Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

|  |
| --- |
| *К защите допустить*: |
| Заведующая кафедрой информатики |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Волорова |

Пояснительная записка

к дипломному проекту

на тему

**ПРИЛОЖЕНИЕ ПЕРСОНАЛЬНЫЙ МЕНЕДЖЕР ФИНАНСОВ**

БГУИР ДП 1-40 04 01 054 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | В.А. Попов |
| Руководитель |  | И.А. Удовин |
| Консультанты: |  |  |
| *от кафедры информатики* |  | И.А. Удовин |
| *по экономической части* |  | Т.А. Рыковская |
|  |  |  |
| Нормоконтролер |  | Н.Н. Бабенко |
|  |  |  |
| Рецензент |  | И.П. Иванов |

Минск 2021

**Р Е Ф Е Р А Т**

Пояснительная записка 42 с., 9 рис., 2 табл., 14 источников

ПРИЛОЖЕНИЕ ПЕРСОНАЛЬНЫЙ МЕНЕДЖЕР ФИНАНСОВ

Министерство образования Республики Беларусь

+----------------

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учреждение образования БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| Факультет | КС и С | | | Кафедра | | | информатики | | | | | | | | | | |
| Специальность | 1-40 04 01 | | | Специализация | | | | | | 00 | | | | | | | |
| УТВЕРЖДАЮ | | | | | | | | | | | | | | | | | |
|  | | | | | | | | |  | | | | | Н.А.Волорова | | | |
| « | | | | | | | | |  | | » | |  | | | 20 | г. |
| ЗАДАНИЕ | | | | | | | | | | | | | | | | | |
| по дипломному проекту студента | | | | | | | | | | | | | | | | | |
| **Попова Вадима Андреевича** | | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | | | | | | | | | |
| 1. Тема проекта: | | Приложение персональный менеджер финансов | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| утверждена приказом по университету от | | | | « |  | » | |  | | | | 20 г. | | | № |  | |
| 2 . Срок сдачи студентом законченной работы | | | | | 01 июня 2021 года | | | | | | | | | | | | |
| 3. Исходные данные к проекту | | | Тип операционной системы – ОС Windows 10; | | | | | | | | | | | | | | |
| Язык программирования – JavaScript; Перечень выполняемых функций: | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| Назначение разработки: подсчёт личных доходов/расходов, создание заметок и напоминаний, просмотр ленты новостей | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
| 4. Содержание пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | | | | | | | | | |
| Введение | | | | | | | | | | | | | | | | | |
| 1 Обзор предметной области. Постановка задачи | | | | | | | | | | | | | | | | | |
| 2 Используемые технологии | | | | | | | | | | | | | | | | | |
| 3 Проектирование программного средства | | | | | | | | | | | | | | | | | |
| 4 Создание программного средства | | | | | | | | | | | | | | | | | |
| 5 Тестирование программного средства | | | | | | | | | | | | | | | | | |
| 6 Технико-экономическое обоснование | | | | | | | | | | | | | | | | | |
| Заключение | | | | | | | | | | | | | | | | | |
| Список использованных источников | | | | | | | | | | | | | | | | | |
| Приложение А Текст программы | | | | | | | | | | | | | | | | | |
| Приложение Б Иллюстрации работы программы | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. Перечень графического материала (с точным указанием наименования) и обозначения | | | |
| вида и типа материала) | |  | |
| Порядок разработки программных модулей. Схема данных – формат А4, лист 1. | | | |
| Программа обработки данных. Схема программы – формат А4, лист 1. | | | |
| Схема базы данных персонального менеджера. Схема алгоритма – формат А4, лист 1. | | | |
| Диаграмма классов программного средства. Плакат – формат А1, лист 1. | | | |
| Результаты проверки работоспособности. Плакат – формат А1, лист 1. | | | |
| Экранные формы программы. Плакат – формат А1, лист 1. | | | |
|  | | | |
|  | | | |
| 6. Содержание задания по технико–экономическому обоснованию | | | |
| Расчет экономической эффективности от внедрения программного средства | | | |
| Задание выдал |  | | / Т.А. Рыковская / |
|  | | | |
|  | | | |
|  | | | |
|  | | | |
| Задание выдал: |  | |  |

