс может быть обязательным и необязательным. Если каждый экземпляр сущности участвует в связи, то класс принадлежности — обязательный, иначе — необязательный.

Семантическая модель (концептуальная модель, инфологическая модель) – модель предметной области, предназначенная для представления семантики предметной области на самом высоком уровне абстракции. Это означает, что устранена или минимизирована необходимость использовать понятия «низкого

уровня», связанные со спецификой физического представления и хранения данных.

Проектирование баз данных для данного проекта было относительно просто. Все, что должно быть сделано это сохранение ряда данных для цифрового отпечатка и привязывание данных к аудио, где я мог бы получить детали об исходном аудио. Я также решил добавить к таблице дополнительную информацию о цифровом отпечатке, такую как путь к файлу.

Для метода снятия отпечатков пальцев, который я решил реализовать, цифровой отпечаток будет сохранен как максимальная частота в диапазоне частот. Таким образом, таблица print_data хранит значения для каждой полосы, число блоков, таким образом, я могу искать последовательные части цифрового отпечатка, идентификатор дорожки и могу связать эту таблицу с таблицей дорожек и print_run_id, и получу цифровой отпечаток.

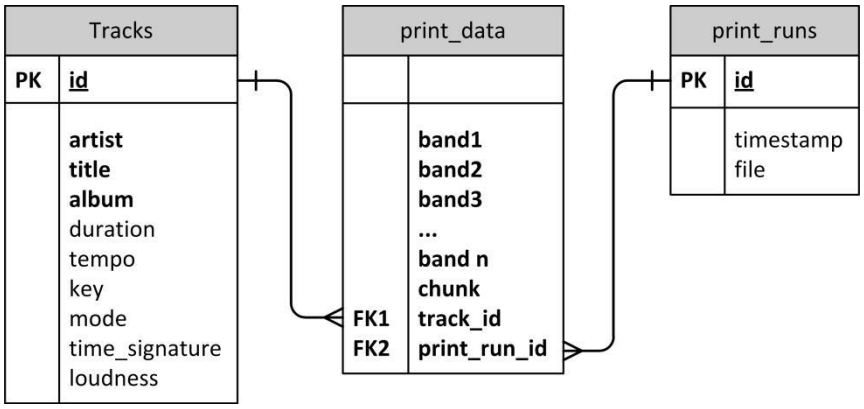


Рисунок 17 - Схема отношений объектов

Рисунок 17 - Схема отношений объектов. SQL скрипт для создания базы данных я не включал в данную работу, но можно использовать БД в приложенном ПО на диске.

5.5 Сторонние библиотеки

Чтобы проанализировать аудио и рассчитать другие факторы, необходимые для увеличенного цифрового отпечатка, у меня было два пути. Первый - нужно было разработать аналитический код самостоятельно, который был бы единственным и его было бы достаточно, чтобы гарантировать один или несколько заключительных годовых отчетов по проекту. Проанализировав трудоёмкость первого способа, я решил следовать другим путём - использование сторонней библиотеки.

Было малое количество библиотек, публично доступных для данного вида аудио анализа доступных, был небольшой класс VAMP, написанный в Лондонском университете. VAMP является немного более техническим и имеет определенные аудио аналитические плагины, есть только обертки для C++ и Python, и я решил создать библиотеку на java, служащую оберткой. Другие библиотеки, имея меньше фокуса на аудио анализе, действительно предлагают некоторые аналитические функции с помощью их собственных API серверов. Поэтому, эти библиотеки не подошли, т.к. были сторонними аналитической службами.
!

5.6 Процесс разработки

Чтобы разработать программное обеспечение для этого проекта, я смотрел на различные процессы разработки, чтобы узнать, будет ли какой-либо из них подходящим. Почти все стандартные методологии разработки, такие как водопад или v-процесс, разрабатываются с многочисленными командами разработчиков, основанных на пользователе требований и процедур развертывания в памяти. У моего проекта не было никаких требований пользователя получить однако кроме моих собственных целей. Я был также единственным разработчиком, таким образом, я не должен был запланировать заранее, чтобы удостовериться, что люди не ожидали на моем коде, который будет закончен или зарегистрирован к репозитарию исходного кода. Поэтому большая часть универсальной модели Жизненного цикла Разработки программного обеспечения; Сбор требований, Проект, Сборка, Тест и Реализация, не применялся ко мне.

Вместо того, чтобы идти для простого, но часто подверженный ошибкам, маршрут Сборки и Фиксируют модель, я вместо этого решил разработать использование метода Extreme Programming. Экстремальное программирование было как создатель Кент Бек, сказанный в его книге “Объясненное Экстремальное программирование: Изменение Объятия”, “*разработанный, чтобы адресовать определенные потребности разработки программного обеспечения, проводимой малочисленными командами перед лицом неопределенных и изменяющихся требований*”(ПРИВЕТСТВИЕ, Кент, 1995). Это удовлетворяло моим потребностям как, в то время как проект делается и известные требования, в то время как я прогрессирую через проект, я ожидаю много изменений к требованиям и буду нуждаться в методе разработки, который позволяет мне сохранять быстрым и гибким.

В статье, написанной Microsoft, они выделяют это “*В Экстремальном программировании, вместо того, чтобы разработать всю систему в начале*

проекта, предварительная дизайнерская работа уменьшается до решения простых задач, которые были уже идентифицированы”(MICROSOFTCORPORATION, 2005). Они также отмечают, что Экстремальное программирование должно использоваться, когда разработчики делают маленькие, инкрементные выпуски с процессом очень разработки через тестирование, работают с новыми технологиями и имеют маленький размер команды. Все из которых подходят мне и этому проекту.

6 Реализация

Чтобы разработать программное обеспечение, я разделял работу на несколько ключевых задач, базируя мои решения о результатах моего экспериментирования ранее и после принципалов разработки, размеченных моделью разработки Экстремального программирования, которая призывает к маленьким, инкрементным выпускам, которые запускаются, создавая ядро системы и разворачивая это, чтобы удовлетворить требования, поскольку это разрабатывается.

