Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

**Отчёт по преддипломной практике**

Выполнил студент гр. 753503 Попов В. А.

Руководитель практики от предприятия: начальник отдела Калиновский А. А.

Руководитель практики от университета: доцент кафедры информатики Удовин И. А.

Минск 2021

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc68459137)

[1. Обзор предметной области. Постановка задачи. 4](#_Toc68459138)

[1.1. Менеджер финансов 4](#_Toc68459139)

[1.2. RSS-Feed 4](#_Toc68459140)

[1.3. Облачное хранение данных 4](#_Toc68459141)

[1.4. Обзор существующих аналогов 4](#_Toc68459142)

[1.5. Постановка задачи 4](#_Toc68459143)

[1.6. Перспективы развития программного средства 4](#_Toc68459144)

[2. Используемые технологии 5](#_Toc68459145)

[2.1. Язык разметки HTML 5](#_Toc68459146)

[2.2. Каскадные таблицы стилей CSS 7](#_Toc68459147)

[2.3. Язык программирования JavaScript 9](#_Toc68459148)

[2.4. Программная платформа Node.js 10](#_Toc68459149)

[2.5. Облачная база данных Firebase 12](#_Toc68459150)

[Технико-экономическое обоснование 13](#_Toc68459151)

[Описание функций, назначения и потенциальных пользователей ПО 13](#_Toc68459152)

[Расчёт затрат на разработку ПО 14](#_Toc68459153)

[Оценка результата (эффекта) от использования (или продажи) ПО 16](#_Toc68459154)

[Расчёт показателей эффективности инвестиций в разработку ПО 18](#_Toc68459155)

[Выводы по технико-экономическому обоснованию 19](#_Toc68459156)

[3. Индивидуальное задание 20](#_Toc68459157)

[4. Этапы выполнения индивидуального задания 21](#_Toc68459158)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 22](#_Toc68459159)

# Введение

1. Обзор предметной области. Постановка задачи.
   1. Менеджер финансов
   2. RSS-Feed
   3. Облачное хранение данных
   4. Обзор существующих аналогов
   5. Постановка задачи
   6. Перспективы развития программного средства
2. Используемые технологии
   1. Язык разметки HTML

HTML (Hypertext Markup Language) - это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных.

*История HTML*

HTML был изобретён [**Тимом Бернерсом-Ли**](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D1%80%D0%BD%D0%B5%D1%80%D1%81-%D0%9B%D0%B8,_%D0%A2%D0%B8%D0%BC), физиком из исследовательского института ЦЕРН в Швейцарии. Он придумал идею интернет-гипертекстовой системы.

Hypertext означает текст, содержащий ссылки на другие тексты, которые зрители могут получить немедленно. Он опубликовал первую версию HTML в 1991 году, состоящую из 18 тегов HTML. С тех пор каждая новая версия языка HTML появилась с разметкой новых тегов и атрибутов (модификаторов тегов).

Согласно Справочнику [**HTML Element Reference**](https://developer.mozilla.org/ru/docs/Web/HTML/Element) от Mozilla Developer Network, в настоящее время существует 140 тегов HTML, хотя некоторые из них уже устарели (не поддерживаются современными браузерами).

Самым большим обновлением языка стало внедрение **HTML5** в 2014 году. Было добавлено несколько новых семантических тегов к разметке, которые показывают смысл их собственного контента, например **<article>**, **<header>** и **<footer>**.

*Как работает HTML?*

HTML-документы — это файлы, которые заканчиваются расширением **.html** или **.htm**. Вы можете просматривать его с помощью любого веб-браузера (например, Google Chrome, Safari или Mozilla Firefox). Браузер читает HTML-файл и отображает его содержимое, чтобы пользователи интернета могли его просматривать.

Каждая HTML-страница состоит из набора **тегов** (также называемых **элементами**), которые вы можете назвать строительными блоками веб-страниц. Они создают иерархию, которая структурирует контент по разделам, параграфам, заголовкам и другим блокам контента.

Большинство элементов HTML имеют открытие и закрытие, в которых используется синтаксис **<tag> </tag>**.

Рассмотрим элемент абзаца более подробно.



