Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики

К защите допустить:

Заведующая кафедрой
информатики

_____ Н. А. Волорова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему:

СЕРВИС ДЛЯ ИЗВЛЕЧЕНИЯ ХАРАКТЕРИСТИК МУЗЫКАЛЬНОЙ КОМПОЗИЦИИ С ПРИМЕНЕНИЕМ МАШИННОГО ОБУЧЕНИЯ

БГУИР ДП 1-40 01 01 03 097 ПЗ

Студент	Д. В. Шматков
Руководитель	П. С. Саттарова
Консультанты:	
от кафедры Информатики	П. С. Саттарова
по экономической части	К. Р. Литвинович
Нормоконтролёр	Н. Н. Бабенко

Рецензент

Минск 2017

РЕФЕРАТ

Пояснительная записка 40 с., 0 рис., 5 табл., 30 формул, 4 источников.
ПРОГРАММНОЕ СРЕДСТВО, ВЕБ-СЕРВИС, УНИВЕРСИТЕТ, АНАЛИЗ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ, МАШИННОЕ ОБУЧЕНИЕ, ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Цель настоящего дипломного проекта состоит в разработке программной системы, предназначенной для автоматизации извлечения характеристик музыкальных произведений.

В процессе анализа предметной области были выделены основные характеристики музыкальных композиций и способы их извлечения, которые в настоящее время практически не охвачены автоматизацией. Было проведено их исследование и моделирование. Кроме того, рассмотрены существующие средства, разрозненно разрабатываемые и применяемые сотрудниками университетов и отдельными компаниями (так называемые частичные аналоги). Выработаны функциональные и нефункциональные требования.

Была разработана архитектура программной системы, для каждой ее составной части было проведено разграничение реализуемых задач, проектирование, уточнение используемых технологий и собственно разработка. Были выбраны наиболее современные средства разработки, широко применяемые в индустрии.

Полученные в ходе технико-экономического обоснования результаты о прибыли для разработчика, пользователя, уровень рентабельности, а также экономический эффект доказывают целесообразность разработки проекта.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

API – Application Programming Interface (программный интерфейс приложения) [1].

ВВЕДЕНИЕ

Подъем сервисов электронного распространения медиаинформации дал беспрецендентный доступ пользователям к записям музыкальных композиций. Сейчас такие сервисы, как iTunes, Google Music, Яндекс Музыка, Spotify и другие, предоставляют мгновенный доступ к миллионам записей. Перед пользователями возникает проблема выбора следующей композиции для прослушивания. Для решения этой проблемы, а так же для того, чтобы облегчить ориентацию в этом большом количестве информации, сервисы электронного распространения записей предоставляют пользователям системы рекомендации контента.

Современные системы рекомендации контента обычно используют статистику прослушивания композиций пользователями, чтобы сделать рекомендации более точными. Однако на такие системы часто оказывают влияние смещения популярности в больших масштабах, из-за чего системы не могут рекомендовать композиции, которые менее популярны. Так же это затрудняет создание индивидуальных рекомендаций для пользователей, исходя из их предпочтений. Поэтому возникает необходимость извлекать информацию из композиций, а так же классифицировать сами копозиции. На сегодняшний день извлечение характеристик и классификация музыкальных композиций является интересной и соревновательной задачей, которой занимаются различные компании и исследователи. На данный момент сильное развитие получают способы, основанные на машинном обучении.

Целью данного дипломного проекта является создание сервис для извлечения музыкальных характеристик, который будет пригоден для дальнейшей интеграцией с другими системами. В ходе работы предстоит выполнить следующие задачи:

а) выбрать способ извлечения базовых характеристик музыкальных композиций (таких как спектр, ритм, продолжительность);

б) построить модель машинного обучения для излечения высокоуровневых характеристик;

в) собрать и упорядочить данные для обучения модели;

г) построить эффективную архитектуру сервиса;

д) построить API для интеграции с другими системами.

Реализация сервиса для извлечения характеристик музыкальных композиций позволит упростить задачу классификации контента для сервисов электронного распространения записей, а так же создать систему для создания более точных и индивидуальных рекомендаций для пользователей.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Извлечение музыкальной информации - небольшая, но интересная и быстрорастущая область исследований, которая охватывает большой диапазон продуктов, используемых по всему миру. Эта область охватывает несколько дисциплин научных знаний: музыковедение, психологию, обработку сигналов, машинное обучение и комбинации этих дисциплин. Задача, которую будет решать разрабатываемый сервис, относится к задачам извлечения музыкальной информации.

Несмотря на то, что извлечение музыкальной информации является еще небольшой областью исследований, для извлечения характеристик музыкальных приложений применяется достаточное количество методов. Эти методы различаются как на основаниии применяемых подходов, так и на основании уровней извлекаемых характеристик. На основании характеристик выделяют методы, которые извлекают низкоуровневые характеристики (спектр, ритм), и методы которые извлекают высокоуровневые характеристики (жанр, настроение). В основном, нас будут интересовать методы, которые извлекают высокоуровневые характеристики.

Для извлечения высокоуровневых характеристик выделяют два класса методов: основанные на математических функциях, в основе которых лежит машинное обучение. В музыке появляются новые направления, а старые могут изменяться, поэтому наиболее перспективными являются методы, которые способны постраиваться под переменчивую природу музыки. Под такие условия в большей степени подходят методы, в основе которых лежит машинное обучение.

Машинное обучение - класс методов, характерной чертой которых является не прямое решение задачи, а решение, которое строится в процессе обучения при применении решения к множеству сходных задач. В последнее время этот класс методов получил активное развитие и поддержку со стороны крупных компаний и университетов.

Цифровой аудиосигнал можно представить в виде изображения звуковой волны.

ТУТ ДОЛЖЕН БЫТЬ РИСУНОК ЗВУКОВОЙ ВОЛНЫ!!!

Цифровой аудиосигнал представляет собой множество точек, которые расставлены на одинаковом расстоянии друг от друга. Расстояние между точками называется частотой дискретизации. Чем меньше интервал (выше частота дискретизации), тем шире частотный диапазон, который можно закодировать таким образом.

ТУТ ДОЛЖЕН БЫТЬ РИСУНОК ЗВУКОВОЙ ВОЛНЫ В БОЛЬШОМ

РАЗРЕШЕНИИ!!!

Амплитуда колебаний зависит от времени звука и коррелирует с громкостью звука. А частота колебаний напрямую связана с высотой звука. Для того, чтобы получить информацию о частоте колебаний, необходимо применить преобразование Фурье. Преобразование Фурье позволяет разложить периодическую функцию в сумму гармонических с разными частотами. Коэффициенты гармонических функций при сложении будут давать нам те частоты, которые мы хотим получить.

При применении преобразования Фурье ко всей звуковой дорожке мы получим "смазанный"во времени спектр. Для того, чтобы получить спектр, не теряя временной составляющей, необходимо применить к сигналу оконное преобразование Фурье. Оконное преобразование Фурье отличается от обычного тем, что мы делим наш сигнал на короткие отрезки (окна) и применяем к каждому преобразование Фурье. По-сути мы получаем набор спектров - отдельно для каждого отрезка. В результате мы получим картинку, которой можно описать звуковую дорожку.