6.1 Аудио считыватели

Первая задача состояла в том, чтобы разработать устойчивую систему для того, чтобы считать аудио файлы в. В то время как я только хотел реализовать чтение wav файлов для этого проекта, я знал, что у меня могло бы быть время, чтобы развернуть систему позже, чтобы поддерживать другие форматы. С расширением и работа будущего в памяти я решил создать интерфейс под названием AudioReader. Этот интерфейс определяет общепринятые методики, которые были бы обязаны читать любой аудио формат, такой как чтение в буфер, получение информации о файле и закрытии его. Поскольку разработка класса чтения в wav данных файла не была в пределах проекта, и обеспеченный java кода не работал достоверно, я использовал сторонний класс (GREENSTED, доктор Эндрю, 2010), который был изменен, чтобы реализовать интерфейс, который я определил. Я отключал много перегруженных методов в третьем классе части, которого я не потребовал, и изменил тех, которых оставили соответствовать моим требованиям интерфейсов. Я также взял класс исключений, что wav используемый класс чтения и сделал его универсальным так, чтобы любые будущие аудио классы чтения могли выдать исключения таким же образом и быть пойманы кодом, который использовал его.

Как только аудио чтение было реализовано и протестировано, заставляя это возвратить амплитудные данные стандарту, следующая задача состояла в том, чтобы повторно реализовать дисплей спектрограммы и форма волны. В то

время как эти дисплеи не требуются для работы системы, они обеспечивают очень полезную информацию о работах системы и являются большим представлением, отлаживая методы снятия отпечатков пальцев позже.

6.2 Обзор графика волн

Окно формы волны вызывают из главного окна, быть переданным, ссылкой, буфер, содержащий аудиоданные и максимальную амплитуду всех выборок. В то время как я должен, возможно, передавать только регистрационный номер и делать буферизованное чтение в подпрограмме рисования формы волны, чтобы сократить долгосрочное использование памяти, было более просто сделать это этот путь к маленьким аудио сэмплам, с которыми я имею дело.

Окно формы волны тогда создает новое окно и вызывает подпрограмму ничьей. Подпрограмма ничьей устанавливает объект Graphics2D, который используется, чтобы потянуть к буферизованному изображению, которое помещается в JLabel в окне JFrame. Подпрограмма ничьей тогда, после обнаружения, сколько выборок должно быть принято на вертикальную строку пространства окна, начинает тянуть. Для каждой вертикали 1 пиксельная строка в изображении окно амплитудных выборок берутся и, проведенные между ними. В то время как циклы работают, чтобы потянуть данные на буферизованный объект изображения, новая ветвь дискуссии инстанцировали, который вызывает метод обновления каждые 50 мс. Метод обновления помещает буферизованное изображение в JLabel, очищает JFrame и затем помещает JLabel в это и затем вызывает перерисовку JFrame. Это делает для дисплея в реальном времени, поскольку форма волны оттягивается. Рисунок 18 - пример окна формы волны, выводящего на экран форму волны в течение первых 30 секунд.

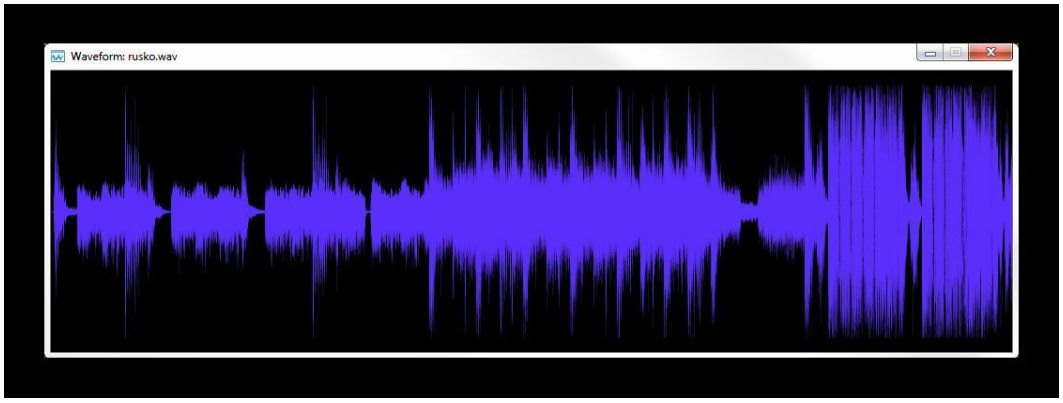


Рисунок 18 - Окно Формы волны

6.3 Окно спектрограммы

Окно спектрограммы работает почти таким же способом, только прежде, чем вызвать функцию ничьей, это преобразовывает амплитудные данные временного интервала в данные частоты домена частоты. Чтобы сделать это, это заполняет буфер объектов комплексного числа, сложный класс, прибывающий из (SEDGEWICK, Роберт и Уэйн, Кевин, 2011), который может тогда быть выполнен через функцию FFT, класс FFT, также прибывающий из (SEDGEWICK, Роберт и Уэйн, Кевин, 2011). С блоками данных теперь в домене частоты, 2dimensional массив данных частоты передают к функции ничьей. Метод ничьей устанавливает объект Graphics2D снова, но на сей раз циклы через каждый блок обработанных данных. Для каждого блока это пробегает каждую частоту, чтобы узнать характерные точки, которые классы снятия отпечатков пальцев будут позже базироваться вокруг. Затем это тянет 5x1 пиксельный прямоугольник для каждой группы частоты в блоке на буферизованное изображение, используя или полную красноту в качестве цвета для характерной точки, темно-зеленого цвета для границы диапазона частот или синюю насыщенность относительно величины частоты. Рисунок 19 - окно спектрограммы в течение первых 30 секунд.

