

ОГЛАВЛЕНИЕ

ВЕДОМОСТЬ ОБЪЕМА ДИПЛОМНОГО ПРОЕКТА	
ВВЕДЕНИЕ.....	7
1 ОБЗОР ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ.....	9
2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	10
3 ПОСТАНОВКА ЗАДАЧИ	14
3.1 Общие определения.....	14
3.2 Определение требований к системе	14
4 МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА	16
4.1 Варианты использования системы	16
4.2 Логическая модель данных	17
4.3 Экраны мобильного клиента	19
4.4 Модель взаимодействия клиентского приложения и сервера	20
5 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА	22
5.1 Выбор интегрированной среды разработки и языка программирования.....	22
5.1.1 Интегрированная среда разработки Xcode	22
5.1.2 Язык программирования Swift.....	23
5.1.3 Программная платформа Cocoa Touch	24
5.2 Выбор архитектуры	25
5.2.1 MVC.....	25
5.2.2 MVP	27
5.2.3 MVVM.....	28
5.3 Реализация выбранной архитектуры.....	31
6 РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА	34
6.1 Развертывание мобильного клиента	34
6.1.1 Настройка аккаунта	34
6.1.2 Оформление приложения для магазина	35
6.1.3 Настройка проекта, сборка и выгрузка	36
6.1.4 Отправка приложения на рассмотрение.....	37
6.2 Тестирование мобильного клиента	38
6.2.1 Критическое тестирование.....	39
6.2.2 Углубленное тестирование	41
7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	43
8 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА	49
8.1 Расчет сметы затрат, цены и прибыли на программное средство.....	49
8.1.1 Исходные данные.....	50
8.1.2 Общая характеристика разрабатываемого программного средства.....	50

8.1.3 Определение объема программного средства.....	51
8.1.4 Расчет трудоемкости выполняемой работы	52
8.1.5 Расчет основной заработной платы исполнителей	53
8.1.6 Расчет дополнительной заработной платы исполнителей.....	55
8.1.7 Расчет отчислений в фонд социальной защиты населения	56
8.1.8 Расчет налога на ликвидацию последствий чернобыльской катастрофы	56
8.1.9 Расчет расходов по статье «Материалы»	56
8.1.10 Расчет расходов по статье «Спецоборудование»	57
8.1.11 Расчет расходов по статье «Машинное время»	58
8.1.12 Расчет расходов по статье «Научные командировки»	58
8.1.13 Расчет расходов по статье «Прочие затраты»	58
8.1.14 Расчет расходов по статье «Накладные расходы»	59
8.1.15 Расчет общей суммы расходов на разработку	59
8.1.16 Расчет прибыли и отпускной цены	60
8.2 Расчет экономического эффекта от применения ПС у пользователя.....	61
8.2.1 Исходные данные.....	61
8.2.2 Расчет объема работ	62
8.2.3 Расчет капитальных затрат.....	62
8.2.4 Расчет экономии основных видов ресурсов в связи с использованием нового программного средства	63
8.2.5 Расчет экономического эффекта.....	65
9 ОХРАНА ТРУДА	68
9.1 Производственная санитария, техника безопасности и пожарная профилактика.....	68
9.1.1 Метеоусловия	68
9.1.2 Вентиляция и отопление.....	69
9.1.3 Освещение	70
9.1.4 Шум	71
9.1.5 Электробезопасность	71
9.1.6 Излучение	72
9.1.7 Пожарная безопасность	73
9.2 Требования к организации медицинского обслуживания пользователей ВДТ, ЭВМ и ПЭВМ	74
ЗАКЛЮЧЕНИЕ	75
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	
ГРАФИЧЕСКАЯ ЧАСТЬ	
ПРИЛОЖЕНИЕ	

ВВЕДЕНИЕ

Управление разработкой программного обеспечения (англ. Software project management) – особый вид управления проектами, в рамках которого происходит планирование, отслеживание и контроль за проектами по разработке программного обеспечения [1]. Ключевым моментом в управлении проектом по разработке программного обеспечения является правильный выбор метода разработки.

В связи с быстрым увеличением мощностей компьютеров в 60-е и 70-е годы XX века проблемы, которые могли быть решены с их помощью, становились сложнее. Поэтому требовались более масштабные проекты, включавшие в себя координацию труда большего числа людей и написание гораздо большего объёма кода. Однако методы, применявшиеся к управлению такими проектами, были рассчитаны на решение задач в рамках намного меньших проектов. Отсутствие необходимой методологии привело к огромному числу провальных проектов. Попытки изменить положение к лучшему привели к созданию новой модели процесса разработки, концентрировавшей больше внимания на соответствие конечного программного продукта изначальным требованиям заказчика.

Исследования проектов, окончившихся неудачей, показали, что самыми распространёнными причинами провалов были:

- 1) невыполнимые или неясно сформулированные цели проекта;
- 2) ошибочный подсчет необходимых ресурсов;
- 3) некорректно определённые системные требования;
- 4) неосведомлённость управляющего проектом о точном состоянии проекта;
- 5) неуправляемые риски;
- 6) слабое взаимодействие между заказчиком, разработчиком и пользователем;
- 7) использование слишком новых, нестабильных технологий;
- 8) неспособность справиться со сложностями проекта;
- 9) слабое управление проектом;
- 10) финансовые ограничения.

С тех пор было представлено несколько усовершенствований уже существующих (итеративный подход) и совершенно новых (разработка через тестирование) методов управления разработкой программного обеспечения. Ввиду того что процессы управления разработкой ПО тем или иным образом повторялись от проекта к проекту, были предприняты попытки автоматизации управленческой деятельности. Так появились системы управления программными проектами.

Первоначально такие системы представляли из себя веб-приложения, доступ к которым для мобильных устройств осуществлялся через мобильный веб-браузер. Однако со временем многие разработчики таких программных систем стали также предоставлять мобильные приложения для доступа к своим веб-системам.

Мобильное приложение (англ. «Mobile app») – программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах [2].

Первоначально мобильные приложения использовались для быстрой проверки электронной почты, но их высокий спрос привел к расширению их назначений и в других областях, таких как игры для мобильных телефонов и GPS, общение, просмотр видео и пользование интернетом.

На сегодняшний день число пользователей мобильных устройств стремительно растет. В то время, как многие компании уже успешно ведут бизнес с использованием мобильных технологий, другие только начинают присматриваться к этой области. И перед ними встает вопрос, сделать ли выбор в пользу разработки мобильного веб-сайта или же мобильного приложения [3].

Разработка мобильных бизнес-приложений должна исходить в первую очередь из задач для этих приложений – в этом основная стратегия выбора. К примеру, корпоративный сайт компании полностью удовлетворяет всем задачам и потребностям, и необходимо лишь обеспечить доступ к его сервисам со смартфонов и планшетных ПК, тогда мобильный веб-сайт отлично подойдет. Приложение требует скачивания и установки, доступ к сервисам и информации через браузер осуществим быстрее. Но, если сотрудники компании или клиенты пользуются вашими сервисами постоянно, скорее всего, им будет удобнее получать доступ, просто нажав на иконку вашего мобильного бизнес-приложения на своем устройстве, чем запускать браузер, искать адрес. Так что регулярное использование предоставляемых ресурсов – плюс для выбора разработки приложения.

В плане производительности, чаще всего в выигрыше оказываются мобильные приложения, поскольку они имеют большую интеграцию с платформой и напрямую могут использовать ресурсы устройства, а также задействовать функции, которые просто недоступны для мобильного сайта (например, доступ к телефонной книге смартфона).

Часто основным доводом в пользу разработки мобильных веб-сайтов считают их кроссплатформенность. Это конечно так, но не стоит забывать, что очень часто мобильные сайты невозможно легко адаптировать под все типы устройств, под все разрешения экранов без потери качества и юзабилити. В свою очередь, и для разработки мобильных бизнес-приложений существуют кроссплатформенные решения, что позволяет не писать приложения под различные системы с нуля.

Таким образом, целью данного дипломного проекта является решение вышеперечисленных проблем управления процессом разработки программных проектов с использованием достоинств мобильных устройств и преимуществ мобильных приложений.

1 ОБЗОР ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

Книга «iOS Apprentice: Beginning iOS development with Swift 4.2» Мэтьюса Холлеманса предназначена для людей, которые не знакомы как с разработкой под платформу iOS на языке программирования Swift, так и с программированием в целом. В данную книгу включены четыре раздела, каждый из которых является туториалом для создания полноценного приложения под платформу iOS. Каждый из туториалов описывает шаг за шагом процесс создания мобильного приложения, в который также включены выноски с кодом и скриншоты промежуточных результатов. Каждое последующее приложение является более сложным и функциональным, чем предыдущее, что позволяет постепенно переходить от более простых тем к более сложным. Прочитав всю книгу и следя инструкциям по реализации приложений, читатель получит все необходимые знания для возможности создать свое собственное приложение.

Книга «iOS 12 App Development Essentials» Нэйл Смита, последнее издание популярной серии книг, которое было обновлено в соответствии с выходом iOS 12 SDK, Xcode 10 и языка программирования Swift 4.2. Книга начинается с введения в архитектуру мобильной ОС iOS 12 и основы программирования на языке Swift, с дальнейшим переходом к обзору способов проектирования iOS приложений и пользовательского интерфейса. В книге представлены для изучения такие углубленные темы, как:

- обработка файлов;
- работа с базами данных;
- работа с графикой и анимацией;
- обработка нажатий экрана;
- многозадачность;
- управление локальными уведомлениями;
- инструменты автопозиционирования;
- и др.

Так же в деталях описаны новые ключевые особенности iOS12 и Xcode 10. Главной целью книги является обучение необходимым навыкам для создания своего собственного полноценного iOS приложения.

2 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Система управления программными проектами – комплексное программное обеспечение, включающее в себя приложения для планирования задач, распределения ресурсов, совместной работы, общения, документирования и администрирования системы, которое используются совместно для управления программными проектами.

Задачи программного обеспечения для управления проектами:

- планирование;
- расчет критического пути;
- управление данными и предоставление информации;
- управление коммуникациями команды проекта.

Одной из наиболее распространенных возможностей является возможность планирования событий и управления задачами. Требования могут различаться в зависимости от того, как используется инструмент. Наиболее распространенными являются:

- планирование различных событий, зависящих друг от друга;
- идентификация крупных составных частей проекта и их декомпозиция, посредством которой создается структура декомпозиции работ, также называемая иерархической структурой работ;
- планирование расписания работы сотрудников и назначение ресурсов на конкретные задачи;
- расчет времени, необходимого на решение каждой из задач;
- сортировка задач в зависимости от сроков их завершения;
- презентация графика работ по проекту в виде различных графиков и диаграмм;
- управление несколькими проектами одновременно.

В основе метода критического пути лежит определение наиболее длительной последовательности задач от начала проекта до его окончания с учетом их взаимосвязи. Задачи, лежащие на критическом пути (критические задачи), имеют нулевой резерв времени выполнения, и, в случае изменения их длительности, изменяются сроки всего проекта. В связи с этим, при выполнении проекта критические задачи требуют более тщательного контроля, в частности, своевременного выявления проблем и рисков, влияющих на сроки их выполнения и, следовательно, на сроки выполнения проекта в целом. В процессе выполнения проекта критический путь проекта может меняться, так как при изменении длительности задач некоторые из них могут оказаться на критическом пути.

Программное обеспечение для управления проектами предоставляет большое количество требуемой информации, такой как:

- список задач для сотрудников и информацию распределения ресурсов;

- обзор информации о сроках выполнения задач;
- ранние предупреждения о возможных рисках, связанных с проектом;
- информации о рабочей нагрузке;
- информация о ходе проекта, показатели и их прогнозирование.

К возможностям управления коммуникациями команды проекта относят:

- обсуждение и согласование рабочих вопросов проекта;
- фиксация проблем проекта и запросов на изменения, их обработка;
- ведение рисков проекта и проактивное управление ими;
- предоставление доступа к информации о ходе проекта в виде живой ленты событий.

В настоящее время на рынке представлено огромное количество систем управления программными проектами. Рассмотрим наиболее популярные из них, для которых так же существуют клиентские приложения под мобильную операционную систему iOS.

Trello – программа для управления проектами небольших групп, разработанное Fog Creek Software [4]. Trello использует парадигму для управления проектами, известную как канбан (система организации производства и снабжения, позволяющая реализовать принцип «точно в срок»), метод, который первоначально был популяризирован Toyota в 1980-х для управления цепочками поставок. Скриншот клиентского приложения под мобильную ОС iOS приведен на рисунке 2.1.



Рисунок 2.1 – мобильное приложение Trello под iOS

Достоинства:

- низкий порог входления;
- удобна для использования в небольших проектах.

Недостатки:

- недостаточность функционала для управления большими проектами.

Еще одним довольно популярным представителем среди данных видов программных систем является Jira – коммерческая система отслеживания ошибок, предназначена для организации взаимодействия с пользователями, хотя в некоторых случаях используется и для управления проектами. Разработана компанией Atlassian.

Основной элемент учёта в системе – задача (англ. ticket или issue) [5]. Задача содержит название проекта, тему, тип, приоритет, компоненты и содержание. Задача может быть расширена дополнительными полями (также и новые пользовательские поля могут быть определены), приложениями (например – фотографиями, скриншотами) или комментариями. Задача может редактироваться или просто изменять статус, например, из «открыт» в «закрыт». Какие переходы между состояниями возможны, определяется через настраиваемый поток операций. Любые изменения в задаче протоколируются в журнал.

Mobile for Jira – клиентское приложение системы Jira под мобильную операционную систему iOS. Разработана компанией Infosysta.

Скриншот версии мобильного клиента под iPhone приведен на рисунке 2.2, под iPad – на рисунке 2.3.

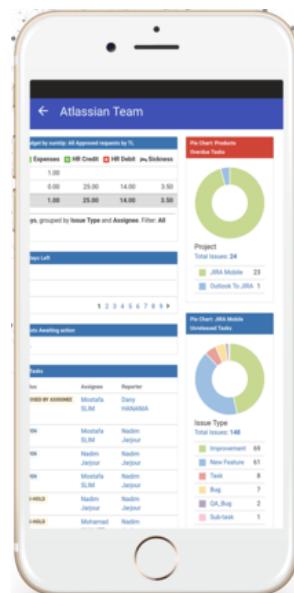


Рисунок 2.2 – Mobile for Jira для iPhone

Данный мобильный клиент предоставляет пользователям следующие возможности:

- создание, редактирование, удаление, просмотр, связывание задач и переходы между ними;
- добавление, редактирование, удаление комментариев;

- добавление от одного до нескольких приложений (документы, файлы изображений и т.д.) к задачам;
- получение оповещений о важных событиях;
- расширенный поиск по различным атрибутам задач;
- списывание времени, затраченного на выполнение задачи;
- использование фильтров по задачам.

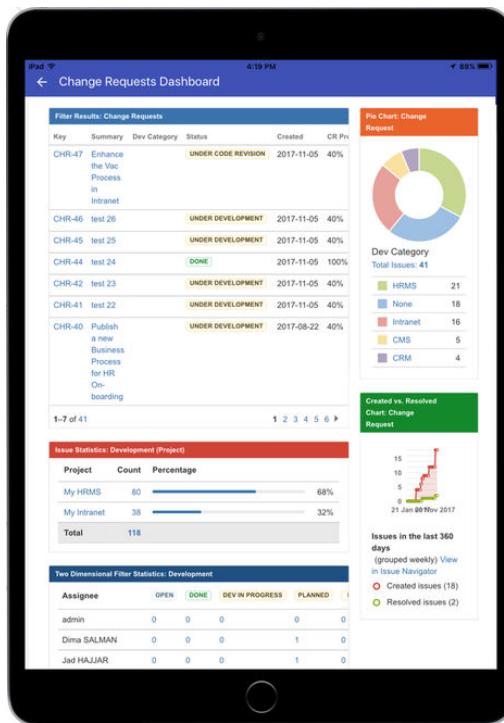


Рисунок 2.3 – Mobile for Jira для iPad

Достоинства:

- поддержка большинства возможностей богатого функционала системы Jira.

Недостатки:

- реализовано и поддерживается сторонним разработчиком, что не гарантирует быстрое появление нового функционала, реализованного в самой системе Jira;
- некоторый функционал доступен лишь в премиум версии (например: создание задач);
- для возможности использования мобильного клиента необходимо устанавливать дополнительные компоненты к серверному приложению;
- для ведения документации используется отдельный программный продукт Confluence, с которым данный мобильный клиент не имеет никакой интеграции;
- периодические проблемы с производительностью приложения.

3 ПОСТАНОВКА ЗАДАЧИ

3.1 Общие определения

Пользователь – информация, связанная с реальным человеком, имеющим доступ к использованию данной системы. Набор функционала, которым может воспользоваться пользователь, зависит от назначенных ему прав доступа. Для различных проектов пользователь может иметь различные права доступа.

Программный проект – может означать идею, описание цели или чего-либо другого, по отношению к чему могут выполняться какие-либо задачи.

Задача – означает конкретную единицу работы, которую нужно выполнить конкретным исполнителем по отношению к определенному проекту.

Новая задача, или новая функция, или новое свойство системы – задача, связанная с реализацией чего-то нового.

Дефект – задача, связанная с исправлением определенных проблем.

Иная задача – абстрактный тип задачи, обычно используется для тех случаев, когда другие типы задач не подходят.

Подзадача – дочерняя задача. Необходима для детализации или разбиения некоторой задачи на несколько подзадач (подзадачи могут быть созданы для любого типа задачи кроме типа «подзадача»).

Исполнитель – конкретный пользователь, ответственный за выполнение задачи.

Приоритет – приоритет задачи. Означает важность этой задачи по отношению к другим.

Предполагаемое время – запланированное время необходимое для выполнения задачи.

Затраченное время – фактически потраченное время на задачу.

3.2 Определение требований к системе

Необходимо разработать мобильный клиент для системы управления программными проектами ZIRO под мобильную ОС iOS. Разработанное приложение должно отвечать следующим требованиям:

- поддерживать устройства с установленной мобильной ОС iOS 12 и выше;
- поддерживать мобильные устройства, начиная с iPhone 6;
- поддерживать планшетные компьютеры, начиная с iPad Pro 2017;
- обрабатывать информацию, введенную пользователем;
- предоставлять информацию по запросам пользователя;
- оповещать пользователя об ошибочных действиях, связанных с работой в

приложении;

- принимать и передавать данные на серверную часть системы ZIRO в формате JSON.