КАРТИНКА СПЕКТРАЛЬНОЙ ХАРАКТЕРИСТИКИ!!!!

Жанр композиции является высокоуровневой характеристикой. Для определения жанра необходимо разобраться, как человек воспринимает звук.

Ухо человека устроено так: звуковые волны смещают барабанную перепонку при взаимодействии с ней. Вибрации передаются во внутреннее ухо и считываются им. Смещение барабанной перепонки зависит от звукового давления, при том зависимость является не линейной, а логарифмической. Для измерения громкости принято использовать относительную шкалу - уровень звукового давления (измеряется в децибелах). Так же воспринимаемая громкость зависит и от частоты звука. Для оценки громкости звука используется логарифмическая единица измерения - фон. В шкале фонов, в отличие от децибелов, значения громкости связаны с чувствительностью на разных частотах человеческого уха. Частота 1000 Гц является чистым тоном, и уровень фона для неё численно равен уровню в децибелах. Для остальных частот используют поправки, которые представляют собой стандартизированное семейство кривых, называемых изофонами (ISO 226).

РИСУНОК КОНТУР РАВНЫХ ГРОМКОСТЕЙ!!!!

Аналогично с громкостью, частота так же воспринимается человеческим ухом нелинейно. Для измерения воспринимаемой частоты человеческим ухом используется мел-шкала. Шкала основана на статистической обработке субъективного восприятия звука на больших данных. За 1000 мел взят звук с частотой 1000 Гц при уровне громкости 40 фон. За 0 мел взят звук частотой 20 Гц при уровне громкости 40 фон.

РИСУНОК С МЕЛ-ШКАЛОЙ!!!!

2 ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГОВ

Существует ряд сервисов, которые решают задачу извлечения характеристик музыкальных композиций и задачи, достаточно близкие к данной. Рассмотрим наиболее популярные и известные из них.

2.1 The Echo Nest

The Echo Nest - компания из Соммервиля, которая занимается разработкой сервиса для анализа музыкальных композиций и составлении рекомендаций. Их продукт собирает информацию, используя акустический и текстовый анализ. Текстовый анализ состоит в том, что любое упоминание музыкальной композици, которое найдено в интернете, проходит через системы The Echo Nest. Эти системы настроены на то, чтобы извлекать ключевые слова и термины. Этим данным потом присвается собственный вес, который говорит об их важности. Акустический анализ начинается с того, что композиция разбивается на небольшие кусочки. Затем для каждого сегмента определяются громкость, тембр и другие характеристики. Далее полученные данные объединяются и анализируются. Анализ проводится с помощью методов машинного обучения, что позволяет понять композицию на высоком уровне. Объединение акустического и текстового анализа позволило создать мощную технологию, которая сделала компанию мировым лидером в алгоритмах анализа музыки. В 2014 году компанию купил мировой гигант стримминга музыки - Spotify.

2.2 Niland

Niland - компания из Парижа, которая занимается поиском и реккомендациями музыкальных композиций. Их продукт анализирует композицию, используя акустический анализ. Решаемые задачи разделяются на 2 типа: оценка похожести и классификация. Для оценки похожести используется подход music2vec. То есть преобразование композиции в вектор из значений характеристик. Для классификации композиций используется преобразование музыки в вектор для дальнейшей обработки с помощью алгоритмов машинного обучения. Преобразование музыки в вектор включает в себя:

а) вычисление спектрограммы, т.е. вычисление интенсивности различных частот в различные моменты времени;

б) извлечение кратковременных особенностей, т.е. свойств сигнала, которые имеют высокую частоту обновления;

в) моделирование статического распределения кратковременных особенностей и объединение их в вектор.

В результате получается вектор большой размерности, порядка тысячи элементов, который впоследствие используется для классификации композиций и извлечения характеристик высокого уровня.

2.3 Music Technology Group

Music Technology Group - исследовательская группа с факультета информации и коммуникационных технологий университета Pompeu Fabra, Барселона. Специализируется на вычислительных задачах в области музыки и звука. Свои исследования группа основывает на знаниях из других областей, таких, как обработка сигналов, машинное обучение и взаимодействие человека с компьютером. Темы исследований, которыми группа занимается в данный момент:

а) обработка аудиосигналов, т.е. спектральное моделирование для синтеза и трансформации звука;

б) описание звука и музыки, т.е. семантический анализ и классификация аудио;

в) продвинутое взаимодействие с музыкой, т.е. разработка интерфейсов для создания и изучения музыки;

г) звуковые и музыкальные сообщества, т.е. технологии социальных сетей для музкальных и звуковых приложений.

Группа создала библиотеку для определения степени похожести и классификации музыки, для описания музыки с помощью высокоуровневых характеристик - Gaia. Обрабатываются данные в несколько этапов: подготовка данных, обучение на подготовленных данных, проверка результатов. Подготовка данных включает в себя:

а) удаление метаданных;

б) извлечение тональных характеристик;

в) нормализацию выделенных характеристик;

г) преобразование низкоуровневых характеристик в нормально распределенные величины.

Для обучения используется метод опорных векторов с полиномиальным ядром (однородным). В качестве метрики точности используется н-кратная кроссвалидация.

2.4 DeepMind

DeepMind - компания, которая занимается искусственным интеллектом. Ранее она была известна под названием Google DeepMind. Компания занимается решением проблем, связанных с интеллектом, т.е. занимается разработкой обучающихся алгоритмов общего назначения. На данный момент компания сосредоточила свои усилия на разработке интеллекта, который способен играть в компьютерные игры - от стратегических до аркад.

Сандер Дилеман, ученый из этой компании, стал соавтором статьи (ССЫЛКА НА СТАТЬЮ В ЛИТЕРАТУРЕ), в которой утверждается, что глубинное обучение способно гораздо лучше справляться с рекомендациями, чем колаборативная фильтрация. В качестве инструмента использовалась сверточная нейронная сеть с 7-8 слоями. Для визуализации данных использовался алгоритм t-SNE. Алгоритм определял интструменты, аккорды, и даже гармонии и прогрессии.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Требования к проектируемому программному средству

По результатам изучения предметной области, анализа литературных источников и обзора существующих систем-аналогов сформулируем требования к проектируемому программному средству.

Сервис должен предоставлять простой API для извлечения характеристик из аудиозаписей. Для поддержки большего числа пользователей и эффективного процесса интеграции с системами пользователя сервису необходимо поддерживать основные форматы взаимодействия между сервисами:

а) JSON;

б) XML.

Сервис будет являться модулем, который достаточно просто встраивать в существующие системы. Для повышения удобства использования развертывание модуля должно быть простым процессом, который не занимает много времени.

Для подготовки композиций к дальнейшему анализу, а так же для их анализа, в сервисе реализованы сложные математические вычисления. Для того, чтобы повысить комфортность использования, сервису следует анализировать композиции за приемлемое время. Так же необходимо снизить требовательность к характеристикам компонентов вычислительной системы, на которой будет запущен сервис. Поэтому необходимо оптимизировать вычисления, производимые для анализа композиций.