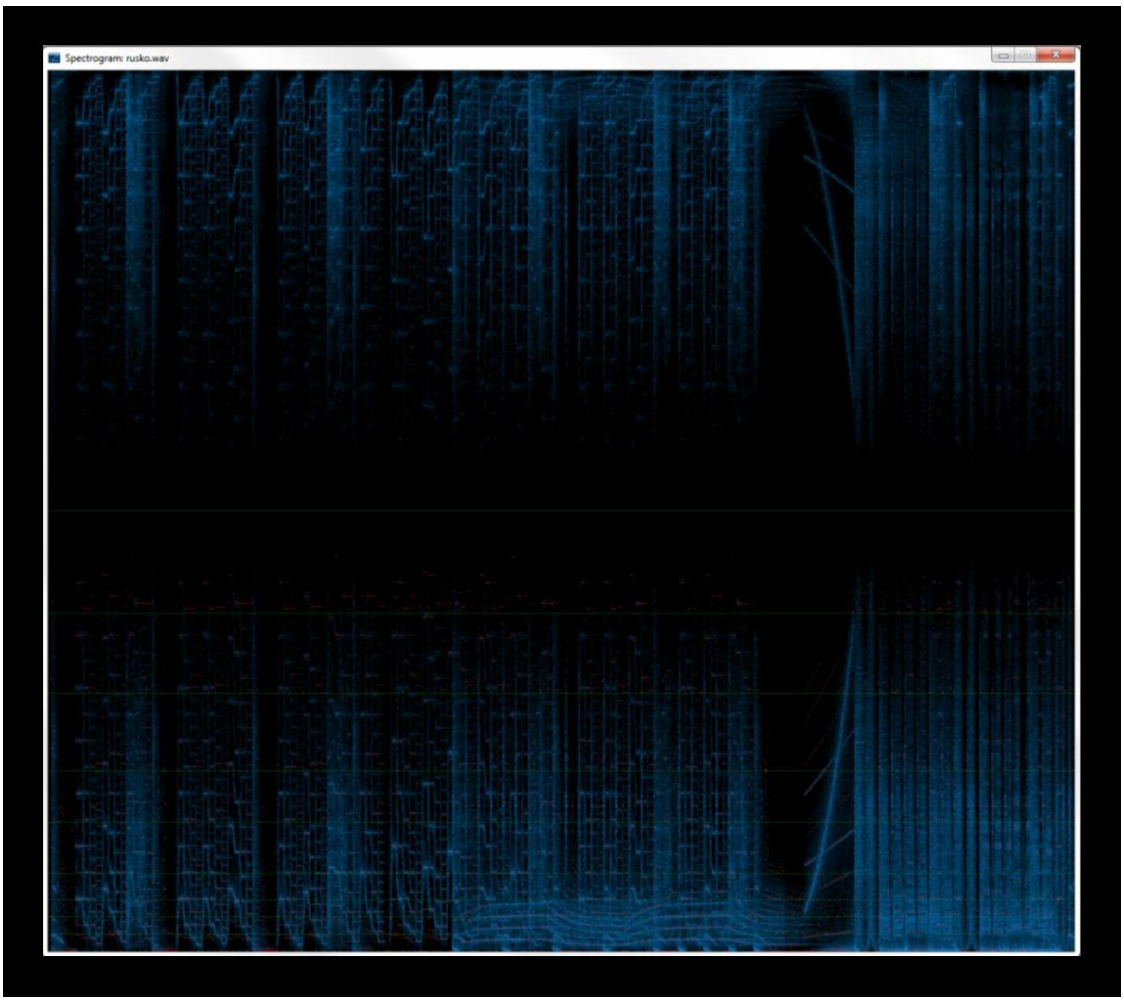


Рисунок 19 - Окно Спектрограммы

Увеличенный масштаб раздел от вышеупомянутой спектрограммы, показывая частоты от аудио подробно так же как выделяя характерные точки в красном показывают ниже в рисунке 20.

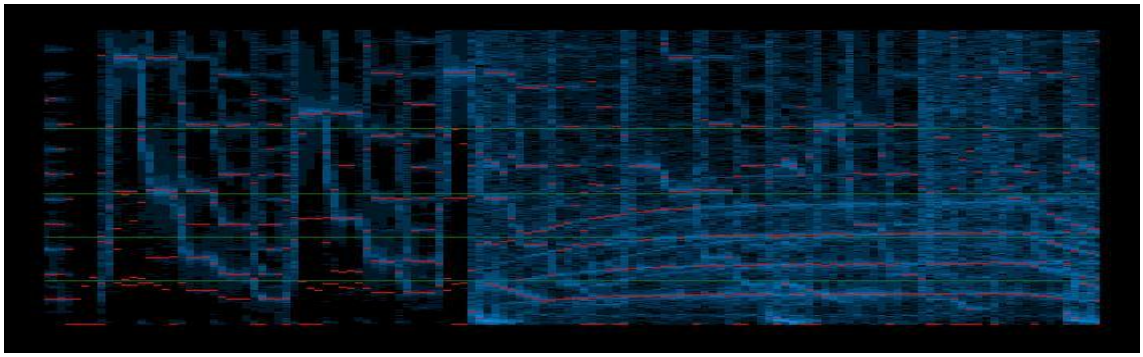


Рисунок 20 - Увеличенный масштаб Спектрограмма

6.4 Снятие аудио отпечатков

Теперь у меня была вся интерфейсная работа, разработанная, следующей задачей сделать был начальный алгоритм снятия отпечатков пальцев. Чтобы сделать это, я не боролся за совершенный алгоритм снятия отпечатков пальцев, я был сразу после довольно устойчивого и простого метода, который мог идентифицировать аудио. Чтобы сделать это, я ссылался на свое исследование от литературы, рассматривают и видел, что все методы, найденные характерными точками и, базировали свои цифровые отпечатки вокруг тех. Самый простой метод, чтобы найти характерные точки был после разделения аудио буфера в блоки и выполнение их через код FFT, разделите блоки на полосы частот. При наличии нескольких полос частоты я мог стать стандартным, и, мы надеемся, повторимым, функции, найденные на выборках. Для каждой полосы я нахожу частоту с самой высокой величиной и беру это в качестве моей характерной точки. В то время как это может быть восприимчиво к ошибке, когда шум происходит на ранее пустой полосе, я добавил в более низком пределе величины, чтобы попытаться уменьшить эту проблему.

С характерными точками, сохраненными, поскольку, частота оценивает на полосу за каждый блок, я, возможно, использовал метод Shazam использования характерной точки как точка привязки и вычисление векторов между этим и другими характерными точками, но вместо этого я решил только сохранить значения частоты на полосу для каждого блока. Это - неэффективная база данных, мудрая, поскольку это означает хранить многократные значения на блок. Это более просто, однако, и позвольте мне видеть точно, что происходило, поскольку цифровой отпечаток создавался. Это также делает алгоритмы поиска немного более простыми, хотя это будет означать более медленные поиски, поскольку есть больше данных, чтобы искать. Чтобы реализовать это в коде, я решил создать класс, FingerprinterBasic.java со статическим методом, чтобы генерировать цифровой отпечаток так не было никакой потребности рассмотреть параллелизм, как система может только брать отпечатки пальцев у одного файла за один раз. Метод цифрового отпечатка разделяет аудио буфер на блоки, вычисляет FFT и находит характерные точки для полос, которые могут тогда быть добавлены к массиву результатов. Этот массив тогда передают конструктору класса BasicFingerprint.java. Класс BasicFingerprint реализует общий интерфейс для методов цифрового отпечатка, которые совместно используют основные и увеличенные классы цифрового отпечатка. Это хранит результаты для цифрового отпечатка и содержит методы, чтобы вывести на экран цифровой отпечаток как строку, искать цифровой отпечаток в базе данных или сохранить цифровой отпечаток в базе данных.