Главными частями нашего элемента являются:

1. **Открывающий тег (Opening tag)**: Состоит из имени элемента (в данном случае, "p"), заключённого в открывающие и закрывающие **угловые скобки**.  Открывающий тег указывает, где элемент начинается или начинает действовать, в данном случае — где начинается абзац.
2. **Закрывающий тег (Closing tag):** Это то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента. Закрывающий элемент указывает, где элемент заканчивается, в данном случае — где заканчивается абзац. Отсутствие закрывающего тега является одной из наиболее распространённых ошибок начинающих и может приводить к странным результатам.
3. **Контент (Content)**: Это контент элемента, который в данном случае является просто текстом.
4. **Элемент(Element)**: Открывающий тег, закрывающий тег и контент вместе составляют элемент.

Элементы также могут иметь атрибуты, которые выглядят так:



Атрибуты содержат дополнительную информацию об элементе, которую вы не хотите показывать в фактическом контенте. В данном случае, class это *имя* *атрибута,* а editor-note это *значение атрибута*. Класс позволяет дать элементу идентификационное имя, которое может позже использоваться, чтобы обращаться к элементу с информацией о стиле и прочих вещах.

Атрибут всегда должен иметь:

1. Пробел между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов).
2. Имя атрибута, за которым следует знак равенства.
3. Значение атрибута, заключённое с двух сторон в кавычки.
   1. Каскадные таблицы стилей CSS

CSS – это язык, с помощью которого описывается внешний вид документа HTML, XML, XHTML. Название означает «каскадная таблица стилей», или Cascading Style Sheets. CSS-стили незаменимы при оформлении страниц сайтов: в одном файле содержатся сведения об отображении всех элементов документа.

*Для чего используется CSS*

По сути, таблица стилей – это файл, где описывается, как будет выглядеть каждый из элементов на странице. В HTML-документе, таким образом, остается только структура странички: сами блоки, их содержимое и расположение. Создать страницу и оформить ее можно и без использования таблиц, прописывая визуальные свойства каждого элемента в его описании. Но, если страниц сотни и тысячи, применять такой метод неудобно: при изменении оформления приходится менять множество документов, вдобавок это загромождает верстку. Поэтому использование CSS считается золотым стандартом оформления сайтов: так получилось благодаря гибкости и многообразию возможностей каскадных таблиц.

*Преимущества CSS*

* Это существенно упрощает верстку и снижает временные затраты. Один созданный файл стилей можно распространить на множество страниц, так что внешний вид элементов достаточно описать один раз.
* Если что-то нужно изменить, достаточно внести правки в один файл. Это касается и изменений в оформлении, и найденных ошибок.
* Применение CSS серьезно облегчает структуру документа, что хорошо и для пользователей, и для поисковых программ.
* Вариативность оформления становится шире. CSS поддерживает намного больше возможностей, чем имеется при использовании чистого HTML, вдобавок к одной странице можно применить несколько стилей в зависимости от обстоятельств (размер монитора пользователя, устройство, с которого выполнен вход, – ПК или мобильное).
* Страницы начинают загружаться быстрее: браузер кеширует таблицу стилей при первом посещении сайта, при последующих подгружаются только данные, что намного быстрее.

*Как развивалась технология*

Начало было положено в 1990-х, когда консорциум W3C решил, что технология, позволяющая разделять содержание и представление документов, необходима. Стандарт CSS1 появился в 1996 году и позволял изменять с помощью таблиц параметры шрифтов, цвета элементов, свойства блоков и текстов, такие как отступы и выравнивание. Длина и ширина блоков задавались там же. С развитием интернета появились новые уровни:

* CSS2. Стандарт расширил технические возможности, дал возможность работать с аудио и страничными носителями (например, при печати документов), включил в себя поддержку блочной структуры и генерируемого содержимого;
* CSS3. Еще более масштабное расширение, находится в разработке до сих пор, поддерживает сглаживание, градиенты, тени и анимацию, для этого не приходится использовать JavaScript;
* CSS4. Находится в разработке, новые модули пока доступны как черновики. Дополняют предыдущие версии новыми расширениями и значениями.