Разрабатываемое мобильное приложение должно предоставлять пользователю такие возможности, как:

- авторизация;
- просмотр информации о пользователе;
- просмотр списка задач;
- просмотр задачи;
- добавление комментария к задаче;
- просмотр списка проектов;
- просмотр проекта.

Входная информация – это данные, поступающие на вход задачи и используемая для ее решения.

Входными данными для разрабатываемого мобильного приложения являются:

- e-mail пользователя;
- имя пользователя;
- пароль пользователя;
- запросы пользователя просмотр учетной записи;
- запросы пользователя на просмотр задач;
- запросы пользователя на просмотр проектов;
- запросы пользователя на просмотр, добавление комментариев;

Выходными данными разрабатываемого мобильного приложения являются:

- список задач;
- информация о задаче;
- список комментариев к задаче;
- список проектов;
- информация о проекте;
- информация об учетной записи пользователя;
- документация проекта;
- сообщения об успешном выполнении операций авторизации пользователя в приложении;
- сообщения об ошибках, связанных с выполнением операции авторизации пользователя в приложении;
- сообщения об успешном выполнении операций добавления комментариев;
- сообщения об успешном выполнении операций добавления комментариев.

4 МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

4.1 Варианты использования системы

Пользователем системы является непосредственно человек, непосредственно участвующий в работе над проектом. Данный пользователь должен иметь доступ исключительно к проектам, участником которых он является и только к тем задачам, над которыми непосредственно работает. В соответствии с этим определим необходимый функционал для пользователя в системе:

- просмотр списка задач;
- фильтрация задач;
- просмотр задачи;
- просмотр журнала изменений задачи;
- просмотр комментариев к задаче;
- добавление комментария;
- просмотр проектов пользователя;
- поиск проекта;
- сортировка проектов;
- просмотр профиля.

Диаграмма вариантов использования вышеписанного функционала представлена на рисунке 4.1.

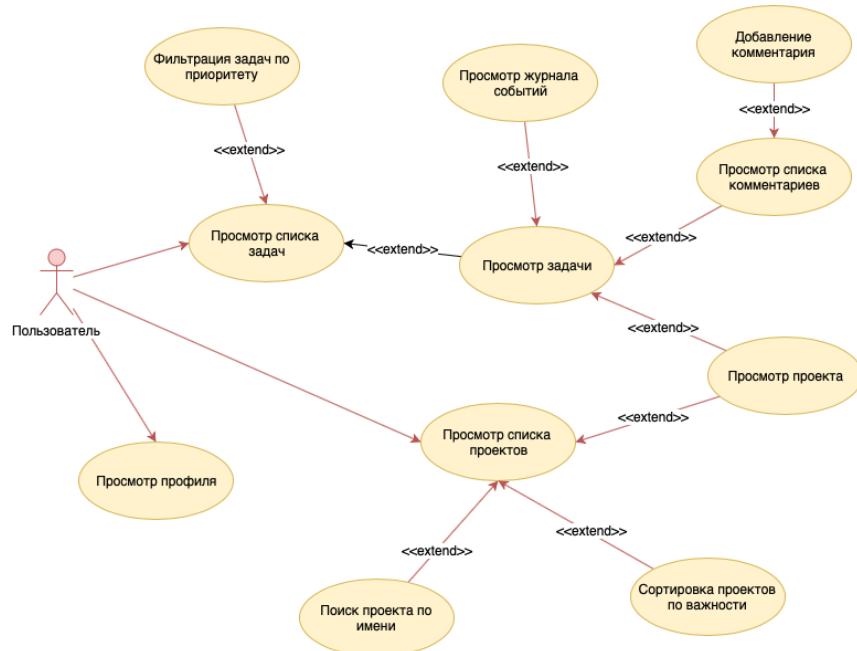


Рисунок 4.1 – Диаграмма вариантов использования

4.2 Логическая модель данных

Определим основные сущности системы:

- проект;
- задача;
- запись журнала изменений задачи;
- комментарий;
- пользователь.

Работа над развитием проекта представляет собой набор отдельных задач, каждая из которых представляет описание некоторой функциональности, которая должна быть разработана. В свою очередь, на протяжении выполнения определенной задачи в ней происходят какие-либо изменения, например такие как перевод в другой статус, отметка о затраченном времени и т.д. Так же, может имеется необходимость сделать какую-то пометку к задаче в виде комментария.

Задачи содержат специализированные атрибуты, которые могут повторяться от задачи к задаче. Определим их как вспомогательные сущности:

- статус;
- приоритет;
- тип.

На рисунке 4.2 приведена логическая модель данных на уровне сущностей.

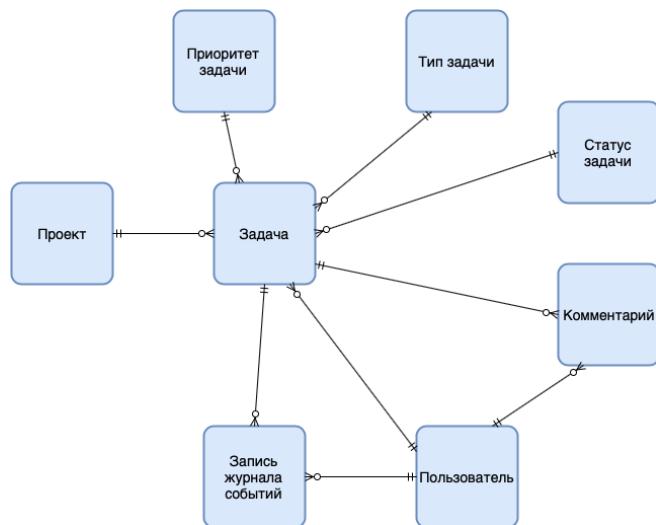


Рисунок 4.2 – Логическая модель данных на уровне сущностей

Определим основные атрибуты сущности «Проект»:

- имя;
- сокращенное имя;
- описание.

Определим основные атрибуты сущности «Задача»:

- номер;
- тип;
- статус;
- заголовок;
- описание;
- приоритет;
- оценочное время;
- затраченное время;
- дата создания;
- проект.

Определим основные атрибуты сущности «Комментарий»:

- автор;
- текст;
- дата.

Определим основные атрибуты сущности «Запись журнала событий»:

- автор;
- текст;
- затраченное время;
- дата.

Сущности «Тип задачи», «Статус задачи» и «Приоритет задачи» являются вспомогательными справочными сущностями и имеют лишь атрибут «Название».

На рисунке 4.3 представлена логическая модель на уровне атрибутов.

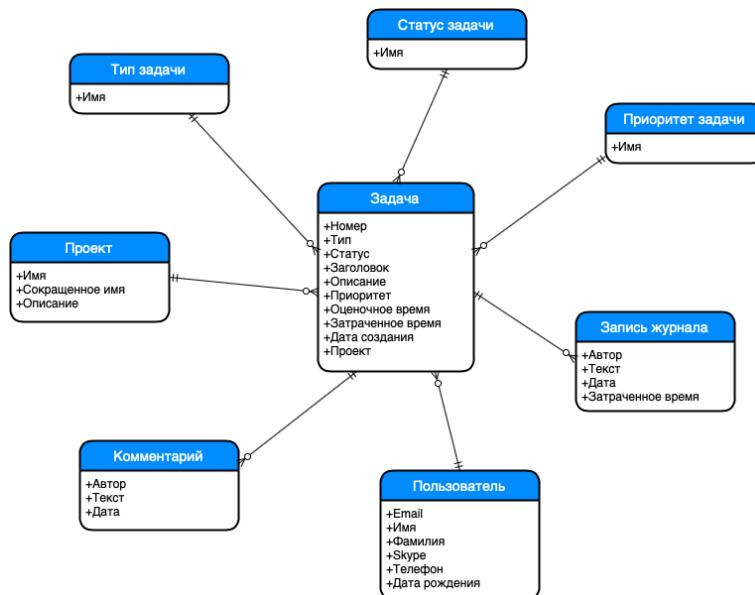


Рисунок 4.3 – Логическая модель на уровне атрибутов

4.3 Экраны мобильного клиента

При запуске приложения пользователю должно быть представлено экран авторизации, для обеспечения возможности его входа в систему. В случае успешной авторизации пользователь должен перейти на главный экран системы. В случае возникновении ошибки при авторизации, пользователю должно показано сообщение об ошибке.

Главный экран приложения должен представлять собой страницу со следующими вкладками:

- список задач;
- список проектов;
- профиль пользователя.

Экран со списком задач должен предоставлять возможность перейти к детализированной информации конкретной задачи. В свою очередь с данного экрана пользователь должен иметь возможность получить доступ к следующей информации:

- проект, к которому относится задача;
- журнал изменений задачи;
- комментарии к задаче.

Экран со списком проектов должен предоставлять доступ к информации о конкретном проекте, а вкладка профиля пользователя – личные данные текущего пользователя системы.

Полученная карта вышеописанных экранов приведена на рисунке 4.4.

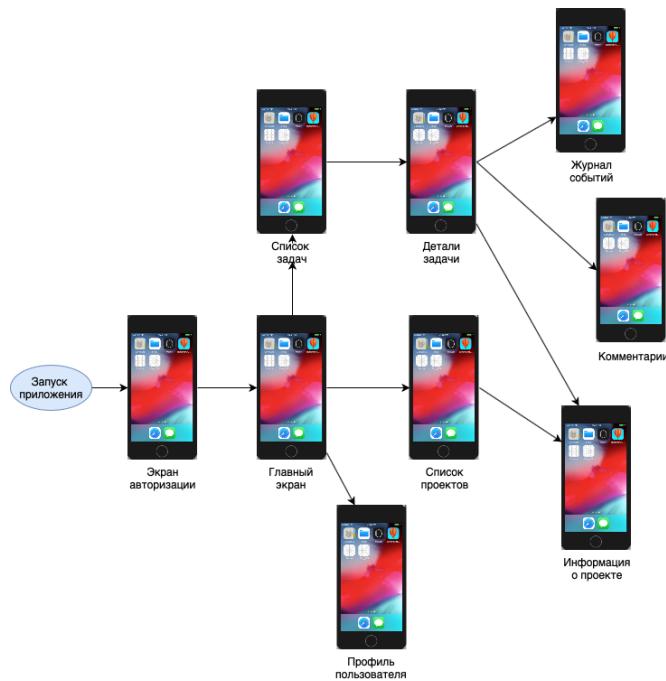


Рисунок 4.4 – Карта экранов мобильного клиента

4.4 Модель взаимодействия клиентского приложения и сервера

Для того чтобы веб-сервер мог взаимодействовать с клиентскими приложениями различных типов он должен предоставлять свой собственный API интерфейс.

API (программный интерфейс приложения, интерфейс прикладного программирования) – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой [6]. Т.е. API интерфейс скрывает за собой некоторый функционал по обработке данных, предоставляя в открытом виде просто способ воспользоваться этим функционалом другим программам.

В контексте веба, взаимодействие между клиентским приложением и сервером должно осуществляться по протоколу HTTP, а значит должен соблюдаться некий набор правил форматов запроса, отправляемых клиентскими приложениями, и форматов ответов, отправляемых сервером в ответ на запрос.

В связи с этим, взаимодействие между клиентом и сервером должно соответствовать архитектуре, называемой REST. REST (сокращение от англ. Representational State Transfer – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине.

В сети Интернет обращение к API может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса. Пример REST-запроса представлен на рисунке 4.5

```
GET http://ctldl.windowsupdate.com/msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?6ac2b6592d164b9e HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Microsoft-CryptoAPI/10.0
Host: ctldl.windowsupdate.com
```

Рисунок 4.5 – Формат REST-запроса

Основные секции запроса:

- заголовки – здесь находится информация о типе запроса (Get, Post и др.), идентификационная информация и другие мета-данные;
- адрес, по которому отправляется запрос (также может содержать параметры, если тип запроса GET);

- тело запроса – необходимые данные и параметры (данная секция существует только для типа запроса POST).

Исходя из этого, необходимо разработать Restfull-веб-сервер, т.е. веб-сервер работающий по протоколу http с поддержкой архитектуры REST.

Реализация API представляет собой совокупность API-методов доступных клиентским приложениям. Каждый API-метод имеет свой тип, адрес и конкретный способ обработки информации.

Таким образом, веб-сервер должен предоставлять информацию о всех реализованных API-методах с указанием следующей информации:

- адрес API-метода – адрес конкретного метода в интерфейсе API, выполняющий определенную обработку (например api/user/getallusers), данный адрес должен объединяться с именем веб-сервера в Интернете и http протоколом, для того чтобы можно было обратиться к данному методу;
- тип принимаемого запроса (Get, Post, Put или Delete);
- перечень всех входных параметров и их форматы;
- формат данных отправляемых в ответе.

На веб-сервер будет использоваться всего два типа http-запросов:

- Get – для получения данных;
- Post – для изменения данных либо запроса данных с передачей большого количества параметров.

Общие схемы отправки и обработки Get запросов приведены на рисунке 4.6.

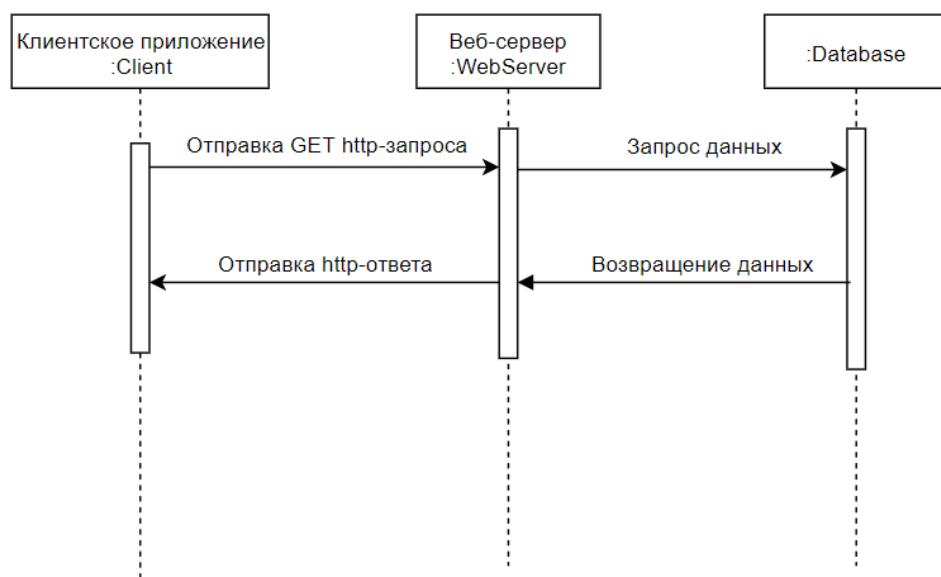


Рисунок 4.6. – Общая схема отправки и обработки Get запроса

5 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА

5.1 Выбор интегрированной среды разработки и языка программирования

Поскольку целью данного дипломного проекта является разработка мобильного приложения для платформы iOS, то необходимо проанализировать доступные на рынке интегрированные среды разработки и языки программирования для данной платформы, и после этого сравнить их и сделать выбор.

5.1.1 Интегрированная среда разработки Xcode

Xcode – интегрированная среда разработки (IDE) программного обеспечения для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple [7]. Первая версия выпущена в 2001 году. Стабильные версии распространяются бесплатно через Mac App Store. Скриншот среды разработки приведен на рисунке 5.1.

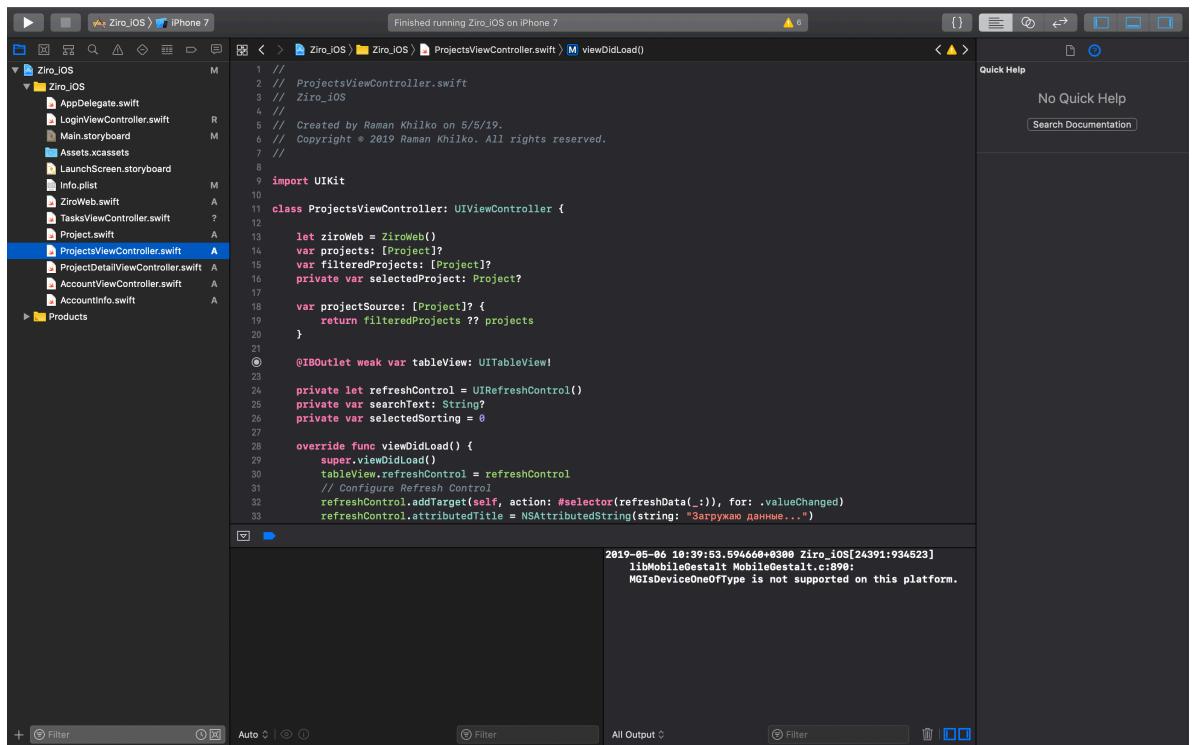


Рисунок 5.1 – Интерфейс среды разработки Xcode

Xcode включает в себя большую часть документации разработчика от Apple и Interface Builder – приложение, использующееся для создания графических интерфейсов (рисунок 5.2). Interface Builder упрощает создание пользовательского интерфейса (UI). С его помощью можно легко, без написания кода, создать слои из окон, различные кнопки, ползунки и другие элементы управления. Затем вы можете

превратить этот прототип UI в реальное приложение, добавив новые возможности. Xcode работает с Interface Builder в режиме реального времени, так что вы видите в графическом интерфейсе (Interface Builder) то, что вы пишете в Xcode.