6.5 Хранение и поиск цифрового отпечатка

Функция хранилища в классе BasicFingerprint может тогда быть вызвана на экземпляре цифрового отпечатка, который сохранен в главном окне после снятия отпечатков пальцев. Чтобы сохранить цифровой отпечаток, пользователь запрашивается детали дорожки; если пользователь вводит детали и не щелкает по отмене, функция тогда создает новую строку в print_runs таблице, чтобы сохранить расположение файла, которое является будучи бравшимся отпечатки пальцев наряду со временем. Затем это хранит новую строку в таблице дорожки, чтобы сохранить художника дорожки, заголовок и название альбома. Как только дорожка устанавливается, она может запустить цикличное выполнение через каждый блок данных результата и сохранить значения для каждой полосы в столбце, который будет разыскиваться позже. Если никакие ошибки SQL не бросаются и пользователь, предоставленный деталям дорожки, истинное значение возвращается функцией, чтобы позволить главному окну знать, что хранилище было успешно.

Поисковая функция в классе BasicFingerprint может также быть вызвана, как только цифровой отпечаток был сгенерирован. Поисковый реализованный метод очень сыр в настоящий момент, но работает, удивительно хорошо рассматривая. Это циклично выполняется через каждый блок данных цифрового отпечатка, просматривая все строки в print_data таблице, у которых есть значения в пределах определенных пределов каждой полосы. Если какой-либо блок соответствует один от базы данных, это получает track_id соответствующей строки и постепенно увеличивает значение в карте счета. Карта счета тогда позже выполняется с помощью итераций через, чтобы получить детали дорожки и заполнить карту хеша, которую она может возвратить главному окну для дисплея.

Теперь, когда у меня было это берущий отпечатки пальцев у файлов, я копировал классы FingerprinterBasic и BasicFingerprint, переименовал основной к увеличенному и добавил код, чтобы увеличить цифровой отпечаток. Чтобы сделать это, я добавил блок кода до конца метода цифрового отпечатка в FingerprinterAugmented.java, который создал новый экземпляр API Вложенного множества Эха и загружает трек к серверам Вложенного множества Эха для анализа. API Вложенного множества Эха также хеширует файл с нормальным хешемMD5, чтобы проверить, нуждается ли файл в загрузке. Если файл был уже загружен основанный на хеше MD5, это использует предыдущие результаты анализа, чтобы сэкономить время загрузки. Как только трек был загружен, или идентифицирован как загруженный ранее, новый экземпляр TrackAnalysis создается, который содержит все значения, которые я использую для того, чтобы увеличить аудио цифровой отпечаток.

Экземпляр TrackAnalysis тогда используется, чтобы создать HashMap, который тогда передают в конструктора AugmentedFingerprint рядом с нормальными результатами цифрового отпечатка так, чтобы AugmentedFingerprint мог позже сохранить и искать использование ставок, которые сделало Вложенное множество Эха. Я сохранил значения в HashMap здесь вместо того, чтобы пройти через ссылку на объект TrackAnalysis, потому что вызовы, чтобы получить аналитические значения на объекте TrackAnalysis идут в серверы Вложенного множества Эха каждый раз. Кэшируя значение на незначительном временем цифровом отпечатке генерируют этап, который я могу сэкономить времени позже, ища цифровой отпечаток.

AugmentedFingerprint.java работает почти таким же способом, как BasicFingerprint.java сделал, поскольку они оба реализуют тот же самый интерфейс Цифрового отпечатка. Увеличенный класс цифрового отпечатка, однако, теперь вставляет все данные, которые это имеет о дорожке, когда это хранит строку в таблице дорожек.

При поиске цифрового отпечатка также проверяется, что увеличенные значения в таблице дорожек, связанные с током в строке print_data, в пределах диапазона текущих значений цифровых отпечатков, если метод класса возвращает достаточно высокое значение уверенности в своем анализе. Управление системой счета выполняется таким же образом как и с основным методом поиска цифрового отпечатка и приводит к hashmap информации об аудио для вывода на экран.

6.6 Обработка базы данных

Есть также файл DBHandler.java, включенный в проект. Он обрабатывает открытие, создание и ссылку на базу данных Tokyo Tyrant. Классы и открытие функций вызывается, когда главное окно инициализируется. Это служило общим местом для доступа к базе данных, в которой вызываются классы цифрового отпечатка, с использованием, try/catch.

6.7 Клиентские приложения

В проекте реализованы 3 клиентских приложения. Первое приложение служит для системы в роли API приложения, которое реализует всё API на языке Python, предоставляемое системой, служит, в основном, для снятия отпечатков и внесение этих цифровых аудио данных в БД.

Второй клиент – мобильный клиент, реализованный на C++ для платформы Maemo/Meego (Linux). Основной задачей клиента является запись аудио фрагмента и отправка на сервер. Клиент имеет простой, понятный и удобный пользовательский интерфейс.

Третий клиент – веб-приложение, предоставляющее доступ к сервису через браузеры, выполняет те же функции, что и второй клиент.

7 Тестирование

Чтобы протестировать это, программное обеспечение, которое я разрабатываю, удовлетворяет моим целям, я проведу два набора тестов. Один, чтобы подтвердить мое основное аудио fingerprinter может достоверно идентифицировать аудио, основанное на музыкальных функциях восприятия и втором наборе тестирования, чтобы видеть, есть ли какой-либо эффект, мы надеемся, положительный, при использовании увеличенного цифрового отпечатка, чтобы искать с.