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объём этапа в % | Срок выполнения этапа | Примечание |
| Анализ предметной |  |  |  |
| области, разработка технического задания | 15 | 01.02–14.02 |  |
| Разработка функциональных требований, |  |  |  |
| проектирование архитектуры программы | 15 | 15.02–06.03 |  |
| Разработка схемы программы, алгоритмов, |  |  |  |
| схемы данных | 20 | 07.03–27.03 |  |
| Разработка программного средства | 20 | 28.03–01.05 |  |
| Тестирование и отладка | 10 | 02.05–08.05 |  |
| Оформление пояснительной записки |  |  |  |
| и графического материала | 20 | 09.05–31.05 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | 23 января 2021 г. | | Руководитель | |  | | /И.А. Удовин/ |
| Задание принял к исполнению | |  | | / В.А. Попов / | |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc71989530)

[1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ 7](#_Toc71989531)

[1.1 Менеджер финансов 7](#_Toc71989532)

[1.2 RSS-Feed 7](#_Toc71989533)

[1.3 Облачное хранение данных 9](#_Toc71989534)

[1.3.1 Файловое хранилище 11](#_Toc71989535)

[1.3.2 Блочное хранилище 11](#_Toc71989536)

[1.3.3 Объектное хранилище 12](#_Toc71989537)

[1.3.4 База данных 12](#_Toc71989538)

[1.4 Обзор существующих аналогов 13](#_Toc71989539)

[1.4.1 Monefy 13](#_Toc71989540)

[1.4.2 Coinkeeper 14](#_Toc71989541)

[1.4.3 Toshl 14](#_Toc71989542)

[1.4.4 Money Manager 15](#_Toc71989543)

[1.5 Постановка задачи 15](#_Toc71989544)

[1.6 Перспективы развития программного средства 16](#_Toc71989545)

[2 ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ 17](#_Toc71989546)

[2.1 Язык разметки HTML 17](#_Toc71989547)

[2.2 Каскадные таблицы стилей CSS 18](#_Toc71989548)

[2.3 Язык программирования JavaScript 20](#_Toc71989549)

[2.4 Программная платформа Node.js 21](#_Toc71989550)

[2.5 Облачная база данных Firebase 22](#_Toc71989551)

[2.6 Git / GitHub 23](#_Toc71989552)

[3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 27](#_Toc71989553)

[4 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА 28](#_Toc71989554)

[5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 29](#_Toc71989555)

[6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 30](#_Toc71989556)

[6.1 Описание функций, назначения и потенциальных пользователей программного обеспечения 30](#_Toc71989557)

[6.2 Расчёт затрат на разработку программного обеспечения 30](#_Toc71989558)

[6.3 Оценка эффекта от продажи программного обеспечения 33](#_Toc71989559)

[6.4 Расчёт показателей эффективности инвестиций в разработку программного обеспечения 34](#_Toc71989560)

[ЗАКЛЮЧЕНИЕ 37](#_Toc71989561)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 38](#_Toc71989562)

[ПРИЛОЖЕНИЕ А 40](#_Toc71989563)

[ПРИЛОЖЕНИЕ Б 41](#_Toc71989564)

# ВВЕДЕНИЕ

Люди ведут персональный финансовый учет с различными целями: выплатить кредиты, организовать накопления, сократить расходы, спланировать даты платежей и поступлений, если с трудом хватает денег до зарплаты.

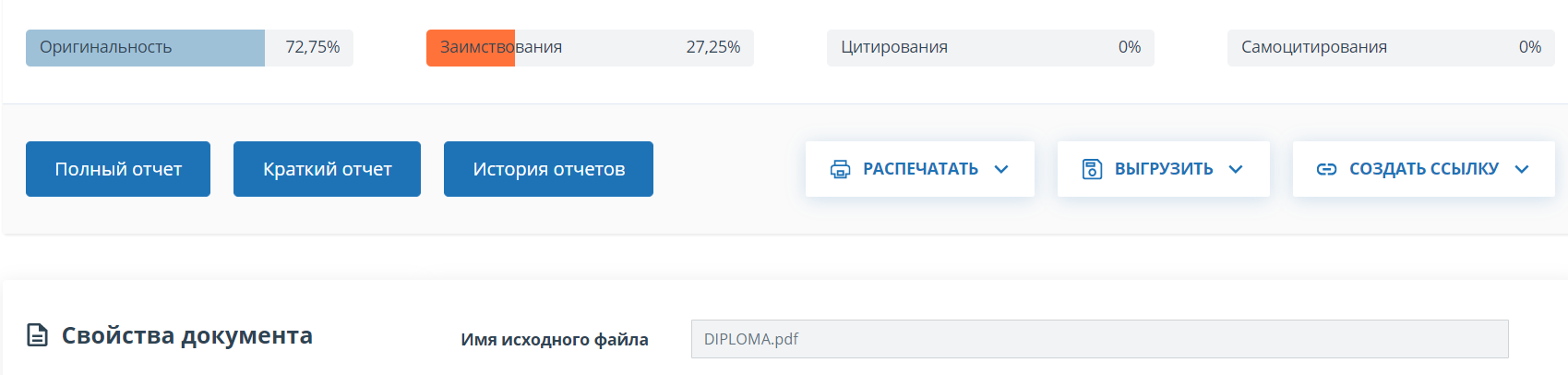
Мобильный банкинг уже давно умеет делить ваши расходы по категориям. Однако, ему ничего не известно о ваших наличных средствах, и он не знает, как планировать бюджет наперед. А учет финансов – первый шаг на пути к богатству. Магазинные чеки собирать неудобно, а переносить из них цифры в блокнот или документ Excel долго и скучно. Поэтому проблему решают мобильные приложения и веб-сервисы.