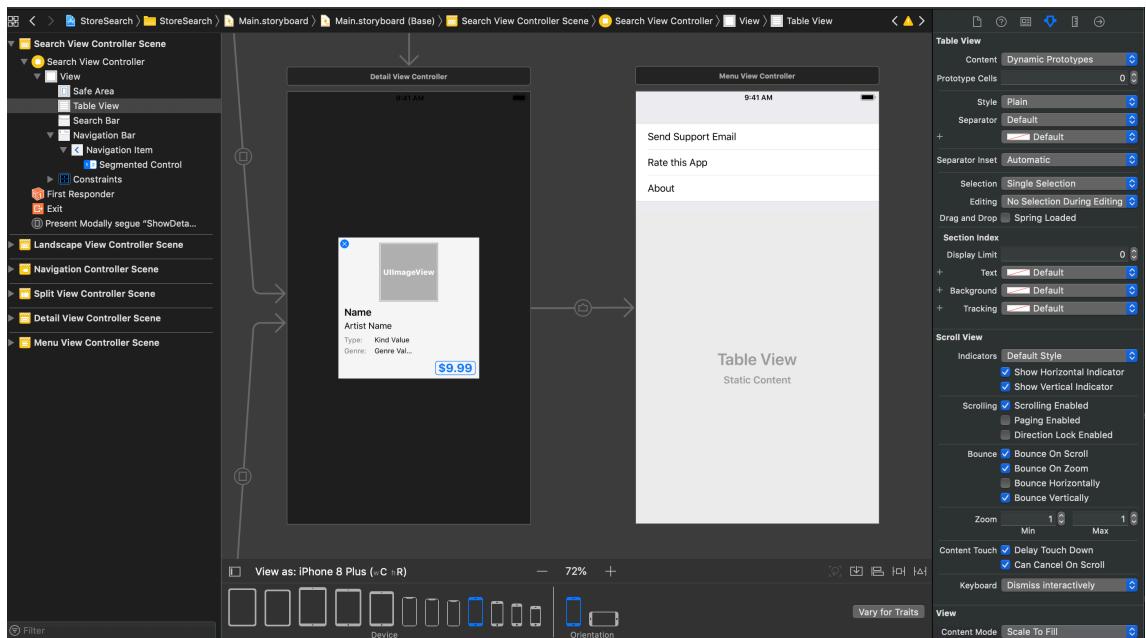


Рисунок 5.2 – Interface Builder

Ключевые особенности:

- полностью бесплатная среда разработки;
- встроенные возможности для работы с кодом;
- встроенные инструменты для работы с графическим интерфейсом;
- интеграция с системами контроля версий;
- поддержка языков программирования C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez), и Swift.

5.1.2 Язык программирования Swift

Swift – мультипарадигмальный компилируемый язык программирования общего назначения [8]. Создан компанией Apple в первую очередь для разработчиков iOS и macOS. Swift работает с фреймворками Cocoa и Cocoa Touch и совместим с основной кодовой базой Apple, написанной на Objective-C. Swift задумывался как более лёгкий для чтения и устойчивый к ошибкам программиста язык, нежели предшествовавший ему Objective-C. Программы на Swift компилируются при помощи LLVM, входящей в интегрированную среду разработки Xcode 6 и выше. Swift может использовать рантайм Objective-C, что делает возможным использование обоих языков (а также C) в рамках одной программы.

Swift заимствовал довольно многое из Objective-C, однако он определяется не указателями, а типами переменных, которые обрабатывает компилятор. По аналогичному принципу работают многие скриптовые языки. В то же время, он предоставляет разработчикам многие функции, которые прежде были доступны в C++ и Java, такие как определяемые наименования, обобщения и перегрузка операторов.

Часть функций языка выполняется быстрее по сравнению с другими языками программирования. Например, сортировка комплексных объектов выполняется в 3,9 раз быстрее, чем в Python, и почти в 1,5 раза быстрее, чем в Objective-C.

Код, написанный на Swift, может работать вместе с кодом, написанным на языках программирования C и Objective-C в рамках одного и того же проекта.

5.1.3 Программная платформа Cocoa Touch

Cocoa Touch – это фреймворк для создания приложений под iPhone, iPod touch, и iPad [9].

Библиотека Cocoa Touch предоставляет уровень абстракции для iOS (операционной системы iPhone, iPad и iPod touch). Cocoa Touch основана на классах фреймворка Cocoa, используемого в Mac OS X, и, аналогично ей, использует язык Objective-C. Cocoa Touch следует шаблону проектирования Model-View-Controller.

Инструменты для разработки приложений с использованием Cocoa Touch включены в iOS SDK.

OS-технологии можно рассматривать как набор слоев, где Cocoa Touch находится на самом высоком уровне, а Core OS и ядро macOS – на более низких. Это позволяет реализовывать многие сложные задачи, сокращая объём работы, которую пришлось бы проделывать разработчикам, работая они на более низком уровне. Тем не менее, некоторые низкие слои абстрагирования могут быть доступны разработчикам по мере необходимости.

Расположение слоев абстрагирования можно представить в следующем виде:

- Cocoa Touch;
- Media / Application Services;
- Core Services;
- Core OS / ядро Mac OS X.

Основные технологии и возможности, присутствующие в Cocoa Touch:

- Core Animation;
- Мультизадачность;
- Распознаватели мультитач-жестов.

Cocoa Touch предоставляет основные фреймворки для разработки приложений на устройствах под управлением iOS. Некоторые из них:

- Foundation Kit Framework – основная библиотека, содержащая классы с префиксом NS;
- UIKit Framework (основывается на Application Kit) – библиотека, содержащая специфические для iOS GUI-классы;
- Game Kit Framework – библиотека для взаимодействия с сервисом Game Center;
- iAd Framework – библиотека для реализации сервисов контекстной рекламы iAd внутри приложений;
- MapKit Framework – библиотека, осуществляющая взаимодействие с картами и навигационными возможностями iOS-устройств.

5.2 Выбор архитектуры

Выбор архитектуры является очень важным вопросом в процессе разработке мобильного приложения под платформу iOS. Если не провести должного анализа и сделать неправильный выбор, то в будущем можно столкнуться со следующими проблемами:

- архитектура является избыточной для данного проекта. Временные затраты на ее поддержку превышают затраты на разработку новой функциональности;
- сложность кода растет, классы содержат большое количество методов и свойств, становится проблематично отлаживать и исправлять ошибки в подобных классах. Такие классы довольно сложно держать в голове, поэтому разработчик всегда будет упускать какие-нибудь важные детали.

Для последующего определения подходящей архитектуры определим следующие признаки:

- простота использования и низкая стоимость обслуживания;
- скорость разработки;
- сбалансированное распределение обязанностей между сущностями с жесткими ролями;
- возможность покрытия этих отдельных сущностей модульными тестами.

В качестве возможных вариантов архитектуры для данного дипломного проекта рассмотрим семейство шаблонов проектирования MV(X): MVC, MVP и MVVM.

5.2.1 MVC

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента

может осуществляться независимо [10].

Версия традиционной реализации концепции «Модель-Представление-Контроллер» приведена на рисунке 5.3.

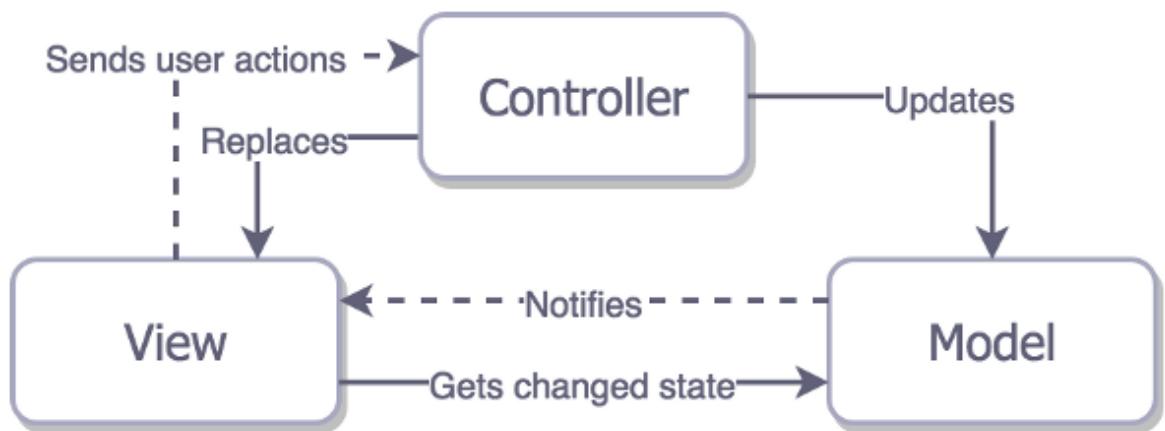


Рисунок 5.3 – Традиционная реализация MVC

В традиционном MVC представление не хранит состояния в себе. Контроллер просто «отрисовывает» представление при изменениях модели. Например, веб-страница полностью перегружается после того, как вы нажмете на ссылку для перехода в другое место. Хотя можно реализовать традиционный MVC в среде iOS, это не имеет особого смысла из-за архитектурной проблемы: все три компонента тесно связаны, каждый компонент знает о двух других. Это сильно снижает возможность повторного использования каждого из элементов. По этой причине традиционный MVC является неприменимым к современной iOS разработке.

В свою же очередь Apple предоставляет для разработчиков свою реализацию шаблона MVC, которая называется Cocoa MVC (рисунок 5.4).

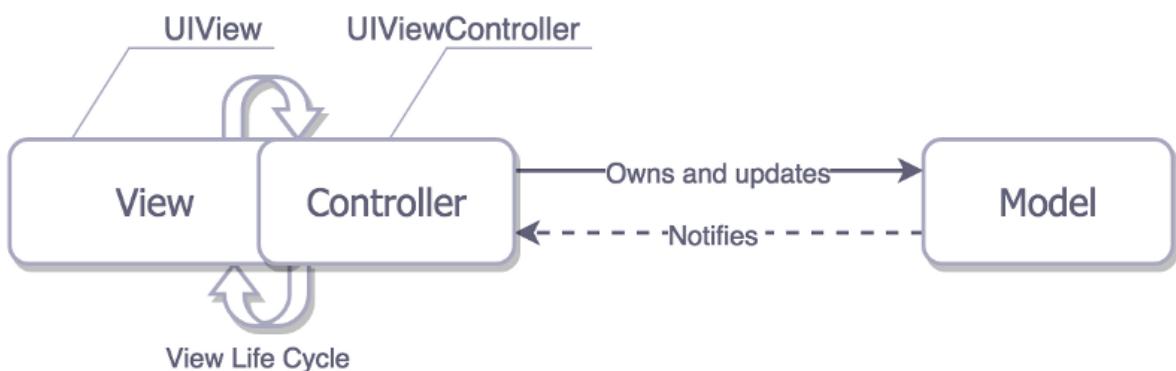


Рисунок 5.4 – Cocoa MVC

Использование Cocoa MVC приводит к проблеме «массивного контроллера», потому что контроллер настолько вовлечен в жизненный цикл представления, что

трудно сказать, что он является отдельной сущностью. В большинстве случаев вся ответственность представления состоит в том, чтобы отправить действия к контроллеру. В итоге все заканчивается тем, что контроллер становится делегатом и источником данных. Хотя все еще сохраняется возможность определить бизнес-логику приложения и логику преобразования данных в модели.

Проблема не очевидна, пока дело не доходит до модульного тестирования. Так как контроллер тесно связан с представлением, его становится трудно тестировать, и приходится идти изощренным путем, заменяя представление Mock-объектами и имитируя их жизненный цикл, а также писать код контроллера таким образом, чтобы бизнес-логика была по максимуму отделена от кода.

После всего сказанного может показаться что, Сосоа MVC является довольно плохим выбором в качестве архитектуры приложения. Но давайте оценим его с точки зрения признаков хорошей архитектуры, определенных в начале данного подраздела:

- распределение: представление и модель на самом деле разделены, но представление и контроллер тесно связаны;
- тестируемость: из-за плохого распределения для наименьших затрат логичным решением является тестировать только модель;
- простота использования: наименьшее количество кода среди других шаблонов проектирования. К тому же он выглядит понятным, поэтому его легко может поддерживать даже неопытный разработчик.

Таким образом Сосоа MVC является лучшим архитектурным шаблоном с точки зрения скорости разработки и небольшого размера проекта.

5.2.2 MVP

Model-View-Presenter (MVP) – шаблон проектирования, производный от MVC, который используется в основном для построения пользовательского интерфейса. Реализация данного шаблона в среде iOS приведена на рисунке 5.5.

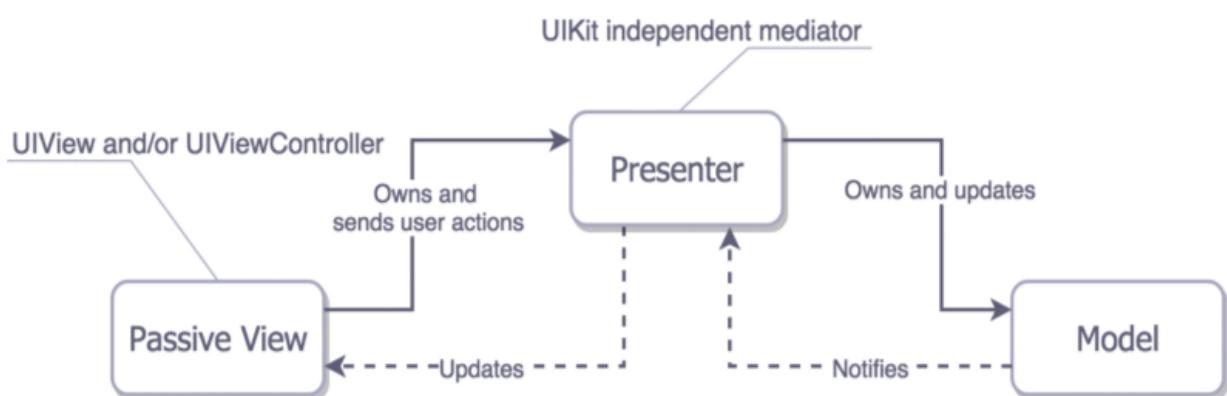


Рисунок 5.5 – Реализация шаблона MVP в среде iOS

Исходя из данного рисунка можно заметить некоторые схожести с Cocoa MVC. Но стоит отметить, что Cocoa MVC нельзя назвать MVP, поскольку в Cocoa MVC представление и контроллер тесно связаны, в то время как посредник в MVP – Presenter – не имеет никакого отношения к жизненному циклу представления. Presenter не содержит никакой логики, связанной с отрисовкой представления, но отвечает за обновление представления в соответствии с новыми данными и состоянием.

С точки зрения MVP подклассы UIViewController на самом деле являются представлением, а не Presenter. Это различие обеспечивает превосходную тестируемость, которая идет за счет уменьшения скорости разработки, поскольку появляется необходимость связывать вручную данные и события именно между представлением и Presenter.

Оценим шаблон проектирования MVP по признакам хорошей архитектуры:

- распределение: большая часть ответственности разделена между Presenter и моделью, а представление ничего не делает;
- тестируемость: отличная, поскольку имеется возможность проверить большую часть бизнес-логики благодаря бездействию представления;
- простота использования: сама по себе идея шаблона MVP является очень простой, но использование данной концепции ведет к увеличению кода по сравнению с Cocoa MVC.

Таким образом шаблон проектирования MVP в iOS означает превосходную возможность обеспечить покрытие кода модульными тестами, но также приводит к увеличению количества кода, что в свою очередь ведет к значительному понижению скорости разработки.

5.2.3 MVVM

Model-View-ViewModel (MVVM, «Модель – Представление – Модель представления») – шаблон проектирования архитектуры приложения, один из новейших шаблонов проектирования в семействе MV(X).

Шаблон MVVM делится на три части:

- модель (англ. Model) (так же, как в классической MVC) представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения;
- представление (англ. View) – графический интерфейс (окна, списки, кнопки и т. п.). Выступает подписчиком на событие изменения значений свойств или команд, предоставляемых моделью представления. В случае, если в модели представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и представление, в свою очередь, запрашивает обновлённое значение свойства из модели

представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную моделью представления;

- модель представления (англ. ViewModel) – с одной стороны, абстракция представления, а с другой – обёртка данных из модели, подлежащие связыванию. То есть, она содержит модель, преобразованную к представлению, а также команды, которыми может пользоваться представление, чтобы влиять на модель.

Реализация концепции MVVM в среде iOS приведена на рисунке 5.6.