Музыкальные жанры подвержены изменению. Для того, чтобы поддерживать аналитическую составляющую в актуальном состоянии, необходимо иметь возможность обновлять части системы. Поэтому сервис должен быть построен из независимых компонентов, согласованных на уровне интерфейсов.

Аудиозаписи могут храниться в различных форматах, поэтому необходимо реализовать поддержку обработки основных форматов распростанения цифровых аудиозаписей, таких как mp3, wav, flac, m4a.

Для предоставления как можно более полной информации об аудиозаписи извлекаемые характеристики должны быть различных уровней, т.е. высокоуровневые и низкоуровневые. Определим основные характеристики композиции для извлечения:

а) длина;

б) громкость;

в) количество ударов в минуту;

г) жанр.

Так же необходимо учитывать, что человеческий слух воспринимает информацию несколько иначе, чем компьютер. Поэтому нужно делать поправку на особенности человеческого слуха при анализе аудиозаписей.

3.2 Построение архитектуры

Сформулировав требоваяния к программному продукту, приступим к проектированию.

Необходимо определиться с методом преобразования адиозаписей к виду, который будет точно описывать композицию. Существует большая разница между характеристиками композиции, которые влияют на предпочтения пользователей, и характеристиками, которыми обладает сам сигнал. Поэтому работать непосредственно со звуковыми сигналами недостаточно эффективно, а так же достаточно долго. Спектрограммы ускоряют процесс работы, но так же не настолько эффективны, поскольку не учитывают особенности человеческого уха. Человеческое ухо, как говорилось ранее, воспринимает звук несколько иначе, что требует определенных корректировок. Для того, чтобы учесть поправки на восприятие звука ухом человека, можно отобразить спектрограмму на мел-шкалу. В результате мы получим мел-спектрограмму. Работать с мел-спектрограммой достаточно долго ввиду её большого размера. Необходимо найти способ сжать информацию до приемлемых размеров, при этом не потеряв в точности.

Для того, чтобы сжать информацию и учесть поправки на человеческое ухо, было решено использовать мел-частотные кепстральные коэффициенты. Достоинствами этого метода являются:

а) используется спектр сигнала, что позволяет учесть волновую природу звука;

б) спектр проецируется на мел-шкалу, что позволяет учесть восприятие частот человеческим ухом;

в) позволяет сжать количество информации количеством вычисляемых коэффициентов.

Осталось понять как преобразовать сигнал в набор коэффициентов. Первым делом нам нужен спектр исходного сигнала, который мы получаем с помощью преобразования Фурье. Для того, чтобы не потерять временной составляющей, воспользуемся оконным преобразованием Фурье. Теперь полученный спектр нам нужно расположить на мел-шкале. Для этого мы используем окна, равномерно расположенные на мел-оси.

РИСУНОК ОКОН ИЗ СТАТЬИ НА ХАБРЕ!!!

Если перевести график (ССЫЛКА НА КАРТИНКУ ВЫШЕ) в частотную шкалу, можно увидеть картину (ССЫЛКА НА КАРТИНКУ НИЖЕ).

РИСУНОК ОКОН В НОРМ ШКАЛЕ ИЗ СТАТЬИ НА ХАБРЕ!!!!

На этом графике заметно, что окна «собираются» в области низких частот, обеспечивая более высокое «разрешение» там, где оно необходимо для извлечения. Простым перемножением векторов спектра сигнала и оконной функции найдем энергию сигнала, которая попадает в каждое из окон анализа. Мы получили некоторый набор коэффициентов, но это еще не те коэффициенты, которые мы ищем. Пока их можно было бы назвать мел-частотными спектральными коэффициентами. Возводим их в квадрат и логарифмируем. Нам осталось только получить из них кепстральные, или «спектр спектра». Для этого мы могли бы еще раз применить преобразование Фурье, но лучше использовать дискретное косинусное преобразование. В результате мы видим рисунок (ССЫЛКА НА РИСУНОК НИЖЕ).

Таким образом мы имеем очень небольшой набор значений, который при извлечении успешно заменяет тысячи отсчетов речевого сигнала.

Преобразовав сигнал, необходимо создать инструмент, который будет извлекать характеристики из полученных данных. Для этого решено использовать методы, основанные на машинном обучении. Для того, чтобы применять машинное обучение, необходимо построить модель, которая будет являться искомым инструментом. Для построения модели необходимо разработать структуру модели, а так же выбрать библиотеку для обучения.

В качестве библиотеки для обучения была выбрана TensorFlow от компании Google. Она предоставляет большой набор инструментов, при помощи которых можно обучать модели на различных системах. Так же TensorFlow помогает работать с высокой степенью оптимизации, что ускоряет процесс обучения и обработки данных.

Для того, чтобы ускорить процесс прототипирования и повысить уровень абстракции поверх TensorFlow будет использоваться библиотека Keras.

Связка Keras - TensorFlow позволит оптимально расходовать ресурсы системы на которых будет запускаться сервис, при этом будет сохранен высокий уровень абстракции, что повысит читабельность и универсальность кода.

Построим архитектуру модели, которая будет извлекать высокоуровневые характеристики из музыкальных композиций. Для извлечения признаков хорошо показывают глубинные нейронные сети. Для извлечения признаков из матриц больших размерностей, к которым относится полученный список мел-кепстральных коэффициентов, лучше всего использовать сверточные нейронные сети. Но для извлечения временных паттернов лучше всего себя

заркоммендовали рекурентные нейронные сети. Для получения наилучшего результата, скомбинируем сверточные нейронные сети и рекурентные нейронные сети в глубинной нейронной сети. В результате получим нейронную сеть, в которой будет 6 слоев. Первые 4 слоя являются 4-ех слойной сверточной нейронной сетью, которая будет извлекать локальные признаки. Следующие 2 слоя являются 2-ух слойной рекурентной нейронной сетью, которая предназначена для агреггирования временных шаблонов. Это более эффективный подход, чем, например, использовать усреднение результатов.

РИСУНОК НЕЙРОННОЙ СЕТИ, ПОЛУЧЕННОЙ В РЕЗУЛЬТАТЕ!!!!

Полученная нейронная сеть может быть больших размеров. Для того, чтобы сократить время на загрузку нейронной сети из дискового пространства в память следует держать сеть в оперативной памяти. Наиболее эффективно хранить сеть в отдельном процессе, в который будут поступать данные для обработки. Это снизит затраты по памяти, а так же позволит обрабатывать большее количество данных за единицу времени. Процесс, который хранит и использует нейронную сеть, решено реализовать в виде демона.

Для того, чтобы предоставить API, необходимо реализовать сервер, который обеспечит сетевое взаимодействие по заданному протоколу. Для реализации части сервера, которая отвечает за бизнес-логику, будет использоваться микрофреймворк Flask. Flask является микрофреймворком, что обеспечит необходимую функциональность с потреблением минимального количества ресурсов. Эта часть сервера будет использоваться для маршрутизации запросов, сериализации и десериализации данных, а так же для обмена данных с демоном, который отвечает за обработку данных. Для того, чтобы повысить пропускную способность, часть сервера, которая отвечает за сетевое взаимодействие, будет использоваться HTTP сервер Gunicorn. Он поддерживает использование несколько рабочих процессов для обработки запросов и одного мастер процесса для балансировки нагрузки между рабочими процессами.