Протестировать программное обеспечение может достоверно идентифицировать аудио, основанное на музыкальных функциях восприятия, я создал серию файлов. Я взял аудио различных типов: Классическая музыка, Dupstep, Металл, Появляется, Чиптюн и Джаз. Каждая часть аудио была получена в высшем качестве, которое я мог найти, часто FLAC (Бесплатный кодек без потерь), и преобразовал это в формат wav. Поскольку wav формат файла без потерь, я не теряю данных из-за сжатия, и тест остается допустимым. После снятия отпечатков пальцев у каждой высококачественной версии файлов и хранения их в моей базе данных я могу тогда приступить к обнаружению пределов распознавания. Я ухудшу файл, добавляя различные объемы статического шума или отфильтровывая определенные частоты, чтобы моделировать различные версии того же самого файла, который могут иметь много людей. Сравнивая цифровой отпечаток ухудшенного файл к той из

основных копий я могу проверить, чтобы видеть, насколько устойчивый и надежный мое программное обеспечение снятия отпечатков пальцев. Так же как записывая количество соответствия блоков я также запишу любые ложные положительные соответствия или файлы, которые почти достигли того, чтобы быть положительной ложью.

Чтобы протестировать эффект добавления в других музыкальных факторах, таких как ключ, громкость или темп в мой цифровой отпечаток, я сравню те же самые файлы и проверю, есть ли какое-либо изменение, вовремя потраченное на поиск соответствий и числа ложных положительных результатов / не соответствия.

7.1 Результаты

Во-первых, чтобы установить базовую линию проекта и иметь что-то, чтобы сравнить мои результаты испытаний с я выполнял все свои тестовые файлы через основной fingerprinter и искал соответствия после только вставки исходных, высококачественных цифровых отпечатков в базу данных. Результаты показывают в рисунке 22.

	Basic Fingerprint												
	Chiptune		Classical		Dubstep		Jazz		Metal		Rock		
	Time	Top Match	Time	Top Match	Time	Top Match	Time	Top Match	Time	Top Match	Time	Top Match	
Clean	2794	7261	2616	32934	2432	2234	2491	12235	2925	3949	2776	4637	
Static 0.2	2461	2152	2642	MISSING	2248	-178	2503	MISSING	2425	2135	2456	MISSING	
Static 0.4	3064	1407	2499	MISSING	2267	-109	2626	MISSING	2466	1618	2504	MISSING	
Static 0.6	2848	869	2415	MISSING	2398	-292	2444	MISSING	2476	1222	2327	MISSING	
Static 0.8	2583	759	2396	MISSING	2397	-250	2327	MISSING	2433	955	2316	MISSING	
HPF 200Hz	2498	7308	3139	31305	2256	2124	2557	10977	2362	3357	2416	4259	
HPF 400Hz	2419	6760	2715	28947	2224	2071	2413	8666	2362	2718	2435	3933	
HPF 600Hz	2476	6114	2578	26146	2267	2007	2381	6791	2417	2395	2466	3566	
HPF 800Hz	2245	5715	2616	23865	2283	1916	2576	5839	2399	2201	2344	3215	
HPF 1000Hz	2513	5327	3295	22391	2192	1790	2349	5316	2395	2076	2463	3163	Overall Avg.
AVG	2590		2691		2296		2467		2466		2450		2493

Рисунок 22 - Основные Результаты Цифрового отпечатка

Данные времена являются миллисекундами, взятыми от пользователя, щелкающего “по Поисковой Базе данных”, пока она не возвратила результат. Для тестирования каждый поиск был сделан 3 раза на поиск, и результат времени был усреднен. Столбец TopMatch показывает, сколько блоков соответствовало, делая поиск. Это число может быть больше исходное соответствие файла, если определенные блоки заканчивают тем, что соответствовали больше чем один блок дорожки в базе данных.

Результатами, отмеченными как без вести пропавшие, являются результаты, где fingerprinter сделал ответные матчи, но ни один из них не был правильным соответствием. Результаты отрицательно от того, где соответствия

были найдены, но главное соответствие было неправильным, различие между намеченным результатом и главным результатом - показанная отрицательная величина.

От вышеупомянутой таблицы можно идентифицировать это, времена были примерно непротиворечивыми через все поиски и только тихое демонстрационное аудио, такими как джаз, классическими и горные выборки, были абсолютно неспособны соответствовать после того, как шум был добавлен.и шумовые тесты в стиле дабстеп вышли отрицательные как каждый раз система снятия отпечатков пальцев, неправильно идентифицированная выборка как выборка чиптюна, только маленьким полем как бы то ни было. Это не является слишком большим из удивления, поскольку обеим дорожкам, используемым в этом тесте, действительно использовали подобные частоты в запуске, но выборка чиптюна использует музыкальный эффект на 8 битов, который походит на грубый, искаженный, низкий бас, который показывает на спектрограмме как полосы белого шума после примечаний сопрано.

Видеть, увеличивая цифровой отпечаток со значениями ключа; например. С, Г, Фа-диез; темп и музыкальный размер, улучшенный точность или улучшенная поисковая скорость, я запускал повторно тесты с теми же самыми выборками и теми же самыми исходными демонстрационными цифровыми отпечатками в базе данных. Результаты вышли как показано в рисунке 23.

	Augmented Fingerprint												
	Chiptune		Classical		Dubstep		Jazz		Metal		Rock		
	Time	Top Match	Time	Top Match	Time	Top Match	Time	Top Match	Time	Top Match	Time	Top Match	
Clean	2568	7261	1709	32934	2496	2234	2239	12235	2241	3949	2257	4637	
Static 0.2	2636	2152	2568	MISSING	MISSING	MISSING	MISSING	MISSING	2363	2135	2434	MISSING	
Static 0.4	2208	1407	MISSING	MISSING	2087	-109	2166	MISSING	2332	1618	2291	MISSING	
Static 0.6	2304	869	2213	MISSING	2174	-292	2056	MISSING	2305	1222	2318	MISSING	
Static 0.8	2287	-115	2174	MISSING	2225	MISSING	2157	MISSING	2354	955	2257	MISSING	
HPF 200Hz	2156	7308	MISSING	MISSING	2061	2124	2188	10977	2140	3357	2564	4259	
HPF 400Hz	2090	6760	MISSING	MISSING	2050	2071	2173	8666	2157	2718	2386	MISSING	
HPF 600Hz	2151	6114	2221	26146	2028	2007	2111	6791	2941	2395	2310	MISSING	
HPF 800Hz	2179	5715	2196	23865	2029	1916	2050	5839	3031	2201	2430	MISSING	
HPF 1000Hz	2097	5327	2211	22391	2083	1790	2159	5316	2071	2076	2215	MISSING	Overall Avg.
AVG	2268		2185		2137		2144		2394		2346		2246

Рисунок 23 - Увеличенные Результаты Цифрового отпечатка

Как можно видеть на первый взгляд по таблице есть намного больше недостающих значений. Это, кажется, противоречит моей идее, что увеличенные значения улучшили бы соответствующую точность. Есть причины этого, как бы то ни было.