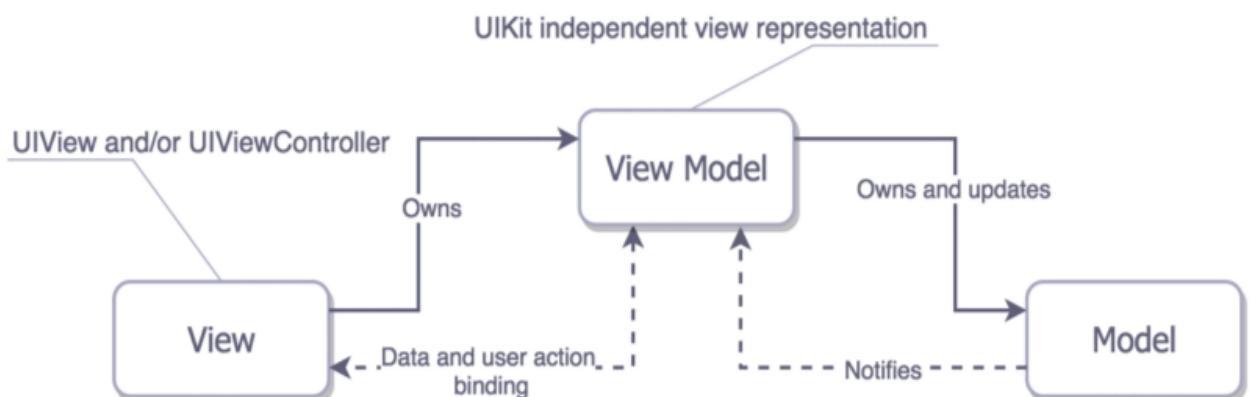


Рисунок 5.6 – Реализация шаблона MVVM в среде iOS

Шаблон проектирования MVVM тесно связан с понятием связывания данных.

Связывание данных – это процесс, который устанавливает соединение между UI (пользовательским интерфейсом) приложения и бизнес-логикой. Если настройки и уведомления установлены правильно, данные отражают изменения, когда они сделаны. Это может также значить, что, когда UI изменяется, лежащие в его основе данные будут отражать эти изменения.

По умолчанию в среде iOS не реализовано никаких средств для поддержки связывания данных. Поэтому, для того чтобы избежать реализации данной возможности самостоятельно, можно прибегнуть к следующим способам:

- одну из биндинг-библиотек, основанных на KVO (например, RZDataBinding или SwiftBond);
- полноразмерный фреймворк для функционального реактивного программирования, такой как ReactiveCocoa, RxSwift или PromiseKit.

Использование полноценного фреймворка для функционального реактивного программирования приводит к следующим проблемам:

- сложность написания кода, и, как следствие, высокий порог входления в подобные фреймворки;
- очень сложный процесс отладки приложения.

Оценим шаблон проектирования MVVM по признакам хорошей архитектуры:

- распределение: в MVVM представление имеет больше обязанностей, чем представление из MVP. Потому что первая обновляет свое состояние с моделью представления за счет механизма связывания данных, тогда как вторая направляет все события в Presenter и не обновляет себя (это делает Presenter);
- тестируемость: модель представления не знает ничего о представлении, это позволяет нам с легкостью тестировать ее. Для представление также можно реализовать покрытие модульными тестами;
- простота использования: меньшее количество кода по сравнению с концепцией MVP, поскольку механизм связывания данных избавляет от ручного обновления представления, которое производится Presenter в MVP. В то же время для того, чтобы иметь возможность использования механизма связывания данных, необходимо реализовать его либо вручную, либо использовать готовые решения, которые приносят свои сложности в разработку.

Таким образом, шаблон проектирования MVVM является очень привлекательной альтернативой, по сравнению с концепцией MVP, не требует дополнительного кода для обновления представления в связи с механизмом связывания данных на стороне представления, и в то же время тестируемость отдельных компонентов все еще находится на хорошем уровне. Но использования данного подхода заставляет прибегнуть к функционально реактивному программированию, что приводит к написанию более сложного кода и увеличению времени на отладку приложения.

Таким образом мы рассмотрели основные архитектуры семейства MV(X). Теперь же проведем их сравнение (таблица 5.1).

Таблица 5.1 – Сравнение архитектур по ключевым признакам

	MVC	MVP	MVVM
Простота использования и стоимость обслуживания	+	+/-	-
Скорость разработки	+	-	+/-
Распределение обязанностей	+/-	+/-	+
Тестирование кода	+/-	+	+

В качестве главных ключевых признаков для данного дипломного проекта были выделены: простота использования и стоимость обслуживания, и скорость разработки. Поскольку разрабатываемое мобильное приложение не является довольно большим, распределение обязанностей и степень тестируемости не являются критическими характеристиками в вопросе выбора архитектуры. В соответствии с этим, для данного дипломного проекта в качестве архитектуры был выбран шаблон MVC, а точнее, его реализацию в среде iOS – Cocoa MVC.

5.3 Реализация выбранной архитектуры

На основании выбранной архитектуры Сосоа MVC было разработано мобильное приложение, которое включает в себе следующие группы компонентов:

- ViewControllers;
- модели.

Каждый ViewController представляет собой пару View + ViewController. Каждый ViewController является сущностью отдельного экрана приложения. В данном приложении были разработаны следующие объекты ViewControllers:

- LoginViewController – экран входа в систему;
- TasksViewController – экран списка задач пользователя;
- ProjectsViewController – экран списка проектов пользователя;
- AccountViewController – экран профиля пользователя;
- ProjectDetailsViewController – экран детализированной информации о проекте;
- TaskDetailsViewController – экран детализированной информации о задаче;
- LogsViewController – экран журнала изменений задачи;
- CommentViewController – экран комментариев к задаче.

В свою очередь, группа «Модели» делиться на следующие подгруппы компонентов:

- сервисы;
- сущности.

Сервисы представляют специальные объекты, главное обязанность которых является манипулирование с данными мобильного приложения. В рамках данного дипломного приложения был разработан один сервис, главной задачей которого является обеспечение взаимодействия с сервером.

Сущности представляют собой объекты, описывающие данные, используемые в системе. В процессе реализации данного мобильного клиента были разработаны следующие объекты-сущности:

- Project – сущность, описывающая проект в системе;
- Task – сущность, описывающая задачу в системе;
- AccountInfo – сущность, описывающая профиль пользователя;
- TaskDetail – сущность, описывающая детальную информацию о задаче в системе;
- Comment – сущность, описывающая комментарий оставленный к задаче;
- LogItem – сущность, описывающая определенное изменение задачи в процессе ее существования в системе.

Таким образом, для данного мобильного приложения была реализована архитектура, соответствующая основным принципам шаблона Сосоа MVC.

Разработанная архитектура приведена на рисунке 5.7.

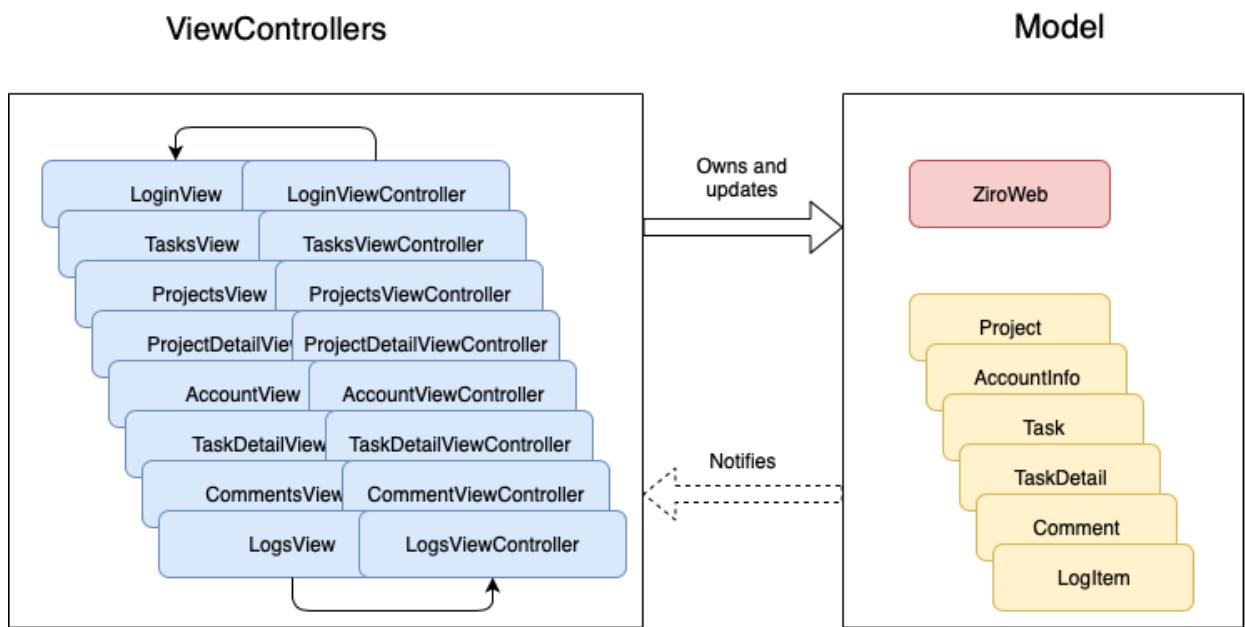


Рисунок 5.7 – Разработанная архитектура мобильного приложения

Схема аппаратного развертывания приведена на рисунке 5.8

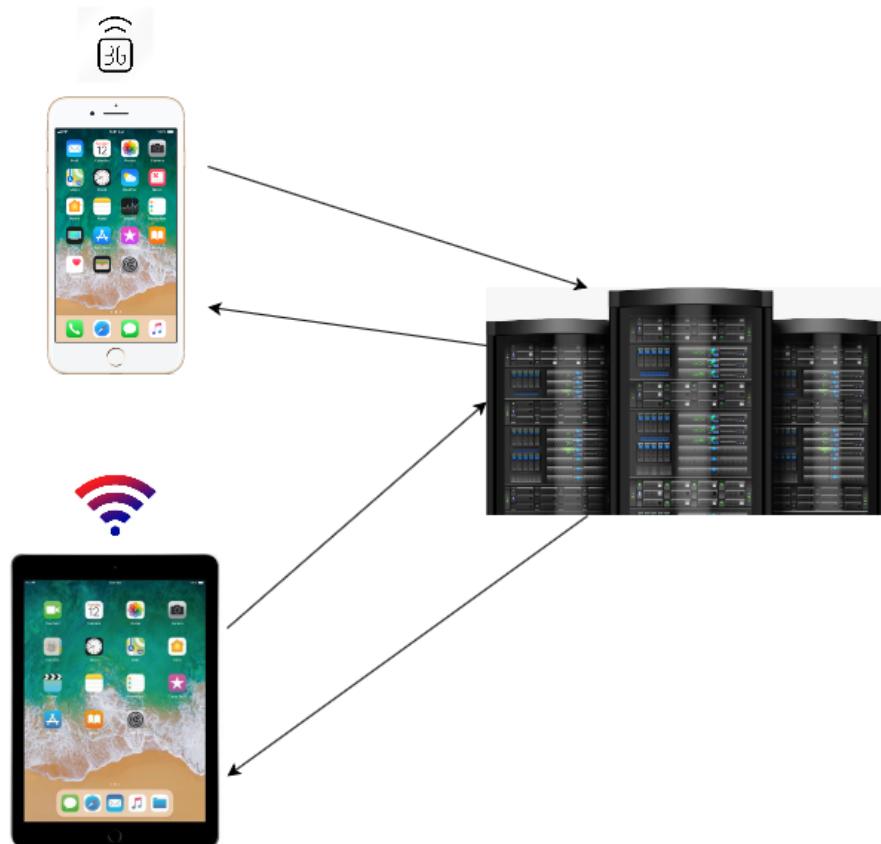


Рисунок 5.8 – Схема аппаратного развертывания

Для возможности использования разработанного программного продукта необходимо мобильное устройство или планшетный ПК, работающие на базе мобильной ОС Apple iOS.

Также используемое устройство должно иметь способность подключаться к Интернету, что может быть обеспечено одним из следующих технических средств:

- встроенный беспроводной модем с поддержкой сетей сотовой связи;
- встроенный модуль Wi-Fi.

Для возможности установки разработанного программного продукта на устройстве необходимы следующие программные средства:

- мобильная ОС Apple iOS 12 и выше;
- мобильное приложение App Store;
- персональная учетная запись Apple ID.

Схема программного развертывания разработанного мобильного приложения приведена на рисунке 5.9.

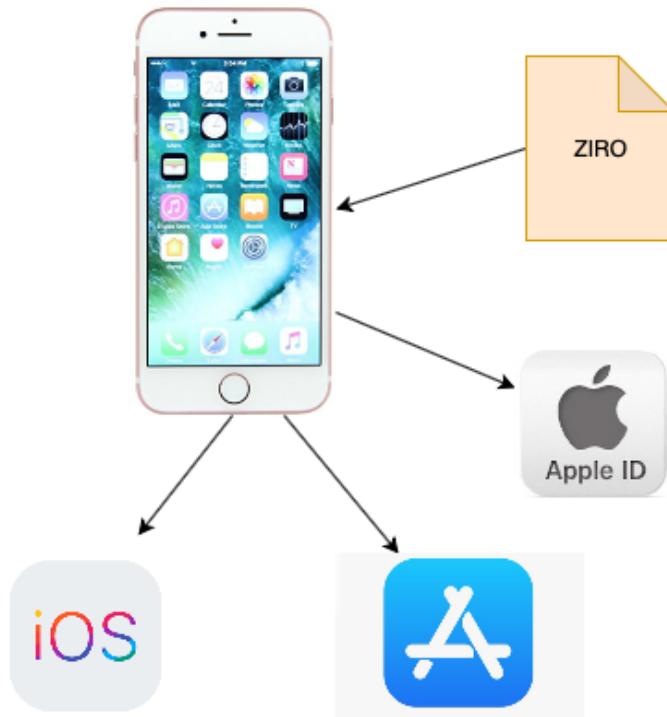


Рисунок 5.9 – Схема программного развертывания

6 РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

6.1 Развортывание мобильного клиента

Для того чтобы выложить приложение в App Store, потребуется оплаченный аккаунт разработчика, среда разработки XCode и исходный код вашего приложения. Разобьем процесс на четыре этапа:

- 1) настройка аккаунта;
- 2) оформление приложения для магазина;
- 3) настройка проекта, сборка и выгрузка;
- 4) отправка на рассмотрение.

6.1.1 Настройка аккаунта

Все приложения перед выгрузкой в App Store должны быть подписаны сертификатом разработчика. Это нужно, чтобы пользователи App Store были уверены, что скачивают конкретное приложение от конкретного разработчика, а не подделку от чужого имени. Процедура подписывания (code signing) приложения позволяет операционной системе узнать, кто является разработчиком. И удостовериться в том, что приложение не было изменено с момента сборки. Точнее, с того момента, как разработчик его подписал. В этой процедуре участвуют три объекта: сертификат разработчика, AppID и Provisioning profile.

Сертификат представляет собой пару ключей асимметричного шифрования: приватный и публичный. В процессе сборки XCode формирует цифровую подпись для сборки на основании данных приватного ключа. Проверить подпись можно с помощью публичного ключа, который доступен и для Apple, который этот сертификат выдала.

Чтобы создать сертификат, нужно:

- перейти на developer.apple.com, перейти в раздел «сертификаты», нажать на плюсик;
- выбрать тип сертификата Production – App Store and Ad Hoc;
- следуя приведенным инструкциям сформировать Certificate request и загрузить его на портал.

Сохраните сгенерированный сертификат на компьютер, откройте его (дважды кликнув). Сертификат будет помещен в системное хранилище и доступен для XCode.

AppID – уникальный строковый идентификатор приложения среди всех приложений. Он нужен для однозначной идентификации во всех системах: iTunes Connect, App Store и пр. Он состоит из двух частей: TeamID и BundleID. TeamID – идентификатор разработчика, выдается Apple на этапе регистрации аккаунта разработчика и не меняется. BundleID задается разработчиком при регистрации

приложения в аккаунте.

Что бы создать его, необходимо выполнить следующие шаги:

- перейдите в раздел AppIDs, нажмите на плюсик справа вверху
- заполните поле App ID Description – условное наименование приложения для разработчика, не видимое в App Store;
- заполните поле Explicit App ID – тот самый BundleID;
- в разделе App Services подключите те сервисы, которые понадобились в процессе разработки приложения.

Provisioning profile – это профиль, который однозначно связывает AppID и сертификат разработчика (публичный ключ сертификата). В случае development provisioning profile он также содержит UDID всех устройств, на которых возможен запуск приложения.

Процесс создания:

- перейдите в раздел iOS Provisioning Profiles, нажмите на плюсик справа вверху;
- тип профайла – App Store;
- затем необходимо указать AppID, для которого создается профайл – укажите ранее созданный AppID;
- следующий шаг – выбор сертификата. Отметьте галочкой сертификат, который мы создали ранее;
- последним шагом необходимо дать наименование профайлу.

Сохраните сгенерированный профайл на компьютер, откройте его, дважды кликнув. Теперь сертификат доступен для XCode.

6.1.2 Оформление приложения для магазина

Теперь нужно добавить приложение в iTunes Connect.

Переходим на <https://itunesconnect.apple.com> в раздел MyApps. Чтобы добавить приложение, нажмите плюсик слева вверху. Далее необходимо заполнить появившуюся форму (рисунок 6.1):

- указываем платформу – iOS;
- наименование приложения, будет отображаться в App Store. До 30 знаков;
- основной язык приложения;
- выберите из выпадающего списка AppID приложения.

Если все заполнено правильно, попадаем на страницу приложения.

На вкладке слева «Pricing and Availability» заполняем информацию о стоимости приложения в сторе. На вкладке слева с номером версии заполняем маркетинговую информацию:

- видео-превью и скриншоты для различных устройств;

- ключевые слова для поиска в магазине;
- описание;
- возрастной рейтинг;
- наименование правообладателя;
- контактную информацию.

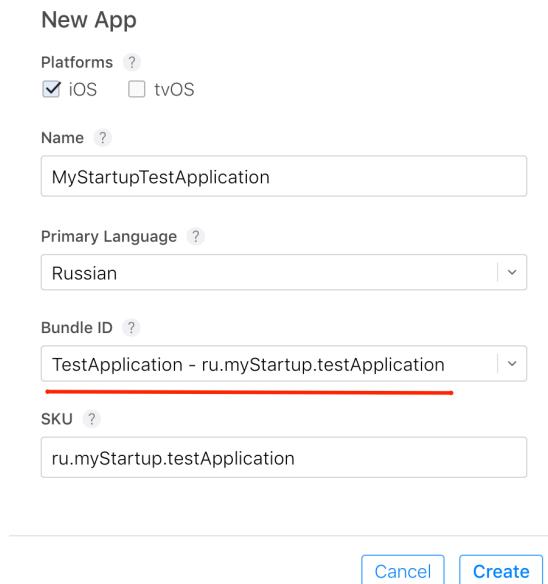


Рисунок 6.1 – Форма добавления нового приложения

6.1.3 Настройка проекта, сборка и выгрузка

Откройте проект вашего приложения в XCode, перейдите к настройкам проекта. Необходимо, чтобы Bundle Identifier совпадал с BundleID, который вы указали при создании AppID. Также необходимо отключить функцию автоматического управления подписыванием в XCode. В выпадающем списке Provisioning Profile выберите тот, который недавно создали.