*Синтаксис и структура*

Файл CSS сводится к набору правил, описанных по определенному синтаксису. Правило состоит из селекторной части и блока объявлений: ими описываются всевозможные элементы страницы. Формат примерно таков: селектор { параметр: значение }.

**Селекторы**. Указывают, к каким элементам будут применяться те или иные параметры стиля. Пишутся в начале строки, по сути, являются названиями тегов, для которых справедливо правило.

* Можно использовать любой тег, написанный латиницей.
* Если вариантов стиля для одного типа элементов несколько, используются так называемые классы. У одного тега их может быть несколько (применяются все стили, что описаны в таблице). Запись в этом случае выглядит так: тег.Класс { параметр: значение }.
* Есть возможность видоизменить только один конкретный элемент. Это делается с помощью идентификаторов – уникальных имен, которые можно присвоить элементам. Идентификатор будет использоваться как селектор.
  1. Язык программирования JavaScript

[JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript)  — это полноценный [динамический язык программирования](https://developer.mozilla.org/en-US/docs/Glossary/Dynamic_programming_language), который применяется к [HTML](https://developer.mozilla.org/en-US/docs/Glossary/HTML) документу, и может обеспечить динамическую интерактивность на веб-сайтах сайт (например: игры, отклик при нажатии кнопок или при вводе данных в формы, динамические стили, анимация). Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

* Программные интерфейсы приложения ([API](https://developer.mozilla.org/en-US/docs/Glossary/API)), встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком, работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов.
* Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook.
* Также вы можете применить к вашему HTML сторонние фреймворки и библиотеки, что позволит вам ускорить создание сайтов и приложений.

Базовой особенностью этого языка отмечается то, что на него повлияли другие (Python, Java и др.) языки программирования с целью придания максимального комфорта JavaScript и лёгкости в понимании его теми пользователями, которые не имеют соответствующего образования и глубинных знаний – не программистами. JavaScript – официально зарегистрированная торговая марка компании Oracle.

С помощью него доступны к исполнению следующие функции:

* возможность изменять страницы браузеров;
* добавление или удаление тегов;
* изменение стилей страницы;
* информация о действиях пользователя на странице;
* запрос доступа к случайной части исходного кода страницы;
* внесение изменений в этот код;
* выполнение действия с cookie-файлами.

Область применения этого языка удивительно обширна и ничем не ограничена: среди программ, которые используют JS, присутствуют и тестовые редакторы, и приложения (как для компьютеров, так и мобильные и даже серверные), и прикладное ПО.

*Преимущества JavaScript*

* Ни один современный браузер не обходится без поддержки JavaScript.
* С использованием написанных на JavaScript плагинов и скриптов справится даже не специалист.
* Полезные функциональные настройки.
* Взаимодействие с приложением может осуществляется даже через текстовые редакторы – Microsoft Office и Open Office.
* Перспектива использования языка в процессе обучения программированию и информатике.

*Недостатки JavaScript*

* Пониженный уровень безопасности ввиду повсеместного и свободного доступа к исходным кодам популярных скриптов.
* Множество мелких раздражающих ошибок на каждом этапе работы. Большая часть из них легко исправляется, но их наличие позволяет считать этот язык менее профессиональным, сравнительно с другими.
* Повсеместное распространение. Своеобразным недостатком можно считать тот факт, что часть активно используемых программ (особенно приложений) перестанут существовать при отсутствии языка, поскольку целиком базируются на нем.
  1. Программная платформа Node.js

Node.js — среда выполнения кода JavaScript вне браузера. Эта платформа позволяет писать серверный код для динамических веб-страниц и веб-приложений, а также для программ командной строки. С помощью Node.js реализуется парадигма «JavaScript для всего». Она предполагает использование одного языка программирования для разработки веб-приложений вместо применения разных языков для работы над фронтендом и бэкендом.

Node.js — не отдельный язык программирования, а платформа для использования JavaScript на стороне сервера. Если говорить о языке, то как для фронденда, так и для бэкенда используется один и тот же JavaScript. Разница только в наборе API, которые используют фронтендеры и бэкендеры.