В рамках данного проекта поставлена задача разработать веб-сервис для анализа статистики личных доходов/расходов, добавления заметок и напоминаний, а также отображения статей новостных веб-сайтов в виде RSS-ленты.

При разработке программного обеспечения были поставлены следующие задачи:

* разработать веб-приложение, содержащее в себе финансовый менеджер, заметки/напоминания, и RSS-ленту;
* спроектировать хранилище данных, способное оперировать достаточно большими объёмами данных;
* создать интуитивно понятный и дружелюбный пользовательский интерфейс.

Дипломный проект выполнен самостоятельно, проверен в системе «Антиплагиат.ру». Процент оригинальности соответствует норме, установленной кафедрой информатики. Цитирования и заимствования обозначены ссылками на публикации, указанные в «Списке использованных источников».



*Скриншот с результатом проверки на плагиат*

1. **ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ**
   1. **Менеджер финансов**

Учет личных финансов – первый шаг на пути к богатству. Так считают бизнесмены и консультанты по финансовой грамотности. Но собирать чеки неудобно, переносить в памяти цифры из магазина в компьютер долго и скучно. Записывать все в тетрадку у кассы – неловко. Поэтому проблему решают мобильные приложения и веб-сервисы.

Такое приложение должно предоставлять возможность:

* разделять бюджет на доходы и расходы;
* разделять доходы/расходы по категориям;
* разделять доходы/расходы по временному периоду;
* устанавливать лимит на расход личного бюджета;
* уведомлять пользователя, когда он приближается к лимиту;
* предоставлять итоговую статистику виде графиков;
* иметь интуитивно понятный интерфейс.
  1. **RSS-Feed**

RSS ленты на сегодняшний день одна из наилучших способностей держать руку в пульсе событий. К примеру, подбор последних новостей сможет помочь находиться в курсе дел вашей ниши либо увлечения. Не нужно «лопатить» десятки ресурсов, для того чтобы отыскать актуальную информацию, достаточно подписаться на RSS каналы, а также получить сортированные по категориям новости.

Этот формат прекрасно подойдет для инфопорталов, где отсутствует высокой вовлеченности аудитории в использованный материал. То есть новости никак не комментируют, никак не делятся в соцсетях. RSS поток – это ознакомление с информацией. То, что с ней делать далее решает непосредственно абонент: идти на блог и как-то реагировать на статью, просто принимать данные к сведению либо пропускать «мимо ушей». В 90% случаев формат RSS с успехом применяется порталами, блогами (в качестве анонсирования свежих материалов), библиотечными ресурсами, досками объявлений.

Для начала разберемся, что такое RSS технически. Изнутри этот файл включает в себя три блочные структуры:

1. метаданные – сюда входят заголовки, подзаголовки, изображения, описания, основной текст;
2. гиперссылки на целевой ресурс для ознакомления с полным вариантом новости;
3. категории статей и метаданных, объединенных по тематике публикаций.

Принцип работы: как только на целевом ресурсе/источнике появляется новая информация, скрипт создает RSS файл, который считывают онлайновые или программные агрегаторы и оповещают пользователей об изменениях.

RSS начал с намерения распространять заголовки новостей. Потенциал для RSS значительно больше и может использоваться в любой точке мира.