Теперь проект можно собрать и отправить в iTunes Connect. Для этого нужно нажать меню Product – Archive. По итогу сборки будет показано окно организатора XCode, нажмите там кнопку «Upload To App Store» (рисунок 6.2).



Рисунок 6.2 – Окно организатора Xcode

На следующем шаге нужно выбрать provisioning profile из выпадающего списка. Затем XCode подготовит архив для выгрузки в iTunes Connect. На этом экране будут отображены параметры, которые были установлены до этого (рисунок 6.3). Далее необходимо нажать кнопку Upload.

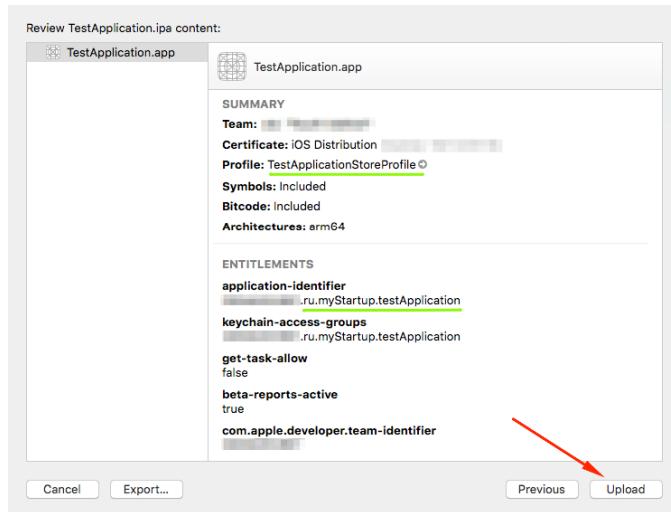


Рисунок 6.3 – Окно параметров

В зависимости от скорости соединения нужно будет подождать некоторое время. Если все в порядке, XCode сообщит об успешном завершении выгрузки в iTunes Connect. Можно переходить к последнему этапу.

6.1.4 Отправка приложения на рассмотрение

В iTunes Connect на вкладке Activity можно увидеть отправленную сборку. Для проектов на Swift автоматическая проверка сборки занимает примерно полчаса. До тех пор сборка будет со статусом Processing.

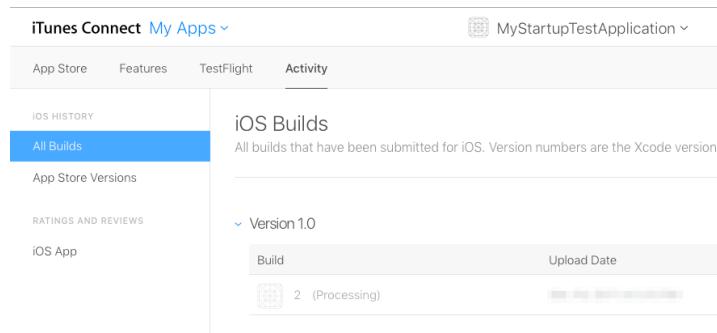


Рисунок 6.4 – Процесс автоматической сборки

После окончания проверки сборка доступна для выбора на странице информации

о версии приложения (рисунок 6.5).

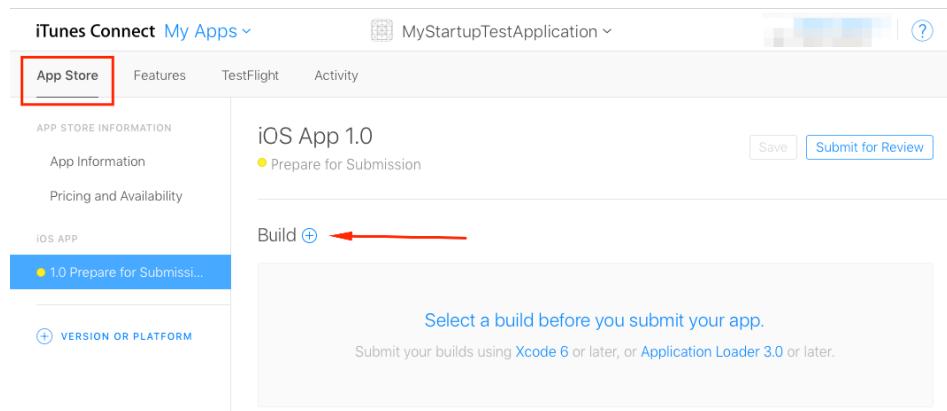


Рисунок 6.5 – Информация о версии приложения

Далее необходимо выбрать сборку и сохранить изменения, после чего сборку можно отправить на рассмотрение. Данный процесс обычно занимает несколько дней.

6.2 Тестирование мобильного клиента

Тестирование программного обеспечения – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Тестирование программного обеспечения будет проводится в рамках функционального тестирования, разделив его на критическое и углубленное.

Тестирование программы состоит из разработки тестовых случаев (тестов), их запуска и анализа полученных результатов.

Тестовые случаи – это алгоритмы проверки функциональности программы.

Каждый тестовый случай должен обладать следующими свойствами:

- четкой целью проверки;
- известными начальными условиями тестирования;
- строго определенной средой тестирования;
- тестовыми данными и ожидаемым результатом тестирования.

Тестирование проводилось на устройствах, указанных в таблице 6.1.

Таблица 6.1 – Матрица конфигурации мобильного приложения

	iPhone 6	iPhone XR	iPad Pro 9.7"
iOS 12.2	+	+	+

6.2.1 Критическое тестирование

Тест критического пути (critical path test) – основной тип тестовых испытаний, во время которого значимые элементы и функции приложения проверяются на предмет правильности работы при стандартном их использовании.

Чаще всего на практике на данном уровне тестирования проверяется основная масса требований к продукту. Пример: возможность набора текста, вставки картинок, возможность авторизации, создать запись, и т.д.

Тест критического пути является одним из самых распространенных видов функционального тестирования. Частота проведения данного тестирования обусловлена в первую очередь необходимостью периодической проверки всего приложения в сжатые сроки. А также позволяет выявить самые быстро находимые дефекты и исправить приложение в более сжатые сроки.

Для данного дипломного проекта была выделена следующая функциональность, для которой необходимо разработать тестовые случаи критического тестирования:

- авторизация пользователя;
- профиль пользователя;
- список проектов;
- детали задачи.

Разработанные тестовые случаи приведены в таблице 6.2.

Таблица 6.2 – Тестовые случаи для критического тестирования

№	Название модуля/экрана	Описание тестового случая	Ожидаемые результаты	Тестовый случай пройден Да/Нет
1	2	3	4	5
1	Экран авторизации	Успешная авторизация 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения	Да
2	Профиль пользователя	Просмотр профиля пользователя 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Профиль»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся экран с профилем пользователя	Да

Продолжение таблицы 6.2

1	2	3	4	5
3	Список проектов	Поиск проекта по имени 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Проекты» 6 В поисковое поле вводим значение «Twi» 7 Кликаем по кнопке поиска	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список проектов 6 В поисковом поле отобразилось значение «Twi» 7 В списке проектов отображен один проект с названием «Twitter»	Да
4	Список проектов	Сортировка списка проектов по критерию «Загруженность» 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Проекты» 6 В списке сортировок выбираем «По загруженности»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список проектов 6 Список проектов отсортирован по критерию «Загруженность»	Да
5	Список задач	Фильтрация задач по критерию «Важность» 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Задачи» 6 В списке фильтраций выбираем «Важные»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список задач 6 Список содержит только задачи, отвечающие критерию «Важные»	Да
6	Детали задачи	Просмотр проекта, к которому относится задача 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Задачи» 6 В списке выбираем задачу с именем «TWIT-1» 7 В секции «Проект» кликаем по полю «Название»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список задач 6 Открылся экран с деталями задачи «TWIT-1» 7 Открылся экран с деталями проекта «Twitter»	Да

Продолжение таблицы 6.2

1	2	3	4	5
7	Детали задачи	Просмотр журнала изменений 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Задачи» 6 В списке выбираем задачу с именем «TWIT-1» 7 В секции «Журнал изменений» кликаем по полю «Кол-во изменений»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список задач 6 Открылся экран с деталями задачи «TWIT-1» 7 Открылся экран с журналом изменений	Да
8	Список комментариев	Добавление комментария 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В текстовое поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Задачи» 6 В списке выбираем задачу с именем «TWIT-1» 7 В секции «Комментарии» кликаем по полю «Количество комментариев» 8 Нажимаем кнопку «Добавить» 9 В текстовое поле вводим значение «Текст» 10 Жмем кнопку «Сохранить»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список задач 6 Открылся экран с деталями задачи «TWIT-1» 7 Открылся экран со списком комментариев 8 Появилось всплывающее окно с текстовым полем 9 В текстовом поле отобразилось значение «Текст» 10 В список был добавлен комментарий с текстом «Текст»	Да

6.2.2 Углубленное тестирование

Углубленное (расширенное) тестирование – это процесс поиска ошибок в программе в нестандартных, непредвиденных ситуациях (например, при некорректно вводимых данных).

Углубленное тестирование используется для определения таких параметров работы приложения, как: проверка обработки исключительных ситуаций, исследование поведения приложения в незапланированном режиме работы и других возможных нефункциональных атрибутов, которые могут возникать как в следствии ошибок со стороны пользователей, так и их преднамеренных действий.

Примеры тестовых случаев для углубленного тестирования представлены ниже в таблице 6.3.

Таблица 6.3 – Тестовые случаи для углубленного тестирования

№	Название модуля/ экрана	Описание тестового случая	Ожидаемые результаты	Тестовый случай пройден Да/Нет
1	Экран авторизации	Авторизация при отсутствии интернета Предусловие: на устройстве необходимо отключить интернет 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Отобразилось всплывающее окно с текстом ошибки «Отсутствует интернет соединение»	Да
2	Экран авторизации	Авторизация с пустыми значениями email и пароля 1 Запускаем приложение 2 Нажимаем кнопку «Войти»	1 Открылся экран авторизации 2 Отобразилось всплывающее окно с текстом ошибки «Email и пароль не могут быть пустыми»	Да
3	Экран авторизации	Авторизация с неверным паролем 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В поле «Пароль» вводим «5231» 4 Нажимаем кнопку «Войти»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось четыре символа «*» 4 Отобразилось всплывающее окно с текстом ошибки «Неверный логин или пароль»	Да
4	Список проектов	Поиск проекта с пустым именем 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Проекты» 6 Кликаем по поисковому полю 7 Кликаем по кнопке поиска	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список проектов 6 Появилась клавиатура устройства 7 В списке проектов отображены все проекты	Да
5	Список комментариев	Добавление пустого комментария 1 Запускаем приложение 2 В текстовое поле «Email» вводим «testUser@mail.com» 3 В поле «Пароль» вводим «523» 4 Нажимаем кнопку «Войти» 5 На списке вкладок кликаем по вкладке «Задачи» 6 В списке выбираем задачу с именем «TWIT-1» 7 В секции «Комментарии» кликаем по полю «Количество комментариев» 8 Нажимаем кнопку «Добавить» 9 Жмем кнопку «Сохранить»	1 Открылся экран авторизации 2 В поле «Email» отобразилось значение «testUser@mail.com» 3 В поле «Пароль» отобразилось три символа «*» 4 Открылся главный экран приложения 5 Открылся список задач 6 Открылся экран с деталями задачи «TWIT-1» 7 Открылся экран со списком комментариев 8 Появилось всплывающее окно с текстовым полем 9 Отобразилось всплывающее окно с текстом ошибки «Комментарий не может быть пустым»	Да

Таким образом, в результате проведение критического и углубленного тестирования не было выявлено каких-либо ошибок. Заявленный функционал реализован полностью и работает в соответствии с требованиями.

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске приложения пользователю необходимо авторизоваться для того, чтобы иметь возможность начать работу с приложением. Для этого необходимо ввести свои данные на экране входа в приложения, который изображен на рисунке 7.1.

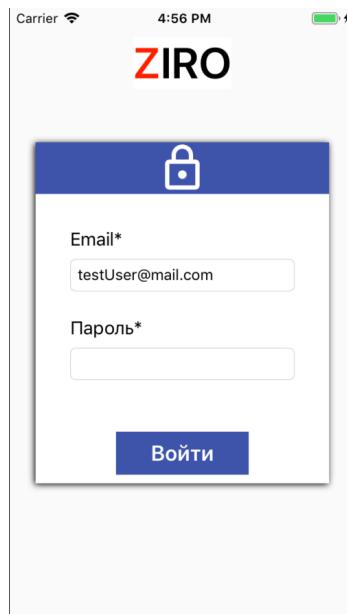


Рисунок 7.1 – Экран авторизации

Если email и пароль введены верно, то пользователь перейдет на главный экран приложения, который приведен на рисунке 7.2.

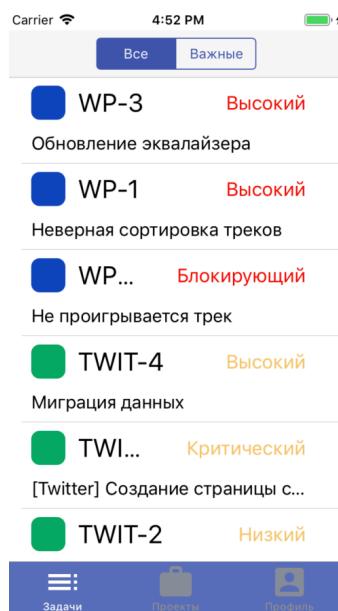


Рисунок 7.2 – Главный экран приложения

Данный экран состоит из нескольких вкладок, переключатель расположен в самом низу экрана. Всего данный экран содержит три вкладки: список задач текущего пользователя, список проектов текущего пользователя и профиль текущего пользователя.

По умолчанию открывается вкладка со списком задач. Каждая задача представлена отдельным элементом, который приведен на рисунке 7.3. Данный элемент содержит следующую информацию о задаче:

- номер задачи;
- заголовок;
- приоритет;
- статус.

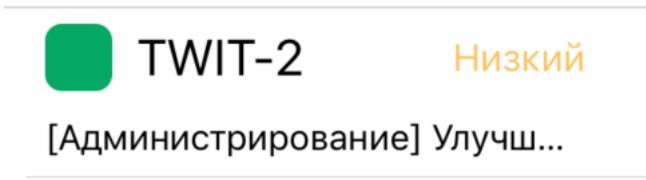


Рисунок 7.3 – Элемент «Задача»

Также, экран со списком задач содержит одну доступную фильтрацию – показать только важные задачи. Задача является важной в том случае, когда ее приоритет «Высокий» или выше.

Каждая из задач является кликабельной, что позволяет открыть ее детализированную версию. Пример экрана с деталями задачи приведен на рисунке 7.4.

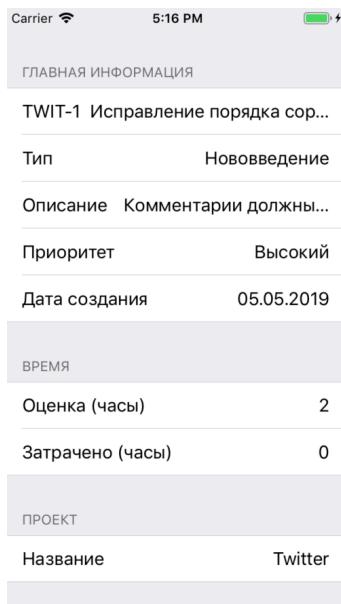


Рисунок 7.4 – Детали задачи

Данный экран содержит много различной информации о задаче пользователя. Стоит отметить, что такие поля, например, как описание задачи, являются довольно длинными и очень часто не влезают на экран. Поэтому реализована возможность просмотра таких полей, путем нажатия на них. В результате появится всплывающее окно, приведенное на рисунке 7.5.

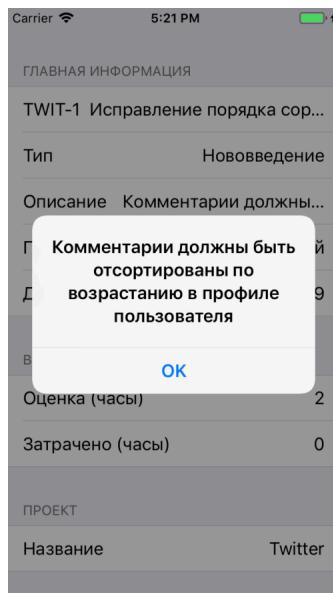


Рисунок 7.5 – Полный текст поля «Описание»

Данный экран имеет возможность прокрутки, поскольку не всю информацию можно уместить на всех устройствах. Пример оставшейся информации о вышеописанной задачи приведен на рисунке 7.6.

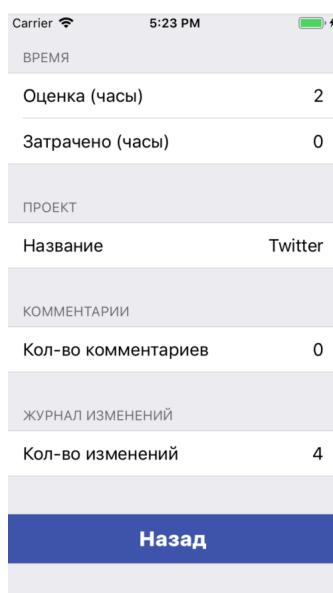


Рисунок 7.6 – Детали задачи

Поля «Название проекта», «Кол-во комментариев» и «Кол-во изменений» так же являются кликабельными. При нажатии на название проекта откроется экран с детализированной информацией о проекте, который приведен на рисунке 7.7.