Браузерный JavaScript использует Web API, которые обеспечивают доступ к DOM и пользовательскому интерфейсу страниц и веб-приложений. Серверный JavaScript использует API, обеспечивающие доступ к файловой системе приложений, http-запросам, потокам.

То есть Node.js — это технология для использования JS на бэкенде. С особенностями и перспективами развития языка JavaScript можно ознакомиться в [соответствующей статье](https://ru.hexlet.io/blog/posts/stoit-li-uchit-javascript-perspektivy-situatsiya-na-rynke-truda-mneniya-ekspertov), а здесь речь идёт об одной из технологий данного языка.

*Экскурс в историю: с 2009 до 2019*

Платформа Node.js была представлена в 2009 году. Её создал инженер Райан Дал, а спонсором разработки выступила компания Joyent. Компания известна поддержкой опенсорсных проектов, включая Node.js, Illumos, SmartOS.

Райан Дал использовал для создания Node.js движок V8. Платформа реализована с низкоуровневой неблокирующей моделью ввода/вывода, которая построена на событийно-ориентированной модели.

*Сфера применения: где используют Node.js*

Как отмечалось выше, Node.js применяется для [бэкенд-разработки на JavaScript](https://ru.hexlet.io/professions/backend" \t "_blank). Если в браузере JavaScript тотально доминирует, и конкурентов этого языка на фронтенде не видно даже на горизонте, то в бэкенд-разработке ситуация другая. Здесь JS конкурирует с PHP, Python и другими языками.

Выбор серверного JavaScript для бэкенда обеспечивает проекту ряд преимуществ:

* рост эффективности разработки благодаря использованию одного языка для фронт- и бэкенда и возможности переиспользования кода;
* возможность использовать [npm](https://www.npmjs.com/" \t "_blank) — самый большой пакетный менеджер;
* более простой по сравнению с другими стеками поиск исполнителей, так как JavaScript входит в число самых популярных языков программирования.

Node.js хорошо подходит для разработки RTA — веб-приложений, реагирующих на действия пользователя в режиме реального времени. Например, это может быть онлайн-редактор типа Google Docs, который позволяет работать над одним документом нескольким пользователям одновременно.

**Node.js легко обрабатывает большое количество запросов одновременно и обеспечивает быстродействие приложения. Поэтому серверный JavaScript часто используют для создания SPA — одностраничных веб-приложений, в которых рендеринг выполняется на стороне клиента. Node.js на бэкенде используют Netflix, Uber, eBay, Groupon, Yahoo и другие известные организации и проекты.**

* 1. Облачная база данных Firebase

Firebase – это платформа разработки мобильных приложений с огромным функционалом. Начиналась она как стартап, а сегодня ее используют при разработке лучших кроссплатформенных приложений. Главное достоинство платформы в том, что она позволяет разработчику не отвлекаться на создание бэкенда, то есть скрытой от пользователя программной части проекта, например, серверного кода. И это упрощает и ускоряет создание мобильных приложений, дает возможность полностью сосредоточиться именно на UX/UI, то есть, на пользовательском интерфейсе и опыте.

Firebase – это одно из BaaS-решений (Backend as a Service), которое дает разработчику массу возможностей.

Это и сервер, и база данных, и хостинг, и аутентификация в одной платформе. Так, Firebase Realtime Database предоставляет разработчикам API, который синхронизирует данные приложения между клиентами и хранит их в облачном хранилище.

Приложение подключается к базе данных через WebSocket, который отвечает за синхронизацию данных в течение всего сеанса.

Также Firebase выступает в качестве хранилища файлов. Firebase Storage обеспечивает надежную загрузку и выгрузку файлов для приложения. Облачное хранение файлов видео, аудио или любого другого типа поддерживается Google Cloud Storage. Содержимое облачного хранилища надежно защищено собственной системой безопасности.

Создавать систему аутентификации каждый раз с нуля довольно затратно, причем затраты эти чаще всего не оправданы. Справится с большинством вызовов позволяет система аутентификации Firebase Auth, в которой возможна аутентификация пользователя приложения по паролю и электронной почте. Поддерживает Firebase Auth также открытый протокол авторизации OAuth 2.0, используемый Google, Twitter, Facebook. Система аутентификации Firebase интегрируется непосредственно в базу данных.