Большое количество без вести пропавших значений является результатом погрешности анализа, который Вложенное множество Эха делает вместе со строгостью моего алгоритма поиска. В то время как Вложенное множество Эха действительно возвращает значение уверенности, утверждающее, насколько

точный результат, вероятно, будет с представленным шумом или фильтр высоких частот, вычисление важных значений, таких как ключ, может быть дико извращено. Это неверное истолкование значений, с которыми я увеличиваю цифровой отпечаток тогда, устраняет корректные дорожки из того, чтобы быть соответствующим приводящего к недостающим значениям соответствия, поскольку мой алгоритм поиска принимает увеличивающиеся значения с 50%-ой уверенностью. Это действительно означает однако, что, хотя корректные дорожки отсутствуют в списке результатов, ложные положительные стороны сокращаются также, хотя это не действительно измеримо с размером небольшой выборки.

Однако, действительно кажется, что мои надежды на увеличенный цифровой отпечаток, являющийся быстрее, чтобы искать, были корректны. Хотя среднее полное время составляет только приблизительно 0.02 секунды быстрее в среднем, это было непротиворечивое среднее число. Кроме того, небольшой размер базы данных не отражает, как хорошо увеличенный метод цифрового отпечатка масштабировался бы по сравнению с основным цифровым отпечатком с точки зрения времен поиска. Однако ту гипотезу трудно протестировать точно с этой системой, поскольку SQLite не разрабатывается, чтобы обработать большие наборы данных хорошо.

ЗАКЛЮЧЕНИЕ

Была разработана распределённая система снятия и распознавания отпечатков аудио файлов. В процессе разработки были получены знания и опыт работы с физическими характеристиками аудио. Программное обеспечение было имеет клиент-серверную распределённую архитектуру.

Из результатов я не могу окончательно определить, лучше ли мой увеличенный цифровой отпечаток или хуже по сравнению со стандартным методом снятия отпечатков пальцев. Различие между временем поиска незначительно, и в таких маленьких значениях может легко влиять на другие системные процессы при тестировании. Увеличенный метод снятия цифрового отпечатка не возвращал много ложных соответствий, но также и бывало, что соответствия не были корректными, и этот факт мешал сравнению и оценке метода.

Представленная база данных была разработана для более крупных наборов данных, таким образом время поиска заметно уменьшилось, даже с увеличенными цифровыми отпечатками, добавленными к базе данных. Я пришёл к выводу, что увеличенный цифровой отпечаток был действительно улучшением, нежели неоправданным увеличением данных в БД. Но в настоящий момент, для точной оценки необходимо продолжать тестирование.

Лично я сказал бы, что сокращение ложных соответствий при поиске, даже за счет дополнительного является явным достижением в этой сфере. Система не сообщает ни о каком соответствии, если искажение цифрового отпечатка очень сильное из-за самого снятия отпечатков, которое напрямую влияет на точность. Однако, для улучшения точности я пытался сократить ложные соответствия, но не сокращать сам файл.

Если бы я заново сел за этот проект, я произвел бы несколько изменений в своем проекте и в моем плане тестирования.

В проекте я изменил использование сторонней библиотеки для аудио анализа и изучил реализацию аналитических функций и встроил их в проект путём создания библиотек java, которые обертывают систему плагина ВАМПА, таким образом, я получил более точный анализ и, надеюсь, уменьшил возникновения отсутствия соответствий на фазе тестирования.

Я также добавил опции для улучшения удобства пользования системы, такие как поддержка других форматов файлов и других методов снятия отпечатков. Хорошо поддерживать различные форматы файлов, потому что это

сэкономит время создания данных для тестирования, и позволяет ускорить снятие цифровых отпечатков всей музыки в БД.

На будущее я выделил бы больше времени тому, как я сохранил и искал соответствия в своей реализации цифрового отпечатка. Возможно, переписал бы некоторые компоненты поиска для алгоритма и продолжал бы тестировать способ идентификации устойчивых характеристик.

Список литературы

1. L. Lu, M. Wang, and H.J. Zhang. Repeating pattern discovery and struc-ture analysis from acoustic music data. In Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval, pages 275– 282, New York, NY, 2004. ACM Press.

2. L. Lu and H.J. Zhang. Unsupervised speaker segmentation and tracking in real-time audio content analysis. Multimedia Systems, 10(4):332–343, Apr. 2005.

3. L. Lu, H.J. Zhang, and S.Z. Li. Content-based audio classification and segmentation by using support vector machines. Multimedia Systems, 8(6):482–492, Apr. 2003.

4. C. Lvy, G. Linars, and P. Nocera. Comparison of several acoustic model-ing techniques and decoding algorithms for embedded speech recognition systems. In Proceedings of the Workshop on DSP in Mobile and Vehicular Systems, Nagoya, Japan, Apr. 2003.

5. N. Maddage, C. Xu, M. Kankanhalli, and X. Shao. Content-based music structure analysis with applications to music semantics understanding. In
6. Proceedings of the ACM International Conference on Multimedia, pages 112–119. ACM, 2004.

7. S. Mallat. A wavelet tour of signal processing. Academic Press, San Diego, California, 1999.

8. H. Malvar. A modulated complex lapped transform and its applications to audio processing. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 3, pages 1421–1424, Phoenix, AZ, Mar. 1999. IEEE, IEEE.

9. M.F. McKinney and J. Breebaart. Features for audio and music classifica-tion. In Proceedings of the International Conference on Music Information Retrieval, Oct. 2003.

10. R. Meddis and L. O’Mard. A unitary model of pitch perception. The Journal of the Acoustical Society of America, 102(3):1811–1820, Sep. 1997.

11. A.Meng, P. Ahrendt, and J. Larsen. Improving music genre classification by short time feature integration. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 5, pages 497–500, Philadelphia, Pennsylvania, Mar. 2005. IEEE, IEEE.