Необходимо организовать передачу данных между демон процессом, который отвечает за обработку данных, и сервером. Для этого следует использовать очередь задач, которая будет использоваться для последовательной обработки данных, так же необходимо использовать кеширующую систему для того, чтобы обеспечить обратное взаимодействие. В качестве очереди задач будет использоваться RedisMQ, которая использует в качестве хранилища данных базу данных Redis. В качестве кеша, который будет обеспечивать обратное взаимодействие, так же будет использоваться Redis. Преимущество Redis в данном случае, заключается в том, что эта технология хранит данные в оперативной памяти, увечиливая её расход, зато это сокращает вре-

мя, которое потребуется для обмена данными.

Для того, чтобы контроллировать и запускать демон и сервер следует использовать систему контроля процессов. Такой системой является Supervisor. С его помощь процессы сервера и демона будут запускаться, так же они будут перезапускаться в случае неожиданного завершения работы какого-либо из процессов. Supervisor так же позволит хранить логи приложений в указанном месте для того, чтобы обеспечить мониторинг работы системы, что поможет так же и для отладки процессов в случае их аварийного завершения.

Чтобы предоставить сервис в виде модуля, который удобно встраивать в сторонние системы, необходимо все компоненты, описанные выше поместить в контейнер, который будет предоставляться пользователю. В качестве системы, которая обеспечит управление и развертывание контейнеров, следует использовать Docker. Преимущество Docker заключается в том, что он предоставляет не полную виртуализацию, что позволяет существенно экономить вычислительные ресурсы. Так же Docker обеспечит переносимость контейнера на другие системы, что позволит развертывать сервис с минимальными затратами. Особенности виртуализации Docker уменьшает размер контейнера, что является дополнительным удобством при переносе сервиса.

4 ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ

4.1 Essentia

Essentia - это библиотека C ++ с открытым исходным кодом, обладающая привязками Python, предназначенная для аудиоанализа. Она выпускается под лицензией Affero GPLv3 и также доступна под собственной лицензией по запросу. Библиотека содержит обширную коллекцию многократно используемых алгоритмов, реализующих функциональность ввода и вывода звука, стандартные блоки цифровой обработки сигналов, статистическую характеристику данных и большой набор спектральных, временных, тональных музыкальных обработчиков.

Essentia - это не фреймворк, а скорее набор алгоритмов, завернутый в библиотеку. Он сконструирован с упором на надежность, производительность и оптимальность предоставляемых алгоритмов, включая скорость вычислений и использование памяти, а также простоту использования. Поток анализа определяется и реализуется пользователем, в то время как Essentia заботится о деталях реализации используемых алгоритмов. Существует специальный режим потоковой передачи, в котором возможно соединять алгоритмы и запускать их автоматически, вместо того чтобы явно указывать порядок выполнения с преимуществом менее стандартного кода и меньшим потреблением памяти. Ряд примеров предоставляется библиотекой, однако их не следует рассматривать как единственный правильный способ делать вещи. Большая часть алгоритмов Essentia хорошо подходит для приложений реального времени.

Предоставляемые функциональные возможности легко расширяемы и позволяют проводить как исследовательские эксперименты, так и разработку крупномасштабных промышленных приложений.

4.2 Python

Python — 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 вычисле-

ний и удобные высокоуровневые структуры данных. Код в Python 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 Python с использованием JIT-компиляции, которая значительно увеличивает скорость выполнения Python-программ.

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

4.3 TensorFlow

TensorFlow - это библиотека программного обеспечения с открытым исходным кодом для машинного обучения по целому ряду задач и разработанная Google для удовлетворения своих потребностей в системах, способных создавать и обучать нейронные сети для обнаружения и расшифровки паттернов и корреляций, аналогичных обучению и рассуждениях, которые используют люди. В настоящее время он используется для исследований и производства продуктов Google, часто заменяя роль предшественника с закрытым исходным кодом, DistBelief. TensorFlow была первоначально разработана командой Google Brain для внутреннего использования Google, прежде чем выпустить ее под лицензией с открытым исходным кодом Apache 2.0 9 ноября 2015 года.

Начиная с 2011 года, Google Brain построила DistBelief как проприетарную систему машинного обучения, основанную на глубоком обучении нейронных сетей. Его использование быстро развивалось в различных компаниях родительской компании Alphabet как в исследовательских, так и в коммерческих целях. Google назначил нескольких компьютерных ученых, включая Джеффа Дина, для упрощения и реорганизации базы кода DistBelief в более быструю и более надежную библиотеку прикладного уровня, которая стала TensorFlow. В 2009 году команда во главе с Джеффри Хинтоном реализовала обобщенное обратное распространение и другие улучшения, которые позво-

лили создать нейронные сети с существенно большей точностью, например, на 25% сократить ошибки распознавания речи.

TensorFlow - это система машинного обучения второго поколения Google Brain, выпущенная как программное обеспечение с открытым исходным кодом 9 ноября 2015 года. Хотя эталонная реализация выполняется на отдельных устройствах, TensorFlow может работать на нескольких процессорах и графических процессорах (с дополнительными расширениями CUDA для универсальных вычислений на графических процессорах). TensorFlow доступен на 64-битных Linux, macOS и мобильных вычислительных платформах, включая Android и iOS.

Вычисления TensorFlow выражаются в виде графов прохождения данных с состоянием. Название TensorFlow происходит от операций, которые такие нейронные сети выполняют с многомерными массивами данных. Эти многомерные массивы называются тензорами. В июне 2016 года Джефф Дин из Google заявил, что 1500 репозиториев на GitHub упомянули TensorFlow, из которых только 5 пренадлежали Google.

В мае 2016 года Google объявил о своем тензорном процессоре (TPU) - специализированной ASIC, разработанной специально для машинного обучения и предназначенной для TensorFlow. TPU представляет собой программируемый ускоритель ИИ, предназначенный для обеспечения высокой пропускной способности арифметики с низкой точностью (например, 8-разрядной) и ориентированной на использование или запуск моделей, а не на их обучение. Google объявила, что они уже более года работают с моделями машинного обучения на основе TPU в своих центрах обработки данных, и обнаружили, что TPU обеспечивает на порядок более оптимизированную производительность на ватт для машинного обучения.

4.4 Keras

Keras - это высокоуровневый API нейронных сетей, написанный на Python и способный работать поверх TensorFlow или Theano. Он был разработан с упором на быстрое проведение экспериментов. Способность идти от идеи к результату с наименьшей задержкой является ключевой особенностью данной библиотеки.

Особенности Keras:

а) позволяет легко и быстро создавать прототипы (благодаря удобству пользователя, модульности и расширяемости);

б) поддерживает как сверточные сети, так и повторяющиеся сети, а также их комбинации;

в) работает без проблем на процессоре и графическом процессоре.