Статические файлы приложения размещаются на хостинге Firebase. Поддерживается хостинг файлов JavaScript, HTML, CSS и других. Через Cloud Functions реализована динамическая поддержка Node.js. Передача файлов осуществляется через сеть доставки контента с использованием защищенных протоколов SSL и HTTPS.

# Технико-экономическое обоснование

## Описание функций, назначения и потенциальных пользователей ПО

Программный модуль предназначен для отслеживания и анализа некоторых данных, описывающих микроклиматические условия помещений, таких как температура воздуха и относительная влажность.

Разрабатываемый продукт представляет собой мобильное приложение, позволяющее:

* Отображать визуализацию климатических изменений в процессе измерения.
* Выбирать наиболее подходящий для восприятия вид отображения данных.
* Хранить данные в удалённой и локальной базах данных.
* Получать рекомендации по улучшению микроклимата помещения.

Среди потенциальной аудитории программного продукта можно выделить как частных лиц, заинтересованных в мониторинге микроклимата помещений, в которых они пребывают, так и компании, выражающие интерес в анализе условий, в которых работают их сотрудники. Своевременное информирование позволит компании обеспечить надлежащие условия труда всем сотрудникам, что окажет положительный экономический эффект. Сотрудники компании, в свою очередь, смогут использовать систему для информирования менеджеров о неблагоприятных условиях труда, запрашивать сопутствующее оборудование для регуляции измеряемых параметров, а также компенсацию медицинских услуг и дополнительные выплаты за вредные и тяжелые условия труда.

Программный продукт может быть также использован в научных целях для исследования микроклиматических условий различных помещений и их влияние на здоровье и жизнедеятельность человека.

Разрабатываемое программное средство предназначено для использования широким кругом потребителей и имеет достаточный спрос для свободной реализации на рынке информационных технологий. Разработка выполняется по договору с негосударственной организаций здравоохранения. Финансирование разработки системы осуществляется за счёт упомянутой организации. Личные неимущественные права принадлежат разработчику программного обеспечения, имущественные (исключительные) права – заказчику (собственнику разработки).

Расчёты приведены по состоянию на май 2020 г.

Источник расчётных формул – методическое пособие [13].

## Расчёт затрат на разработку ПО

Расчет затрат на основную заработную плату команды разработчиков осуществляется исходя из состава и численности команды, размера месячной заработной платы каждого участника команды, а также трудоемкости работ, выполняемых при разработке программного средства отдельными исполнителями по формуле:

где – коэффициент премий (1,75);

*n* – категории исполнителей, занятых разработкой ПС;

– часовая заработная плата исполнителя *i*-й категории, *руб*.;

– трудоёмкость работ, выполняемых исполнителем *i*-й категории, определяется исходя из сложности разработки программного обеспечения и объема выполняемых им функций, *ч*.

Расчёт затрат на основную заработную плату осуществляется в табличной форме (табл. 6.1).

Таблица 6.1– Расчет затрат на основную заработную плату [14]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Участник команды | Месячная заработная плата, *руб.* | Часовая заработная плата, *руб.* | Трудоемкость работ, *ч.* | Зарплата по тарифу, *руб.* |
| 1 | Менеджер проекта | 5 544 | 33 | 252 | 8 316 |
| 2 | Программист (разработчик мобильного приложения) | 4 872 | 29 | 504 | 14 614 |
| 3 | Программист (разработчик базы данных) | 4 872 | 29 | 504 | 14 614 |
| 4 | Тестировщик | 2 856 | 17 | 120 | 2 040 |
| 5 | Дизайнер | 3 696 | 22 | 120 | 2 640 |
| Итого затраты на основную заработную плату | | | | | 42 224 |

Затраты на дополнительную заработную плату команды разработчиков включает выплаты, предусмотренные законодательством о труде (оплата трудовых отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по формуле:

где ‒ норматив дополнительной заработной платы, (15 %).

–затраты на основную заработную плату, руб.;

Затраты на дополнительную заработную плату составят:

= 6 333,6 *руб*.

