**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФГБОУ ВО**

**«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра «Информатика и программное обеспечение»

**«УТВЕРЖДАЮ»**

Зав. кафедрой «ИиПО», к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Копелиович Д.И.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Мобильное приложение ДЛЯ сборникА песен И аккордОВ**

**Дипломная работа**

Документы текстовые

Всего \_\_\_\_ листов в папке

**Руководитель**

\_\_\_\_\_\_\_\_\_\_\_\_ ст. преп. Трубакова А.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Консультанты:**

по экономической части

\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. Титарёв Д.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

по организационной части

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преп. Зяблова Е.С.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Нормоконтролер**

\_\_\_\_\_\_\_\_\_ к.т.н., доц. Булатицкий Д.И.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Студент**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лисицын Д.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Брянск 2022**

**БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Направление 09.03.01 – Информатика и вычислительная техника**

**профиль «Программное обеспечение вычислительной техники**

**и автоматизированных систем»**

**ЗАДАНИЕ**

**на дипломную работу**

студенту группы О-18-ИВТ-1-по-Б

***Лисицыну Давиду Алексеевичу***

1. Тема работы *Мобильное приложение для сборника песен и аккордов*

Утверждена приказом по БГТУ №  *530-3*  от  *17.05.2022* г.

1. Срок предоставления законченной работы  *июнь 2022 г.*
2. Исходные данные

Цель работы *разработка нативного музыкального мобильного приложения для сборника песен и аккордов*

Инструментальные средства  *IntelliJ IDEA Ultimate, Xcode, Xcode Simulator, Git, Sketch, Swift, SwifttUI, Core Data, Java 8, PostgreSQL, Postman, Spring boot, Hibernate, REST API*

Основные требования *приложение должно содержать каталог песен, поиск и сортировку по каталогу, фильтр в каталоге, смену тональности и размер шрифта в песне*

Дата выдачи задания 2022г.

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Копелиович Д.И.

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Трубакова А.А.

Задание принял к исполнению 2022г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лисицын Д.А.

**АННОТАЦИЯ**

Данная выпускная квалификационная работа выполнена по теме «Мобильное приложение для сборника песен и аккордов». Структура данной работы представлена следующими главами и их описанием.

**Анализ требований.** В разделе представлены существующие виды мобильных приложений. Рассмотрены их возможные аналоги, проведён анализ требований, сравнение существующих приложений по реализованному функционалу, на основании которого сделан вывод о разработке нативного мобильного приложения для дальнейшего использования.

**Экономическая часть.** В разделе представлено описание организации работ по созданию ПО и расчет его себестоимости.

**Разработка программного продукта.** Данный раздел определяет архитектуру программного продукта, схемы взаимодействия, описание серверной и клиентской части приложения, разработанный интерфейс, а также в разделе дано описание средств разработки в виде языков, сред разработки и фреймворков.

**Экспериментальная часть.** Описывает методику проводимых испытаний, позволяющих проверить корректность работы мобильного приложения, провести инсталляционное и функциональное тестирование, на основе запланированных и проведенных экспериментов. Провести интеграционное тестирование с помощью дополнительных программ.

**Организационная часть.** Включает различные требования при работе с ПЭВМ, а также проведен расчет искусственного освещения, необходимого для работы с ПЭВМ.

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 4](#_Toc106450159)

[ВВЕДЕНИЕ 6](#_Toc106450160)

[1. АНАЛИЗ ТРЕБОВАНИЙ 8](#_Toc106450161)

[1.1. Обзор предметной области 8](#_Toc106450162)

[1.2. Описание и сравнение программ аналогов 13](#_Toc106450163)

[1.2.1. Мобильное приложение Guitar Tab 13](#_Toc106450164)

[1.2.2. Мобильное приложение Guitar U 14](#_Toc106450165)

[1.2.3. Мобильное приложение Ultimate Guitar: Chords & Tabs 15](#_Toc106450166)

[1.2.4. Сравнение аналогов 17](#_Toc106450167)

[1.3. Функциональная модель разрабатываемой системы 18](#_Toc106450168)

[1.4. Функциональные требования 19](#_Toc106450169)

[1.5. Вывод 20](#_Toc106450170)

[2. ЭКОНОМИЧЕСКАЯ ЧАСТЬ 22](#_Toc106450171)

[3. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 24](#_Toc106450172)

[3.1. Архитектура программного продукта 24](#_Toc106450173)

[3.2. Модель данных 26](#_Toc106450174)

[3.2.1. Описание языков программирования 27](#_Toc106450175)

[3.2.2. Описание сред разработки 28](#_Toc106450176)

[3.2.3. Описание используемых фреймворков 30](#_Toc106450177)

[3.3. Описание структуры клиентской части 32](#_Toc106450178)

[3.4. Описание структуры серверной части 34](#_Toc106450179)

[3.5. Проектирование интерфейса 36](#_Toc106450180)

[3.5.1. Макет страницы каталога 36](#_Toc106450181)

[3.5.2. Макет экрана с плейлистами 37](#_Toc106450182)

[3.5.3. Макет страницы с песней 38](#_Toc106450183)

[4. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ 39](#_Toc106450184)

[4.1. План испытаний 39](#_Toc106450185)

[4.2. Испытания серверной части 40](#_Toc106450186)

[4.3. Испытания клиентской части 43](#_Toc106450187)

[5. ОРГАНИЗАЦИОННАЯ ЧАСТЬ 46](#_Toc106450188)

[ЗАКЛЮЧЕНИЕ 48](#_Toc106450189)

[СПИСОК ЛИТУРАТУРЫ 49](#_Toc106450190)

[Приложение 1 53](#_Toc106450191)

[Приложение 2 60](#_Toc106450192)

[Приложение 3 71](#_Toc106450193)

ВВЕДЕНИЕ

В наше время сложно найти человека, который не слушает музыку. Музыка окружает нас везде и всюду, где бы мы ни были, будто в кафе, в спортзале, парке, магазине, везде мы можем услышать какую-либо музыку. Она постоянно есть в нашей жизни, каждый день по всему миру появляются новые композиции, новые авторы и исполнители. Для современного поколения музыка это некий образ жизни, один из способов, при помощи которого человек может отобразить свою позицию в жизни, внутреннее состояние, переживания, выразить свои эмоции. Часто на каких-то праздниках, мероприятиях, просто дружеских посиделках люди поют песни. Раньше слова и ноты записывали на бумаге и хранили, как сборник, чтобы не приходилось все учить и запоминать, но время не стоит на месте. Сейчас печатные носители не пользуются таким спросом и популярностью. Люди хотят иметь все сразу у себя под рукой. Создание сайтов с огромными библиотеками песен стало следующим шагом в этом процессе, но появились также и ограничения в виде подключения к сети Интернет. Не всегда удобно заходить в браузер и искать нужную песню, а при медленном соединении, это еще и долго, а ведь сети может вообще не быть, к примеру, в лесу, у костра.

Решить эту проблему можно разными способами, но создание мобильного приложения, дает большое количество возможностей и путей решений. Мобильные приложения в наше время набирают все большую популярность, за счет своих преимуществ. Возможности смартфонов с каждым годом становятся все шире, а опыт их использования все лучше. Мобильные приложения используют ресурсы телефона, и не зависят от подключения к сети, в этом их большое преимущество и автономность [1].

Мобильный сборник песен, позволяет хранить любимые композиции на собственном устройстве, имея к ним постоянный прямой доступ без подключения к сети Интернет. Существует немало проектов, связанных с такими мобильными приложениями. Однако, не все эти проекты полностью удовлетворяют запросы современного пользователя. Интерфейс таких приложений не адаптирован под особенности музыкантов и меломанов. Нехватка функционала, нестабильная работа и множество багов не дают пользователю приятно использовать такое приложение. Среди достойных проектов возможно найти подходящее приложение, но оно в основном требует оплаты подписки на месяц или год и постоянной поддержки сети.

Актуальность работы заключается в том, что человеку, любящему музыку или умеющему играть на музыкальных инструментах, необходимо удобное и простое решение для хранения любимых песен под рукой, в частности на своем устройстве. В данное время использование сайтов с мобильных устройств создает некоторые неудобства в использовании и поиске песен.

**Целью** данной выпускной квалификационной работы является разработка нативного музыкального мобильного приложения для сборника песен и аккордов с коллекцией современных композиций для удобного использования обычными людьми или музыкантами.

Учитывая актуальность, было принято решение о разработке мобильного приложения, отвечающего ниже поставленным задачам:

1. Провести анализ предметной области, аналогичных продуктов, доступных для общего пользования и сравнить их.
2. Сформировать требования к разрабатываемой системе.
3. Реализовать мобильное приложение под управлением ОС iOS.
4. Собрать данные с сайтов, имеющих большие базы с песнями.
5. Провести испытания и устранить найденные ошибки.

**Объектом** выпускной квалификационной работы является организация деятельности музыкантов.

**Предметом** выпускной квалификационной работы является разработка мобильного приложения – сборника песен и аккордов для музыкантов.

1. АНАЛИЗ ТРЕБОВАНИЙ
   1. Обзор предметной области

Компьютеры или ноутбуки уже давно находятся позади мобильных устройств. В последнее время мобильные устройства развиваются быстрыми темпами, каждый год производятся новые модели, с более новыми функциями, как аппаратными, так и программными. Люди уже давно не хотят иметь просто телефон или плеер, им нужен умный смартфон с большей функциональностью, нежели у обычного телефона с калькулятором и записной книжкой. В мире все большее количество людей используют мобильные устройства как основные для доступа в интернет, социальные сети, просмотр кино, чтение электронных книг и так далее. Мощности мобильных телефонов уже догнали и опередили мощности старых компьютеров 2000-х годов. Они заменили многим компьютеры, ноутбуки и другие виды электронной техники. С каждым новым годом количество пользователей мобильных устройств растет. Согласно отчету, опубликованному на сайте datareportal.com, который содержит всю последнюю информацию о мобильной статистике и статистике социальных сетей, в январе 2021 года в Российской Федерации было 228,6 млн. подключений с мобильных устройств. Количество таких подключений в РФ на январь 2021 года было эквивалентно 156,7% от всего населения. Многие люди имеют более одного мобильного устройства, поэтому цифры превышают 100% от общей численности населения [1]. Ни для кого не секрет, что мобильные устройства на сегодняшний день получили широчайшее распространение и мобильная разработка уже давно заняла свое место в мире информационных технологий, в каждой крупной компании разрабатывающий свой продукт, есть отдел по мобильной разработке. Разработка мобильных продуктов, очень перспективное направление, так как миллионы людей по всему миру пользуются мобильными устройствами [3]. Это достаточно свежее направление, которое постоянно развивается и совершенствуется. С каждым годом появляются новые технологии и новые возможности по взаимодействию пользователя с устройством.

Согласно статистике в статье бизнес-аналитика Sophia Martin исследователи прогнозируют, что совокупная доходность мобильных приложений во всем мире в 2023 году превысит сумму в $935,2 миллиарда [16]. Более того, с учетом увеличения числа приложений, предлагаемых разными магазинами, прогнозируется, что к 2023 году количество их скачиваний достигнет показателя в 268,2 миллиарда. В качестве обобщения этих статистических данных можно уверенно сказать, что индустрия мобильных приложений процветает и с каждым годом достигает все новых вершин своего развития.

Существую несколько видов разработки под мобильные устройства [16]. Данные направления можно разделить на три категории, такие как:

* Web-App;
* Native;
* Hybrid.

Нативные приложения (Native) – это такие приложения, которые создаются для конкретной платформы, нацеливаясь на пользователей, либо Android, либо iOS. Поскольку каждая из этих платформ имеет совершенно различные стандарты, для разработки двух отдельных приложений, одно для Google Play Store, а второе для Apple App Store нужны разные инструменты и языки программирования.

**Преимущества нативного приложения:**

* Превосходная производительность: нативные приложения выполняются плавно и без зависаний даже в случаях повышенной нагрузки на графический процессор и интеграции сложных вычислений.
* Наличие доступа к индивидуальным возможностям платформы: самое лучшее в этом типе приложений – это то, что они обеспечивают доступ к встроенным возможностям устройств или конкретной платформы.
* Нативный пользовательский интерфейс: плавный опыт использования обеспечивается благодаря тому, что приложения создаются в соответствии со стандартами платформы.

**Недостатки нативного приложения:**

* Необходимы две команды разработчиков: нативные приложения для Android обычно создаются на Java или Kotlin, в то время как приложения для iOS разрабатываются на Objective-C или Swift, в связи с чем вам потребуется нанять команду, имеющую опыт работы именно с этими языками.
* Высокая стоимость разработки: нативные приложения идеально подходят для крупных корпораций с большими бюджетами. Поскольку вам нужно разрабатывать каждое приложение с нуля, то для его итоговой экономической успешности потребуется много ресурсов, в том числе времени.

Гибридные (Hybrid) приложения создаются с единой базой кода, допускающей их запуск на нескольких операционных платформах. Многие относят гибридные приложения к кроссплатформенным, но общее между ними лишь то, что они имеют одну базу кода [8]. Тем не менее кроссплатформенный подход отлично работает для малобюджетных приложений с безопасными, стабильными и легко обслуживаемыми функциями.

**Преимущества гибридного приложения:**

* Быстрый вывод на рынок: гибридные приложения разрабатываются быстрее, так как в ходе процесса используются стандартные веб-технологии, которые легко обслуживать в долгосрочной перспективе.
* Доступ к возможностям устройств: используя гибридные приложения, вы также можете использовать нативные возможности целевых устройств.
* Дистрибуция для нескольких платформ: этот вид приложений распространяется через оба магазина, что позволяет охватить большее число пользователей.

**Недостатки гибридного приложения:**

* Производительность, их производительность ниже, чем у нативных вариантов, так как зависит от качества процессов, отображающих UI и выполняющих код. Поэтому чем быстрее использующее приложение устройство, тем выше его производительность.
* Интеграция сторонних сервисов: Вы не можете разрабатывать гибридное приложение на одном только JavaScript. Вам потребуется интегрировать такие фреймворки для гибридной разработки, как Cordove, Ionic или React Native, каждый из которых требует определенных усилий для освоения.

Веб-приложения (Web-App), в свою очередь, являются мобильными версиями сайтов, которые, благодаря своей функциональности и креативности, создают впечатление нативных приложений [13]. Такие приложения предоставляет вам около нативный опыт и возможность выполнения во всех браузерах и устройствах, включая ноутбуки, планшеты, смартфоны, умные часы и даже ТВ. Единственным требованием является наличие на устройстве браузера. В этом случае вместо разработки отдельных приложений для каждой платформы можно нацелиться на все сразу, создав всего одно.

**Преимущества веб-приложения:**

* Совместимость с несколькими платформами: создав web-приложение вы можете тут же запускать его на любой платформе, не вкладывая дополнительных средств и времени в дополнительную разработку.
* Мгновенные обновления: при использовании гибридных приложений пользователи всегда имеют доступ к их последним версиям и скачивать обновления им не приходится.
* Использование распространенных технологий: поскольку web-приложения можно разрабатывать при помощи различных технологий, выбор наиболее подходящей компании-разработчика не представляет для стартапов сложности.

**Недостатки веб-приложения:**

* Ограниченный доступ к нативным функциям платформы: web-приложения не имеют доступа к встроенным возможностям устройств, таким как камера, хранилище, контакты и прочее.
* Базовая производительность: эти приложения работают плавно в простых случаях применения, таких как новые издательства и онлайн магазины.

Нативные приложения для мобильных устройств обладают неоспоримыми плюсами в сравнении с другими решениями. Производительность приложения – это одна из важнейших его составляющих, определяющая продолжительность использования этого приложения пользователями. В ходе опроса выяснилось, что наиболее распространенными причинами удаления приложений являются следующие: 59% пользователей назвали низкую скорость, 76% назвали фризы экрана, а 71% сбои в работе [16]. Когда доходит до оценки типов приложений в отношении их производительности, нативные варианты могут обеспечить несопоставимые с другими показатели. Поскольку они создаются под конкретную платформу, приложения Android и iOS размещаются в соответствующих этим платформам магазинах. Это позволяет задействовать возможности устройств и пользоваться системой рейтинга магазинов.

В рамках данной работы будет разработано нативное приложение под iOS. На данный момент операционная система iOS от компании Apple занимает второе место среди самых распространенных операционных системах на мобильные устройства, а магазин приложений App Store одним из самых крупных на этой платформе.

* 1. Описание и сравнение программ аналогов

Для того чтобы выявить ключевые моменты, на которые стоит обратить внимание при разработке приложения, был проведен анализ и сравнение похожих по функционалу программ. Из них для сравнения были выбраны три приложения:

* Guitar Tab;
* Guitar U;
* Ultimate Guitar: Chords & Tabs.