Руководящие принципы:

а) удобство для пользователя (Keras - это API, предназначенный для людей, а не для машин, это ставит удобство использования в центр внимания и обеспечивает ясную обратную связь из-за пользовательской ошибки);

б) модульность (под моделью понимается последовательность или граф автономных, полностью конфигурируемых модулей, которые могут быть подключены вместе с минимальными ограничениями);

в) легкая расширяемость (новые модули легко добавлять, как новые классы и функции, а существующие модули предоставляют достаточно примеров, чтобы иметь возможность легко создавать новые модули);

г) работа с Python (нет отдельных конфигурационных файлов моделей в декларативном формате, модели описаны в коде Python, который является компактным, более легким для отладки и позволяет легко расширять).

4.5 Flask

Flask - это небольшой фреймворк написанный на языке Python с весьма большим сообществом, и множеством модулей на все случаи жизни. В отличии от, скажем, Django, Flask не навязывает определенное решение той или иной задачи. Вместо этого, он предлагает использовать различные сторонние или собственные решения по вашему усмотрению.

Одним из проектных решений во Flask является то, что простые задачи должны быть простыми; они не должны занимать много кода, и это не должно ограничивать вас. Поэтому были сделаны несколько вариантов дизайна, некоторые люди могут посчитать это удивительным и не общепринятым. Например, Flask использует локальные треды внутри объектов, так что вы не должны передавать объекты в пределах одного запроса от функции к функции, оставаясь в безопасном треде. Хотя это очень простой подход и позволяет сэкономить много времени, это также может вызвать некоторые проблемы для очень больших приложений, поскольку изменения в этих локальных тредах-объектов может произойти где угодно в том же треде. Для того, чтобы решить эти проблемы, разработчики не скрывают от вас локальные треды-объекты, вместо этого охватывают их и предоставляем вам много инструментов, чтобы сделать работу с ними настолько приятным насколько это возможно.

4.6 Gunicorn

Gunicorn - это HTTP-сервер Python WSGI для UNIX. Это модель рабочих процессов, перенесенная из проекта Ruby Unicorn. Сервер Gunicorn в целом совместим с различными веб-фреймворками, прост в использовании, не нагружает ресурсы сервера и является достаточно быстрым.

Особенности:

а) поддерживает WSGI, web2py, Django и Paster;

б) автоматическое управление рабочими процессами;

в) простая конфигурация Python;

г) несколько рабочих конфигураций;

д) различные серверные триггеры для расширяемости;

е) Совместимость с Python версий 2.6 и выше, а также Python версий 3.2 и выше.

4.7 Supervisor

Supervisor - это система клиент-сервер, которая позволяет своим пользователям управлять несколькими процессами в UNIX-подобных операционных системах.

Часто сложно получить точный статус (запущен или не запущен) в процессах в UNIX. Файлы с идентификаторами процессов часто не соответствуют действительности. Supervisord запускает процессы как подпроцессы, поэтому он всегда знает истинный статус своих подпроцессов, и предоставляет возможность узнать статусы для его дочерних процессов.

Supervisor предоставляет вам одно место для запуска, остановки и мониторинга процессов. Процессами можно управлять индивидуально или в группах. Вы можете настроить Supervisor для предоставления локальной или удаленной командной строки и веб-интерфейса.

Supervisor запускает свои подпроцессы через fork/exec, а подпроцессы не запускает как демоны. Операционная система сразу же сигнализирует Supervisor, когда процесс завершается, в отличие от некоторых решений, которые полагаются на неточные файлы идентификаторов процессов и периодический опрос процессов, чтобы перезапустить завершившиеся процессы.

4.8 Docker

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы. Позволяет поместить приложение со всем его окружением и

зависимостями в контейнер, который может быть перенесен на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами. Изначально использовал возможности LXC, с 2015 года применял собственную библиотеку, абстрагирующую виртуализационные возможности ядра Linux — libcontainer. С появлением Open Container Initiative начался переход от монолитной к модульной архитектуре.

Цели использования Docker:

а) абстрагирование хост-системы от контейнеризованных приложений;

б) простота масштабирования;

в) простота управления зависимостями и версиями приложения;

г) чрезвычайно легкие, изолированные среды выполнения;

д) совместно используемые слои;

е) возможность компоновки и предсказуемость.

Приложения, реализующие этот подход к проектированию, должны иметь следующие характеристики:

а) они не должны полагаться на особенности хост-системы;

б) каждый компонент должен предоставлять консистентный API, который пользователи могут использовать для доступа к сервису;

в) каждый сервис должен принимать во внимание переменные окружения в процессе первоначальной настройки;

г) данные приложения должны храниться вне контейнера на примонтированных томах или в отдельных контейнерах с данными.

5 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ВНЕДРЕНИЯ ПРОГРАММНОГО СРЕДСТВА

6.1 Характеристика программного средства

...

Разработки проектов программных средств связана со значительными затратами ресурсов. В связи с этим создание и реализация каждого проекта программного обеспечения нуждается в соответствующем технико-экономическом обосновании [2], которое и описывается в данном разделе.

6.2 Определение объема и трудоемкости ПС

Целесообразность создания ПС требует проведения предварительной экономической оценки. Экономический эффект у разработчика ПС зависит от объема инвестиций в разработку проекта, цены на готовый продукт и количества проданных копий, и проявляется в виде роста чистой прибыли.

Оценка стоимости создания ПС со стороны разработчика предполагает составление сметы затрат, вычисление цены и прибыли от реализации разрабатываемого программного средства.

Исходные данные, которые будут использоваться при расчете сметы затрат, представлены в таблице 6.1.

Перед определением сметы затрат на разработку программного средства необходимо определить его объём. Однако, на стадии ТЭО нет возможности рассчитать точные объемы функций, вместо этого с помощью применения действующих нормативов рассчитываются прогнозные оценки.

В качестве метрики измерения объема программных средств используется строка их исходного кода (LOC – lines of code). Данная метрика широко распространена, поскольку она непосредственно связана с конечным продуктом, может применяться на всём протяжении проекта и, кроме того, может использоваться для сопоставления размеров программного обеспечения. Далее под строкой исходного кода будем понимать количество исполняемых операторов.

Расчет объема функций программного средства и общего объема приведен в таблице 6.2.

Исходя из определенной 3-ей категории сложности и общего объема ПС $V_o = 28\,940$, нормативная трудоемкость $T_н = 520$ чел./д. [2, приложение 3]. Перед определением общей трудоемкости разработки необходимо определить несколько коэффициентов.