RSS растет в популярности. Причина довольно проста. RSS — это бесплатный и простой способ продвижения сайта и его контента без необходимости рекламы или создания сложных партнерств по обмену контентом.

Говорить о плюсах и минусах использования необходимо в разрезе возможностей для пользователя и владельца сайта.

Для пользователя преимущества следующие:

* RSS ленты собирают всю информацию с подключенных источников в единый поток, который удобно быстро просматривать без лишних телодвижений;
* RSS полностью бесплатная опция. Выгода сайтов в новых заинтересованных читателях. Если анонс новости интересен и актуален, то продолжение захотят прочитать на источнике;
* быстрота уведомлений – сигнал о выходе новой публикации или новости сразу же передается пользователю. Это позволяет быть в курсе событий и вовремя реагировать на изменения;
* для мобильного интернета – экономия трафика (выгоднее подключить RSS, чем открыть десяток страниц в браузере);

Для сайта преимущества следующие:

* рост трафика – что такое RSS лента для сайта – это клики по ссылкам в новостях, переходы на ресурс и повышение вовлеченности пользователей;
* популярность ресурса – связка: полезный контент + RSS повышает узнаваемость сайта среди целевой аудитории;
* теплая ЦА – прочитав интересный анонс новости, человек в 80% случаев перейдет на сайт, чтобы продолжить ознакомление.

Недостатки для ресурсов и пользователей:

* переизбыток информации – зачастую пользователи подписываются на большое количество лент, которые как-то отвечает теме их интересов, в результате информация «бьет ключом», но обработать ее и, тем более, как-то отреагировать не хватает времени;
* поисковики раньше индексируют RSS, чем сам первоисточник;
* кража контента – XML язык считается самым простым и удобным для парсинга. Трастовые ресурсы могут спокойно находить уникальный контент и выдавать его за свой. Даже наличие ссылки не спасает от воровства, а если её и вовсе нет – то доказать статус первоисточника проблематично.

Технология RSS выгодна и для владельцев сайтов, и для интернет-пользователей. Благодаря RSS-подписке читатели могут собрать в одном сервисе все свои любимые сайты и быть в курсе публикации новых статей, не заходя на каждый из них самостоятельно. Более того, богатый функционал позволяет пользователям настраивать чтение новостей так, как они пожелают.

В свою очередь, вебмастера благодаря RSS получают постоянную целевую аудиторию, которой интересен контент, публикуемый на сайте. Все это может сказывать положительно на популярности ресурса [1].

* 1. **Облачное хранение данных**

Отличительной чертой нашего времени является постоянный рост объема деловой информации. Дизайнеры, маркетологи, копирайтеры, представители IT-профессий, а также компании, работающие с огромными массивами данных, постоянно нуждаются в надежном месте, в котором можно было бы хранить ценные файлы. Если раньше их держали на дискетах, флешках и компакт-дисках, то сейчас лучше всего отправлять их в облачные хранилища.

В общем и целом, облачное хранилище - это специально выделенное место на серверах, куда любой пользователь может закачать различные документы: текстовые файлы, любимые аудиозаписи и видеоролики, картинки, гиф-картинки, переписку из мессенджеров и многое другое. При этом серверы могут находиться где угодно: в Европе, Азии или Северной Америке.

Механизм облачного хранилища очень прост: нужно установить клиентское приложение и зарегистрироваться в нем. После чего можете спокойно сбрасывать в «облако» любую информацию, обмениваться ей с коллегами, обновлять ее, просматривать и так далее. Доступ к нему можно получить с любого устройства и из любого места, в котором есть Интернет.

У модели две стороны: клиент и поставщик услуги (провайдер). Клиент арендует место на серверах провайдера, сохраняет на них документы, приложения, статический контент сайтов, получая доступ к ним удаленно. Поставщик организует хранение, обслуживание, безопасность и доступ к данным. Эта модель имеет несколько преимуществ:

1. экономия – в случае с облачным хранилищем большинство операционных расходов (помещение, серверы, охлаждение, затраты на резервное копирование с покупкой ПО и дополнительные накопители, обслуживание) сокращаются, а капитальные отсутствуют;
2. надежность – облачные хранилища обслуживаются инженерами со специализированным опытом в эксплуатации систем такого типа. Администраторы провайдеров регулярно обновляют железо, улучшают ПО, работают над безопасностью. При этом данные хранятся «с запасом»: для хранения 1 Гб данных клиента поставщик резервирует 2 Гб. Серверы часто распределены по нескольких городам или странам, что добавляет отказоустойчивости при форс-мажорах;
3. доступность – данными можно управлять через графические интерфейсы, консоль или API;
4. масштабирование **–** объем быстро увеличивается за счет подключения дополнительных серверов и СХД;
5. управление расходами – платить нужно ровно столько, сколько потреблять ресурсов. В объектных хранилищах есть классы — стандартное, «холодное», «ледяное». Классы помогают управлять стоимостью хранения. Например, когда к данным нужно часто обращаться, можно платить дороже за хранение, но дешевле за трафик (обращения). Для архивов наоборот — можно платить за работу с файлами дороже, но за хранение дешевле, потому что к ним редко обращаются;
6. бизнес-процессы упрощаются, когда облачное хранилище доступно для сотрудника, например, из дома на выходных. А еще не забываем о восстановлении данных, когда бизнес-процессы не прерываются форс-мажорами из-за потери документов или репозитория.

Облачное хранилище во многих случаях может стать хорошей альтернативой традиционным решениям по хранению в корпоративной системе (on-premise). Однако, во многих случаях резервирование файлов в облаке имеет некоторые минусы:

1. **зависимость от Интернет-соединения;**
2. **зависимость от провайдера;**
3. **безопасность;**
4. **защита данных.**

Хранение в облаке используется:

* для хранения массивных данных;
* в качестве репозиториев контента, например, мультимедиа ресурсов;
* для Big Data, «Интернета вещей» и машинного обучения;
* для архивации или хранения для последующей аналитики;
* для хранения данных игровых платформ, вроде Google Stadia;
* видеохостинги используют хранилища для потоковой раздачи контента;
* в качестве хостинга интернет-магазинов, порталов, блогов.

Но чаще встречается пять сценариев.

1. резервное копирование и восстановление – большинство файловых систем облаков совместимы с базами данных, поэтому хранилища часто используют для резервирования, например, при обновлениях;
2. разработка ПО и тестирование – часто разработка требует дублирования сред, которые потом нужно удалять, и совместной работы. Использование облачных ресурсов для этого — стандартная практика среди разработчиков ПО. Также, облака интегрируются с разными приложениями без дополнительных «костылей»;
3. совместный доступ – например, для команд разработки и тестирования из разных офисов или городов. Если данные хранятся на сервере внутри сети предприятия, часто нужен VPN. Но можно обойтись без этого и перенести часть общих файлов, к которым обычно и нужен доступ, в облачное хранилище;
4. миграция данных в облако облегчает обслуживание своей инфраструктуры, но это серьезная задача, требующего многолетнего опыта у системного администратора;
5. Big Data и IoT – например, для Big Data массив данных в 100 Терабайт не так уж много, но держать на локальных серверах такой объем дорого, поэтому для этого часто используют облака. Хранить в «облаке» массивы удобно: в облачных сервисах обычно высокая пропускная способность, низкие задержки, и возможность настроить запросы не извлекая данные [2].

Поскольку данные бывают разные, то и хранить их лучше в подходящих для этого местах. По типу организации облачные хранилища делятся на:

* файловые;
* блочные;
* объектные;
* базы данных.
  + 1. **Файловое хранилище**

В основе файловой системы лежит иерархическая структура: корневая запись, от которой отходят данные о файлах и их атрибутах. Все они, в свою очередь, организованы в удобную структуру каталогов – зная имя того или иного документа, доступ к нему можно получить, щелкнув мышью по его имени. С ними можно осуществлять любые операции – открывать, изменять, переименовывать, удалять, копировать, перемещать в другую папку.

Файловое хранилище может быть двух видов: физическим и виртуальным. В первом случае данные сохраняются на жестком диске, во втором – на виртуальном. Последний имеет намного больший объем чем жесткий, а еще туда можно настроить удаленный доступ. В качестве примера можно привести Dropbox, «Облако Mail.Ru», «Google Диск», «Яндекс. Диск» и другие аналогичные им сервисы.

**Преимущества:**

* простая и понятная структура;
* в таком хранилище легко ориентироваться, искать нужные документы.

**Недостатки:**

* ограниченность в объеме, по мере заполнения которого падает скорость доступа, а вместе с ней и производительность.