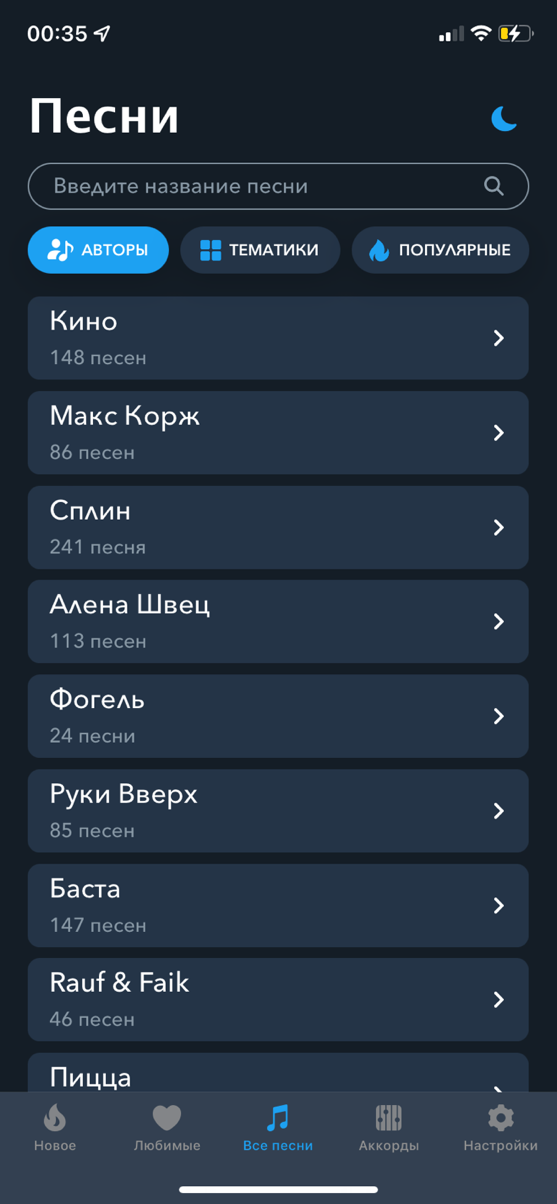
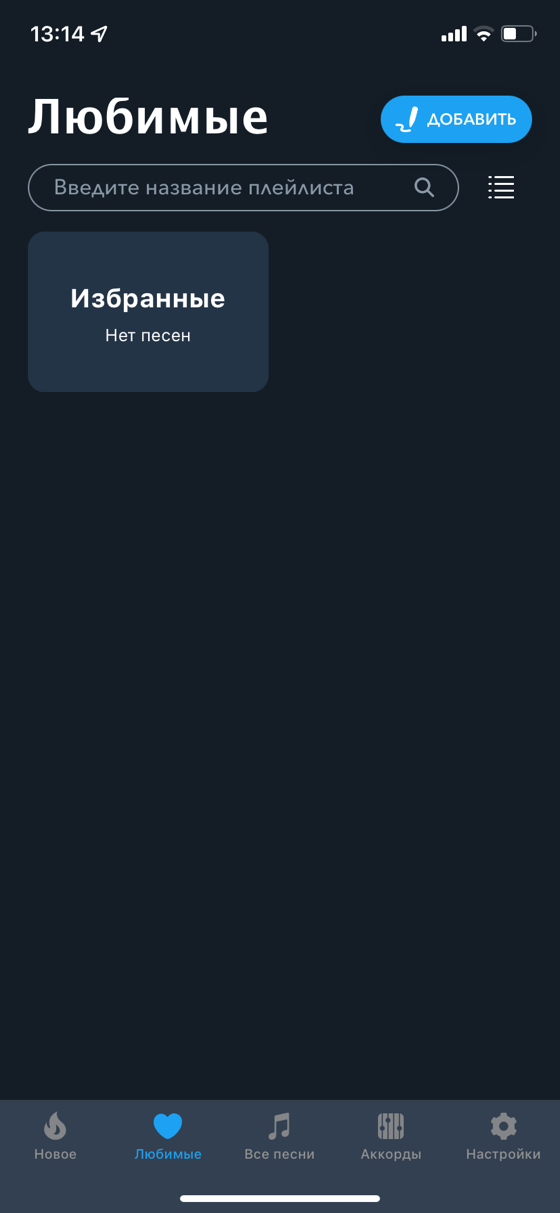
Приложения бесплатны для скачивания, имеют положительные оценки от пользователей и наибольший рейтинг среди других приложений, такой-же тематики.

* + 1. Мобильное приложение Guitar Tab

Мобильное приложение Guitar Tabs (см. рис. 1) представляет собой сборник новых и популярных песен для гитары. Имеет оценку в 4,5 звезды в магазине App Store.

Простой и интуитивно понятный интерфейс. Имеется своя база песен, в каталоге песен можно перейти в раздел с песнями на разную тематику и популярные песни. Приложение имеет внутренний рейтинг среди песен. Во вкладке новое можно найти свежие добавленные песни и новинки. Приятно радуем вкладка с аппликатурами для аккордов, где можно посмотреть, как берется какой аккорд на гитаре. По функционалу, приложение имеет все основные нужные функции, на странице с песней можно менять тональность, увеличивать текст, также возможно импортировать в PDF формат.

Что отталкивает от приложения, это невозможность открытия всего списка песен, так как приложение начинает предлагать платную подписку и ограничивает доступ к песням. Также в приложении в основном большинстве новые песни, и нет поиска по тексту песни.

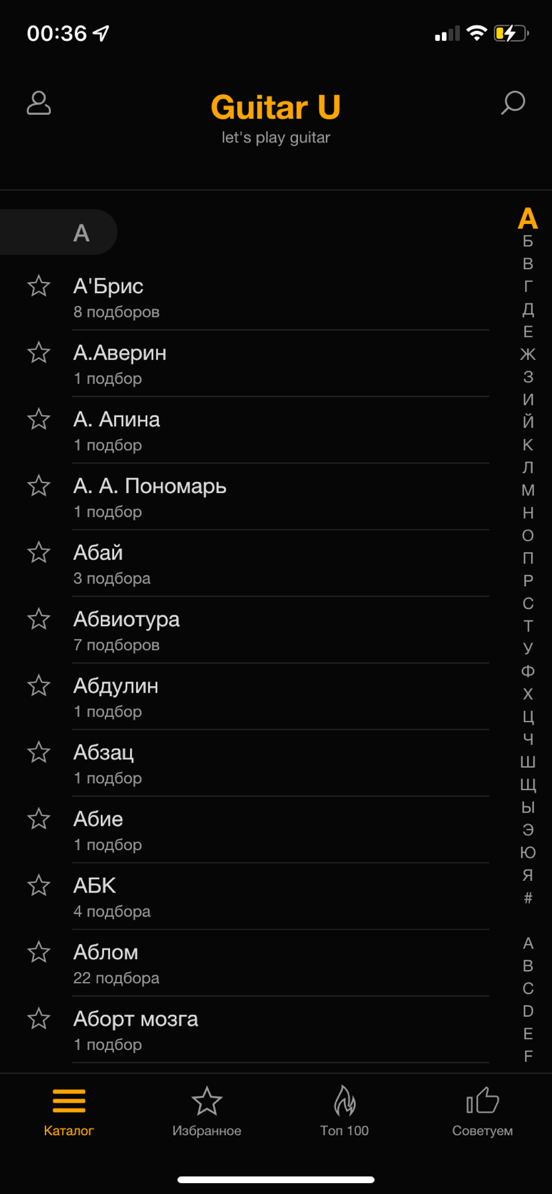
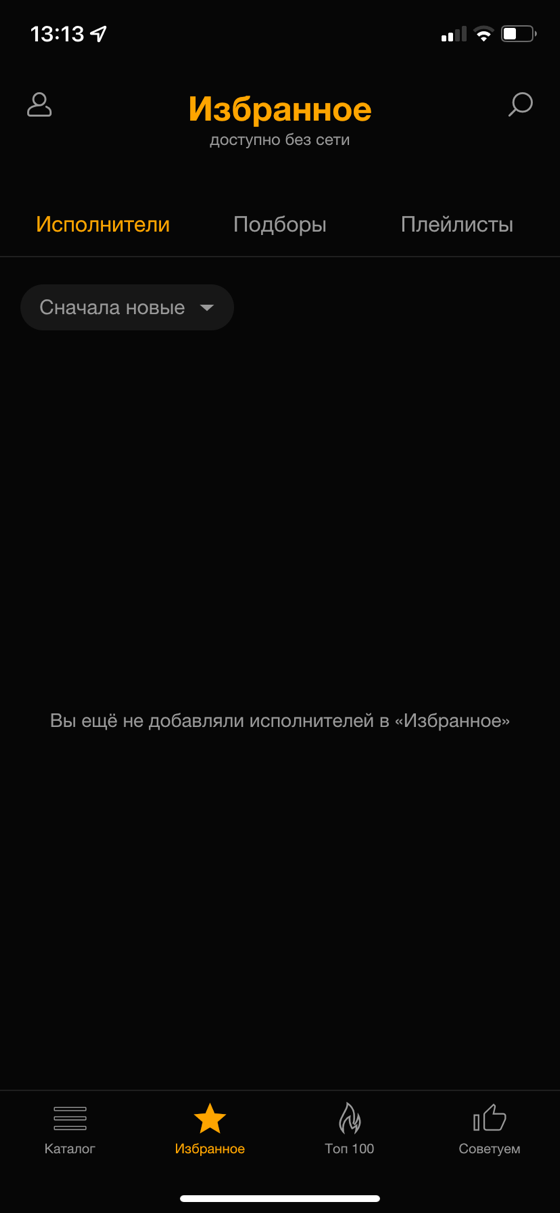
**а) б)**

**Рис. 1. Мобильное приложение Guitar Tabs:**

**а – экран все песни; б – экран любимые**

* + 1. Мобильное приложение Guitar U

Мобильное приложение Guitar U (см. рис. 2), также представляет собой сборник песен для гитары. Имеет оценку в 4 звезды в магазине App Store. Является почти бесплатным приложением, имеющем большое количество песен с аккордами в одном облачном каталоге. В основном песни представлены в виде подборов. Подбор – это песня с определенными изменениями в виде текста или мелодии и аккордов. Поэтому песни могут повторятся, но в разных вариациях, это интересно. Каждый такой вариант песни имеет свой рейтинг. В самом приложении есть проблема с разделами, так как то, что могло быть в одном месте разделено на 3 вкладки. Это каталог с поиском, вкладка топ 100 песен и вкладка советуем. Не совсем понятно, песни в разделах повторяются. В приложении также нужна подписка, но она дает только дополнительные функции. Все основные функции присутствуют в хорошем виде. В комментариях пользователи жалуются на нестабильную работу приложения.

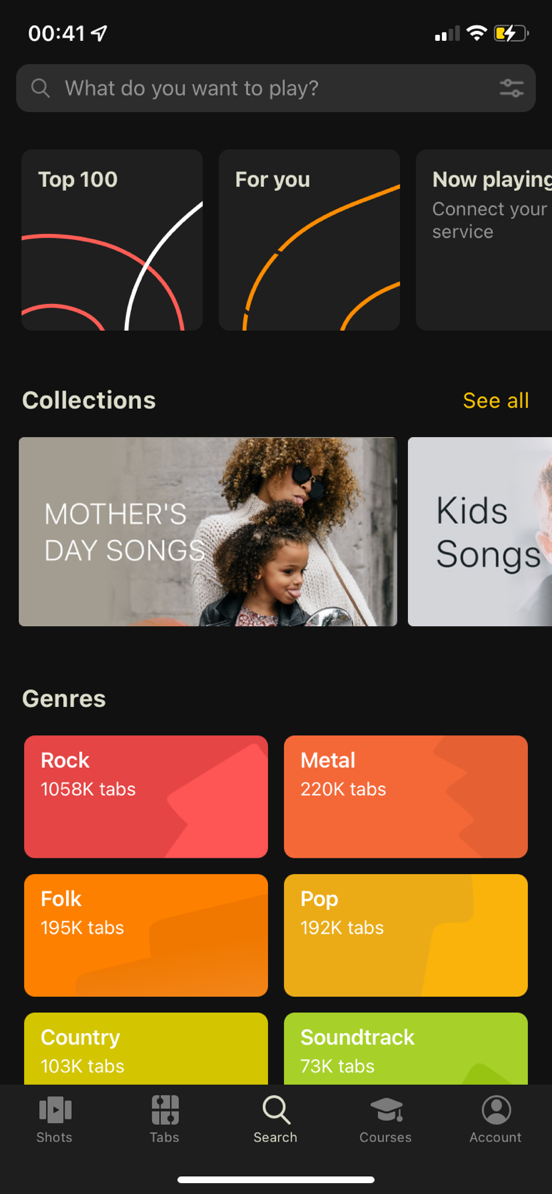
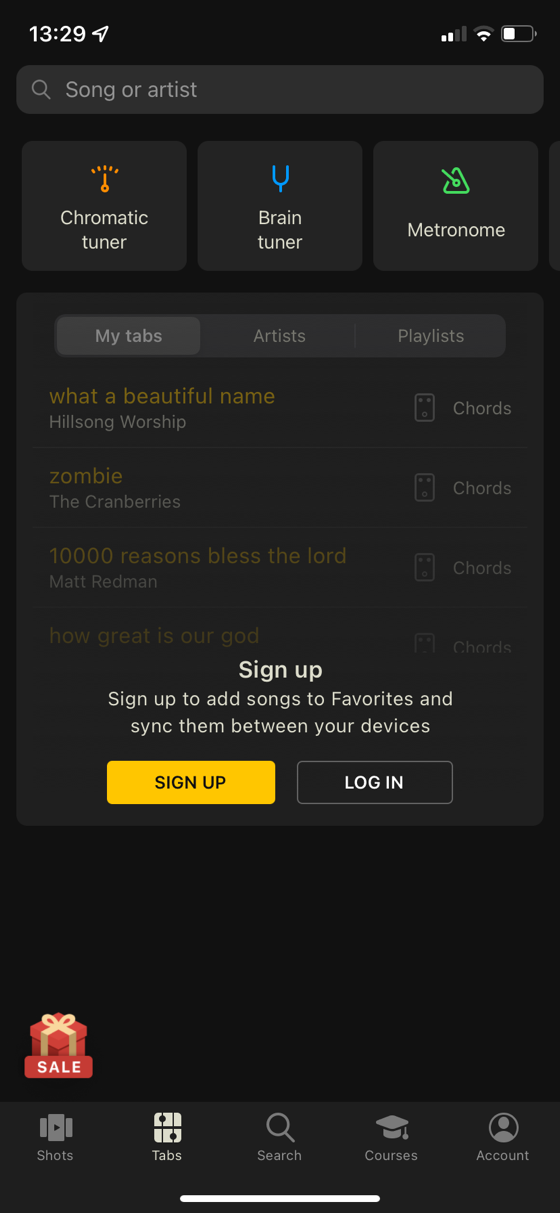
**а) б)**

**Рис. 2. Мобильное приложение Guitar U:**

**а – экран каталог; б – экран избранное**

* + 1. Мобильное приложение Ultimate Guitar: Chords & Tabs

Мобильное приложение Ultimate Guitar: Chords & Tabs (см. рис. 3), это большой каталог аккордов, табулатур и текстов для гитары, баса и укулеле. Имеет оценку в 4,7 звезды в магазине App Store. Приложение немного напоминает социальную сеть. В нем есть лента с видео от пользователей. Есть обучающие курсы встроенные прямо в приложение. Большое разнообразие разделов, по которым группируется музыка и разделение по сложности. Само приложение – это дополнение к сайту. Есть расширенный поиск по типу, сложности, настройке и рейтингу. Дизайн приложения проработан и понятен. Приложение предоставляет, огромный функционал, начиная от разных партий для разных инструментов, заканчивая персональными курсами. Есть большой минус, платная подписка, отдельно для курсов, отдельно для приложения. Многие пользователи жалуются на работу этой подписки. Дополнительно приложение ориентировано на англоговорящую часть людей, поэтому русских песен мало.

а) б)

**Рис. 3. Мобильное приложение Ultimate Guitar: Chords & Tabs:**

**а – экран Search; б – экран Tabs**

* + 1. Сравнение аналогов

В таблице 1 приведены сравнительные характеристики аналогов.

**Таблица 1**

***Сравнительная таблица***

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | Guitar Tabs | Guitar U | Ultimate Guitar |
| Сортируемый каталог | 3 | 2 | 5 |
| Поиск | 3 | 3 | 3 |
| Смена тональности | 5 | 5 | 5 |
| Изменение размера текста | 3 | 5 | 4 |
| Импорт в другие форматы | 4 | 2 | 5 |
| Обучающие материалы | 3 | 0 | 5 |
| Стоимость подписки  (больше = дороже) | 2 | 3 | 5 |
| Контент  (разнообразие и кол-во песен) | 4 | 3 | 3 |

Подводя итоги, можно сказать, что все три аналога в целом выполняют поставленные задачи. С некоторыми из них более приятно работать (за счет приятного дизайна, удобного интерфейса, разделения на категории), а с другими более быстро.