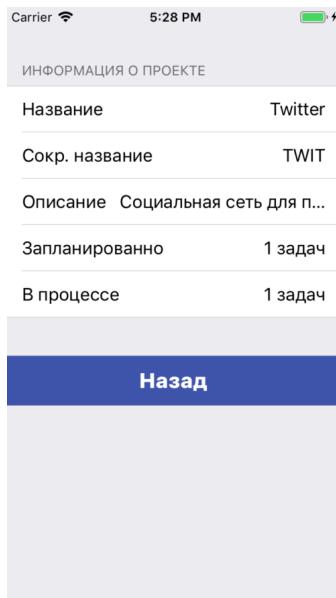


Рисунок 7.7 – Экран «Детали проекта»

При нажатии на поле «Кол-во комментариев» откроется экран с комментариями к данной задаче, который приведен на рисунке 7.8

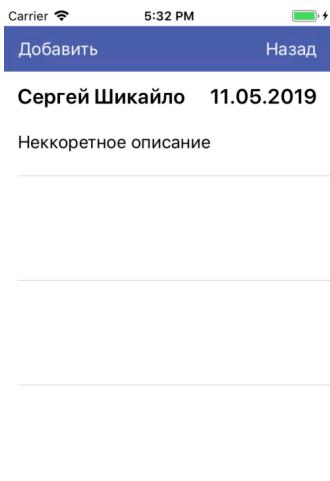


Рисунок 7.8 – Список комментариев к задаче

Здесь же пользователь может оставить свой комментарий к задаче. Для этого необходимо кликнуть на кнопку «Добавить» и ввести текст в всплывающее окно, после

чего нажать кнопку «Сохранить». Пример добавления комментария приведен на рисунке 7.9.

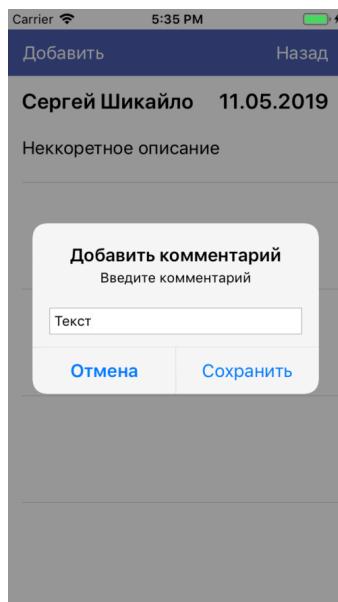


Рисунок 7.9 – Добавление комментария

Вернемся к главному экрану приложения. Следующая вкладка содержит список проектов, в которых участвует данный пользователь. Данный экран содержит функционал для возможности поиска по имени проекта и сортировки по имени и важности проекта. Важность проекта подразумевает под собой количество запланированных задач и задач, которые на данный момент находятся в процессе выполнения. Пример списка проектов приведен на рисунке 7.10.

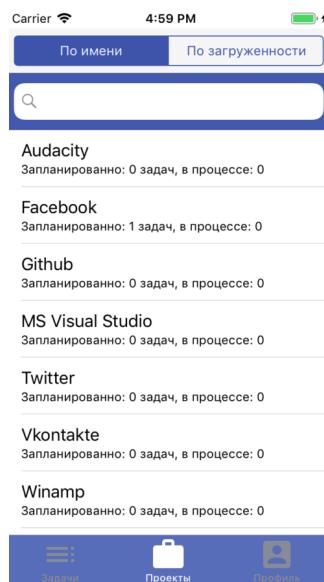


Рисунок 7.10 – Список проектов

И, наконец, третьей вкладкой главного экрана является профиль пользователя, который содержит всю основную информацию о текущем пользователе. Пример данного экрана приведен на рисунке 7.11.

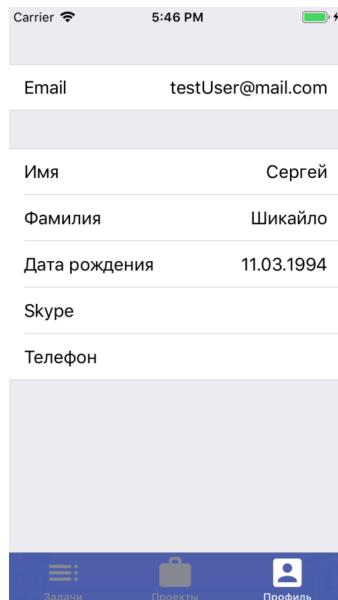


Рисунок 7.11 – Профиль пользователя

Стоит также отметить, что мобильное приложение является адаптированным для работы на планшетах iPad (рисунок 7.12).

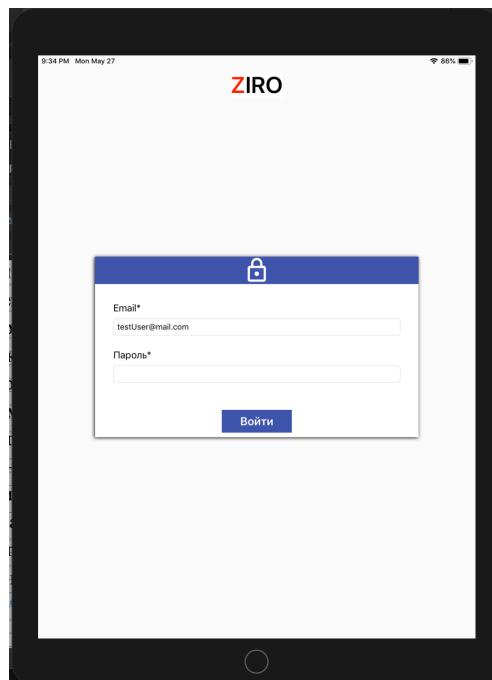


Рисунок 7.12 – Экран авторизации на устройстве iPad Pro

8 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

8.1 Расчет сметы затрат, цены и прибыли на программное средство

В современных рыночных экономических условиях программное средство выступает преимущественно в виде продукции научно-технических организаций, представляющей собой функционально завершенные и имеющие товарный вид программные средства, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки программных средств являются научно-технической продукцией.

Выбор эффективных проектов связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли, остающейся в распоряжении предприятия от реализации программного средства, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

- снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ за счет использования программного средства в процессе разработки автоматизированных систем обработки данных;
- сокращения расходов на оплату машинного времени и других ресурсов на отладку программ;
- снижения расходов на материалы (магнитные ленты, магнитные диски и прочие материалы);
- ускорение ввода в эксплуатацию новых систем;
- улучшения показателей основной деятельности предприятий в результате использования программного средства.

Стоймостная оценка программного средства у разработчиков предполагает составление сметы затрат, которая включает следующие статьи:

- заработка платы исполнителей основная (Z_0) и дополнительная (Z_d);
- отчисления в фонд социальной защиты населения ($Z_{СЗ}$);
- отчисления на развитие здравоохранения и охрану здоровья ($Z_{ОЗ}$);
- налоги, от фонда оплаты труда (H_e);
- материалы и комплектующие (M);
- спецоборудование (P_C);
- машинное время (P_M);
- расходы на научные командировки ($P_{НК}$);

- прочие прямые затраты (Пз);
- накладные расходы (Рн).

На основании сметы затрат рассчитывается себестоимость и отпускная цена программного средства.

8.1.1 Исходные данные

Исходные данные для расчета заносятся в таблицу (таблица 8.1).

Таблица 8.1 – Исходные данные для расчета

№ пп	Наименование показателя	Единица измерения	Условные обозначения	Норматив
1.	Коэффициент изменения скорости обработки информации	ед.	K _{ск}	0,5
2.	Численность разработчиков	чел.	Ч _р	1
3.	Тарифная ставка 1-го разряда в организации	руб.	T _{м1}	36,4
4.	Тарифный коэффициент	ед.	T _к	2,48
5.	Норматив дополнительной заработной платы рассчитывается по формуле	%	H _д	15
6.	Норматив отчислений в фонд социальной защиты населения	ед.	H _{сз}	35
7.	Норматив налога, уплачиваемого единым платежом	%	H _{не}	4
8.	Норматив расходов на командировки в целом по научной организации	%	H _{рнк}	30
9.	Норматив прочих затрат в целом по научной организации	%	H _{пз}	20
10.	Норматив накладных расходов в целом по научной организации	%	H _{рн}	100
11.	Уровень рентабельности программного средства	%	У _{рн}	55
12.	Норматив расходов на сопровождение и адаптацию	%	H _{рса}	4
13.	Норматив отчислений в местный и республиканский бюджеты	%	H _{мр}	3,9
14.	Норматив НДС	%	H _{дс}	20
15.	Налог на прибыль при отсутствии льгот	%	H _н	24
16.	Норматив расхода машинного времени на отладку 100 строк исходного кода	час	H _{мв}	12
17.	Количество дней в году	день	Д _г	365
18.	Количество праздничных дней в году	день	Д _п	9
19.	Количество выходных дней в году	день	Д _в	104
20.	Количество дней отпуска	день	Д _о	24

8.1.2 Общая характеристика разрабатываемого программного средства

Данное программное средство является программным средством общего назначения, основные функции:

- организация ввода информации;
- контроль, предварительная обработка и ввод информации;
- управление вводом/выводом;
- организация ввода/вывода информации в интерактивном режиме;
- обработка ошибочных и сбойных ситуаций;
- справка и обучение.

Разрабатываемое программное средство входит в третью группу сложности.

8.1.3 Определение объема программного средства

Объем программного средства определяется путем подбора аналогов на основании классификации типов программных средств, каталога функции программного средства и аналогов программного средства в разрезе функций, которые постоянно обновляются и утверждаются в установленном порядке. На основании информации о функциях разрабатываемого программного средства по каталогу функций определяется объем функций. Общий объем программного средства рассчитывается по формуле:

$$V_o = \sum_{i=1}^n V_i, \quad (8.1)$$

где V_o – общий объем программного средства, строка исходного кода;

V_i – объем функций программного средства, строка исходного кода;

n – общее число функций.

Содержание и объем функций разрабатываемого программного средства приведено в таблице 8.2.

Таблица 8.2 – Содержание и объем функций разрабатываемого программного средства

№ функции	Содержание функции	Объем функций (строки исходного кода)
101	Организация ввода информации	150
102	Контроль, предварительная обработка и ввод информации	450
109	Организация ввода/вывода информации в интерактивном режиме	320
111	Управление вводом/выводом	2 400
506	Обработка ошибочных и сбойных ситуаций	410
604	Справка и обучение	720
	ИТОГО	4 450

$$V_o = 150 + 450 + 320 + 2400 + 410 + 720 = 4450 \text{ строк исходного кода.}$$

8.1.4 Расчет трудоемкости выполняемой работы

На основании общего объема программного средства определяется нормативная трудоемкость (T_n). Нормативная трудоемкость устанавливается с учетом сложности программного средства. Выделяются три группы сложности, в которых учтены следующие составляющие программного средства; языковой интерфейса, ввод-вывод, организация данных, режим работы, операционная и техническая среда. Кроме того, устанавливаются дополнительные коэффициенты сложности программного средства.

С учетом дополнительного коэффициента сложности K_{cl} рассчитывается общая трудоемкость программного средства:

$$T_o = T_n \cdot K_{cl}, \quad (8.2)$$

где T_o – общая трудоемкость ПС, человеко-день;

T_n – нормативная трудоемкость ПС, человеко-день;

K_{cl} – дополнительный коэффициент сложности ПС.

Поскольку разрабатываемое ПС относится к группе сложности 3, то нормативная трудоемкость T_n равна 112. ПС обладает характеристикой функционирования в расширенной операционной среде, поэтому коэффициент сложности K_{cl} равен 0.08.

$$T_o = 112 * 0,08 = 8,96 \text{ человеко – дней.}$$

При решении сложных задач с длительным периодом разработки ПС трудоемкость определяется по стадиям разработки (техническое задание – ТЗ, эскизный проект – ЭП, технический проект – ТП, рабочий проект – РП и внедрение – ВН) с учетом новизны, степени использования типовых программ и удельного веса трудоемкости стадий разработки ПС и общей трудоемкости разработки ПС. При этом на основании общей трудоемкости рассчитывается уточненная трудоемкость с учетом распределения по стадиям (T_y).

$$T_y = \sum_{i=1}^m T_i, \quad (8.3)$$

где T_i – трудоемкость разработки ПС на i -й стадии, человеко-день;

m – количество стадий разработки.

Трудоемкость ПС по стадиям определяется с учетом новизны и степени

использования в разработке типовых программ и ПС.

$$T_{cti} = d_{cti} \cdot K_H \cdot K_T \cdot T_o, \quad (8.4)$$

где T_{cti} – трудоемкость разработки ПС на i -й стадии (технического задания, эскизного проекта, технического проекта, рабочего проекта и внедрения), человеко-день;

K_H – поправочный коэффициент, учитывающий степень новизны ПС;

K_T – поправочный коэффициент, учитывающий степень использования в разработке типовых программ и ПС;

d_{cti} – удельный вес трудоемкости i -й стадии разработки ПС в общей трудоемкости разработки ПС.

На основании данных в приложениях разрабатываемое ПС относится к 3-ей группе сложности и имеет степень новизны В, при этом $K_{сл}=0,08$, а $K_H=0,7$. Коэффициенты удельных весов трудоемкостей для каждой стадии: $T_3=0,09$; $\mathcal{E}П=0,07$; $ТП=0,07$; $РП=0,61$; $ВН=0,16$.

Исходя из этих данных, можно рассчитать трудоемкость ПС на каждой стадии:

$$T_{ctt3} = 0,09 * 0,7 * 0,7 * 8,96 = 0,4 \text{ человека – дней.}$$

$$T_{ct\mathcal{E}П} = 0,07 * 0,7 * 0,7 * 8,96 = 0,31 \text{ человека – дней.}$$

$$T_{cttп} = 0,07 * 0,7 * 0,7 * 8,96 = 0,31 \text{ человека – дней.}$$

$$T_{ctrп} = 0,61 * 0,7 * 0,7 * 8,96 = 2,68 \text{ человека – дней.}$$

$$T_{ctvн} = 0,16 * 0,7 * 0,7 * 8,96 = 0,7 \text{ человека – дней.}$$

Тогда уточненная трудоемкость с учетом распределения по стадиям равна:

$$T_y = 0,4 + 0,31 + 0,31 + 2,68 + 0,7 = 4,39 \text{ человека – дня.}$$

8.1.5 Расчет основной заработной платы исполнителей

Эффективный фонд времени работы одного работника ($\Phi_{\mathcal{E}Ф}$) рассчитывается по формуле:

$$\Phi_{\mathcal{E}Ф} = D_g - D_p - D_v - D_o, \quad (8.5)$$

где D_g – количество дней в году, день;

D_p – количество праздничных дней в году, день;

D_v – количество выходных дней в году, день;

До – количество дней отпуска, день.

При утверждении плановой численности разработчиков продолжительность разработки определяется по формуле:

$$T_p = \sum_{i=1}^m \frac{T_i}{\Psi_{pi} \cdot \Phi_{\text{эф}}}, \quad (8.6)$$

где T_p – срок разработки ПС (лет);

T_i – трудоемкость разработки ПС на i -й стадии (человеко-дней);

Ψ_{pi} – численность разработчиков ПС на i -й стадии (чел.);

m – число стадий.

Уточненная трудоемкость и общая плановая численность разработчиков служат базой для расчета основной заработной платы. По данным о специфике и сложности выполняемых функций составляется штатное расписание группы специалистов-исполнителей, участвующих в разработке ПС, с определением образования, специальности, квалификации и должности.

Таким образом, можно рассчитать эффективный фонд времени:

$$\Phi_{\text{эф}} = 365 - 9 - 104 - 24 = 228 \text{ дней.}$$

А также срок разработки ПС при количестве разработчиков на всех стадиях разработки ПС равным 1.

$$T_p = \frac{4,39}{1 * 228} = 0,019 \text{ лет.}$$

В соответствии с «Рекомендациями по применению «Единой тарифной сетки» рабочих и служащих народного хозяйства» и тарифными разрядами и коэффициентами должностей руководителей научных организаций и вычислительных центров, бюджетных учреждений науки непроизводственных отраслей народного хозяйства каждому исполнителю устанавливается разряд и тарифный коэффициент.

Месячная тарифная ставка каждого исполнителя (T_m) определяется путем умножения действующей месячной тарифной ставки 1-го разряда (T_{m1}) на тарифный коэффициент (T_k), соответствующий установленному тарифному разряду:

$$T_m = T_{m1} * T_k. \quad (8.7)$$

Часовая тарифная ставка рассчитывается путем деления месячной тарифной ставки на установленный при семичасовом рабочем дне фонд рабочего времени Φ_p :

$$T_q = \frac{T_m}{\Phi_p}, \quad (9.8)$$

где T_q – часовая тарифная ставка, руб.;

T_m – месячная тарифная ставка, руб.