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле:

где ‒ ставка отчислений в ФСЗН и Белгосстрах (34,6 %).

Затраты на социальные нужды составят:

(42 224 + 6 333,6) 0,346 = 16 800,93 *руб*.

Прочие затраты включаются в себестоимость разработки ПО в процентах от затрат на основную заработную плату команды разработчиков по формуле:

где ‒ норматив прочих расходов, (35 %).

Прочие затраты составят:

14 778,4 *руб*.

Расходы на реализацию рассчитываются по формуле:

где ‒ норматив расходов на реализацию, (5%).  
 Расходы на реализацию составят:

= 42 224 0,05 = 2 111,2 *руб*.

Полная информация о формировании затрат для производства программного средства приведена в таблице 6.2.

Таблица 6.2. – Формирование цены программного средства на основе затрат

|  |  |
| --- | --- |
| Статья затрат | Сумма, *руб*. |
| Основная заработная плата команды разработчиков, | 42 224 |
| Дополнительная заработная плата команды разработчиков, | 6 333,6 |
| Отчисления на социальные нужды, | 16 800,93 |
| Прочие затраты, | 14 778,4 |
| Расходы на реализацию, | 2 111,2 |
| Общая сумма затрат на разработку, *)* | 82 248,13 |

Таким образом, общая сумма затрат на разработку программного средства составит 82 248,13 *руб*.

## Оценка результата (эффекта) от использования (или продажи) ПО

Экономический эффект организации-разработчика программного средства представляет собой прирост чистой прибыли от его продажи на рынке потребителям, величина которого зависит от объема продаж, цены реализации и затрат на разработку программного средства.

В течение трех месяцев будут введены все основные функции и программное средство будет полностью готово к запуску. В первый год выхода на рынок ожидается более 10 тыс. пользователей. Был проведен социологический опрос среди потенциальных пользователей приложения (совершеннолетних работающих людей), в результате которого было выявлено, какую сумму они готовы ежемесячно тратить на данный продукт.