**Для чего подходит:**для работы с небольшими объемами разных данных.

* + 1. **Блочное хранилище**

В блочном хранилище структура размещения та же, но все попадающие туда файлы делятся системой на блоки, каждому из которых присваивается свой идентификатор. С его помощью система собирает файлы в случае надобности.

**Преимущества:**

* каждая пользовательская среда находится отдельно, за счет чего можно рассортировывать данные и обеспечить отдельный доступ к ним;
* БХ обеспечивает повышенную производительность: благодаря хост-адаптеру шины, который разгружает процессор и освобождает его ресурсы для выполнения других задач.

**Недостатки:**

* оно дороже, и им трудно управлять, поскольку работа с блоками создает дополнительную нагрузку на базу данных;
* оно, как и файловое, ограничено в объеме.

**Для чего подходит:**для работы с корпоративными базами данных

* + 1. **Объектное хранилище**

Это самый популярный тип хранилища. Вместо файловой системы в нем есть плоское пространство, состоящее из множества объектов, каждый из которых состоит из идентификатора и метаданных. Идентификатор – это присвоенный адрес, в роли которого выступает 128-битное число. Зная его можно без труда найти нужный файл. Метаданные (информация о файле) – его имя, размер, координаты и другая информация.

**Достоинства:**

* возможность работы с колоссальным объемом информации. Общий объем данных, хранящихся в Haystack Facebook, оценивается в 357 петабайт;
* возможность хранения резервных копий данных, особенно тех, от которых зависит жизнедеятельность системы (например, файлы для аварийного восстановления);
* возможность проверки корректности файлов и обеспечения быстрого доступа к ним.

**Недостатки:**

* в большинстве нет интерфейса для загрузки и управления файлами.

**Для чего подходит:** для хранения больших данных, текстовых документов, изображений, медиафайлов, переписок и многого другого.

* + 1. **База данных**

База данных – это совокупность определенной информации, хранящаяся в строго установленном порядке на физических или виртуальных носителях. Она управляется специальной программой под названием СУБД (Система Управления Базами Данных). СУБД позволяет обрабатывать любые тексты, графику, медиа; с ними можно делать все что угодно: хранить, анализировать, тестировать продукты и обновления, запускать новые проекты.

Базы данных могут находиться либо на сервере, либо в облаке. Облачные СУБД сегодня являются самыми популярными в своей области. Согласно исследованиям Market Realist, их используют 35% респондентов, экспериментируют с ними 14%, планируют внедрение – 12%.

**Преимущества:**

* облачные базы данных имеют почти неограниченный объем хранения;
* есть функция резервного копирования;
* они обладают высоким внешним и внутренним уровнем безопасности;
* поддержка многозадачного и многопользовательского режимов.

**Недостатки:**

* сложность управления, что требует затрат на соответствующий персонал и ПО;
* в случае нахождения их на физическом носителе имеют ограниченный объем, так что может потребоваться увеличение дискового пространства;
* высокая стоимость разработки и эксплуатации.

**Для чего подходят:**для управления однородными массивами данных [3].

* 1. **Обзор существующих аналогов**

Рассмотрим существующие аналоги программных средств для мониторинга и анализа микроклиматических условий и выявим их преимущества и недостатки, чтобы в последствие учесть их при составлении технических требований к разрабатываемому продукту. Источники информации об аналогах – магазины мобильных приложений Play Market и App Store, а также поисковые системы Google и Yandex.

* + 1. Monefy

Monefy – крайне простое и удобное приложение для тех, кто хочет тщательно следить за расходами. Траты распределяются по категориям, а для каждой категории можно присвоить собственную иконку. Расходы отображаются в виде диаграммы. В приложении есть встроенный калькулятор и нет рекламы (рисунок 1.4.1).



Рисунок 1.4.1 - Интерфейс Monefy

* + 1. Coinkeeper

У приложения необычный интерфейс, напоминающий монетницу. Контролировать расходы и доходы можно простым перетаскиванием монет из кошелька в расходные статьи. Облачная синхронизация поможет семейным парам вести общий бюджет на разных устройствах (рисунок 1.4.2).