12. A.Mesaros, E. Lupu, and C. Rusu. Singing voice features by time-frequency representations. In Proceedings of the International Symposium on Image and Signal Processing and Analysis, volume 1, pages 471–475, Rome, Italy, Sep. 2003. IEEE, IEEE.
13. I.Mierswa and K. Morik. Automatic feature extraction for classifying audio data. Machine Learning Journal, 58(2-3):127–149, Feb. 2005.

14. N. Minematsu, M. Sekiguchi, and K. Hirose. Automatic estimation of one's age with his/her speech based upon acoustic modeling techniques of speakers. In Proceedings of the IEEE International Conference on Acous-tics, Speech, and Signal Processing, volume 1, pages 137–140, Orlando, FL, May 2002. IEEE, IEEE.

15. MIREX. Music information retrieval evaluation exchange. http://www.music-ir.org/ mirexwiki, 2007. last visited: September, 2009.

16. H. Misra, S. Ikbal, H. Bourlard, and H. Hermansky. Spectral entropy based feature for robust asr. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 1, pages 193–196, Montreal, Canada, May 2004. IEEE, IEEE.

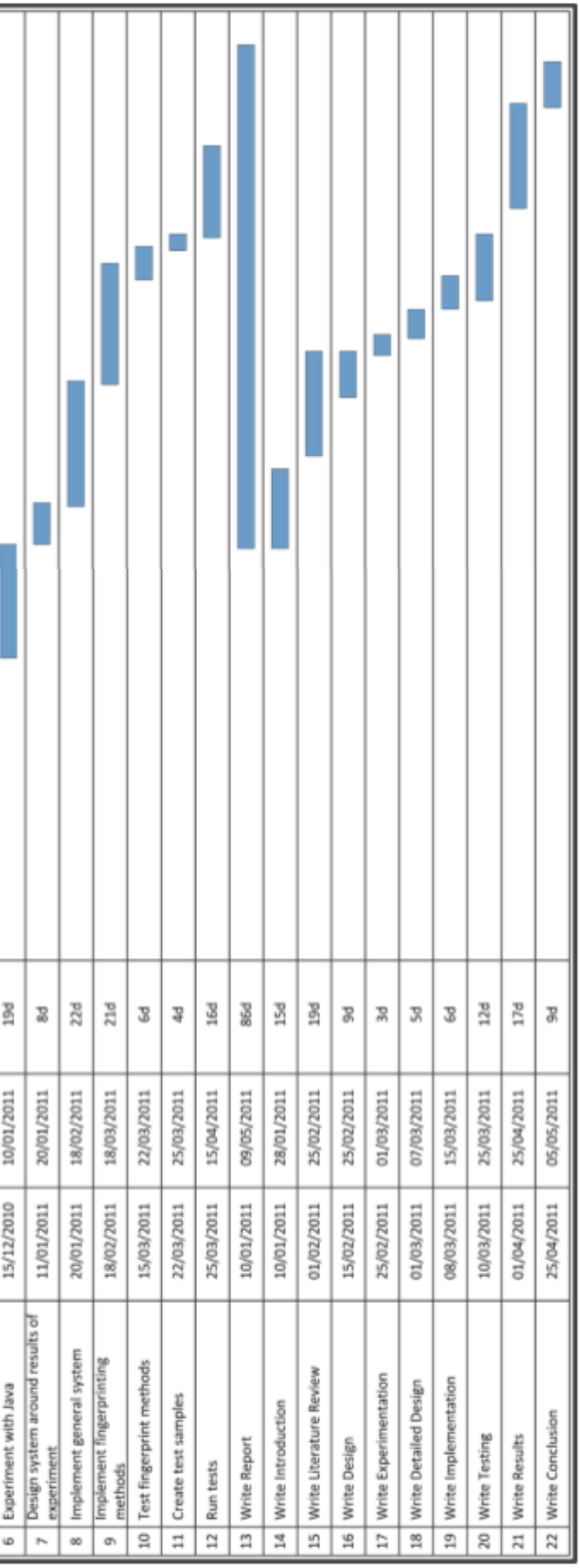
17. H. Misra, S. Ikbal, S. Sivadas, and H. Bourlard. Multi-resolution spectral entropy feature for robust asr. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 1, pages 253–256, Philadelphia, Pennsylvania, Mar. 2005. IEEE, IEEE.

18. D. Mitrovic, M. Zeppelzauer, and C. Breiteneder. Discrimination and retrieval of animal sounds. In Proceedings of IEEE Multimedia Modelling Conference, pages 339–343, Beijing, China, Jan. 2006. IEEE, IEEE.

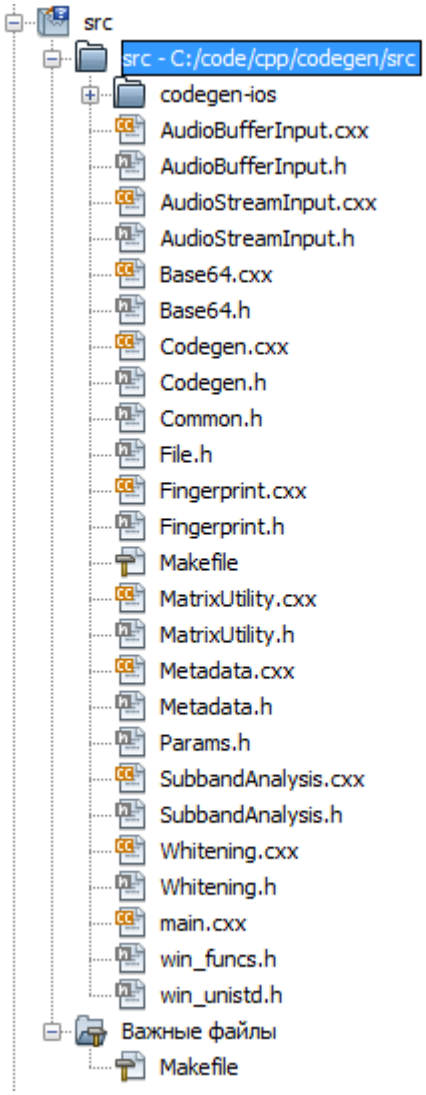
19. B.C.J. Moore. An Introduction to the Psychology of Hearing. Academic Press, Amsterdam, The Netherlands, 5th edition, 2004.