Работник принадлежит 10-му разряду. Тарифная месячная и часовая ставки для него будут равны:

$$T_m = 36,4 * 2,48 = 90,27 \text{ руб.}$$

$$T_q = \frac{90,27}{133} = 0,68 \text{ руб.}$$

Основная заработная плата исполнителей на конкретное ПС рассчитывается по формуле

$$Z_{oi} = \sum_{i=1}^n T_{qi} \cdot T_q \cdot \Phi_{ei} \cdot K, \quad (8.9)$$

где n – количество исполнителей, занятых разработкой конкретного ПС;

T_{qi} – часовая тарифная ставка i -го исполнителя (руб.);

Φ_{ei} – эффективный фонд рабочего времени i -го исполнителя (дней);

T_q – количество часов работы в день (ч);

K – коэффициент премирования.

$$Z_o = 1 * 0,68 * 133 * 1 = 90,27 \text{ руб.}$$

8.1.6 Расчет дополнительной заработной платы исполнителей

Дополнительная заработка на конкретное ПС (Z_{di}) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате:

$$Z_{di} = \frac{Z_{oi} \cdot H_d}{100}, \quad (8.10)$$

где Z_{di} – дополнительная заработка исполнителей на конкретное ПС, руб.;

H_d – норматив дополнительной заработной платы, %.

$$Z_d = \frac{90,27 * 15}{100} = 13,54 \text{ руб.}$$

8.1.7 Расчет отчислений в фонд социальной защиты населения

Отчисления в фонд социальной защиты населения (Z_{czi}) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу, установленному в целом по организации:

$$Z_{czi} = \frac{(Z_{oi} + Z_{di}) \cdot H_{cz}}{100}, \quad (8.11)$$

где H_{cz} – норматив отчислений в фонд социальной защиты населения, %.

$$Z_{czi} = \frac{(90,27 + 13,54) * 35}{100} = 36,33 \text{ руб.}$$

8.1.8 Расчет налога на ликвидацию последствий чернобыльской катастрофы

Налоги рассчитываемые от фонда оплаты труда определяются в соответствии с действующими законодательными актами по нормативам в процентном отношении к сумме всей заработной платы, относимой на ПС (налог, уплачиваемый единым платежом, включая налог на ликвидацию последствий чернобыльской катастрофы и отчисления в фонд занятости (H_{ei})):

$$H_{ei} = \frac{(Z_{oi} + Z_{di}) \cdot H_{ne}}{100}, \quad (8.12)$$

где H_{ne} – норматив налога, уплачиваемого единым платежом (%).

$$H_{ei} = \frac{(90,27 + 13,54) * 4}{100} = 4,15 \text{ руб.}$$

8.1.9 Расчет расходов по статье «Материалы»

Расходы по статье «Материалы» (M) определяются на основании сметы затрат,

разрабатываемой на ПС с учетом действующих нормативов.

По статье «Материалы» отражаются расходы на магнитную носители, перфокарты, бумагу, красящие ленты и другие материалы, необходимые для разработки ПС. Нормы расхода материалов в суммарном выражении (H_M) определяются в расчете на 100 строк исходного кода. Сумма затрат материалов рассчитывается по формуле:

$$M_i = H_{Mi} \cdot \frac{V_{oi}}{100}, \quad (8.13)$$

где H_{Mi} – норма расхода материалов в расчете на 100 строк исходного кода ПС, руб.;

V_{oi} – общий объем ПС на конкретное ПС, строка исходного кода.

$$M_i = 0,038 * \frac{4\,450}{100} = 1,69 \text{ руб.}$$

8.1.10 Расчет расходов по статье «Спецоборудование»

Расходы по статье «Спецоборудование» (P_{ci}) включает затраты средств на приобретение вспомогательных специального назначения технических и программных средств, необходимых для разработки конкретного ПС, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию.

Затраты по этой статье определяются в соответствии со сметой расходов, которая составляется перед разработкой ПС. Данная статья включается в смету расходов на разработку ПС в том случае, когда приобретаются специальное оборудование или специальные программы, предназначенные для разработки и создания только данного ПС:

$$P_{ci} = \sum_{i=1}^n \Pi_{ci}, \quad (8.14)$$

где Π_{ci} – стоимость конкретного специального оборудования, руб.;

n – количество применяемого специального оборудования.

Необходимо учесть следующие расходы по данной статье:

- аккаунт разработчика Apple – 17 рублей.

$$P_{ci} = 17 \text{ руб.}$$

8.1.11 Расчет расходов по статье «Машинное время»

Расходы по статье «Машинное время» (P_{Mi}) включают оплату машинного времени, необходимого для разработки и отладки ПС, которое определяется по нормативам (в машино-часах) на 100 строк исходного кода (H_{MB}) машинного времени в зависимости от характера решаемых задач и типа ПЭВМ:

$$P_{Mi} = \Pi_{Mi} \cdot \frac{V_{oi}}{100} \cdot H_{MB}, \quad (8.15)$$

где Π_{Mi} – цена одного машино-часа, руб.;

V_{oi} – общий объем ПС, строка исходного кода;

H_{MB} – норматив расхода машинного времени на отладку 100 строк исходного кода, машино-час.

Цена одного машино-часа (Π_{Mi}) составляет 2 рубля.

$$P_{Mi} = 2 * \left(\frac{4450}{100} \right) * 12 = 1068 \text{ руб.}$$

8.1.12 Расчет расходов по статье «Научные командировки»

Расходы по статье «Научные командировки» (P_{Hki}) на конкретное ПС определяются по нормативу, разрабатываемому в целом по научной организации, в процентах к основной заработной плате:

$$P_{Hki} = \frac{3_{oi} \cdot H_{pHK}}{100}, \quad (8.16)$$

где H_{pHK} – норматив расходов на командировки в целом по научной организации, %.

$$P_{Hki} = \frac{90,27 * 30}{100} = 27,08 \text{ руб.}$$

8.1.13 Расчет расходов по статье «Прочие затраты»

Расходы по статье «Прочие затраты» (Π_{3i}) на конкретное ПС включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы.

Определяются по нормативу, разрабатываемому в целом по научной организации, в процентах к основной заработной плате:

$$\Pi_{3i} = \frac{3_{oi} \cdot H_{pz}}{100}, \quad (8.17)$$

где H_{pz} – норматив прочих затрат в целом по научной организации.

$$\Pi_{3i} = \frac{90,27 * 20}{100} = 18,05 \text{ руб.}$$

8.1.14 Расчет расходов по статье «Накладные расходы»

Затраты по статье «Накладные расходы» (P_{hi}), связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды (P_{ph}), относятся на конкретное ПС по нормативу (H_{ph}) в процентном отношении к основной заработной плате исполнителей. Норматив устанавливается в целом по научной организации:

$$P_{hi} = \frac{3_{oi} \cdot H_{ph}}{100}, \quad (8.18)$$

где P_{hi} – накладные расходы на конкретное ПС, руб.;

H_{ph} – норматив накладных расходов в целом по научной организации.

$$P_{hi} = \frac{90,27 * 100}{100} = 90,27 \text{ руб.}$$

8.1.15 Расчет общей суммы расходов на разработку

Общая сумма расходов по всем статьям сметы (C_{pi}) на ПС рассчитывается по формуле:

$$C_{pi} = 3_{oi} + 3_{di} + 3_{czi} + H_{ei} + M_i + P_{ci} + P_{mi} + P_{hki} + \Pi_{3i} + P_{hi}. \quad (8.19)$$

Кроме того, организация-разработчик осуществляет затраты на сопровождение и адаптацию ПС (P_{CAi}), которые определяются по нормативу (H_{PCA})

$$\begin{aligned} C_{pi} &= 90,27 + 13,54 + 36,33 + 4,15 + 1,69 + 17 + 1068 + 27,08 + 18,05 + 90,27 \\ &= 1\ 366,4 \text{ руб.} \end{aligned}$$

$$P_{cai} = \frac{C_{pi} \cdot H_{pca}}{100}, \quad (8.20)$$

где H_{pca} – норматив расходов на сопровождение и адаптацию, %.

$$P_{cai} = \frac{1\ 366,4 * 4}{100} = 54,66 \text{ руб.}$$

Общая сумма расходов на разработку (с затратами на сопровождение и адаптацию программного средства) как полная себестоимость ПС (C_{pi}) определяется по следующей формуле:

$$C_{pi} = C_{pi} + P_{cai}. \quad (8.21)$$

$$C_{pi} = 1\ 366,4 + 54,66 = 1\ 421,06 \text{ руб.}$$

8.1.16 Расчет прибыли и отпускной цены

Рентабельность и прибыль по создаваемому ПС определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены,

Отпускная цена включает дополнительно налог на добавленную стоимость и отчисления на содержание ведомственного жилого фонда. Прибыль рассчитывается по формуле:

$$\Pi_{nci} = \frac{C_{pi} \cdot Y_{ppi}}{100}, \quad (8.22)$$

где Π_{nci} – прибыль от реализации ПС, руб.;

Y_{ppi} – уровень рентабельности ПС, %;

C_{pi} – себестоимость ПС, руб.

$$\Pi_{nci} = \frac{1\ 421,06 * 55}{100} = 781,58 \text{ руб.}$$

Прогнозируемая цена ПС без налогов ($\Pi_{\text{пi}}$):

$$\Pi_{\text{пi}} = C_{\text{пi}} + \Pi_{\text{ci}}. \quad (8.23)$$

$$\Pi_{\text{пi}} = 1\,421,06 + 781,58 = 2\,202,64 \text{ руб.}$$

Отчисления и налоги в местный и республиканский бюджеты единым платежом ($O_{\text{мрi}}$):

$$O_{\text{мрi}} = \frac{\Pi_{\text{пi}} \cdot H_{\text{мр}}}{100\%}, \quad (8.24)$$

$$O_{\text{мрi}} = \frac{(2\,202,64 * 3,9)}{100} = 85,9 \text{ руб.}$$

где $H_{\text{мр}}$ – норматив отчислений в местный и республиканский бюджеты (%).

Налог на добавленную стоимость (НДС_i):

$$\text{НДС}_i = \frac{(\Pi_{\text{пi}} + O_{\text{мр}}) \cdot H_{\text{дс}}}{100\%}, \quad (8.25)$$

где $H_{\text{дс}}$ – норматив НДС, %.

$$\text{НДС}_i = \frac{(2\,202,64 + 85,9) * 20}{100} = 457,71 \text{ руб.}$$

Прогнозируемая отпускная цена ($\Pi_{\text{оi}}$):

$$\Pi_{\text{оi}} = \Pi_{\text{пi}} + O_{\text{мрi}} + \text{НДС}_i. \quad (8.26)$$

$$\Pi_{\text{оi}} = 2\,202,64 + 85,9 + 457,71 = 2\,746,25 \text{ руб.}$$

8.2 Расчет экономического эффекта от применения ПС у пользователя

8.2.1 Исходные данные

Исходные данные для расчета экономического эффекта от применения ПС у

пользователя сведены в таблицу 8.3.

Таблица 8.3 – Исходные данные для расчета экономического эффекта

Наименование показателей	Обозначения	Единицы измерения	Значение показателя	
			в базовом варианте	в новом варианте
1. Средняя трудоемкость работ в расчете на 100 КБ	T_{c1} T_{c2}	человеко-час на 100 КБ	1,8	1,7
2. Средний расход машинного времени в расчете на 100 КБ	M_{v1} M_{v2}	машино-час на 100 КБ	9,0	8,91
3. Средний расход материалов в расчете на 100 КБ	M_{t1} M_{t2}	руб. на 100 КБ	0,032	0,028
4. Количество типовых задач, решаемых за год	Z_{r2}	задача	9	
5. Ставка налога на прибыль	H_p	%	18	

8.2.2 Расчет объема работ

Объем работ в зависимости от функциональной группы и назначения ПС можно определить по формуле:

$$A = V_{nc} \cdot K_{nc}, \quad (8.27)$$

где V_{nc} – объем ПС в натуральных единицах измерения;
 K_{nc} – коэффициент применения ПС.

$$A = 4\,450 * 0,5 = 2\,225.$$

8.2.3 Расчет капитальных затрат

Общие капитальные вложения (K_o) заказчика (потребителя), связанные с приобретением, внедрением и использованием ПС, рассчитываются по формуле:

$$K_o = K_{np} + K_{oc} + K_{tc} + K_{ob}, \quad (8.28)$$

где K_{np} – затраты пользователя на приобретение ПС по отпускной цене разработчика с учетом стоимости услуг по эксплуатации и сопровождению, руб.;

K_{oc} – затраты пользователя на освоение ПС, руб.;

K_{tc} – затраты на доукомплектацию ВТ техническими средствами в связи с внедрением нового ПС, руб.;

K_{ob} – затраты на пополнение оборотных средств в связи с использованием нового ПС, руб.

$$K_o = 2\ 746,25 + 27,46 + 0 + 27,46 = 2\ 801,17 \text{ руб.}$$

Затраты на освоение ПС и на пополнение оборотных средств рекомендуется рассчитывать по формулам:

$$K_{oc} = K_{pr} * H_{koc}, \quad (8.29)$$

где H_{koc} - норматив затрат пользователя на освоение ПС, равный 0,01.

$$K_{oc} = 2\ 746,25 * 0,01 = 27,46 \text{ руб.}$$

$$K_{ob} = K_{pr} * H_{kob}, \quad (8.30)$$

где H_{kob} - норматив затрат на пополнение оборотных средств в связи с использованием нового ПС, равный 0,01.

$$K_{ob} = 2\ 746,25 * 0,01 = 27,46 \text{ руб.}$$

8.2.4 Расчет экономии основных видов ресурсов в связи с использованием нового программного средства

Экономия затрат на заработную плату при использовании нового ПС в расчете на объем выполненных работ:

$$C_3 = C_{ze} * A_2, \quad (8.31)$$

где C_{ze} – экономия затрат на заработную плату при решении задач с использованием нового ПС в расчете на 100 КБ, руб.;

A_2 – объем выполненных работ с использованием нового ПС (100 КБ).

$$C_3 = 0,0678 * 20\ 025 = 1\ 357,17 \text{ руб.}$$

Экономия затрат на заработную плату в расчете на 100 КБ (C_{ze}):

$$C_{3e} = \frac{Z_{cm} \cdot (T_{c1} - T_{c2}) \cdot T_q}{D_p}, \quad (8.32)$$

где Z_{cm} – среднемесячная заработка одного программиста, руб.;
 T_{c1}, T_{c2} – снижение трудоемкости работ в расчете на 100 строк кода (человеко-часов);
 T_q – количество часов работы в день (ч);
 D_p – среднемесячное количество рабочих дней.

$$C_{3e} = \frac{90,27 * (1,8 - 1,7)}{133} = 0,0678 \text{ руб.}$$

Объем выполненных работ с использованием нового ПС (100 КБ):

$$A_2 = A_o \cdot Z_{t2}, \quad (8.33)$$

где A_o – объем работ необходимый для решения одной задачи (100 КБ);
 Z_{t2} – количество типовых задач, решаемых за год (задач).

$$A_2 = 2\ 225 * 9 = 20\ 025.$$

Экономия затрат на оплату машинного времени (C_m) в расчете на выполненный объем работ в результате применения нового ПС:

$$C_m = C_{me} \cdot A_2, \quad (8.34)$$

где C_{me} – экономия затрат на оплату машинного времени при решении задач с использованием нового ПС в расчете на 100 КБ.

$$C_m = 0,18 * 20\ 025 = 3\ 604,5 \text{ руб.}$$

Экономия затрат на оплату машинного времени в расчете на 100 КБ (C_{me}):

$$C_{me} = \Pi_m \cdot (M_{b1} - M_{b2}), \quad (8.35)$$

где Π_m – цена одного машино-часа работы ЭВМ;
 M_{b1}, M_{b2} – средний расход машинного времени в расчете на 100 КБ при применении соответственно базового и нового ПС.

$$C_{Me} = 2 * (9 - 8,91) = 0,18 \text{ руб.}$$

Экономия затрат на материалы (C_{MT}) при использовании нового ПС в расчете на объем выполненных работ:

$$C_{MT} = C_{Mte} \cdot A_2, \quad (8.36)$$

где C_{Mte} – экономия затрат на материалы в расчете на 100 КБ.

$$C_{MT} = 0,004 * 20\,025 = 80,2 \text{ руб.}$$

$$C_{Mte} = C_{m1} - C_{m2}, \quad (8.37)$$

где C_{m1} , C_{m2} – средний расход материалов у пользователя в расчете на 100 КБ при использовании соответственно базового и нового ПС, руб.

$$C_{Mte} = 0,032 - 0,028 = 0,004 \text{ руб.}$$

Общая годовая экономия текущих затрат, связанных с использованием нового ПС (C_o):

$$C_o = C_3 + C_{o3} + C_m + C_{MT}. \quad (8.38)$$

$$C_o = 1\,359,17 + 3\,604,5 + 80,1 = 5\,043,77 \text{ руб.}$$

8.2.5 Расчет экономического эффекта

Внедрение нового ПС позволит пользователю сэкономить на текущих затратах, т.е. практически получить на эту сумму дополнительную прибыль. Для пользователя в качестве экономического эффекта выступает лишь чистая прибыль – дополнительная прибыль, остающаяся в его распоряжении ($\Delta\Pi_q$), которые определяются по формуле:

$$\Delta\Pi_q = C_o - \frac{C_o \cdot H_n}{100}, \quad (8.39)$$

где H_n – ставка налога на прибыль (%).

$$\Delta\Pi_q = 5\,043,77 - \frac{5\,043,77 * 18}{100} = 4\,135,89 \text{ руб.}$$

В процессе использования нового ПС чистая прибыль в конечном итоге возмещает капитальные затраты. Однако, полученные при этом суммы результатов (прибыли) и затрат (капитальных вложений) по годам приводят к единому времени – расчетному году путем умножения результатов и затрат за каждый год на коэффициент привидения ($ALFA_t$), который рассчитывается по формуле:

$$ALFA_t = (1 + E_h)^{t_p - t}, \quad (8.40)$$

где E_h – норматив привидения разновременных затрат и результатов;
 t_p – расчетный год, $t_p=1$;
 t – номер года, результаты и затраты которого приводятся к расчетному (2019-1, 2020-2, 2021-3, 2022-4).

Норматив приведения разновременных затрат и результатов (E_h) для программных средств ВТ в существующей практике принимается в пределах 0,2–0,4. Например, при нормативе 0,4 коэффициентам приведения ($ALFA_t$) по годам будут соответствовать следующие значения:

$$\begin{aligned} ALFA_1 &= (1 + 0,4)^{1-1} = 1 - \text{расчетный год;} \\ ALFA_2 &= (1 + 0,4)^{1-2} = 0,714 - 2020 \text{ год;} \\ ALFA_3 &= (1 + 0,4)^{1-3} = 0,510 - 2021 \text{ год;} \\ ALFA_4 &= (1 + 0,4)^{1-4} = 0,364 - 2022 \text{ год.} \end{aligned}$$

Результаты расчета экономического эффекта сведены в таблицу 8.4.