Таблица 6.1 – Исходные данные

Наименование показателя	Условное обозначение	Значение
Категория сложности		3
Дополнительный коэффициент сложности	$\sum_{i=1}^{n} К_i$	0,13
Степень охвата функций стандартными модулями	$К_т$	0,7
Коэффициент новизны	$К_н$	0,9
Количество дней в году	$Д_г$	365
Количество праздничных дней в году	$Д_п$	9
Количество выходных дней в году	$Д_в$	103
Количество дней отпуска	$Д_о$	21
Количество разработчиков	$Ч_р$	3
Тарифная ставка первого разряда, руб.	$Т^1_ч$	265
Среднемесячная норма рабочего времени, ч.	$Ф_р$	168,3
Продолжительность рабочей смены, ч.	$Т_ч$	8
Коэффициент премирования	K	1,3
Норматив дополнительной заработной платы	$Н_д$	20%
Норматив отчислений в ФСЗН	$Н_{сз}$	34%
Норматив отчислений по обязательному страхованию	$Н_{ос}$	0,6%
Норматив расходов по статье «Материалы»	$Н_{мз}$	5%
Норматив расходов по статье «Машинное время»	$Н_{мв}$	15%
Понижающий коэффициент к статье «Машинное время»		0,5
Стоимость машино-часа, руб.	$Ц_м$	0,8
Норматив расходов по статье «Научные командировки»	$Н_{рнк}$	15%
Норматив расходов по статье «Прочие затраты»	$Н_{пз}$	20%
Норматив расходов по статье «Накладные расходы»	$Н_{нр}$	50%
Уровень рентабельности	$У_{рп}$	10%
Ставка НДС	$Н_{ДС}$	20%
Норматив затрат на освоение ПО	$Н_о$	10%
Норматив затрат на сопровождение ПО	$Н_с$	20%

Коэффициент сложности, который учитывает дополнительные затраты труда, связанные с обеспечением интерактивного доступа и хранения, и

Таблица 6.2 – Перечень и объём функций программного модуля

№ функции	Наименование (содержание)	Объём функции, LoC
101	Организация ввода информации	100
102	Контроль, предварительная обработка и ввод информации	500
109	Организация ввода/вывода информации в интерактивном режиме	190
111	Управление вводом/выводом	2600
204	Обработка наборов и записей базы данных	1900
207	Манипулирование данными	8000
208	Организация поиска и поиск в БД	7500
304	Обслуживание файлов	500
305	Обработка файлов	800
309	Формирование файла	1000
506	Обработка ошибочных и сбойных ситуаций	500
507	Обеспечение интерфейса между компонентами	750
601	Отладка прикладных программ в интерактивном режиме	4300
707	Графический вывод результатов	300
	Общий объем	28 940

поиска данных в сложных структурах [2, приложение 4, таблица П.4.2]

$$К_с = 1 + \sum_{i=1}^{n} К_i = 1 + 0{,}06 + 0{,}07 = 1{,}13, \tag{6.1}$$

где $К_i$ — коэффициент, соответствующий степени повышения сложности за счет конкретной характеристики;
n — количество учитываемых характеристик.

Коэффициент $К_т$, учитывающий степень использования при разработке стандартных модулей, для разрабатываемого приложения, в котором степень охвата планируется на уровне около 50%, примем равным $0{,}7$ [2, приложение 4, таблица П.4.5].

Коэффициент новизны разрабатываемого программного средства $К_н$ примем равным $0{,}9$, так как разрабатываемом программное средство принадлежит определенному параметрическому ряду существующих программных средств [2, приложение 4, таблица П.4.4].

Исходя из выбранных коэффициентов, общая трудоемкость разработ-

ки $Т_о = Т_н \cdot К_с \cdot К_т \cdot К_н = 520 \cdot 1{,}13 \cdot 0{,}7 \cdot 0{,}9 = 370$ чел./д.

Для расчета срока разработки проекта примем число разработчиков $Ч_р = 3$. Исходя из комментария к постановлению Министерства труда и социальной защиты Республики Беларусь от 05.10.16 №54 «Об установлении расчетной нормы рабочего времени на 2017 год» [3], эффективный фонд времени работы одного человека составит

$$Ф_{эф} = Д_г - Д_п - Д_в - Д_о = 365 - 9 - 103 - 21 = 232 \text{ д.}, \tag{6.2}$$

где $Д_г$ — количество дней в году;
$Д_п$ — количество праздничных дней в году;
$Д_в$ — количество выходных дней в году;
$Д_о$ — количество дней отпуска.

Тогда трудоемкость разработки проекта

$$Т_р = \frac{Т_о}{Ч_р \cdot Ф_{эф}} = \frac{370}{3 \cdot 232} = 0{,}53 \text{ г.} = 194 \text{ д.} \tag{6.3}$$

Исходя из того, что разработкой будет заниматься 3 человека, можно запланировать фонд рабочего времени для каждого исполнителя

$$Ф_{пi} = \frac{Т_р}{Ч_р} = \frac{194}{3} \approx 65 \text{ д.} \tag{6.4}$$

6.3 Расчет сметы затрат

Основной статьей расходов на создание ПО является заработная плата разработчиков проекта. Информация об исполнителях перечислена в таблице 6.3. Кроме того, в таблице приведены данные об их тарифных разрядах, приведены разрядные коэффициенты, а также по формулам 6.5 и 6.6 рассчитаны месячный и часовой оклады.

$$Т_м = Т_м^1 \cdot Т_к, \tag{6.5}$$

$$Т_ч = \frac{Т_м}{Ф_р}, \tag{6.6}$$

где $Т_м$ — месячный оклад;
$Т^1_м$ — тарифная ставка 1-го разряда (положим ее равной 265 руб.);
$Т_к$ — тарифный коэффициент;
$Т_ч$ — часовой оклад;
$Ф_р$ — среднемесячная норма рабочего времени (в 2017 г. составляет 168,3 ч. [3]).

Таблица 6.3 – Работники, занятые в проекте

Исполнители	Разряд	Тарифный коэффициент	Месячный оклад, руб.	Часовой оклад, руб.
Руководитель проекта	17	3,98	1054,70	6,27
Ведущий инженер-программист	15	3,48	922,20	5,48
Инженер-программист II категории	11	2,65	702,25	4,17

Тогда основная заработная плата исполнителей составит

$$\begin{aligned} З_о &= \sum_{i=1}^{n} Т_{чi} \cdot Т_ч \cdot Ф_{пi} \cdot K = \\ &= (6{,}27 + 5{,}48 + 4{,}17) \cdot 8 \cdot 65 \cdot 1{,}3 = 10\,761{,}92 \text{ руб.}, \end{aligned} \tag{6.7}$$

где $Т_{чi}$ — часовая тарифная ставка i-го исполнителя, руб.;
$Т_ч$ — количество часов работы в день;
$Ф_{пi}$ — плановый фонд рабочего времени i-го исполнителя, д.;
K — коэффициент премирования (принятый равным 1,3).