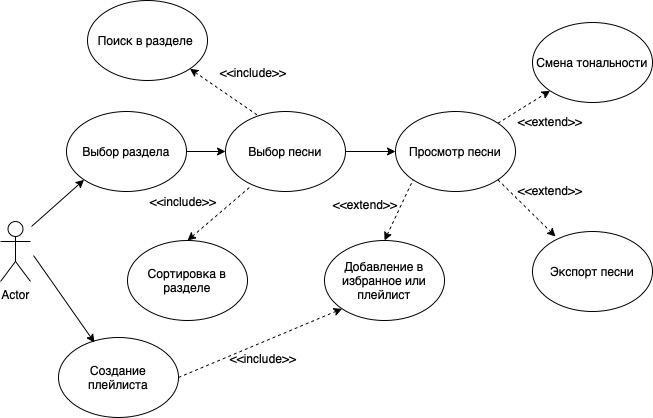
Все взятые аналоги выполняют основную задачу, но в каждом есть найти свои недостатки. Поэтому было принято решения создать собственное приложение.

* 1. Функциональная модель разрабатываемой системы

Для моделирования разрабатываемой системы была выбрана методология Use Cases и построена диаграмма вариантов использования, предназначенная для отображения внешнего функционирования проектируемого приложения. По правилам UML модель спецификации обычно использует диаграммы сценариев использования (Use Cases) для кластеризации требований, а затем – различные механизмы UNL (машины состояния, моделирование сценариев и моделирование активности) для уточнения и устранения неясностей в требованиях. Проектная модель определяет требования низкого уровня (LLR’s) для проверки уточненных требований и модели спецификации. Трассировка требований определяет связи между требованиями, моделью спецификации и проектной моделью. Главная задача моделирования – выяснение требований к программному продукту на начальных этапах, когда решаются наиболее общие задачи его предназначения.

Пользователь с использованием приложения просматривает по указанным параметрам песни, содержащиеся в каталоге. Соответственно, на диаграмме представлены следующие сценарии:

* Выбор раздела;
* Выбор песни;
* Поиск в разделе;
* Сортировка в разделе;
* Просмотр песни;
* Смена тональности;
* Экспорт песни;
* Добавление в избранное или плейлист;
* Создание плейлиста.



**Рис. 4. Диаграмма вариантов использования**

* 1. Функциональные требования

Разрабатываемая система должна равняться на уже существующие продукты, предоставляющие схожий функционал. В качестве примеров пользовательской части приложения могут служить описанные ранее веб-приложения Guitar Tabs, Guitar U, Ultimate Guitar. Данные приложения предоставляют схожий функционал для пользователей: задание параметров для поиска песни, выбор песни из списка найденных, сохранение или добавление в плейлист песни. Каждое из них имеет свои недостатки, в Guitar U нет хорошего сортируемого каталога и импорта в другие форматы, в Guitar Tabs плохо работает поиск, в Ultimate Guitar малое кол-во русских песен, также во всех приложениях имеется платная подписка.

После рассмотрения аналогов можно сделать вывод о том, какой функционал стоит позаимствовать, а также как можно усовершенствовать разрабатываемый программный продукт. Итак, функциональные требования, которые необходимо учитывать при разработке вспомогательных программных компонентов для мобильного приложения, должны быть следующими:

1. Просмотр каталога песен.
2. Поиск среди песен по названию, автору или тексту.
3. Сортировка песен по автору или названию.
4. Настройка шрифта и тональности песни.
5. Возможность скачать песню, добавив в избранное.
6. Возможность создания своих плейлистов и добавление туда песен.

Пользовательский интерфейс для приложения должен быть представлен несколькими страницами. Снизу располагается меню, включающее три основных раздела. Первый раздел содержит рекомендации, историю просмотра композиций, и рейтинг песен в самом приложении. Второй раздел является основным и содержит каталог песен с поиском и фильтром. Третий раздел содержит избранные песни и плейлисты, все песни этого раздела сохранены на самом устройстве. По нажатию на любую композицию должен осуществляться переход на ее страницу. На странице располагаются сам текст песни с аккордами песни и элементы управления, такие как: кнопки для перехода на предыдущую страницу, изменения размера шрифта, изменения тональности песни, добавления в избранное, добавление в плейлист, кнопка поделится с возможностью переслать песню или экспортировать.

Любой пользователь должен быть в состоянии легко разобраться с интерфейсом.

* 1. Вывод

На основании приведенного анализа были сделаны выводы об актуальности разработки мобильного приложения.

**Целью** данной выпускной квалификационной работы является разработка нативного музыкального мобильного приложения для сборника песен и аккордов с коллекцией современных композиций для удобного использования обычными людьми или музыкантами.

Учитывая актуальность, было принято решение о разработке мобильного приложения, отвечающего ниже поставленным задачам:

1. Провести анализ предметной области, аналогичных продуктов, доступных для общего пользования и сравнить их.
2. Сформировать требования к разрабатываемой системе.
3. Реализовать мобильное приложение под управлением ОС iOS.
4. Собрать данные с сайтов, имеющих большие базы с песнями.
5. Провести испытания и устранить найденные ошибки.

**Объектом** выпускной квалификационной работы является организация деятельности музыкантов.

**Предметом** выпускной квалификационной работы является разработка мобильного приложения – сборника песен и аккордов для музыкантов.

1. ЭКОНОМИЧЕСКАЯ ЧАСТЬ

Перед разработкой программного продукта необходимо также провести экономический анализ. Это довольно важный этап, поскольку в нем рассматриваются вопросы организации работ по созданию и внедрению продукта, а также рассчитывается его себестоимость.

**Расчет себестоимости программного продукта**

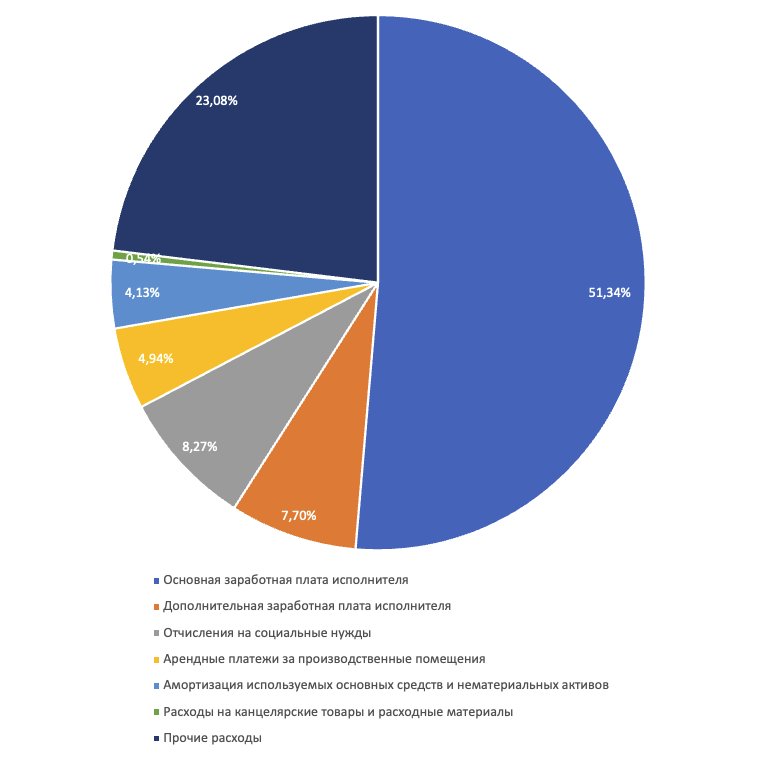
Структура себестоимости программного продукта отражена в табл. 2 и представлена на рис. 5.

Расчеты себестоимости приведены в прил. 1.

**Таблица 2**

***Структура себестоимости программного продукта***

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Элементы себестоимости** | **Сумма (руб.)** | **% в общ. сумме себестоимости** |
| 1 | Основная заработная плата исполнителя | 284 976 | 51,34 |
| 2 | Дополнительная заработная плата исполнителя | 42 746 | 7,70 |
| 3 | Отчисления на социальные нужды (страховые взносы) | 45 881 | 8,27 |
| 4 | Арендные платежи за производственные (офисные) помещения | 31 714 | 4,94 |
| 5 | Амортизация используемых основных средств и нематериальных активов | 26 522 | 4,13 |
| 6 | Расходы на модернизацию и приобретение основных средств | - | - |
| 7 | Расходы на приобретение необходимого ПО | - | - |
| 8 | Расходы на интернет, связь | - | - |
| 9 | Расходы на канцелярские товары и расходные материалы | 3 000 | 0,54 |
| 10 | Прочие расходы | 130 452 | 23,08 |
| **Итого:** | | **565 292** | **100** |

******

**Рис. 5 Структура себестоимости программного продукта**

1. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА
   1. Архитектура программного продукта

Архитектурное проектирование – процесс проектирования архитектуры, целью которой является определение совокупности компонент программного изделия и их интерфейсов, чтобы использовать их в качестве основы для последующей разработки программного изделия. Архитектурный проект должен охватывать все требования, сформулированные на этапе системного анализа.

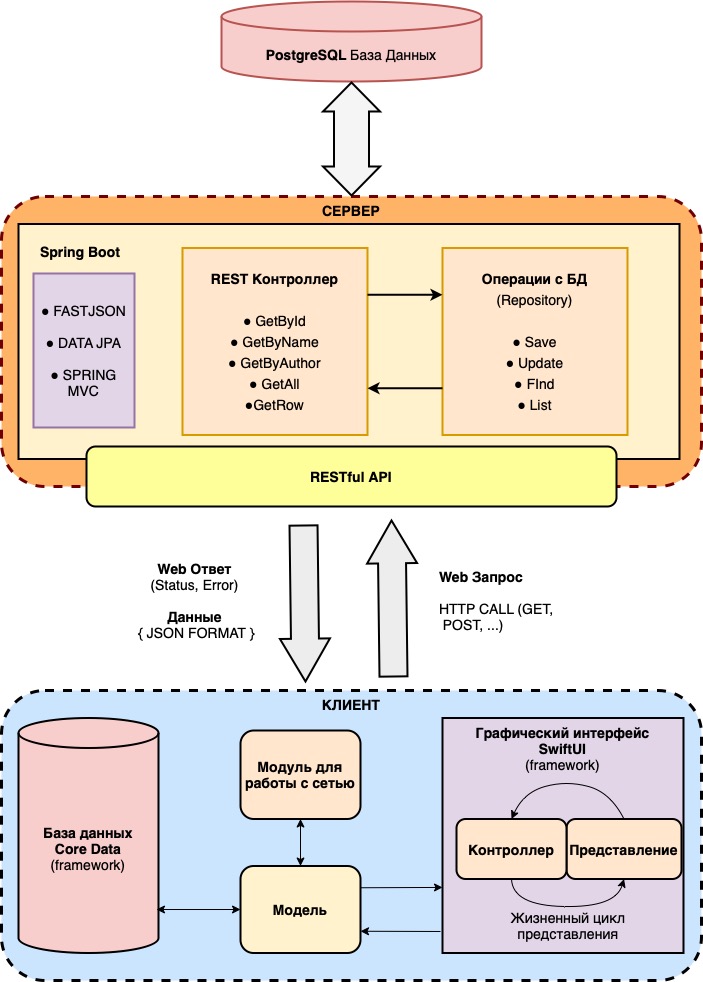
Мобильное приложение разработано в клиент-серверной архитектуре. Данная архитектура помогает решить многие вопросы, связанные с хранением и скоростью доступа к файлам [21]. Этот тип архитектуры не является самым простым и малозатратным, но используя данную архитектуру, мы получаем большое количество преимуществ, среди которых можно выделить:

* Централизованность;
* Производительность;
* Масштабируемость;
* Информационная безопасность.

Среди преимуществ приложения стоит отметить:

* Централизованность, дает возможность хранить все в одном месте, то есть в базе данных сервера, кроме этого, поддерживать информацию в актуальном состоянии, быстро исправлять ошибки, добавлять новую информацию. Работа происходит с данными на сервере, что никак не затрагивает пользователя и клиентскую сторону. Нет необходимости обновлять клиентскую часть или ждать пока исправят какую-либо проблему, связанную с данными.
* Производительность предполагает обособленность, свободу действий, делая упор не на мощности устройства, а на мощности сервера. Таким образом пользователю нет необходимости иметь мощное устройство. Все необходимые операции с большим количеством данных будет производить сервер.

На рис. 6. представлена клиент-серверная архитектура приложения.



**Рис. 6. Архитектурная схема проекта**

Клиент-серверная архитектура приложения представляет из себя две независимых друг от друга самостоятельных программы.

Первая это Restful API сервер, который имеет большую базу данных и программу обрабатывающие запросы к серверу [24]. Сервер устроен достаточно просто, но во всем приложении играет большую роль. Он берет на себя многие функции, которые реализуя на клиентской части, работали бы не так хорошо и быстро.

Вторая часть ‒ это клиентское приложение, которое устанавливается на телефон. Оно содержит в себе бизнес-логику, модуль для связи с сервером, для отправки запросов на сервер и принятию ответов от него, собственное хранилище для сохранения данных на устройство. Клиентская часть может работать в автономном режиме без подключения к сети, так как реализованный функционал позволяет загружать данные с сервера и хранить их на устройстве, также некоторый функционал, реализованный на серверной стороне, имеют и свою реализацию на клиентской, для работы в офлайн режиме [19].

Все приложение имеет следующее взаимодействие между клиентом и сервером:

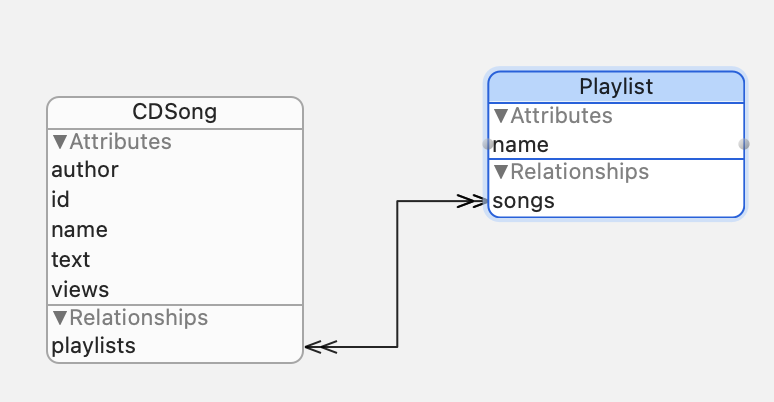
Клиент отправляет HHTP – запрос с выбранными параметрами на сервер, запрос соответствует API для взаимодействия с сервером.

Сервер принимает запрос, делает вызов к базе данных, выполняет логику соответствующего запроса, кодирует данные в формат JSON и передает их обратно клиенту.

Клиент принимает данные в формате JSON, с помощью модуля работы с сетью обрабатывает и выводит их в графическое представление.

* 1. Модель данных

При разработке клиент-серверного приложения необходимо решить вопрос хранения определенного набора данных. Для хранения данных на сервере использовалась одна таблица. Распределение и сортировка песен, производится с помощью бизнес-логики.



**Рис. 7 Модель базы данных приложения**

База данных мобильного приложения (клиентской части) представлена на рис. 7. Она состоит из 2 таблиц, которые связаны между собой.

Таблица «Playlist» служит для хранения основной информации о плейлисте, его название и список песен в нем.

Таблица «CDSong» служит для хранения информации о песне.

* + 1. Описание языков программирования

Для разработки данного мобильного приложения использовалось несколько языков программирования.

В качестве основного языка программирования был использован Swift. Swift ‒ это быстрый и эффективный язык программирования с откликом в реальном времени, который легко можно вставить в готовый код Objective-C [4]. На нем можно не только писать более надёжные и безопасные коды, но также экономить время и создавать приложения с расширенными возможностями. Приложение и весь его интерфейс был разработан на нем. Язык разработан самой компанией Apple на замену Objective-C, для создания программного обеспечения под свои операционные системы. Является достаточно простым, но имеет очень много фреймворков, для создания сложных приложений [14].

Серверная часть приложения написана на Java 17. Java является строго типизированным и объектно-ориентированным языком программирования, на котором разрабатывают достаточно сложные системы и серверы [6]. Благодаря Java-машине может быть почти на любой системе. Обладает огромным набор самых разных дополнений и библиотек, с помощью которых можно реализовать требуемый функционал.

* + 1. Описание сред разработки

Основной средой для разработки приложения, был Xcode.

Xcode – полноценная IDE (Integrated Development Environment - интегрированная среда разработки) от Apple для написания кода и сборки проекта. Среда предназначена для операционной системы macOS. Xcode позволяет создавать любые приложения под такие системы как, iOS, tvOS, watchOs, iPadOS [14]. Поддерживаются такие языки программирования как Swift, Objective-C, C, C++, AppleScript, а также в меньшей степени, Python, Ruby и Java. Использование Xcode является лучшим выбором для разработки под платформу IOS.

Для проверки работы приложения использовался встроенный в Xcode симулятор Simulator, который позволяет запустить приложение на любом устройстве Apple.