Таблица 8.4 – Данные экономического эффекта от использования нового ПС

Показатели	Ед. измерения	2019	2020	2021	2022
1	2	3	4	5	6
Результаты:					
Прирост прибыли за счет экономии затрат (Π_{η})	руб.	4 135,89	4 135,89	4 135,89	4 135,89
То же с учетом фактора времени	руб.	4 135,89	2 953,03	2 109,3	1 505,47

Продолжение таблицы 8.4

1	2	3	4	5	6
Затраты:					
Приобретение, адаптация и освоение ПС ($K_{\text{пр}}$)	руб.	2 746,25	-	-	-
Освоение ПС ($K_{\text{ос}}$)	руб.	27,46	-	-	-
Доукомплектование ВТ техническими средствами (K_{tc})	руб.	-	-	-	-
Пополнение оборотных средств ($K_{\text{об}}$)	руб.	27,46	27,46	27,46	27,46
Всего затрат	руб.	2 801,17	27,46	27,46	27,46
То же с учетом фактора времени	руб.	2 801,17	19,61	14,01	10
Экономический эффект:					
Превышение результата над затратами	руб.	1 334,72	2 933,42	2 095,4	1 495,47
То же с нарастающим итогом	руб.	1 334,72	4 268,14	6 363,44	7 858,92
Коэффициент приведения	единиц	1	0,714	0,510	0,364

Таким образом, отпускная цена предлагаемой разработки составляет 2 750 рублей, окупаемость ее будет достигнута уже на первом году применения. При этом эффект от использования в течение четырех лет составит 7 859 рублей.

9 ОХРАНА ТРУДА

9.1 Производственная санитария, техника безопасности и пожарная профилактика

ПЭВМ применяются как средства массовой автоматизации (в основном для создания на их основе автоматизированных рабочих мест) в социальной и производственных сферах деятельности в различных областях народного хозяйства и предназначенные для пользователей, не обладающих специальными знаниями в области вычислительной техники и программирования.

Работающие с ПЭВМ могут подвергаться воздействию различных опасных и вредных производственных факторов, основными из которых являются повышенные уровни: электромагнитного, рентгеновского, ультрафиолетового и инфракрасного излучения; статического электричества; запыленности воздуха рабочей зоны; повышенное или пониженное содержание аэроионов в воздухе рабочей зоны; повышенный или пониженный уровень освещенности рабочей зоны, содержание в воздухе рабочей зоны оксида углерода, озона, аммиака, фенола, формальдегида и полихлорированных фенилов; напряжение зрения, памяти, внимания; длительное статическое напряжение; большой объем информации, обрабатываемой в единицу времени; монотонность труда; нерациональная организация рабочего места; эмоциональные перегрузки [12].

Работа с ПЭВМ проводится в соответствии с Санитарными нормами и правилами «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» и Гигиеническим нормативом «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», утвержденными постановлением Министерства здравоохранения от 28.06.2013 г. № 59 и Типовой инструкцией по охране труда при работе с персональными ЭВМ, утвержденной постановлением Министерства труда и социальной защиты от 24.12.2013 № 130.

Согласно вышеуказанных документов площадь помещения на одного пользователя ПЭВМ на базе плоских дискретных экранов (жидкокристаллические, плазменные) составляет не менее $4,5 \text{ м}^2$.

9.1.1 Метеоусловия

В производственных помещениях, в которых работа с использованием ПЭВМ является основной (операторские, расчетные, посты управления, залы вычислительной техники), обеспечиваются оптимальные параметры микроклимата для категорий работ 1а и 1б (табл. 9.1) согласно вышеуказанным нормативным документам.

Таблица 9.1 – Оптимальные параметры микроклимата для помещений с ВДТ, ЭВМ и ПЭВМ

Период года	Категория работ	Температура воздуха, °C, не более	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	легкая-1а	22-24	40-60	0,1
	легкая-1б	21-23	40-60	0,1
Теплый	легкая-1а	23-25	40-60	0,1
	легкая-1б	22-24	40-60	0,2

Работа с компьютером относится к категории 1а (работы, производимые сидя и сопровождающиеся незначительным физическим напряжением, при которых расход энергии составляет до 120 ккал/ч, т.е. до 139 Вт).

Интенсивность теплового излучения работающих от нагретых поверхностей технологического оборудования, осветительных приборов, инсоляции на постоянных рабочих местах не превышает значений, указанных в табл. 9.2 согласно Санитарных норм и правил.

Таблица 9.2 – Предельно допустимые уровни интенсивности излучения в инфракрасном и видимом диапазоне излучения на расстоянии 0,5 м со стороны экрана ВДТ, ЭВМ и ПЭВМ

Диапазоны длин волн	400-760 нм	760-1050 нм	свыше 1050 нм
Предельно допустимые уровни	0,1 Вт/м ²	0,05 Вт/м ²	4,0 Вт/м ²

Для создания нормальных метеорологических условий наиболее целесообразно уменьшить тепловыделения от самого источника – монитора, что предусматривается при разработке его конструкции.

В производственных помещениях для обеспечения необходимых показателей микроклимата предусмотрены системы отопления, вентиляции и кондиционирования воздуха.

9.1.2 Вентиляция и отопление

Воздух рабочей зоны помещения соответствует санитарно-гигиеническим требованиям по содержанию вредных веществ и частиц пыли, приведенным в Санитарных нормах и правилах «Требованию к контролю воздуха рабочей зоны», Гигиеническом нормативе «Предельно допустимые концентрации вредных веществ в воздухе рабочей зоны», утвержденных постановлением Министерства здравоохранения от 10.10.2017 г. № 92.

В помещениях, проводится ежедневная влажная уборка и систематическое

проветривание после каждого часа работы.

Уровни положительных и отрицательных аэроионов, а также коэффициент униполярности в воздухе всех помещений, где расположены ПЭВМ, соответствуют значениям, указанным в табл. 9.3.

Таблица 9.3 – Уровни ионизации и коэффициент униполярности воздуха помещений при работе с ВДТ, ЭВМ и ПЭВМ

Уровни	Число ионов в 1 см ³ воздуха		Коэффициент униполярности (У)
	n+	n-	
Минимально допустимые	400	600	$0,4 \leq U < 1,0$
Оптимальные	1500-3000	3000-5000	
Максимально допустимые	50000	50000	

Одним из мероприятий по оздоровлению воздушной среды является устройство вентиляции и отопления. Задачей вентиляции является обеспечение чистоты воздуха и параметров метеорологических условий на рабочих местах. Чистота воздушной среды достигается удалением загрязненного или нагретого воздуха из помещения и подачей в него свежего воздуха. Для поддержания нормального микроклимата необходим достаточный объем вентиляции, для чего в вычислительном центре предусматривается кондиционирование воздуха, осуществляющее поддержание постоянных параметров микроклимата в помещении независимо от наружных условий [13].

Параметры микроклимата поддерживаются в холодный период года за счет системы водяного отопления с нагревом воды до 100°C, а в теплый - за счет кондиционирования, с параметрами отвечающими требованиям СНБ 4.02.01-03.

9.1.3 Освещение

В помещении для эксплуатации ПЭВМ предусмотрены естественное и искусственное освещение. Естественное освещение на рабочих местах осуществляется через световые проемы, ориентированные преимущественно на север, северо-восток, восток, запад или северо-запад и обеспечивает коэффициент естественной освещенности не ниже 1,5 %. Оконные проемы оборудованы регулируемыми устройствами типа жалюзи, занавесей.

Для внутренней отделки интерьера помещений используются материалы с коэффициентом отражения для потолка – 0,7- 0,8; для стен – 0,5- 0,6; для пола – 0,3- 0,5.

Искусственное освещение в помещениях осуществляется системой общего равномерного освещения. При работе с документами применяется система комбинированного освещения, а освещенность поверхности стола в зоне размещения рабочего документа должна составлять 300-500 люкс. Освещенность поверхности

экрана не более 300 люкс. В качестве источников света применяем люминесцентные лампы типа ЛБ. Коэффициент запаса для осветительных установок общего освещения принимается равным 1,4, а коэффициент пульсации – не более 5 %.

9.1.4 Шум

Основными источниками шума в помещениях, оборудованных ЭВМ, являются принтеры, множительная техника и оборудование для кондиционирования воздуха, в самих ЭВМ – вентиляторы систем охлаждения и трансформаторы.

В табл. 9.4 приведены нормированные уровни шума согласно Санитарных норм и правил «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» и Гигиенических нормативов «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», которые обеспечиваются за счет использования малошумного оборудования, применения звукопоглощающих материалов для облицовки помещений, а также различных звукопоглощающих устройств (перегородки и т. п.).

Таблица 9.4 – Предельно-допустимые уровни звука, эквивалентные уровни звука и уровни звукового давления в октавных полосах частот при работе с ВДТ, ЭВМ и ПЭВМ и периферийными устройствами

Категория нормы шума	Уровни звукового давления, дБ, в октавных полосах со среднегеометрическими частотами, Гц								Уровни звука и эквивалентные уровни звука, дБА	
	31.5	63	125	250	500	1000	2000	4000		
I	86	71	61	54	49	45	42	40	38	50
II	93	79	70	63	58	55	52	50	49	60
III	96	83	74	68	63	60	57	55	54	65
IV	103	91	83	77	73	70	68	66	64	75

9.1.5 Электробезопасность

Помещение вычислительного центра по степени опасности поражения электрическим током относится к помещениям без повышенной опасности.

Основные меры защиты от поражения током:

- изоляция и недоступность токоведущих частей;
- защитное заземление ($R_3 = 4 \text{ Ом}$ ГОСТ 12.1.030 - 81).

Первая помощь при поражениях электрическим током состоит из двух этапов: освобождение пострадавшего от действия тока и оказание ему доврачебной медицинской помощи. После освобождения пострадавшего от действия

электрического тока необходимо оценить его состояние. Во всех случаях поражения электрическим током необходимо вызвать врача независимо от состояния пострадавшего.

9.1.6 Излучение

При работе с дисплеем могут возникать следующие опасные факторы: электромагнитные и электростатические поля, ультрафиолетовое и инфракрасное излучение [14].

Уровни физических факторов на рабочих местах пользователей, создаваемые ПЭВМ и периферийными устройствами, не превышают предельно-допустимые уровни: электромагнитных и электростатических полей (табл. 9.5, 9.6), ультрафиолетового (табл. 9.7), установленных Гигиеническим нормативом «Предельно допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами».

Таблица 9.5 – Предельно допустимые уровни электромагнитных полей от экранов ВДТ, ЭВМ и ПЭВМ

Наименование параметра	Предельно-допустимые уровни
Напряженность электрического поля в диапазоне частот: 5 Гц-2 кГц 2-400 кГц	не более 25,0 В/м не более 2,5 В/м
Плотность магнитного потока магнитного поля в диапазоне частот: 5 Гц-2 кГц 2-400 кГц	не более 250 нТл не более 25 нТл
Напряженность электростатического поля	не более 15 кВ/м

Таблица 9.6 – Предельно допустимые уровни электромагнитных полей при работе с ВДТ, ЭВМ, ПЭВМ от клавиатуры, системного блока, манипулятора «мышь», беспроводных системам передачи информации и иных периферийных устройств

Диапазоны частот	0,3-300 кГц	0,3-3 МГц	3-30 МГц	30-300 МГц	0,3-300 ГГц
Предельно допустимые уровни	25 В/м	15 В/м	10 В/м	3 В/м	10 мкВт/см ²

Таблица 9.7. Предельно допустимые уровни интенсивности излучения в УФ диапазоне на расстоянии 0,5 м со стороны экрана ВДТ, ЭВМ и ПЭВМ

Диапазоны длин волн	200-280 нм	280-315 нм	315-400 нм
Предельно допустимые уровни	не допускается	0,0001 Вт/м ²	0,1 Вт/м ²

Наиболее эффективным и часто применяемым методом защиты от

электромагнитных излучений является установка экранов. Экранируют либо источник излучения, либо рабочее место. Часто экран устанавливают непосредственно на монитор.

При работе монитора на экране кинескопа накапливается электростатический заряд, создающий электростатическое поле. При этом персонал, работающий с монитором, приобретают электростатический потенциал. Заметный вклад в общее электростатическое поле вносят электризующиеся от трения поверхности клавиатуры и мыши.

9.1.7 Пожарная безопасность

По взрывопожарной и пожарной опасности помещения и здания для ЭВМ относятся к категории Д согласно ТКП 474-2013. Здания для ВЦ и части зданий другого назначения, в которых предусмотрено размещение ЭВМ, относятся к 2 степени огнестойкости согласно ТКП 45-2.02-315-2018.

Для предотвращения распространения огня во время пожара с одной части здания на другую устраивают противопожарные преграды в виде стен, перегородок, дверей, окон. Особое требование предъявляется к устройству и размещению кабельных коммуникаций.

Нормы первичных средств пожаротушения для вычислительных центров приведены в табл. 9.8.

Таблица 9.8. Примерные нормы первичных средств пожаротушения для вычислительного центра

Помещение	Площадь, м ²	Углекислотные огнетушители ручные	Порошковые огнетушители
Вычислительный центр	100	1	1

Для ликвидации пожаров в начальной стадии применяются первичные средства пожаротушения: внутренние пожарные водопроводы, огнетушители типа ОВП-10, ОУ-2, асbestosовые одеяла и др.

Эвакуация персонала вычислительного центра осуществляется через эвакуационные выходы. Количество и общая ширина эвакуационных выходов определяются в зависимости от максимального возможного числа эвакуирующихся через них людей и предельно допустимого расстояния от наиболее удаленного места возможного пребывания людей до ближайшего эвакуационного выхода согласно ТКП 45-2.02-315-2018.

Расчетное время эвакуации устанавливается по реальному расчету времени движения одного или нескольких потоков людей через эвакуационные выходы из наиболее удаленных мест размещения людей. Необходимое время эвакуации устанавливается на основе данных о критической продолжительности пожара с учетом степени огнестойкости здания, категории производства по взрывной и пожарной опасности. Для успешной эвакуации необходимо, чтобы расчетное время было меньше необходимого.

9.2 Требования к организации медицинского обслуживания пользователей ВДТ, ЭВМ и ПЭВМ

Лица, работающие с ПЭВМ более 50 % рабочего времени (профессионально связанные с эксплуатацией ПЭВМ), проходят обязательные медицинские осмотры. К непосредственной работе с ПЭВМ допускаются лица, не имеющие медицинских противопоказаний.

Женщинам со времени установления беременности и в период кормления ребенка грудью необходимо ограничить время работы с ЭВМ и ПЭВМ до 3 часов за рабочий день (смену) с учетом:

- обязательной организации оптимальных условий труда по тяжести и напряженности;
- обязательной организации оптимальных параметров микроклимата и ионизации воздуха помещений;
- обязательного соблюдения предельно-допустимых уровней параметров физических факторов, создаваемых на рабочем месте при работе с ЭВМ и ПЭВМ;
- регламентированных перерывов.

При невозможности организации работ в соответствии с указанными требованиями по причинам, связанным с особенностями технологического процесса, женщины со времени установления беременности и в период кормления ребенка грудью переводятся на работы, не связанные с использованием ЭВМ и ПЭВМ.

ЗАКЛЮЧЕНИЕ

В процессе выполнения дипломного проекта был создан мобильный клиент системы управления программными проектами ZIRO на базе мобильной ОС Apple iOS, предоставляющий средства для взаимодействия с серверной частью системы.

В процессе разработки и проектирования мобильного приложения учитывались достоинства и недостатки существующих аналогов, а также проанализированы и подобраны подходящие и новейшие технологии и инструменты, позволяющие решить поставленную задачу с наилучшими показателями безопасности, производительности и удобства использования.

Разработанный мобильный клиент успешно интегрировался с результатами разработки серверной части системы. Основной функционал доступный при работе в мобильном приложении:

- авторизация пользователя в системе;
- просмотр профиля;
- просмотр детальной информации о задаче;
- формирование списка задач с помощью различных средств фильтрации и сортировки;
- обсуждение задач;
- контроль потраченного времени и ведение журнала работ;
- просмотр детальной информации о проекте;
- формирование списка проектов с помощью различных средства фильтрации и сортировки.

В качестве архитектуры мобильного приложения был выбран шаблон проектирования MVC («Модель-Представление-Контроллер»), а точнее его реализация для среды iOS – Cocoa MVC, которая имеет хорошую масштабируемость, высокую скорость разработки и низкий порог вхождения, что в свою очередь обеспечивает низкую стоимость как обслуживания данного приложения, так и внедрение новой функциональности.

Мобильное приложение разработано с использованием основных принципов программирования (ООП, SOLID и др.), а также гарантирует высокий уровень безопасности и производительности.

Также стоит упомянуть, что интерфейс программного продукта разработан с высоким уровнем удобства, простоты и эргономичности.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- 1 Управление разработкой программного обеспечения [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Software_project_management – Дата доступа 21.02.2019
- 2 Мобильное приложение [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Мобильное_приложение – Дата доступа 21.02.2019
- 3 Мобильный сайт или мобильное приложение: стратегия выбора [Электронный ресурс]. – Режим доступа: http://rasolution.ru/ru/articles/mobile_business_applications/ – Дата доступа 21.02.2019
- 4 Trello [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Trello> – Дата доступа 22.02.2019
- 5 Jira [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Jira> – Дата доступа 22.02.2019
- 6 API [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/API> – Дата доступа 05.04.2019
- 7 Xcode [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Xcode> – Дата доступа 29.04.2019
- 8 Мэтьюс Холлеманс, «iOS Apprentice: Beginning iOS development with Swift 4.2», 7-е изд., 2018. – 793 с.
- 9 Нэйл Смит, «iOS 12 App Development Essentials», 1-е изд., 2018. – 1509 с.
- 10 Архитектурные паттерны в iOS [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/badoo/blog/281162/> – Дата доступа 10.05.2019
- 11 Model-View-Controller [Электронный ресурс]. – Режим доступа: <https://developer.apple.com/library/archive/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html> – Дата доступа 12.05.2019
- 12 Санитарные нормы и правила «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» и Гигиенический норматив «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», утвержденные постановлением МЗ РБ от 28.06.2013 г. № 59.
- 13 Лазаренков, А. М. Охрана труда в машиностроении: учебное пособие / А. М. Лазаренков. – Минск: ИВЦ Минфина, 2017. – 446 с.
- 14 Лазаренков А.М., Ушакова И.Н. Охрана труда: Учебно-методическое пособие для практических занятий. – Минск: БНТУ, 2011. – 205 с.