Дополнительная заработная плата включает выплаты, предусмотренные законодательство о труде: оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей, и определяется по нормативу, установленному в организации, в процентах к основной заработной плате. Приняв данный норматив $Н_д = 20\%$, рассчитаем дополнительные выплаты

$$З_д = \frac{З_о \cdot Н_д}{100\%} = \frac{10\,761{,}92 \cdot 20\%}{100\%} = 2152{,}38 \text{ руб.} \tag{6.8}$$

Отчисления в фонд социальной защиты населения и в фонд обязательного страхования определяются в соответствии с действующим законодательством по нормативу в процентном отношении к фонду основной и до-

полнительной зарплат по следующим формулам

$$\begin{aligned} З_{сз} &= \frac{(З_о + З_д) \cdot Н_{сз}}{100\%}, \\ З_{ос} &= \frac{(З_о + З_д) \cdot Н_{ос}}{100\%}. \end{aligned} \tag{6.9}$$

В настоящее время нормы отчислений в ФСЗН $Н_{сз} = 34\%$ и в фонд обязательного страхования $Н_{ос} = 0{,}6\%$. Исходя из этого, размеры отчислений

$$\begin{aligned} З_{сз} &= \frac{(10\,761{,}92 + 2152{,}38) \cdot 34\%}{100\%} = 4390{,}86 \text{ руб.}, \\ З_{ос} &= \frac{(10\,761{,}92 + 2152{,}38) \cdot 0{,}6\%}{100\%} = 77{,}49 \text{ руб.} \end{aligned} \tag{6.10}$$

Расходы по статье «Материалы» отражают траты на магнитные носители, бумагу, красящие материалы, необходимые для разработки ПО определяются по нормативу к фонду основной заработной платы разработчиков. Исходя из принятого норматива $Н_{мз} = 5\%$ определим величину расходов

$$М = \frac{З_о \cdot Н_{мз}}{100\%} = \frac{10\,761{,}92 \cdot 5\%}{100\%} = 538{,}10 \text{ руб.} \tag{6.11}$$

Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для разработки и отладки ПО, которое определяется по нормативам на 100 строк исходного кода. Норматив зависит от характера решаемых задач и типа ПК; для текущего проекта примем $Н_{мв} = 15\%$ [2, приложение 6]. Примем величину стоимости машино-часа $Ц_м = 0{,}8$ руб. Тогда, применяя понижающий коэффициент $0{,}5$, получим величину расходов

$$Р_м = Ц_м \cdot \frac{V_о}{100} \cdot Н_{мв} = 0{,}8 \cdot \frac{28\,940}{100} \cdot 15\% \cdot 0{,}5 = 1736{,}40 \text{ руб.} \tag{6.12}$$

Расходы по статье «Научные командировки» определяются по нормативу в процентах к основной заработной плате. Принимая норматив равным $Н_{рнк} = 15\%$ получим величину расходов

$$Р_{нк} = \frac{З_о \cdot Н_{рнк}}{100\%} = \frac{10\,761{,}92 \cdot 15\%}{100\%} = 1614{,}29 \text{ руб.} \tag{6.13}$$

Расходы по статье «Прочие затраты» включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу в процентах к основной зара-

ботной плате. Принимая норматив равным $Н_{пз} = 20\%$ получим величину расходов

$$П_з = \frac{З_о \cdot Н_{пз}}{100\%} = \frac{10\,761{,}92 \cdot 20\%}{100\%} = 2152{,}38 \text{ руб.} \quad (6.14)$$

Затраты по статье «Накладные расходы», связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды, относятся к конкретному ПО по нормативу в процентном отношении к основной заработной плате исполнителей. Принимая норматив равным $Н_{нр} = 50\%$ получим величину расходов

$$Р_н = \frac{З_о \cdot Н_{нр}}{100\%} = \frac{10\,761{,}92 \cdot 50\%}{100\%} = 5380{,}96 \text{ руб.} \quad (6.15)$$

Общая сумма расходов по смете определяется как сумма вышерассчитанных показателей

$$С_п = З_о + З_д + З_{сз} + З_{ос} + М + Р_м + Р_{нк} + П_з + Р_н = \\ = 28\,804{,}78 \text{ руб.} \quad (6.16)$$

Рентабельность определяется из результатов анализа рыночных условий и переговоров с потребителями ПО. Исходя из принятого уровня рентабельности $У_{рп} = 10\%$, прибыль от реализации ПО составит

$$П_о = \frac{С_п \cdot У_{рп}}{100\%} = \frac{28\,804{,}78 \cdot 10\%}{100\%} = 2880{,}48 \text{ руб.} \quad (6.17)$$

На основании расчета прибыли и уровня себестоимости рассчитаем прогнозируемую цену программного средства без учета налогов

$$Ц_п = С_п + П_о = 28\,804{,}78 + 2880{,}48 = 31\,685{,}26 \text{ руб.} \quad (6.18)$$

Далее рассчитаем налог на добавленную стоимость

$$НДС = \frac{Ц_п \cdot Н_{ДС}}{100\%} = \frac{31\,685{,}26 \cdot 20\%}{100\%} = 6337{,}05 \text{ руб.} \quad (6.19)$$

НДС включается в прогнозируемую отпускную цену

$$Ц_о = Ц_п + НДС = 31\,685{,}26 + 6337{,}05 = 38\,022{,}31 \text{ руб.} \quad (6.20)$$

Организация-разработчик участвует в освоении и внедрении ПО и несет соответствующие затраты, которые определяются по нормативу $Н_о = 10\%$

от себестоимости ПО в расчете на три месяца

$$\text{Р}_\text{о} = \frac{\text{С}_\text{п} \cdot \text{Н}_\text{о}}{100\%} = \frac{28\,804{,}78 \cdot 10\%}{100\%} = 2880{,}48 \text{ руб.} \tag{6.21}$$

Кроме того, организация-разработчик осуществляет сопровождение ПО, которое также оплачивается заказчиком. Расчет осуществляется в соответствии с нормативом $\text{Н}_\text{с} = 20\%$ от себестоимости ПО

$$\text{Р}_\text{с} = \frac{\text{С}_\text{п} \cdot \text{Н}_\text{с}}{100\%} = \frac{28\,804{,}78 \cdot 20\%}{100\%} = 5760{,}96 \text{ руб.} \tag{6.22}$$

Экономическим эффектом разработчика будет являться сумма прибыли с вычетом налога на прибыль

$$\text{П}_\text{ч} = \text{П}_\text{о} - \frac{\text{П}_\text{о} \cdot \text{Н}_\text{п}}{100\%} = 2880{,}48 - \frac{2880{,}48 \cdot 18\%}{100\%} = 2361{,}99 \text{ руб.} \tag{6.23}$$

6.4 Оценка экономической эффективности применения ПС у пользователя

В результате применения нового ПО пользователь может понести значительные капитальные затраты на приобретение и освоение ПО, доукомплектованием ЭВМ новыми техническими средствами и пополнение оборотных средств. Однако, если приобретенное ПО будет в достаточной степени эффективнее базового, то дополнительные капитальные затраты быстро окупятся.

Для определения экономического эффекта от использования нового ПО у потребителя необходимо сравнить расходы по всем основным статьям сметы затрат на эксплуатацию нового ПО с расходами по соответствующим статьям базового варианта. При этом за базовый вариант примем ручной вариант. Исходные данные для расчета приведены в таблице 6.4.

Общие капитальные вложения заказчика (потребителя) вычисляются следующим образом

$$\begin{aligned} \text{К}_\text{о} = \text{К}_\text{пр} + \text{К}_\text{ос} + \text{К}_\text{с} = 38\,022{,}31 + 2880{,}48 + 5760{,}96 = \\ = 46\,663{,}75 \text{ руб.} \end{aligned} \tag{6.24}$$

где $\text{К}_\text{пр}$ — затраты пользователя на приобретение ПО по отпускной цене;
$\text{К}_\text{ос}$ — затраты пользователя на освоение;
$\text{К}_\text{с}$ — затраты пользователя на оплату услуг по сопровождению.