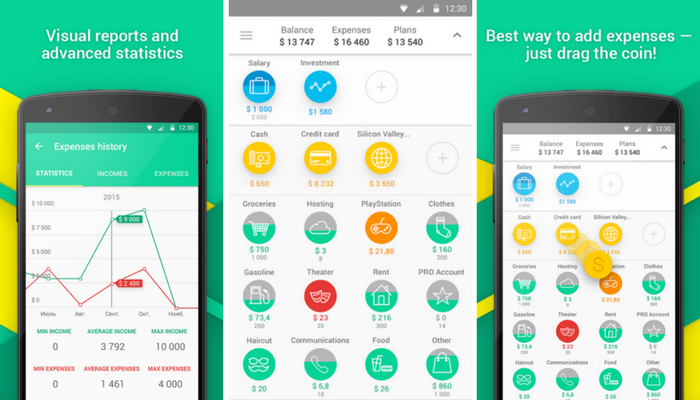


Рисунок 1.4.2 - Интерфейс Coinkeeper

* + 1. Toshl

Самое важное в Toshl это легкость использования. Вам не нужно каждый раз пояснять полное наименование затраты, достаточно пользоваться тегами. К Примеру, все напитки отметить тегом «drink», а затраты на мобильный — «mobile». Подобная концепция существенно упрощает ввод сведений, а также значительно уменьшает период, необходимое для этого (рисунок 1.4.3).

Во вкладке «Бюджет» вы добавляете все ваши доходы. Тут же отображается прогресс-бар со частями денег. Таким Образом же можно выбрать на что выделен тот либо другой «бюджет».

При намерении можно экспортировать всю статистику в отдельный документ формата .cvs, .xls либо .pdf. В Некоторых Случаях программа присылает забавные уведомления наподобие «Я заметил, что вам много денег тратите в выходные, обратите на это внимание». При этом эффективность подобных рекомендаций достаточно высока.