Simulator – вспомогательное програмное обеспечение, основной задачей которого является симуляция реального устройства под управлением операционной системы iOS. Симулятор имеет весь функционал, необходимый для разработки.

Для разработки пользовательского интерфейса использовался Sketch, это векторный редактор для разработки дизайна мобильных приложений.

Для разработки серверной части мобильного приложения требовалась среда, поддерживающая язык Java. Выбор был сделан в пользу продукта компании JetBrains.

InteliJ IDEA – коммерческая IDE (Integrated Development Environment - интегрированная cреда разработки) разработанная компанией JetBrains, в основном предназначенная для Java. Обладает широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Поддерживаются такие языки программирования как Java, JavaScript, CoffeeScript, Python, Ruby/JRuby, Groovy, SQL, PHP.

Для поддержки разрабатываемого приложения использовалась система контроля версий git.

Git – распределённая система управления версиями. Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

* + 1. Описание используемых фреймворков

Фреймворк – это структура, на базе которой можно создать финальный продукт. В данном проекте используются два основных фреймворка.

Для написания пользовательского интерфейса в Swift используется UIkit, который обеспечивает необходимую инфраструктуру для приложений iOS. Он предоставляет архитектуру окна и представления для реализации интерфейса, инфраструктуру обработки событий для доставки Multi-Touch и других типов ввода в ваше приложение, а также основной цикл выполнения, необходимый для управления взаимодействиями между пользователем, системой и вашим приложением. Однако несколько лет назад появился второй фреймворк позволяющий создавать интерфейсы для приложений SwiftUI, позволяющий разрабатывать приложения декларативным способом [17]. Это достаточно простой способ, разработчик сообщаем SwiftUI, как он хочет, чтобы пользовательский интерфейс выглядел и работал, путем добавлений и модификаций определённых структур наследуемых от View.

Для написания интерфейса приложения используется декларативный синтаксис SwiftUI, позволяющий создавать приложения для всех платформ Apple с минимальным количеством кода [27].

Приложению требуется хранение больших объемов данных. Для этого используется фреймворк Core Data, который применяется для управления объектами уровня модели в приложении. Он предоставляет обобщенные и автоматизированные решения общих задач, связанных с жизненным циклом объекта и управлением графом объекта, включая персистентность. Core Data существует в течение многих лет. Он используется в тысячах приложений миллионами людей, как на iOS, так и на OS X. Core Data поддерживается Apple и очень хорошо документирован. Это серьезный фреймворк, который доказывает свою эффективность. Core Data опирается в большей степени на Runtime Objective-C и искусно интегрирован с Core Foundation [18]. В результате использования фреймворка для управления объектным графом, он крайне эффективен в плане потребления памяти.

С помощью редактора модели данных Core Data определяются типы и отношения данных и создаются соответствующие определения классов. В последующем, Core Data может управлять экземплярами объектов во время выполнения, для обеспечения следующих функций. Core Data используется для хранения постоянных данных приложения, для использования в автономном режиме, для кэширования временных данных и для добавления функции отмены в приложение на одном устройстве. Также возможна синхронизация данных на нескольких устройствах в одной учетной записи iCloud, Core Data автоматически отражает схему в контейнере CloudKit.

На сервере используется популярный и удобный фреймворк Spring Boot.

Spring Boot упрощает создание автономных приложений производственного уровня на основе Spring, которые можно «просто запустить». Spring Boot дает возможность начать работу с минимальными усилиями. Большинству приложений на Spring Boot требуется только минимальная конфигурация Spring. Это полезный фреймворк, целью которого является упрощение создания приложений на основе Spring [11]. Он позволяет наиболее простым способом создать приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода обладает обширным функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов. Он обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов. Чтобы ускорить процесс управления зависимостями, Spring Boot неявно упаковывает необходимые сторонние зависимости для каждого типа приложения на основе Spring и предоставляет их разработчику посредством так называемых starter-пакетов (spring-boot-starter-web, spring-boot-starter-data-jpa и т.д.) [12].

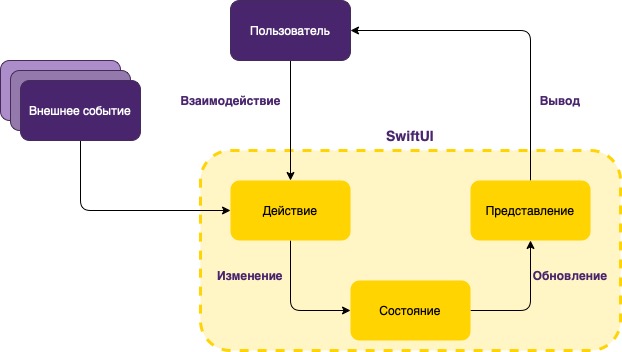
* 1. Описание структуры клиентской части

В SwiftUI реализован декларативный подход к разработке пользовательского интерфейса. Создавая иерархию представлений, указываются зависимости данных для этих представлений. При изменении данных в результате внешних событий, таких как уведомления или в результате действий пользователя, SwiftUI автоматически обновляет все связанные части интерфейса. То есть этот фреймворк выполняет практически всю работу, которую прежде выполняли контроллеры представлений [19].

Это однонаправленный подход: Действие – Состояние – Представление изображен на рис. 8. Больше не нужно задаваться вопросом о синхронизации UI и моделей данных. Обновить Представления может только Состояние, а обновить Состояние могут только Действия. Поскольку Представления не могут напрямую обновлять Состояние, синхронизация UI и моделей данных происходит автоматически.

Такая модель позволяет разрабатывать приложения быстро и не создавать отдельный контроллер к каждому представлению. Однако, в некоторых моментах необходим собственный контроллер, который будет обрабатывать определённые действия так, как нужно разработчику.

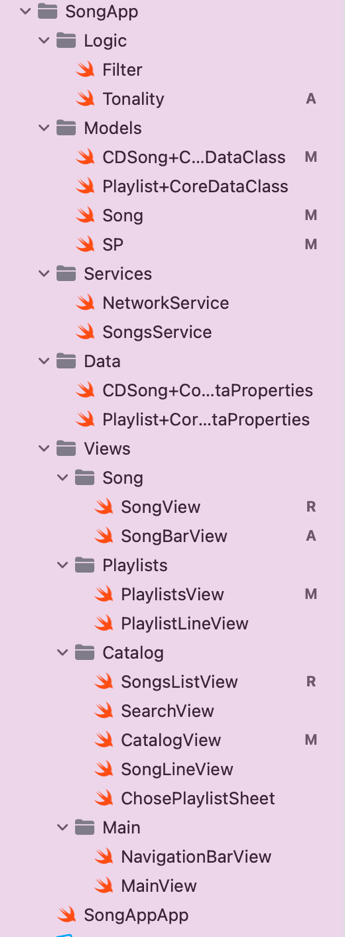
Декларативный подход развивается и дает свои как преимущества, так и недостатки, хоть и приходится привыкать к новым подходам в программировании [20].



**Рис. 8. Декларативный подход**

При разработке приложения структура проекта была выполнена с учетом декларативного подхода, поэтому большая часть логики, которая должна была быть реализована в контроллере, содержится внутри представлений [20].

Структура приложения представлена на рис. 9.



**Рис. 9. Структура приложения**

Краткое описание основных структур и классов:

Views – содержит все элементы и страницы интерфейса приложения.

* MainView.swift – это основное представление, которое открывается при запуске приложения (главный экран приложения).
* CatalogView – представление содержащие каталог песен, загруженных с сервера.
* PlaylistsView – представление, содержащие все созданные плейлисты и раздел избранное.
* NewsView – представление, содержащие новые композиции, историю просмотра и топ песен, по просмотрам.
* SongView – представление, содержащие текст песни и ее аккорды

Services – содержит классы сервисов, которые описывают основную бизнес логику.

* SongsService – класс служит для работы с песнями, преобразование их для вывода в представления, сохранения.
* NetworkService – класс, служащий для работы с сетью, отправки запросов и принятии ответов.
* Tonality – класс, реализующий логику смены тональностей.
* Filter – класс, отвечающий за настройки фильтра и поиска на стороне клиента.
  1. Описание структуры серверной части

Так как все приложение является клиент-серверным и реализует клиент-серверную архитектуру. Разработанный сервер по своей сути является RESTful веб-сервисом [24].

Приложение или архитектура RESTful, обладает следующими характеристиками:

* Состояние и функциональность представлены в виде ресурсов — это значит, что каждый ресурс должен быть доступен через обычные HTTP-запросы GET, POST, PUT или DELETE.
* Архитектура клиент-сервер, отсутствие состояния (stateless) и поддержка кеширования.

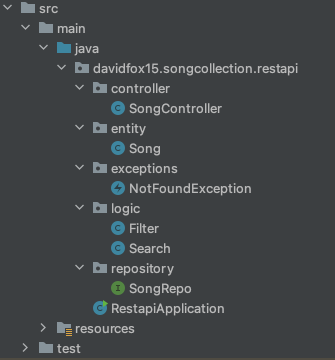
Архитектура без сохранения (stateless) состояния означает, что состояние приложения не сохраняется в REST. Например, если ресурс удален с сервера, то даже при получении положительного кода ответа нет гарантий, что он действительно был удалён. Чтобы убедиться, в действиях, необходимо отправить GET-запрос. С его помощью можно запросить ресурсы, чтобы посмотреть, присутствует ли там удалённый обьект.

Клиент-сервер ‒ архитектура, где сервером может быть веб-сервер, на котором размещено приложение, а клиентом ‒ любое приложение с интерфейсом способное отправлять HHTP запросы и принимать ответ [21]. Часть логики реализовано на сервере, часть на клиентской стороне, чтобы приложение могло работать автономно, без подключения к сети Интернет.

Структура сервера представляет собой контроллер, принимающий HHTP-запросы и работающий с базой данных. В качестве системы управления базами данных выступает PostgreSQL.

PostgreSQL использует объектно-реляционные модели, поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных [28]. Обеспечивает большую ёмкость данных и очень внимательно относится к целостности данных.

Структура сервера представлена на рис. 10.



**Рис. 10. Структура серверной части**

Краткое описание основных классов:

* SongController – контроллер, обрабатывающий запросы.
* Filter – класс, отвечающий за настройки фильтра и поиска на стороне сервера.
* Search – класс, отвечающий за поиск в базе данных.
* SongRepo – интерфейс, описывающий доступ к базе данных.
  1. Проектирование интерфейса

Одна из важнейших вещей для мобильного приложения является его интерфейс.

Клиенты мобильного приложения видят только его графическую составляющую. Простота, интуитивное понимание действий пользователя, не перегруженность различными элементами, скорость отклика являются преимущественными функциями, описывающими дизайн интерфейса.

Существует несколько основных критериев, описывающих качество интерфейса:

* скорость работы пользователей;
* количество человеческих ошибок;
* скорость обучения;
* субъективная удовлетворенность пользователей.

Мобильное приложение имеет несколько основных экрана, объединённых панелью навигации снизу, с помощью которой можно переходить между данными экранами.

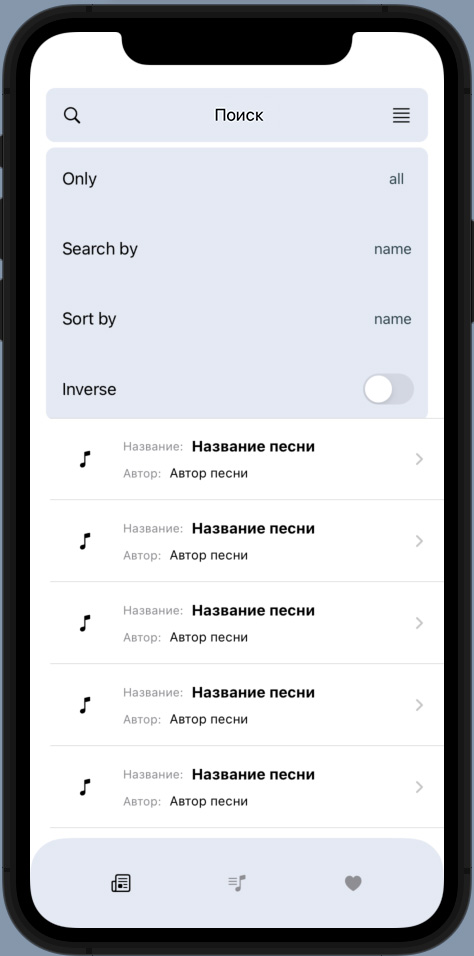
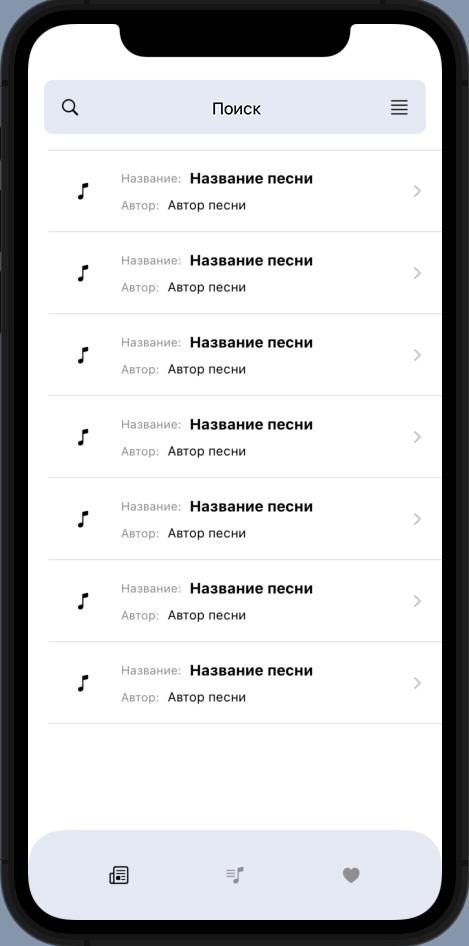
* + 1. Макет страницы каталога

На рис. 11 представлен макет экрана каталога, в котором можно выбрать необходимую композицию. Для удобства над списком песен в каталоге присутствует поисковая строка с надписью ‘поиск’ и значком лупы. Она поможет найти нужную песню.

Поиск может быть выполнен по:

* Названию;
* Автору;
* Тексту песни.

Справа от поисковой строки есть кнопка настройки. С помощью нее можно настроить работу поиска, а также сортировку песен и выбрать необходимые категории (см. рис. 11). Напротив избранных песен есть значок звездочки. Каждую песню в списке можно добавить в избранное или в плейлист, для этого нужно свайпнуть (потянуть) понравившуюся песню влево, после этого появятся 2 кнопки позволяющие это сделать.

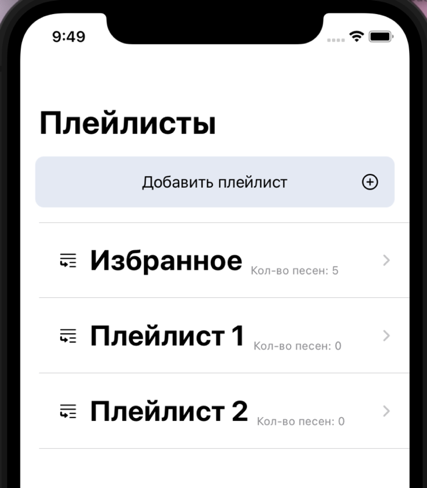
**а) б)**

**Рис. 11. Макет каталога:**

**а – каталог с настройками; б – каталог**

* + 1. Макет экрана с плейлистами

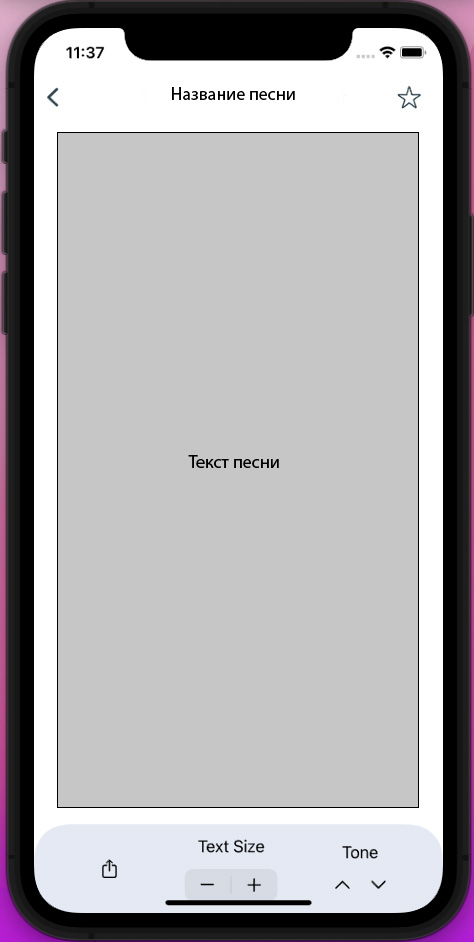
На рис. 12 представлен макет экрана с плейлистами. На нем можно создать новые плейлисты или посмотреть уже существующие с песнями внутри.