[1)] На февраль 2017 г. [4]

Таблица 6.4 – Исходные данные

Наименование показателя	Условное обозначение	Значение в базовом варианте	Значение в новом варианте
Затраты пользователя на приобретение ПО	$К_{пр}$	—	38 022,31 руб.
Затраты пользователя на освоение	$К_{ос}$	—	2880,48 руб.
Затраты пользователя на сопровождение	$К_{с}$	—	5760,96 руб.
Трудоемкость на задачу, чел./ч.	$Т_{с1}$,$Т_{с2}$	2	0,1
Средняя зарплата[1)], руб.	$З_{см}$	716,5	716,5
Количество выполняемых задач	$А_1$,$А_2$	1460	1460
Время простоя сервиса, мин. в день	$П_1$,$П_2$	50	10
Стоимость одного часа простоя, руб.	$С_{п}$	79,8	79,8

В качестве типичного примера использования разрабатываемого ПС предполагается сценарий ее использования ежедневно 4 раза в день, при этом предполагается снижение трудоемкости с $Т_{с1} = 2$ чел./ч. до $Т_{с2} = 0{,}1$ чел./ч.. Приняв среднемесячную заработную плату работника $З_{см} = 716{,}5$ руб., рассчитаем экономию затрат на заработную плату в расчете на одну задачу $С_{зе}$ и за год $С_{з}$

$$С_{зе} = \frac{З_{см} \cdot (Т_{с1} - Т_{с2})}{Ф_{р}} = \frac{716{,}5 \cdot (2 - 0{,}1)}{168{,}3} = 8{,}09 \text{ руб.}, \qquad (6.25)$$

$$С_{з} = С_{зе} \cdot А_2 = 8{,}09 \cdot 1460 = 11\,811{,}4 \text{ руб.}, \qquad (6.26)$$

где $Ф_{р}$ — среднемесячная норма рабочего времени, ч.

Экономия с учетом начислений на зарплату

$$С_{н} = С_{з} \cdot \frac{100\% + K}{100\%} = 11\,811{,}4 \cdot \frac{100\% + 1{,}3}{100\%} = 11\,964{,}95 \text{ руб.}, \qquad (6.27)$$

где K — норматив начислений на зарплату, руб.

Экономия за счет сокращения простоев сервиса

$$С_с = \frac{(П_1 - П_2) \cdot Д_{рг} \cdot С_п}{60} = \frac{(50 - 10) \cdot 300 \cdot 79{,}8}{60} = 15\,960{,}00, \text{ руб.}, \quad (6.28)$$

где $Д_{рг}$ — плановый фонд работы сервиса, д.

Тогда общая годовая экономия текущих затрат, связанных с использованием нового ПО

$$С_о = С_н + С_с = 11\,964{,}95 + 15\,960{,}00 = 27\,924{,}95 \text{ руб.} \quad (6.29)$$

Внедрение нового ПО позволит пользователю сэкономить на текущих затратах, то есть получить на эту сумму дополнительную прибыль. Для пользователя в качестве экономического эффекта выступает лишь чистая прибыль. Принимая размер ставки налога на прибыль $Н_п = 18\%$ получим

$$\Delta П_ч = С_о - \frac{С_о \cdot Н_п}{100\%} = 27\,924{,}95 - \frac{27\,924{,}95 \cdot 18\%}{100\%} = 22\,898{,}46 \text{ руб.} \quad (6.30)$$

В процессе использования нового ПО чистая прибыль в конечном итоге возмещает капитальные затраты. Однако полученные при этом суммы прибыли и затрат по годам приводят к единому времени – расчетному году (за расчетный год принят 2017-й год) путем умножения результатов и затрат за каждый год на коэффициент дисконтирования α. При расчете используются следующие коэффициенты: 2017 г. -1, 2018 г. $-0{,}8696$, 2019 г. $-0{,}7561$, 2020 г. $-0{,}6575$. Все рассчитанные данные экономического эффекта сводятся в таблицу 6.5.

В результате технико-экономического обоснования применения программного средства были получены следующие значения показателей эффективности:

– чистая прибыль разработчика составит $П_ч = 2361{,}99$ руб.;

– затраты заказчика окупятся уже на четвертом году использования;

– экономическая эффективность для заказчика, выраженная в виде чистого дисконтированного дохода, составит $5618{,}02$ руб. за четыре года использования данного ПС; более высокий прирост прибыли заказчик получит по истечению данного срока.

Полученные результаты свидетельствуют об эффективности разработки и внедрения проектируемого программного средства.

Таблица 6.5 – Расчет экономического эффекта от использования нового ПО

Наименование показателя	2017 г.	2018 г.	2019 г.	2020 г.
Результаты				
Коэффициент приведения	1	0,8696	0,7561	0,6575
Прирост прибыли за счет экономии затрат ($П_ч$), руб.	—	22 898,46	22 898,46	22 898,46
То же с учетом фактора времени, руб.	—	19 912,50	17 313,53	15 055,74
Затраты				
Приобретение ПО ($К_{пр}$), руб.	38 022,31	—	—	—
Освоение ПО ($К_{ос}$), руб.	2880,48	—	—	—
Сопровождение ПО ($К_с$), руб.	5760,96	—	—	—
Всего затрат ($К_о$), руб.	46 663,75	—	—	—
То же с учетом фактора времени, руб.	46 663,75	—	—	—
Экономический эффект				
Превышение результата над затратами, руб.	−46 663,75	19 912,5	17 313,53	15 055,74
То же с нарастающим итогом, руб.	−46 663,75	−26 751,25	−9437,72	5618,02

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] ISTQB glossary – Specification [Электронный ресурс]. — Электронные данные. — Режим доступа: http://glossary.astqb.org/search/specification. — Дата доступа: 25.03.17.

[2] Палицын, В.А. Технико-экономическое обоснование дипломных проектов: Метод. пособие для студ. всех спец. БГУИР. В 4-х ч. Ч. 4: Проекты программного обеспечения / В.А. Палицын. — Мн : БГУИР, 2006. — 76 с.

[3] Пещенко, Е.А. Производственный календарь на 2017 год [Электронный ресурс]. — Электронные данные. — Режим доступа: http://www.mintrud.gov.by/system/extensions/spaw/uploads/files/Kommetarij-2017-RV.pdf. — Дата доступа: 06.04.17.

[4] Белстат. О начисленной средней заработной плате работников в феврале 2017 г. [Электронный ресурс]. — Электронные данные. — Режим доступа: http://www.belstat.gov.by/ofitsialnaya-statistika/solialnaya-sfera/trud/operativnaya-informatsiya_8/o-nachislennoi-srednei-zarabotnoi-plate-rabotnikov/o-nachislennoy-sredney-zarabotnoy-plate-rabotnikov-v-fevrale-2017-nbsp-g/. — Дата доступа: 07.04.17.

ПРИЛОЖЕНИЕ А
(обязательное)
Фрагменты исходного кода

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="./node_modules/semantic-ui-css/semantic.min.css"/>
</head>
<body>
    <div id="root"></div>
    <script src="./node_modules/react/dist/react.js"></script>
    <script src="./node_modules/react-dom/dist/react-dom.js"></script>

    <script src="./dist/bundle.js"></script>
</body>
</html>
```