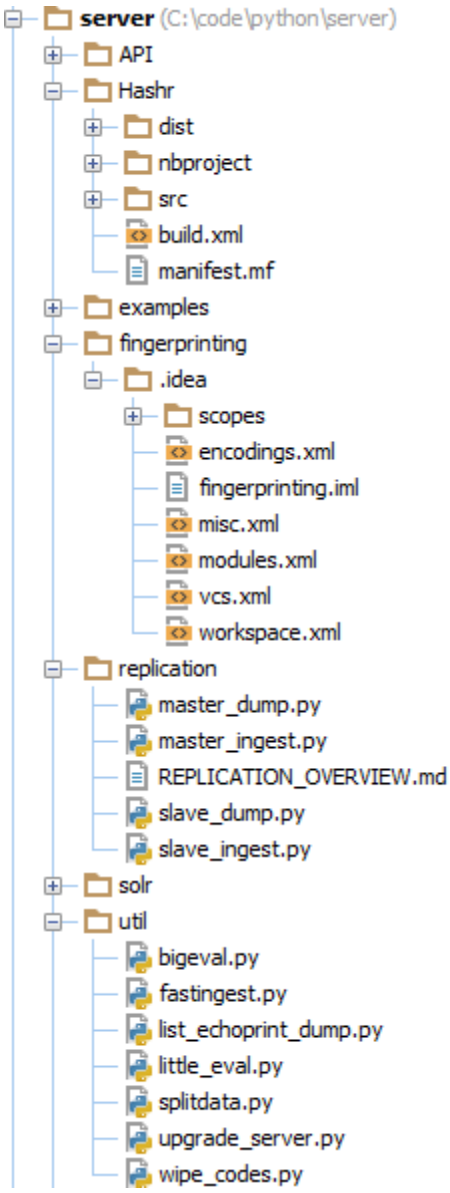
Приложение А – График Ганта

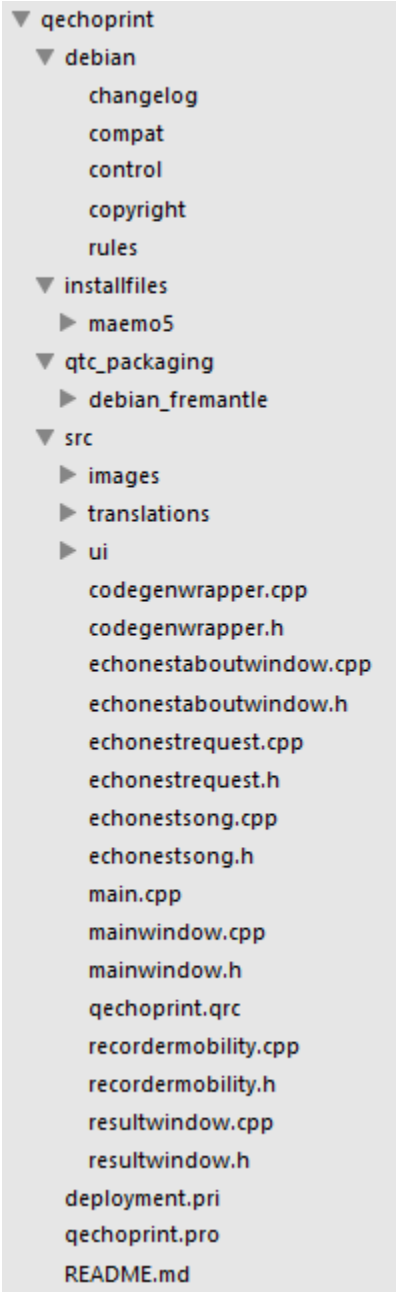
ID	Task Name	Start	Finish	Duration
1	Plan project	04/10/2010	01/11/2010	21d
2	Research general project	04/10/2010	19/11/2010	35d
3	Extended Project Breif	11/10/2010	29/10/2010	15d
4	December Deliverable	29/10/2010	15/12/2010	34d
5	Research fingerprinting methods	22/11/2010	16/12/2010	19d
6	Experiment with Java	15/12/2010	10/01/2011	19d
7	Design system around results of experiment	11/01/2011	20/01/2011	8d
8	Implement general system	20/01/2011	18/02/2011	22d
9	Implement fingerprinting methods	18/02/2011	18/03/2011	21d
10	Test fingerprint methods	15/03/2011	22/03/2011	6d
11	Create test samples	22/03/2011	25/03/2011	4d
12	Run tests	25/03/2011	15/04/2011	16d
13	Write Report	10/01/2011	09/05/2011	86d
14	Write Introduction	10/01/2011	28/01/2011	15d
15	Write Literature Review	01/02/2011	25/02/2011	19d
16	Write Design	15/02/2011	25/02/2011	9d
17	Write Experimentation	25/02/2011	01/03/2011	3d
18	Write Detailed Design	01/03/2011	07/03/2011	5d
19	Write Implementation	08/03/2011	15/03/2011	6d
20	Write Testing	10/03/2011	25/03/2011	12d
21	Write Results	01/04/2011	25/04/2011	17d
22	Write Conclusion	25/04/2011	05/05/2011	9d



Приложение В – IDE, файловая структура




server (C:\code\python\server)
API
Hashr
dist
nbproject
src
build.xml
manifest.mf
examples
fingerprinting
.idea
scopes
encodings.xml
fingerprinting.iml
misc.xml
modules.xml
vcs.xml
workspace.xml
replication
master_dump.py
master_ingest.py
REPLICATION_OVERVIEW.md
slave_dump.py
slave_ingest.py
solr
util
bigeval.py
fastingest.py
list_echoprint_dump.py
little_eval.py
splitdata.py
upgrade_server.py
wipe_codes.py


qechoprint
debian
changelog
compat
control
copyright
rules
installfiles
maemo5
qtc_packaging
debian_fremantle
src
images
translations
ui
codegenwrapper.cpp
codegenwrapper.h
echonestaboutwindow.cpp
echonestaboutwindow.h
echonestrequest.cpp
echonestrequest.h
echonestsong.cpp
echonestsong.h
main.cpp
mainwindow.cpp
mainwindow.h
qechoprint.qrc
recordermobility.cpp
recordermobility.h
resultwindow.cpp
resultwindow.h
deployment.pri
qechoprint.pro
README.md

Приложение С – Архитектура распределённой системы