**Рис. 12 Макет плейлистов**

* + 1. Макет страницы с песней

Тапнув (нажав) по строке песни в каталоге, осуществляется переход на экран с ее текстом и аккордами (см. рис. 13). На экране с песней находится текст песни с аккордами. На данном экране имеются элементы с помощью, которых можно настраивать такие параметры как размер шрифта, менять тональность песни, добавлять ее в избранное. Также есть возможность поделиться песней.



**Рис. 13 Макет экрана с пеней**

1. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Раздел экспериментальной части необходим для того, чтобы понимать, корректно ли работает разработанное приложение и соответствует ли оно функциональным требованиям, составленным ранее. Корректно ли происходит работа по отправке и получению запросов от сервера, взаимодействие модулей приложения. Для этого необходимо провести некоторые испытания.

Главной целью тестирования программы, является проверка того, что реализованные функции работают корректно и соответствуют заявленным требованиям. Второстепенной задачей является поиск ошибок, чтобы ликвидировать нежелательное поведение системы, как, например, крах системы, нежелательное взаимодействие с другими системами, неправильные расчеты, поврежденные данные.

Испытание разработанного программного продукта будет производиться методом «черного ящика». Этот метод используется потому, что для его реализации достаточно лишь информации о требованиях и функциональной спецификации проекта.

* 1. План испытаний

Для проведения более качественного тестирования и устранения нежелательных ошибок во всем приложении, было принято решение провести отдельные испытания для серверной части приложения и для клиентской части.

Испытания для серверной части заключаются в проверке правильного функционирования API, принятия запросов и отправки ответов.

Испытания для клиентской части заключаются в проверке функционирования интерфейса, правильной работе бизнес-логики, тестировании работы сохранения песен и плейлистов на устройстве, а также отправке запросов на сервер.

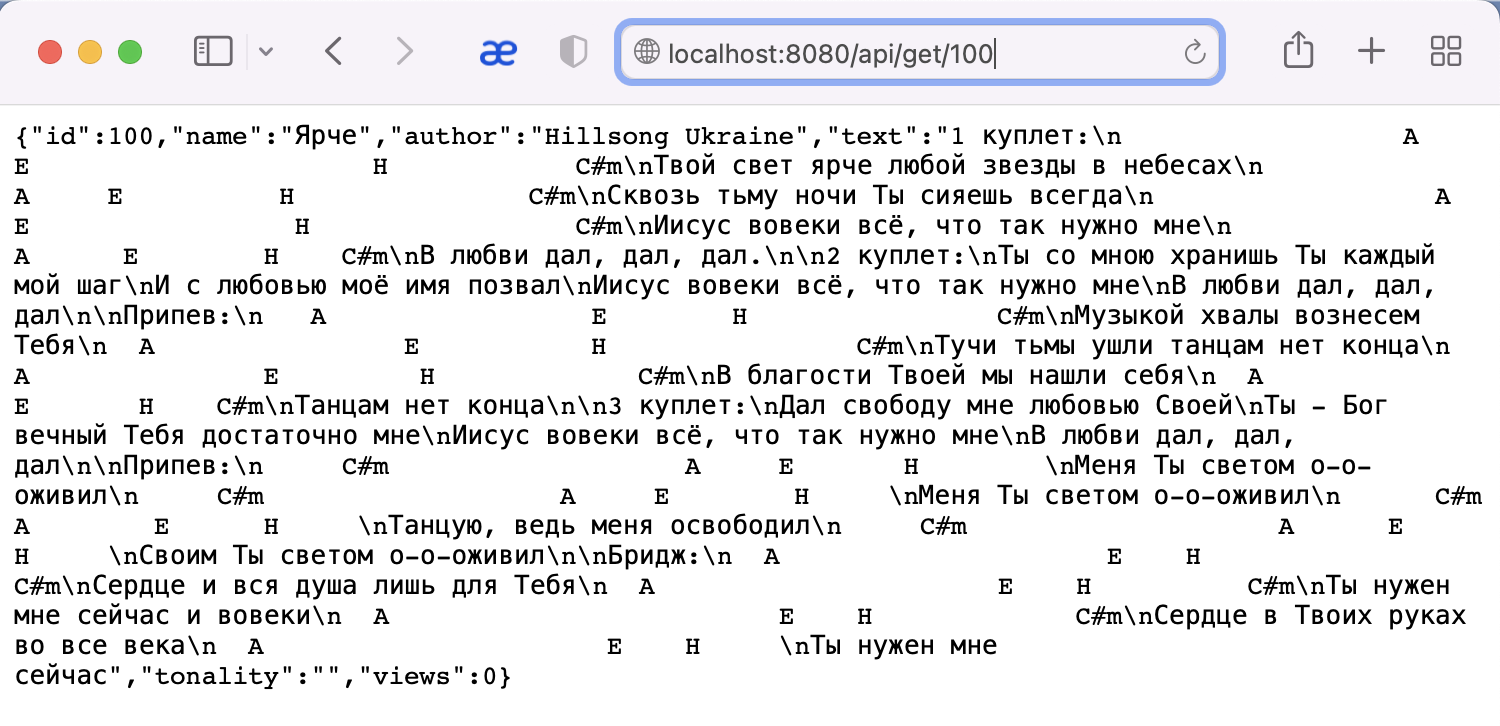
* 1. Испытания серверной части

В ходе выполнения дипломной работы был разработан рабочий сервер приложения. Он работает как Restful API сервер, который принимает запросы от клиентской части приложения обрабатывает их и отправляет ответ

Сервер поддерживает следующий функционал:

* Возвращение ответа на запрос в формате JSON.
* API для взаимодействия с клиентом.
* Сортировка и поиск в базе данных.
* Порционные ответы.

На рис. 14 представлен JSON формат данных, которые присылает сервер в ответ на запрос от клиента. В ответе мы видим название, автора, текст, тональность и идентификатор песни.

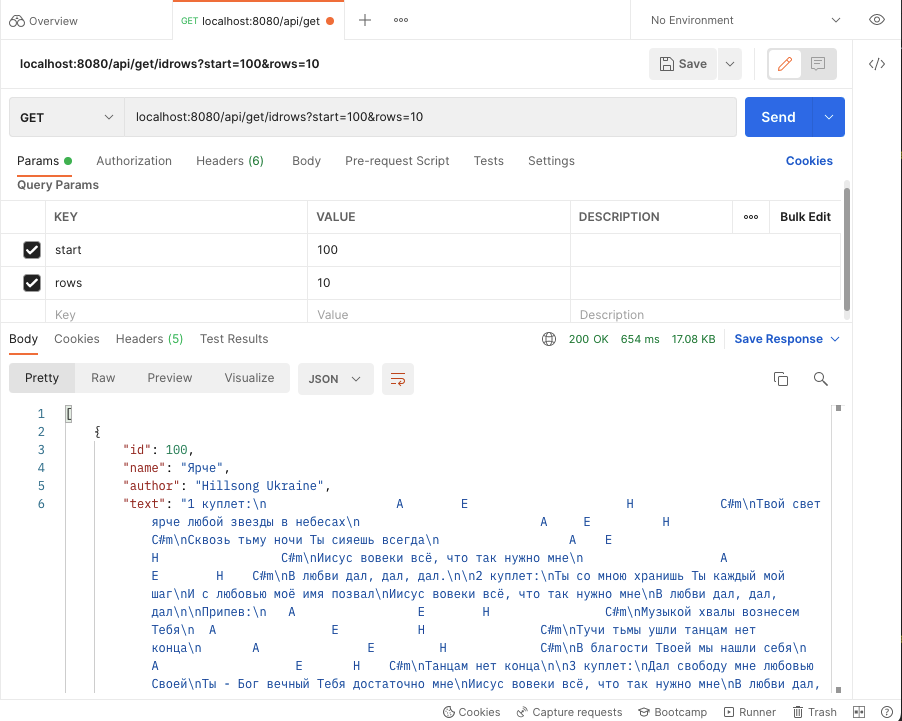


**Рис. 14. JSON ответ**

В REST можно применять как XML, так и JSON.

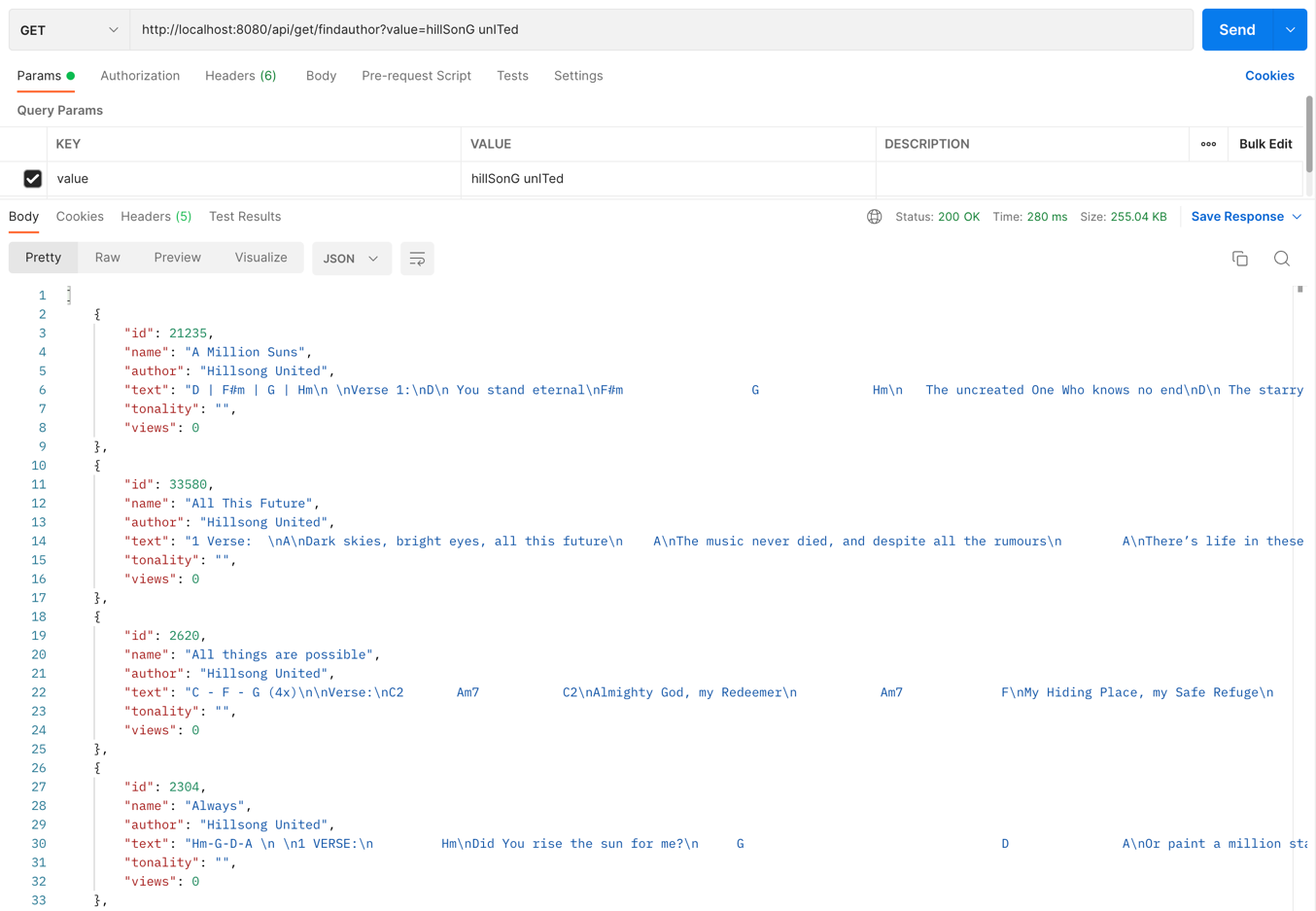
Для удобной работы клиента c сервером был разработан API. API помогает связать клиентскую часть с серверной и получать с помощью HTTP-запросов только нужные нам данные. Более того с помощью данных запросов реализована часть бизнес-логики приложения. При успешном ответе от сервера код ответа будет 200, то есть запрос успешно обработан.

Чтобы протестировать сервер и API приложение на соответствие поставленным целям использовалась программа Postman. Все API были протестированы на корректность работы, и правильность выдачи данных в формате JSON. На рис. 15 представлена работа одного из запросов через API и ответа в виде порции данных.



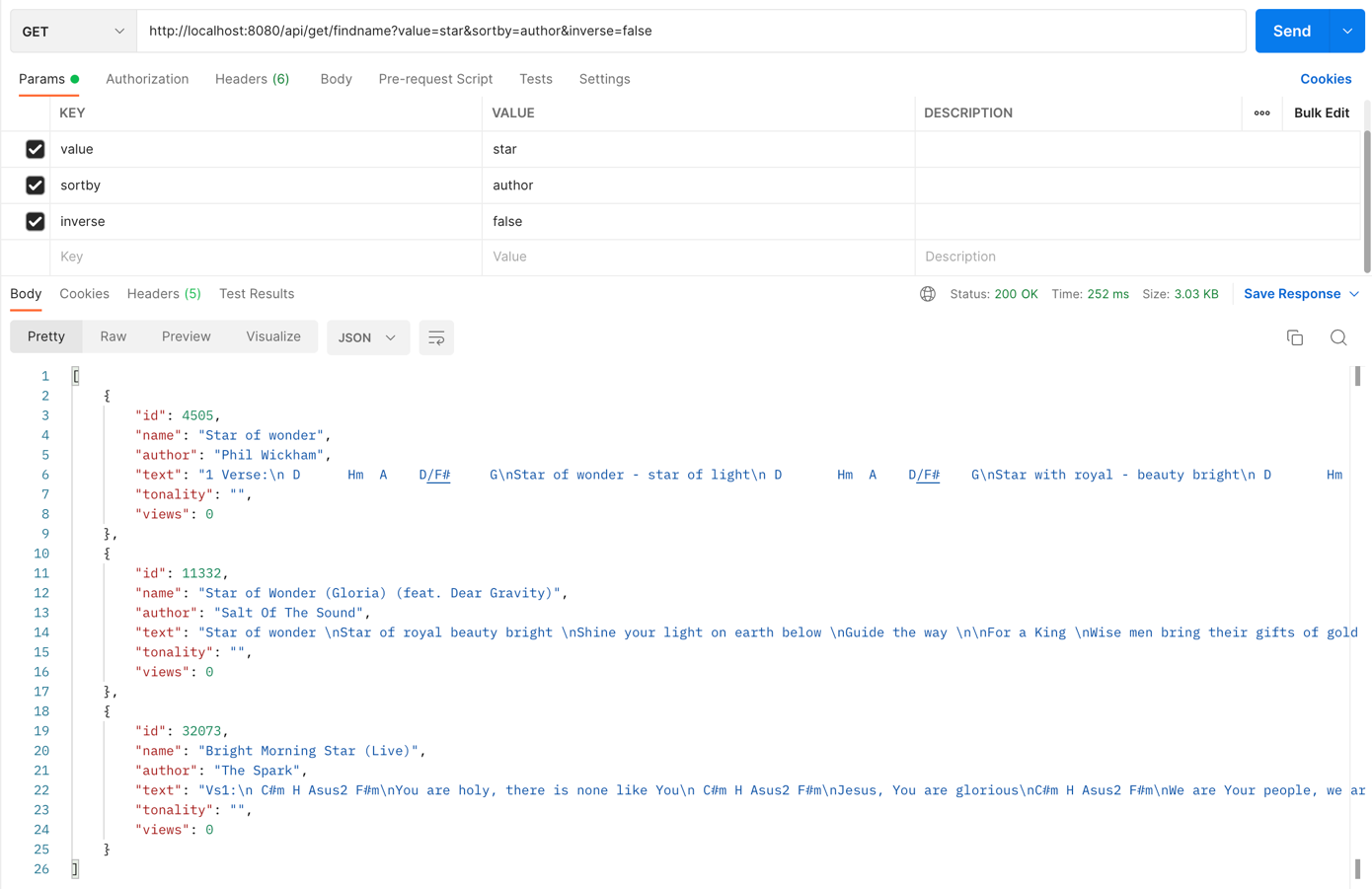
**Рис. 15. API запрос**

На рис. 16 представлен запрос по API на поиск всех песен по определённому автору. Для проверки текст набран с разными символами, строчными и прописными. Как видим поиск не чувствителен к регистру, поэтому в данном случае все выполняется корректно.



**Рис. 16. Поиск по автору**

В параметрах к каждому запросу можно задать тип сортировки и ее параметр, по убыванию или возрастанию. На рис. 17 представлен результат поиска по названию с сортировкой по автору песен в обычном порядке.