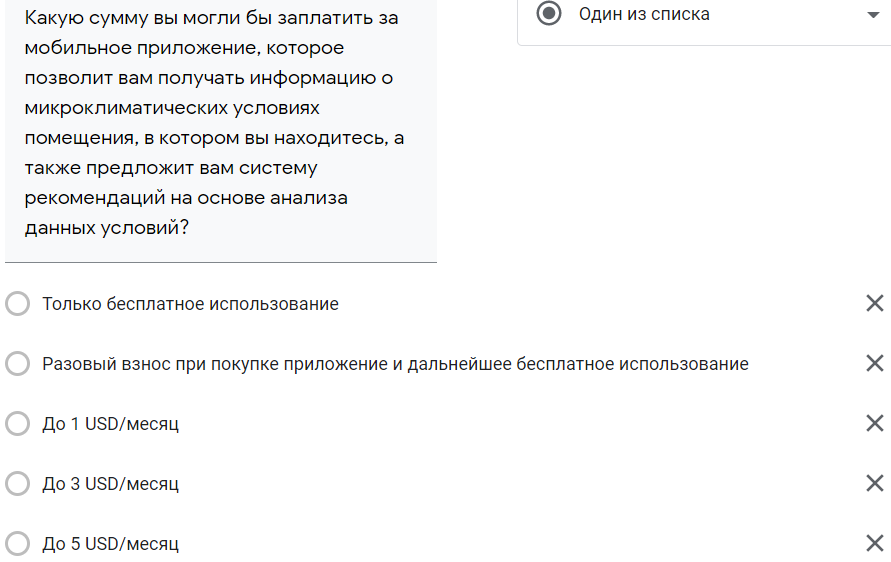


Рис. 6.1. Условие социологического опроса

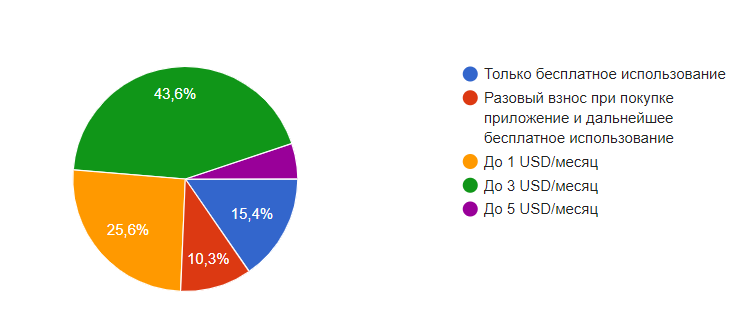


Рис. 6.2. Результаты социологического опроса

Более 43% респондентов указали сумму в 3$ (7 *руб*.) [15], из чего следует, что отпускная цена годовой подписки будет равна = 7 \* 11 = 77 *руб.* (11 месяцев подписки + бесплатный месяц пользования), при количестве годовых подписок равном штук.

Прирост чистой прибыли, полученной разработчиком от реализации программного средства на рынке, можно рассчитать по формуле:

где – отпускная цена подписки, *руб*.;

‒ количество подписок, реализуемое за год, шт.;

‒ сумма налога на добавленную стоимость, *руб*.;

‒ рентабельность продаж подписок, (40 %);

‒ ставка налога на прибыль, (18%).

Налог на добавленную стоимость определяется по формуле:

где – ставка налога на добавленную стоимость в соответствии с действующим законодательством (20 %).

Налог на добавленную стоимость равен:

*руб*.

В таком случае, можем рассчитать прирост чистой прибыли:

= 210 466,67 *руб*.

## Расчёт показателей эффективности инвестиций в разработку ПО

Оценка экономической эффективности разработки и реализации программного средства на рынке зависит от результата сравнения инвестиций в его разработку (модернизацию, совершенствование) и полученного годового прироста чистой прибыли.

Так как сумма инвестиций на разработку меньше суммы годового экономического эффекта, то есть инвестиции окупятся менее чем за год, оценка экономической эффективности инвестиций в разработку программного средства осуществляется с помощью расчета простой нормы прибыли (рентабельности инвестиций) по формуле:

где ‒ прирост чистой прибыли, *руб.;*

‒ затраты на разработку программного средства, *руб*.

Таким образом, рентабельность инвестиций будет равна:

Инвестиции на разработку программного средства и его реализация на рынке информационных технологий будут экономически эффективными, если рентабельность инвестиций превысит 100 % (100 % плюс ставка по банковским долгосрочным депозитам). А поскольку ставка по долгосрочным депозитам не превышает 15% [16], следовательно, программное средство целесообразно разрабатывать и реализовывать по установленной цене, т.к. рентабельность инвестиций превышает 115% более чем в 2 раза.

При проведении технико-экономического обоснования дипломного проекта рассчитывается срок окупаемости инвестиций без учета фактора времени по формуле:

где *n* – расчетный период, количество лет (1 год);

*Зt*– затраты (инвестиции) в году *t*, *руб*.;

*Рср* – среднегодовая сумма результата (экономического эффекта), р.

Без учета фактора времени при условии, что сумма ежегодной прибыли принимается как равноценная, а условия ведения хозяйства стабильные, мы можем говорить, что срок окупаемости будет равен:

## Выводы по технико-экономическому обоснованию

На основании данных результатов можно сделать вывод, что проект представляется выгодным как для разработчика, так и для инвестора: реализация программного средства на рынке экономически эффективна.

Спустя год после внедрения данного программного средства заказчик не только покрывает собственные затраты, но и имеет прибыль. В свою очередь исполнитель также получает прибыль в короткие сроки.

Важно отметить, что статьей дохода приложения также может стать рекламная интеграция с производителями оборудования для поддержания оптимальных климатических условий в помещении.

1. Индивидуальное задание
2. Этапы выполнения индивидуального задания

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

<https://codebra.ru/ru/lessons-html/dating/1/1>

<https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics>

<https://www.hostinger.ru/rukovodstva/shto-takoje-html/>

<https://promo.ingate.ru/seo-wikipedia/css/>

<https://wiki.rookee.ru/css/>

<https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics>

<https://ipipe.ru/info/javascript>

https://promo.ingate.ru/seo-wikipedia/java-script/