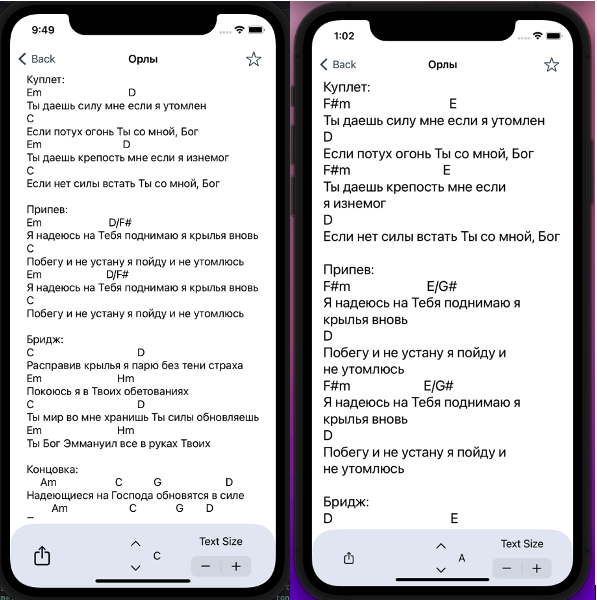
**Рис. 17. Поиск с сортировкой**

* 1. Испытания клиентской части

Реализация модулей клиентской части приложения была проверена на наличие ошибок и соответствию заявленному функционалу, также на удобство использования интерфейса приложения. В ходе выполнения дипломной работы для клиентской части, были протестированы основные экраны приложения.

Экран отображающий текст песни и ее аккорды. Функции, которые были протестированы:

* Смена тональности;
* Изменение размера шрифта;
* Экспорт.



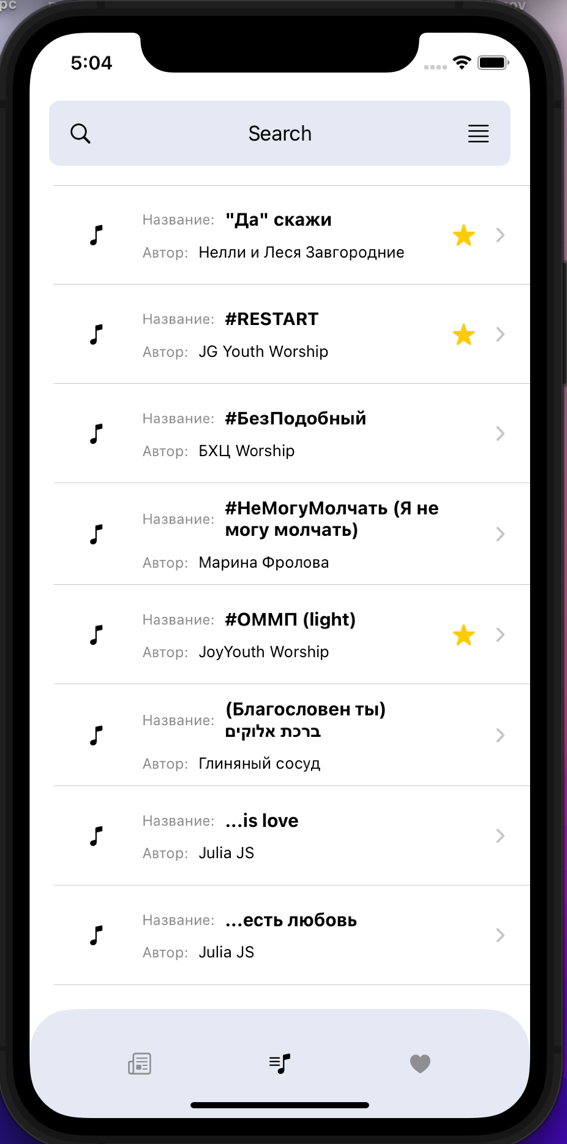
**а) б)**

**Рис. 18. Изменение размера шрифта и смена тональности:**

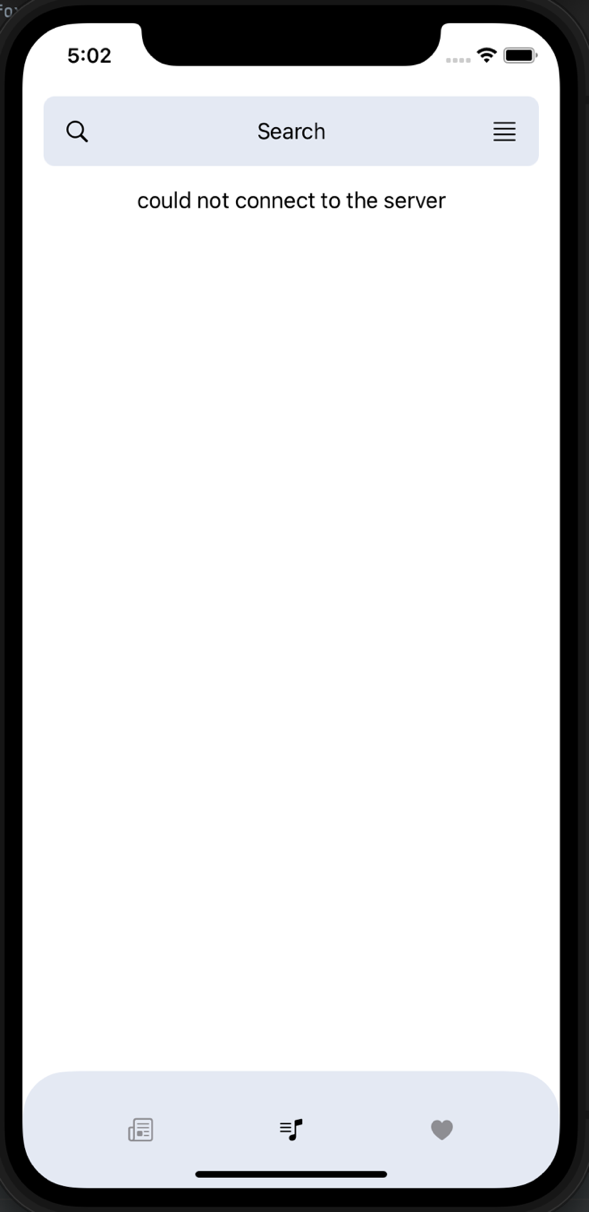
**a – тональность Em; б – тональность F#m**

На рис. 18 представлена наглядная работа функций смены тональностей и функция изменения шрифта. Кнопки изменения размера шрифта имеют ограничения, по максимальному и минимальному размеру. Смена тональностей работает исправно и не сбивается при прохождении полного круга, так как повышая или понижая тональности каждый аккорд идет по кругу переходя только в новую октаву.

На рис. 19 представлен каталог песен, получаемых с сервера. При отсутствии подключения к сети, страница выводит об этом сообщение (см. рис. 20).

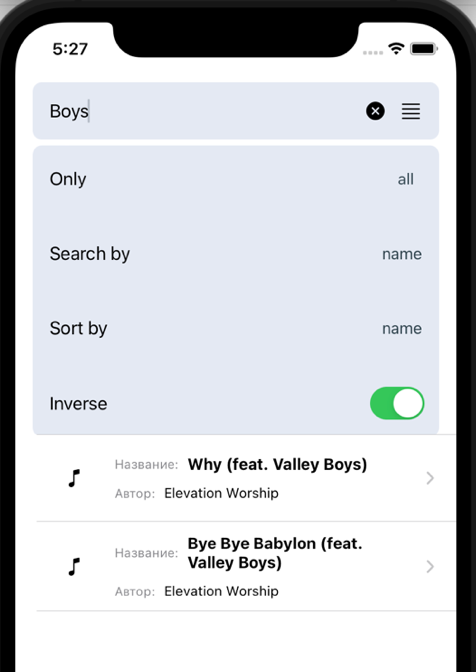


**Рис. 19. Каталог с избранными песнями**



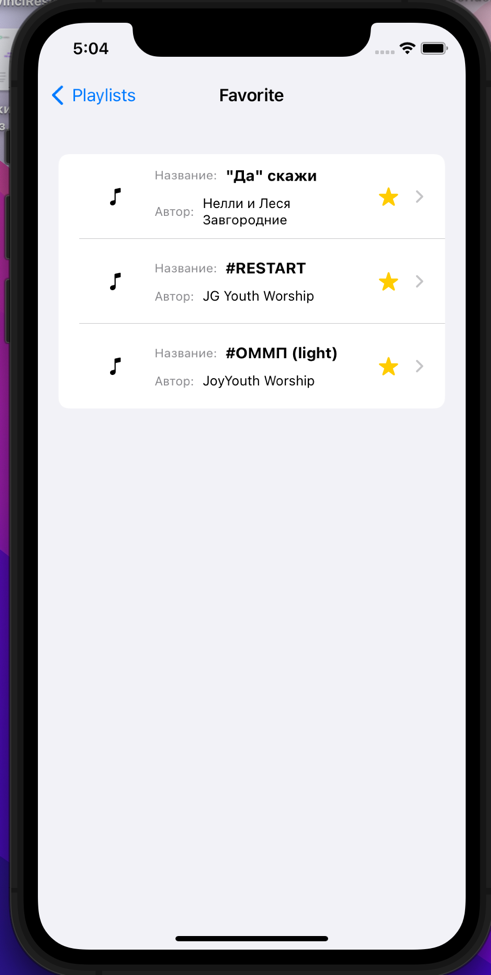
**Рис. 20. Отсутствие подключения**

Также была протестирована работа фильтра с разными параметрами и настройками на странице каталога. Поисковая строка работает корректно вместе с параметрами, выбранными в настройках поиска (см. рис. 21).



**Рис. 21 Поиск по каталогу**

На рис. 19 в избранное добавлено 3 песни. Мы можем их видеть в списке избранных песен на рис. 22. Они хранятся на самом устройстве, следовательно при перезапуске приложения они остаются в памяти устройства и не загружаются снова.



**Рис. 22 Список избранных**

1. ОРГАНИЗАЦИОННАЯ ЧАСТЬ
2. Тип лампы – люминесцентная.
3. Тип светильника – ЛД.
4. Освещенность – *E* = 350 лк.
5. Длина помещения – *a* = 4 м.
6. Ширина помещения – *b* = 6 м.
7. Высота помещения – *H* = 3 м.
8. Высота рабочей поверхности – *hраб. пов.* = 0,8 м.
9. Высота светильника – *hсв.* = 0,2 м.
10. Высота подвеса светильника – *hp* = 2 м.

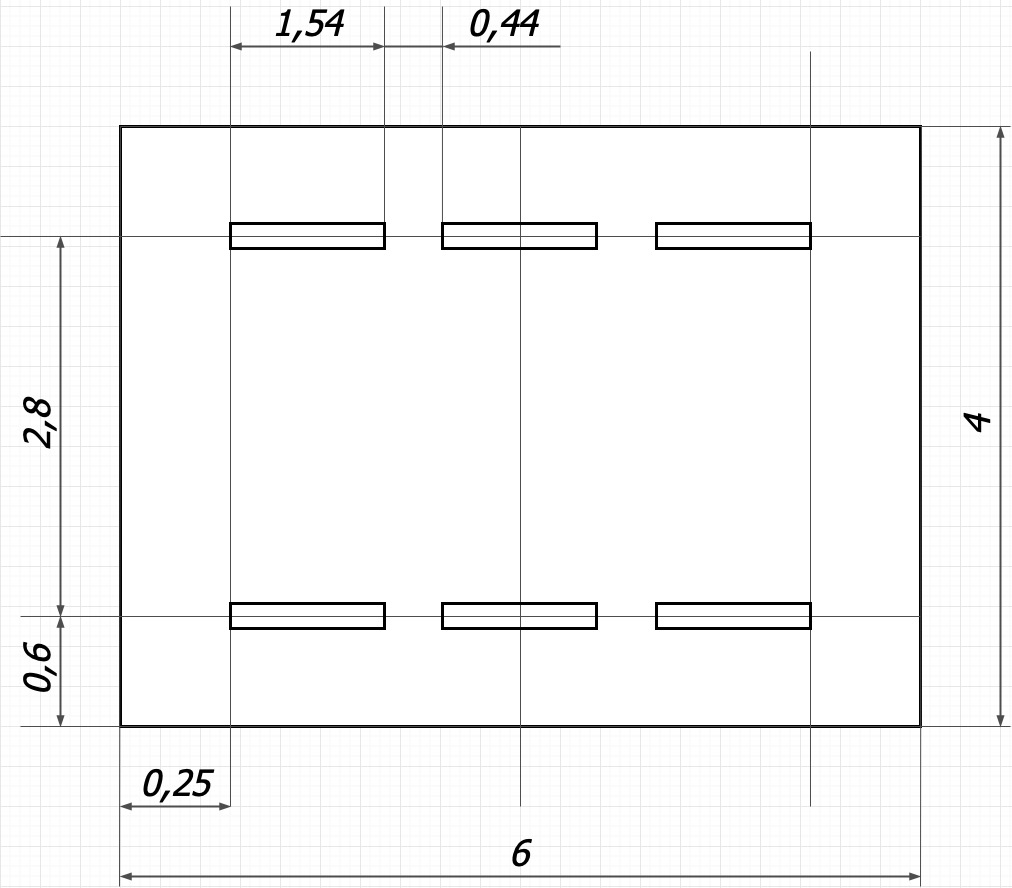
Результаты расчетов освещения обобщены в Таблица 3, а сами расчеты представлены в прил. 3.

**Таблица 3**

***Результаты расчётов параметров осветительной установки***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тип  лампы | Световой поток лампы **Ф**, лм | Количество  светильников | | Отклонение  **nпр** от **nрасч**, % | Мощность  лампы, Вт | Полная мощность **N**, Вт |
| Расчётное **nрасч** | Принятое **nпр** |
| ЛД-65 | 3390 | 5,36 | 6 | 11,9 | 65 | 780 |
| ЛД-80 | 3865 | 4,7 | 6 | 27,6 | 80 | 960 |

Вывод: оптимальным вариантом осветительной установки является установка с лампами ЛБ-65, так как установка другого варианта не обеспечивает заданного уровня освещенности. Схема расположения светильников приведена на рис. 23.



**Рис. 23 Схема расположения светильников**

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы было разработано мобильное приложение для сборника песен и аккордов. В ходе выполнения дипломной работы были произведены следующие действия:

Произведен анализ предметной области, сделаны выводы об актуальности разработки мобильного приложения сборника песен, а также был проведен сравнительный анализ аналогичных приложений. В процессе анализа были выявлены основные требования к программе на основе которых был составлен контрольный список.

Была разработана архитектура программного продукта и требования к среде разработки и программному коду, по этим критериям были разработаны необходимые функции и части клиент-серверного приложения. Были описаны основные средства разработки, структура клиентской части, структура серверной части, проектирование интерфейса.

При разработки клиентской части приложения были получены: навык разработки мобильных приложений, опыт работы в среде Xcode, знания новых фреймворков, таких как SwiftUI и Core Data, подходов к разработке мобильных приложений и построения их внутренней архитектуры.

При разработке серверной части приложения были получены: опыт работы в среде InteliJ IDEA, навыки использования Spring Boot, построения Restful API и настройки сервера.

Разработанное приложение было успешно испытано. В ходе испытаний был протестирован мобильный интерфейс приложения, основная бизнес-логика, протестирован сервер и его API с помощью программы Postman, на корректность выдаваемых данных.

Все поставленные задачи на дипломную работу были выполнены.

Дальнейшими направлениями развития приложения является расширение различных функциональных возможностей: наличие обучающих материалов, добавления прослушивания песни, доработка новостного экрана и добавление новых функций.

СПИСОК ЛИТЕРАТУРЫ

1. Использование мобильных устройств. – Режим доступа: <https://datareportal.com/reports/digital-2021-russian-federation> дата обращения: 17.05.2022.
2. Усов В. Swift. Основы разработки приложений под iOS и macOS. 4-е Издание / Василий Усов – Питер., 2018 – 448с.
3. Аллан. А. Программирование для мобильных устройств на iOS: Профессиональная разработка приложений для iPhone, iPad, andiPodTouch / А. Аллан– СПб.: Питер, 2016. - 416 c.
4. Д. Марк. Swift: разработка приложений в среде Xcode для iPhone и iPad с использованием iOs SDK / Д. Марк. – М.: Вильямс И.Д., 2015. – 816 c.
5. Хорстман К. CoreJava SE 9 для нетерпеливых. 2-е Издание / К. Хорстман 2015.
6. Блох Д. Java. Эффективное программирование. Третье издание / Джошуа Блох пер. с англ. – Диалектика, 2019. – 464с.
7. Хеклер М. SpringBoot по-быстрому / Марк Хеклер. пер. с англ. – Питер, 2022. – 352с.
8. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка / Мартин Клеппман. пер. с англ. – Питер, 2022. – 640с.
9. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих / Бхаргава Адитья. пер. с англ. – Питер, 2022. – 288с.
10. Порселло Е. GraphQL: язык запросов для современных веб-приложений / Порселло Е., Бэнкс А. пер. с англ. – Питер, 2019. – 240с.
11. Раджпут Д. Spring. Все паттерны проектирования / Раджпут Д. пер. с англ. – Питер, 2019. – 320с.
12. Гутьеррес Ф. SpringBoot 2: лучшие практики для профессионалов / Гутьеррес Ф. пер. с англ. – Питер, 2022. – 604с.
13. Хоффман Э. Безопасность веб-приложений / Хоффман Э. пер. с англ. – Питер, 2022. – 336с.
14. Маскри М. Swift 3: разработка приложений в среде Xcode для iPhone и iPad с использованием iOs SDK / М. Маскри. – М.: Диалектика, 2017. – 896 c.
15. Грей Э. Swift. Карманный справочник. Программирование в среде iOS и ОS X. 2-е Издание / Энтони Грей, 2016. – 435с.
16. Native vs. Hybrid vs. Web App. – Which is the best choice for your business and why? – Режим доступа: <https://sophiamartin121.medium.com/native-vs-hybrid-vs-web-app-f95c054a3c02> дата обращения 8.06.2022.
17. SwiftUI по полочкам / Хабр. – Режим доступа: <https://habr.com/ru/post/485548> дата обращения: 8.06.2022.
18. Core Data And SwiftUI 2.0 – Saving, Retrieving, Updating And Deleting Persistent Data. – Режим доступа: <https://blckbirds.com/post/core-data-and-swiftui> − дата обращения: 9.06.2022.
19. SwiftUI 3.0.– Saving, Retrieving, Updating And Deleting Persistent Data. – Режим доступа: <https://swiftbook.ru/post/tutorials/swiftui-3-0-parameter-format-for-text-type> – дата обращения: 13.06.2022.
20. Архитектурные паттерны в iOS. – Режим доступа: <https://habr.com/ru/company/badoo/blog/281162> дата обращения: 13.06.2022.
21. Клиент-серверная архитектура. – Режим доступа: <https://habr.com/ru/post/495698> дата обращения: 13.06.2022.
22. Как начать разрабатывать приложения для iOS: FAQ и полезные ссылки. – Режим доступа: <https://tproger.ru/articles/kak-nachat-razrabatyvat-prilozhenija-dlja-ios-faq-i-poleznye-ssylki> дата обращения: 13.06.2022.
23. Разгоняем производительность iOS-приложений – Режим доступа: <https://habr.com/ru/company/edison/blog/315150> – дата обращения: 14.06.2022.
24. Введение в REST API — RESTful веб-сервисы. – Режим доступа: <https://habr.com/ru/post/483202> дата обращения: 13.06.2022.
25. Apple Developer Documentation. – Режим доступа: <https://developer.apple.com/documentation> – Дата обращения: 26.05.2022.
26. The swift programming language swift. – Режим доступа: <https://docs.swift.org/swift-book> дата обращения: 28.05.2022.
27. IntroducingSwiftUI. – Режим доступа: <https://developer.apple.com/tutorials/swiftui> – Дата обращения: 26.05.2022.
28. Лузанов П. PostgreSQL для начинающих / П. Лузанов, Е. Рогов, И. Левшин – Питер 2017 – 116 с.

**ПРИЛОЖЕНИЕ**

Приложение 1

**ЭКОНОМИЧЕСКАЯ ЧАСТЬ**

Перед началом разработки любой программной системы необходимо выяснить, насколько целесообразна ее разработка, как с точки зрения полезности, так и с точки зрения экономической эффективности. В технико-экономической части работы рассматриваются вопросы организации работ по созданию и внедрению программной системы, а также приводится расчёт ее себестоимости.

***Организационная структура***

Организационная структура проекта (OBS) приведена на рис. 1.

****

**Рис. 1 Организационная структура**

***Календарный план***

Для оценки расходов на реализацию проекта в числе прочих необходимо определить временные затраты на его реализацию. Для определения временных затрат проекта необходимо разработать календарный план проекта. Реализуемый проект является типовым для Компании, исходя из этого, был сформирован состав работ проекта, определена их длительность, а также распределение ресурсов по ним. При разработке календарного плана были учтены ограничения и допущения, накладываемые на проект Заказчиком.

Для реализации проекта необходимо выполнить следующие работы:

1. Сбор требований Заказчика к разрабатываемому ПО.
2. Анализ полученных требований, формирование функциональных требований к разрабатываемому ПО.
3. Разработка дизайна
4. Разработка ПО.
5. Тестирование ПО.

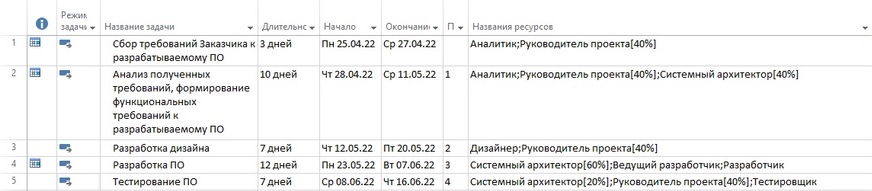
Распределение человеческих ресурсов по работам проекта и степень их загрузки приведены в табл. 1.

**Таблица 1**

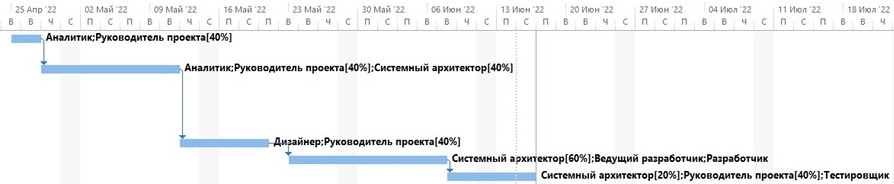
***Структура общего времени на создание программного продукта***

| **№**  **этапа** | **Этап работ** | **Ответственные исполнители (занятость на этапе)** | **Длительность, дней** |
| --- | --- | --- | --- |
| 1 | Сбор требований Заказчика к разрабатываемому ПО | * Аналитик (постановщик задач) [100%] * Руководитель проекта [40%] | 3 |
| 2 | Анализ полученных требований, формирование функциональных требований к разрабатываемому ПО | * Аналитик (постановщик задач) [100%] * Руководитель проекта [40%] * Системный архитектор [40%] | 10 |
| 3 | Разработка дизайна | * Дизайнер [100%] * Руководитель проекта [40%] | 5 |
| 4 | Разработка ПО | * Системный архитектор [60%] * Ведущий разработчик [100%] * iOS Разработчик [100%] | 12 |
| 5 | Тестирование ПО | * Тестировщик [100%] * Системный архитектор [20%] * Руководитель проекта [40%] * iOS Разработчик [30%] | 7 |

При реализации данного проекта работы выполняются последовательно. Диаграмма Ганта приведена на рис. 2 и рис. 3.



**Рис. 2 Табличное представление Диаграммы Ганта**



**Рис. 3 Графическое представление Диаграммы Ганта**

Исходя из длительности работ и коэффициента загрузки членов проектной команды, определим их трудозатраты при реализации проекта (см. табл. 2).

|  |  |  |
| --- | --- | --- |
| **№** | **Исполнитель** | **Трудозатраты, человеко-часов** |
| 1 | Руководитель проекта | 80 |
| 2 | Аналитик (постановщик задач) | 104 |
| 3 | Ведущий разработчик | 96 |
| 4 | Разработчик | 113 |
| 5 | Системный архитектор | 101 |
| 6 | Тестировшик | 56 |
| 7 | Дизайнер | 40 |

**Таблица 2**

***Трудозатраты членов проектной команды***

***Расчёт затрат на разработку продукта***

Расчет затрат на создание и внедрение ПО включает следующие составляющие с последующим их графическим представлением в виде круговой диаграммы:

* заработная плата исполнителей работ по проекту – *ЗПосн*;
* дополнительная заработная плата *ЗПдоп*;
* заработная плата обслуживающего и административного персонала;
* отчисления на социальные нужды (страховые взносы) – *Нзп*;
* арендные платежи за производственные (офисные) помещения – *Апм*;
* амортизация используемых основных средств и нематериальных активов – *А*;
* расходы на модернизацию и приобретение основных средств – *Рмод*;
* расходы на приобретение необходимого ПО – *РПО*;
* расходы на интернет, связь – *Ртел*;
* расходы на канцелярские товары и расходные материалы – *Рр.м.*;
* прочие расходы – *Пр.р.*.

***Расчёт заработной платы исполнителей работ по созданию программного продукта***

Основная ЗП определяется по формуле:

|  |  |
| --- | --- |
|  | (1.1) |

где *M* – месячная зарплата (*руб.*), *T* – общие трудозатраты (*чел.-ч*), *Чр* – число рабочих дней в месяц, *tр.д. –* продолжительность рабочего дня в часах, *П* – процент премии. В данной работе *Чр =* 21 *день*, *tр.д.=*8*ч*, *П*=0.

Значение месячной заработной платы (М), суммарные трудозатраты членов, а также рассчитанная по формуле 1.1 основная заработная плата проектной команды приведены в табл. 3.

**Таблица 3**

***Основная заработная плата членов проектной команды***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Исполнитель** | **Месячная заработная плата (М), руб.** | **Трудозатраты, человеко-часов** | **, руб.** |
| 1 | Руководитель проекта | 120 000 | 80 | 57 143 |
| 2 | Аналитик (постановщик задач) | 100 000 | 104 | 61 905 |
| 3 | Ведущий разработчик | 80 000 | 96 | 45 714 |
| 4 | Разработчик | 65 000 | 113 | 43 643 |
| 5 | Системный архитектор | 80 000 | 101 | 48 000 |
| 6 | Тестировщик | 50 000 | 56 | 16 667 |
| 7 | Дизайнер | 50 000 | 40 | 11 905 |

Суммарное значение основной заработной платы проектной команды на период реализации проекта составит 284 976 (руб.).

Дополнительная заработная плата берется в размере 15% от основной.

ЗПдоп=42 746 (руб.).

Общая заработная плата составит 327 722 (руб.).

Проект реализуется в небольшой ИТ-компании, где доля вспомогательного и административного персонала по отношению к основному персоналу не велика. Большая часть административного персонала задействована в проектной деятельности в качестве руководителей проекта. Кадровый учет, бухгалтерский и налоговый учет в Компании отдан на аутсорсинг. Затраты на аутсорсинг войдут в прочие расходы. В связи с этим примем заработную плату обслуживающего персонала равной 0 руб.

***Расчёт отчислений на социальные нужды (страховые взносы)***

Теперь можно рассчитать величину отчислений на социальные нужды (страховые взносы), которые начисляются на заработную плату и в 2022 г. для организаций, осуществляющих деятельность в области информационных технологий, составляют 14% по выплатам в пределах 309 тыс. руб. Структура отчислений на социальные нужды (страховые взносы) приведена в табл. 4.

**Таблица 4**

Структура отчислений на социальные нужды (страховые взносы)

|  |  |
| --- | --- |
| Пенсионный фонд Российской Федерации | 8,0% |
| *для лиц 1966 года рождения и старше* |  |
| страховые взносы на страховую часть трудовой пенсии | 8,0% |
| *для лиц 1967 года рождения и моложе* |  |
| страховые взносы на страховую часть трудовой пенсии | 2,0% |
| страховые взносы на накопительную часть трудовой пенсии | 6,0% |
| Фонд социального страхования Российской Федерации | 2,0% |
| Федеральный фонд обязательного медицинского страхования | 4,0% |

Таким образом, *Нзп=* 45 881 (*руб.*).

***Арендные платежи за производственные (офисные) помещения***

Компания, реализующая проект по разработке мобильного приложения, арендует офисные помещения в г. Брянск.

Стоимость аренды составляет 600 руб/м2 в месяц.

Арендная плата включает в себя оплату как площади занимаемых Компанией помещений, так и электроэнергии, отопления, водоснабжения, кондиционирования и уборки помещений, вывоза и утилизации технико-бытовых отходов, парковочных мест на автостоянке.

На каждого члена проектной команды приходится 5 м2 арендуемого офисного помещения. На период данного проекта члены проектной команды в других проектах не задействованы.

Исходя из изложенного выше, затраты на аренду помещений, отнесенные на проект составят *Апм* = 31 714 (руб.).

***Амортизация используемых основных средств и нематериальных активов***

При реализации проекта по разработке мобильного приложения задействованы следующие основные средства:

* 7 ноутбук первоначальной стоимостью 70 200 (руб.).

Срок полезного использования для задействованных в проекте основных средств определен в 3 года. Метод начисления амортизации – линейный.

Амортизационные отчисления для ноутбука на 1 месяц составят

70 200 / 36 = 1 950 (руб.).

Амортизационные отчисления по ОС, относящиеся на проект, составят:

*(руб.).*

Ранее (до начала проекта) Компанией были приобретены:

* Корпоративная лицензия на IntelliJ IDEA Ultimate стоимостью 32 500 (руб.).
* Аккаунт разработчика Apple Developer 6 000 (руб.).

Данное ПО принимается Компанией к учету как расходы будущих периодов со сроком списания 3 года. Метод списания – линейный.

В качестве ОС используется установленное по умолчанию ПО MacOS. В качестве сервера БД используется свободно распространяемое ПО – PostgreSQL.

Амортизационные отчисления по РБП, относящиеся на проект, составят:

*(руб.).*

Суммарные амортизационные отчисления составят: А=26 522 руб.

***Расходы на модернизацию и приобретение основных средств***

При реализации проекта по разработке мобильного приложения не планируется приобретение новых и модернизация существующих основных средств.

***Расходы на приобретение необходимого ПО***

При реализации проекта не планируется приобретение ПО.

***Расходы на интернет и связь***

Так как в Компании, реализующей проект не производится биллинг и тарификация телекоммуникационных услуг в разрезе сотрудников, затраты на интернет и связь войдут в прочие затраты, рассчитываемые как процент от прямых затрат.

***Расходы на канцелярские товары и расходные материалы***

Затраты на расходные материалы берутся по факту и составляют = 3 000 (*руб.*). К данным затратам относятся затраты на канцтовары, тонер и бумагу для принтера и т.д.

***Прочие расходы***

Прочие расходы составляют 30% от суммы следующих элементов структуры затрат: *ЗПосн*, *ЗПдоп*, *Нзп*, *Апм*, *А*, *Рмод*, *РПО*, *Ртел* и *Рр.м.*.

|  |  |
| --- | --- |
|  | (1.2) |

Таким образом, *Пр.р.=* 130 452 (*руб.*).

***Расчёт себестоимости программного продукта***

В себестоимость программного продукта входят следующие элементы: *ЗПосн*, *ЗПдоп*, *Нзп*, *Апм*, *А*, *Рмод*, *РПО*, *Ртел*, *Рр.м.* и *Пр.р.*.

Сложив все элементы, можно определить себестоимость программного продукта и услуг по его внедрению: *Сп.п.*= 565 292 (*руб.*).

Приложение 2

**ОРГАНИЗАЦИОННАЯ ЧАСТЬ**

Работа программистов, операторов, технических работников при решении производственных задач сопровождается активизацией внимания и других психологических функций. На сотрудников во время работы влияет множество как вредных, так и опасных факторов. В последние годы совершенствованию условий труда пользователей ПК уделяется значительное внимание.

В связи с особенностями характера и режима работы и из-за значительного умственного напряжения у работников может наблюдаться изменение, как функционального состояния нервной системы, так и нервно-мышечного аппарата рук. Нерациональные конструкция и размещение элементов рабочего места вызывают необходимость поддержки неудобной рабочей позы, а длительный дискомфорт приводит к увеличению напряжения мышц и обуславливает развитие общей усталости и снижение работоспособности. Также от длительной работы за экраном монитора страдает зрительный аппарат, появляется частая головная боль, раздражительность, усталость. Кроме того, могут возникать болезненные ощущения в глазах, пояснице, в области шеи, рук. Особенность опасностей жизнедеятельности человека, исходящих от компьютера, заключается в том, что они невидимы, но оказывают сильное отрицательное воздействие на человека.

Поэтому на рабочем месте должны быть обеспечены комфортные условия, соответствующие нормам СанПиН 1.2.2 3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания», СП 2.2.3670-20 «Санитарно-эпидемиологические требования к условиям труда"».

***Опасные и вредные факторы при работе с ПЭВМ***

При выполнении работ с использованием ПЭВМ можно выделить ряд опасных и вредных производственных факторов. В процессе труда на оператора ПЭВМ оказывают действие следующие опасные и вредные производственные факторы:

1. Физические:

* повышенные уровни электромагнитного, рентгеновского, ультрафиолетового и инфракрасного излучения;
* повышенный уровень статического электричества;
* повышенные уровни запыленности воздуха рабочей зоны;
* пониженная или повышенная влажность и подвижность воздуха рабочей зоны;
* повышенный уровень шума;
* повышенный или пониженный уровень освещенности;
* повышенный уровень прямой и отраженной блесткости;
* повышенный уровень ослепленности;
* неравномерность распределения яркости в поле зрения;
* повышенная яркость светового изображения;
* повышенный уровень пульсации светового потока;
* повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;
* повышенное содержание положительных аэроионов в воздухе рабочей зоны.

1. Химические – повышенное содержание в воздухе рабочей зоны двуокиси углерода, озона, аммиака, фенола, формальдегида и полихлорированных бифенилов;
2. Психофизиологические:

* напряжение зрения и внимания;
* интеллектуальные и эмоциональные нагрузки;
* длительные статические нагрузки и монотонность труда;
* большой объем информации, обрабатываемой в единицу времени;
* нерациональная организация рабочего места.

Биологические – повышенное содержание в воздухе рабочей зоны микроорганизмов.

***Профилактическая гимнастика***

Для снятия статического и нервно-эмоционального напряжения можно использовать обычные физические упражнения, преимущественно для верхней части туловища (рывки руками, повороты, «рубка дров» и т.д. Для снятия напряжения зрения рекомендуется зрительная гимнастика. Даже при небольшой ее продолжительности (1 мин), но регулярном проведении, она является эффективным мероприятием профилактики утомления. Эффективность зрительной гимнастики объясняется тем, что при выполнении специальных упражнений (описаны ниже) обеспечивается периодическое переключение зрения с ближнего на дальнее, снимается напряжение с цилиарной мышцы глаза, активизируются восстановительные процессы аккомодационного аппарата глаза, в результате чего функция зрения нормализуется. Кроме того, есть специальное упражнение (с меткой на стекле), предназначенное для тренировки и развития аккомодационной функции глаза.

Упражнения выполняются сидя или стоя, отвернувшись от экрана при ритмичном дыхании, с максимальной амплитудой движения глаз.

Упражнения для глаз:

1. Закрыть глаза, сильно напрягая глазные мышцы, на счет 1-4, затем раскрыть глаза, расслабив мышцы глаз, посмотреть вдаль на счет 1-6. Повторить 4-5 раз.
2. Посмотреть на переносицу и задержать взор на счет 1-4. До усталости глаза не доводить. Затем открыть глаза, посмотреть вдаль на счет 1-6. Повторить 4-5 раз.
3. Не поворачивая головы, посмотреть направо и зафиксировать взгляд на счет 1-4, затем посмотреть вдаль прямо на счет 1-6. Аналогичным образом проводятся упражнения, но с фиксацией взгляда влево, вверх и вниз. Повторить 3-4 раза.
4. Перенести взгляд быстро по диагонали: направо вверх - налево вниз, потом прямо вдаль на счет 1-6; затем налево вверх направо вниз и посмотреть вдаль на счет 1-6. Повторить 4-5 раз.

Упражнение для снятия локального утомления:

1. И. п. – о. с. 1-2 – встать на носки, руки вверх-наружу, потянуться вверх за руками, 3-4 – дугами в стороны руки вниз и расслабленно скрестить перед грудью, голову наклонить вперед. Повторить 6-8 раз. Темп быстрый.
2. И. п. - стойка ноги врозь, руки вперед, 1 – поворот туловища направо, мах левой рукой вправо, правой назад за спину. 2 и. п. 3-4 – то же в другую сторону. Упражнения выполняются размашисто, динамично. Повторить 6-8 раз. Темп быстрый.
3. И. п. 1 – согнуть правую ногу вперед и, обхватив голень руками, притянуть ногу к животу. 2 – приставить ногу, руки вверх-наружу, 3-4 – то же другой ногой. Повторить 6-8 раз. Темп средний.

Упражнение для улучшения кровообращения мозгового кровообращения:

1. И. п. – о. с. 1 – руки за голову; локти развести пошире, голову наклонить назад. 2 – локти вперед, 3-4 - руки расслабленно вниз, голову наклонить вперед. Повторить 4-6 раз. Темп медленный.
2. И. п. – стойка ноги врозь, кисти в кулаках. 1 – мах левой рукой назад, правой вверх – назад. 2 – встречными махами переменить положение рук. Махи заканчивать рывками руками назад. Повторить 6-8 раз. Темп средний.
3. И. п. – сидя на стуле. 1-2 отвести голову назад и плавно наклонить назад. 3– 4 – голову наклонить вперед, плечи не поднимать. Повторить 4-6 раз. Темп медленный.

Наклоны и повороты головы оказывают механическое воздействие на стенки шейных кровеносных сосудов, повышают их эластичность; раздражение вестибулярного аппарата вызывают расширение кровеносных сосудов головного мозга. Дыхательные упражнения, особенно дыхание через нос, изменяют их кровенаполнение. Все это усиливает мозговое кровообращение, повышает его интенсивность и облегчает умственную деятельность.

***Микроклимат на рабочем месте***

Микроклимат производственных помещений — климат внутренней среды этих помещений, который определяется совместно действующими на организм человека температурой, относительной влажностью и скоростью движения воздуха, а также температурой окружающих поверхностей.

Все требования к помещению и используемому компьютерному оборудованию отражены в Санитарных правилах и нормах СП 2.2.3670-20.

Санитарные правила устанавливают гигиенические требования к показателям микроклимата рабочих мест производственных помещений с учетом интенсивности энерготрат работающих, времени выполнения работы, периодов года.

Оптимальные микроклиматические условия установлены по критериям оптимального теплового и функционального состояния человека. Они обеспечивают общее и локальное ощущение теплового комфорта в течение 8-часовой рабочей смены при минимальном напряжении механизмов терморегуляции, не вызывают отклонений в состоянии здоровья, создают предпосылки для высокого уровня работоспособности и являются предпочтительными на рабочих местах.

Допустимые микроклиматические условия установлены по критериям допустимого теплового и функционального состояния человека на период 8-часовой рабочей смены ( табл. 1). Они не вызывают повреждений или нарушений состояния здоровья, но могут приводить к возникновению общих и локальных ощущений теплового дискомфорта, напряжению механизмов терморегуляции, ухудшению самочувствия и понижению работоспособности.

**Таблица 1**

***Оптимальные величины показателей микроклимата  
на рабочих местах производственных помещений***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Период года | Категория работ по уровню энергозатрат, Вт | Температура воздуха, °С | Температура поверхностей, °С | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | Iа (до 139) | 22-24 | 21-25 | 60-40 | 0,1 |
|  | Iб (140-174) | 21-23 | 20-24 | 60-40 | 0,1 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| Теплый | Iа (до 139) | 23-25 | 22-26 | 60-40 | 0,1 |
|  | Iб (140-174) | 22-24 | 21-25 | 60-40 | 0,1 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Измерения показателей микроклимата в целях контроля их соответствия гигиеническим требованиям должны проводиться в холодный период года - в дни с температурой наружного воздуха, отличающейся от средней температуры наиболее холодного месяца зимы не более чем на 5° С, в теплый период года - в дни с температурой наружного воздуха, отличающейся от средней максимальной температуры наиболее жаркого месяца не более чем на 5° С. Частота измерений в оба периода года определяется стабильностью производственного процесса, функционированием технологического и санитарно-технического оборудования.

Охлаждающий микроклимат способствует развитию не только простудных заболеваний, но и усугубляет отрицательное действие вибрации и вызывает обострение хронических заболеваний опорно-двигательного аппарата.

***Организация рабочего места***

Рабочее место — это часть пространства, в котором инженер осуществляет трудовую деятельность, и проводит большую часть рабочего времени. Рабочее место, хорошо приспособленное к трудовой деятельности инженера, правильно и целесообразно организованное, в отношении пространства, формы, размера обеспечивает ему удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении.

Согласно ГОСТ 12.2.032 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места программиста должны быть соблюдены следующие основные условия:

* оптимальное размещение оборудования, входящего в состав рабочего места;
* достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;
* необходимо естественное и искусственное освещение для выполнения поставленных задач;
* уровень акустического шума не должен превышать допустимого значения.

Главными элементами рабочего места программиста являются: письменный стол и кресло. Основным рабочим положением является положение сидя. Рабочее место для выполнения работ в положении сидя организуется в соответствии с ГОСТ 12.2.032.

Оптимальные размеры на рабочем месте.

* Документацию на рабочем месте необходимо располагать так, чтобы было удобно доставать, не вставая из-за письменного стола.
* Параметры рабочего места нужно выбирать в соответствии с антропометрическими характеристиками. В положении сидя рекомендуется работать только лишь при соблюдении указанных параметров рабочего пространства: ширина и глубина не менее 700 и 400 мм; высота рабочей поверхности стола над полом 700-750 мм.
* Оптимальные размеры стола: высота 710 мм; длина стола 1300 мм; ширина стола 650 мм.
* Письменный стол должен иметь не менее 40 мм в глубину и не менее 600 мм в ширину.
* Под рабочим столом должно быть пространство для ног, которое будет по высоте не менее 600 мм, ширине не менее 500 мм и глубине не менее 400 мм.

Рабочая поза сидя вызывает минимальное утомление программиста. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

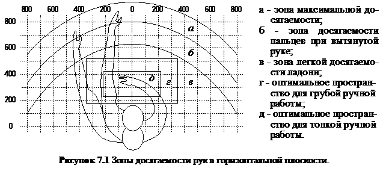
Моторное поле − пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук — это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона − часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.

Максимальная зона досягаемости рук — это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движе­нии их в плечевом суставе. Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.

1. а − зона максимальной досягаемости;
2. б – зона досягаемости пальцев при вытянутой руке;
3. в − зона легкой досягаемости ладони;
4. г − оптимальное пространство для грубой ручной работы;
5. д − оптимальное пространство для тонкой ручной работы.



**Рис. 1 Зоны досягаемости рук**

Оптимальное размещение предметов труда и документации в зонах досягаемости:

* дисплей размещается в зоне а (в центре);
* системный блок размещается в предусмотренной нише стола;
* клавиатура − в зоне г/д;
* «мышь» − в зоне в справа;
* сканер в зоне а/б (слева);
* принтер находится в зоне а (справа);
* документация необходимая при работе − в зоне легкой досягаемости ладони – в, а в выдвижных ящиках стола − литература, неиспользуемая постоянно.



**Рис. 24 Размещение основных и периферийных составляющих ПК**

(1 – сканер, 2 – монитор, 3 – принтер, 4 – поверхность рабочего стола, 5 – клавиатура, 6 – манипулятор типа «мышь».)

На рис. 6.3 показан пример размещения основных и периферийных составляющих ПК на рабочем столе программиста.

***Электробезопасность***

В связи с тем, что для работы ПЭВМ и периферийных устройств используется электрическая энергия, их эксплуатация должна соответствовать «Правилам технической эксплуатации электроустановок потребителей» и «Правилам техники безопасности при эксплуатации электроустановок потребителей». С этой точки зрения ПЭВМ являются потенциальными источниками опасности поражения человека электрическим током. Поражение электрическим током организма человека носит название электро-травмы. Проходя через организм человека, электрический ток оказывает термическое, электролитическое и биологическое действие. Первое заключается в нагреве и ожогах различных частей и участков тела человека, второе — в изменении состава (разложение) и свойств крови и других органических жидкостей. Биологическое действие электрического тока выражается в раздражении и возбуждении живых тканей организма и в нарушении протекания в нем различных внутренних биоэлектрических процессов (прекращение процесса дыхания и остановка сердца).

С целью обеспечения безопасной и надежной работы, снижения вероятности возникновения аварийных (чрезвычайных) ситуаций, таких, например, как поражение электрическим током, пожарная опасность, сбои в работе ПЭВМ и т.п. необходимо обеспечить выполнение следующих условий. Рабочие места с ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации электроустановок и вычислительной техники. Рабочие места с ПЭВМ не следует размещать вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ. Поскольку непосредственно на ПЭВМ должно подаваться стабилизированное электропитание (с отклонением от 220 В не более —10 % +15 %), подачу электроэнергии в компьютерные помещения следует осуществлять от отдельного независимого источника питания. Согласно требованиям ПУЭ корпуса всех распределительных (вводных) щитов в зданиях, сооружениях, квартирах должны быть занулены или заземлены; на вводе в здание должна выполняться система уравнивания потенциалов. Постоянно контролировать надежность соединения контактов трехпроводных розеток. Не ставить системный блок в зоне повышенной влажности и повышенного содержания пыли, на пол, у ног оператора. Во избежание поражения электрическим током запрещается прикасаться к задней панели системного блока и переключать разъемы периферийных устройств работающего компьютера. Необходимо устанавливать ПЭВМ (ПК) только на жестко закрепленной подставке, исключающей даже случайное сотрясение системного блока. Не рекомендуется установка ПЭВМ и его клавиатуры на поверхности, накапливающие статическое электричество (органическое стекло и полированные лаковые поверхности).

Приложение 3

***Расчет общего освещения методом светового потока***

В зависимости от типа светильника выбираем коэффициент *λ*. Он определяет такое соотношение максимального расстояния между светильниками *Lсв(max)* и высотой их подвеса над рабочей поверхностью *hр*, которое обеспечит равномерность освещения в помещении.

*Lсв(max) = λ ⋅ hр =* 1,4 ⋅ 2 = 2,8 м*.*

Определяем расстояние *L1(max)*от стены до первого ряда светильников при наличии рабочих мест у стены:

*L1(max) = (0,2…0,3) ⋅ Lсв(max) =* 0,25 ⋅ 2,8 = 0,7 м*.*

Определяем общее число рядов светильников (по длине помещения):

и число светильников в ряду (по ширине помещения):

где *a* – длина, *b* – ширина помещения, для которого рассчитывается система освещения.

Полученные результаты округляем до ближайшего целого числа, после чего определяем общее расчётное минимальное количество светильников, которое необходимо разместить в помещении:

*nобщ (min) = nш (min) ⋅ nд (min) = 3 ⋅ 2 = 6; S = a ⋅ b = 4 ⋅ 6 = 24.*

По площади помещения *S* и высоте подвески светильника*hр* определяем показатель помещения *i*:

Находим значения коэффициентов отражения потолка *ρп*, стен *ρс* и полов *ρпол* помещения, для которого рассчитывается осветительная установка.

*ρп* = 50%; *ρс* = 30%; *ρпол* = 10%.

В зависимости от типа светильника и вида лампы определяем коэффициент использования светового потока *ηи* по показателю помещения *i* и коэффициентам отражения потолка *ρп*, стен *ρс* и полов *ρпол*.

*ηи*= 0,33.

Определяем коэффициент запаса *k***,** учитывающий снижение уровня освещённости из-за неблагоприятных условий эксплуатации осветительной установки: наличия дыма, копоти, пыли, повышенной концентрации химических веществ и т. д.; из-за старения и выхода из строя ламп. *k* = 1,3.

Решаем сколько источников света *x* будет в светильнике. *x* = 2.

Назначаем коэффициент *z*, характеризующий неравномерность освещённости (коэффициент отношения средней освещённости к максимальной):

z = 1,1 – для люминесцентных ламп.

Рассчитываем требуемый световой поток одной лампы:

По рассчитанному световому потоку лампы *Фрасч* подбираем стандартную лампу со световым потоком *Фтабл*, значение которого близко к значению*Фрасч* (желательно в пределах – 10…+20%).

После выбора стандартных ламп рассчитываем число светильников, необходимых для обеспечения заданной освещённости *E*. Полученное число *nрасч* округляют до ближайшего целого значения *nпр*, при этом отклонение между принятым количеством светильников *nпр* и расчётным *nрасч* допускается в пределах от –10 до +20%.

*Ф1табл =* 3390*; n1пр = 6;*

*.*

*Ф2табл = 3865; n1пр = 6;*

*.*

Рассчитывают полную мощность проектируемой системы освещения:

***Пожарная безопасность***

Согласно Нормам пожарной безопасности НПБ 105-95, помещения с ЭВМ и ПЭВМ относятся к категории В (пожароопасные).

Помещения с ПЭВМ должны оснащаться углекислотными огнетушителями (ОУ-5) и аптечкой первой помощи. Углекислотные огнетушители предназначены для тушения загораний веществ и материалов, за исключением тех, горение которых может происходить без доступа воздуха. Количество и состав огнетушителей выбирается согласно Правилам пожарной безопасности ППБ-01-93 в зависимости от площади защищаемого помещения и класса пожара. Для нескольких помещений одного класса (с небольшой площадью каждого из них) выбирают по суммарной площади этих помещений.

Согласно требованиям Правил ППБ-01-93, если ПЭВМ установлены в общественных зданиях и сооружениях, расстояние от возможного очага возгорания до места размещения огнетушителя не должно превышать 20 м.

Для тушения очагов возгорания офисных помещений применяются порошковые и углекислотные огнетушители. Именно они будут наиболее эффективны для помещения с ПЭВМ. Углекислотные огнетушители с массой заряда 3 кг. (ОУ-3), порошковые огнетушители с массой 4 кг. (ОП-4) будут более подходящими вариантами в офисных помещениях, где расположено большое количество компьютеров и орг. техники.

Применение автоматических средств обнаружения пожаров является одним из основных условий обеспечения пожарной безопасности, поскольку позволяет оперативно известить о пожаре и принять меры по быстрой его ликвидации. Наиболее надежная из таких систем – электрическая пожарная сигнализация Основные элементы таких систем – пожарные извещатели, преобразующие физические параметры, характеризующие пожар, в электрические параметры.