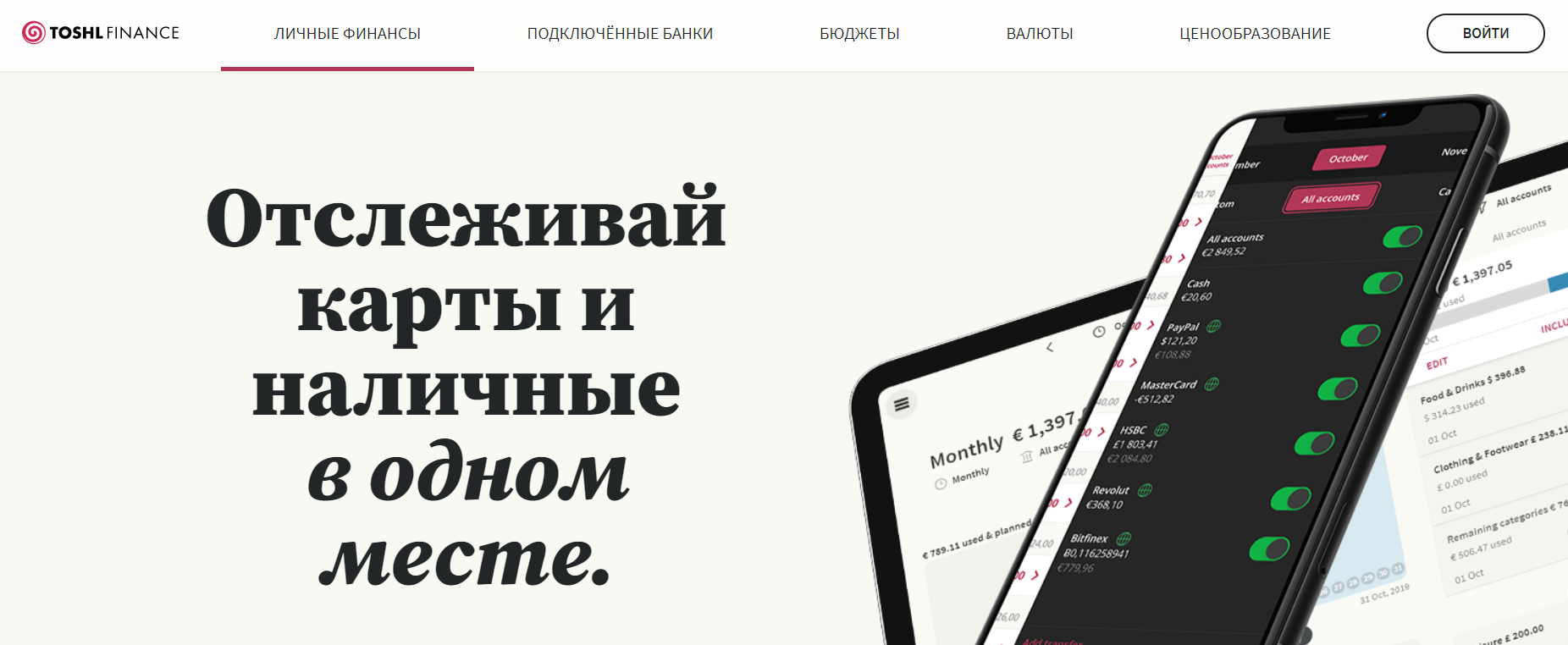


Рисунок 1.4.3 – Интерфейс Toshl

* + 1. Money Manager

Money Manager - одно из наиболее функциональных приложений, с помощью которого можно не только держать под контролем свои траты, но и получать статистику за любой период времени (рисунок 1.4.4). Здесь вы можете управлять кредитными и дебетовыми картами, получать статистику и отслеживать на графиках состояние активов. Кроме того, приложение имеет следующие особенности:

* система двойной записи;
* формирование бюджета по выбранным категориям;
* доступ с ПК;
* защита доступа паролем;
* перевод средств между активами;
* отслеживание расходов и доходов по активам;
* встроенный калькулятор;
* поиск по категориям [4].

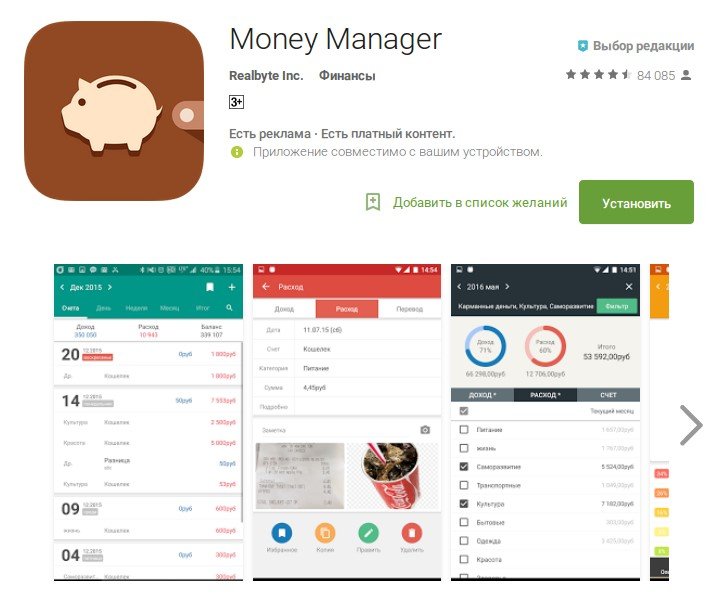


Рисунок 1.4.4 - Интерфейс Money Manager

* 1. **Постановка задачи**

В рамках данного проекта поставлена задача разработать веб-сервис для анализа статистики личных доходов/расходов, добавления заметок и напоминаний, а также отображения статей новостных веб-сайтов в виде RSS-ленты.

Потенциальной целевой аудиторией программного продукта можно считать частных лиц, заинтересованных в удобном отслеживании своего личного бюджета, заметок и напоминаний, просмотра ленты новостей, а также юридических лиц, заинтересованных в качественном ведении бухгалтерского учёта.

При разработке программного обеспечения были поставлены следующие задачи:

* разработать систему, реализующую деятельность персонального менеджера;
* спроектировать хранилище данных, способное оперировать достаточно большими объёмами данных;
* создать дружелюбный и интуитивно понятный пользовательский интерфейс.
  1. **Перспективы развития программного средства**

Разрабатываемый продукт представляет собой веб-сервис, использующий реальные данные пользователя и имеющий реальное практическое применение. Однако, в перспективе, веб-сервис будет дополнен мобильным приложением под мобильные операционные Android и iOS. Таким образом продукт станет масштабируемым для разных платформ. В дополнение к этому, будет добавлена функция парсинга смс, приходящих от банков, для последующего автоматического разделения по категориям доходов и расходов.

1. **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.

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>**.

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> [5]*.*

Рассмотрим элемент абзаца более подробно (рисунок 2.1.1).



Рисунок 2.1.1 – Элемент абзаца

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 (рисунок 2.1.2):



Рисунок 2.1.2 – Атрибут элемента

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 [6].
   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.

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;
* 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.

Начало развития 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:

* второй (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;
* третий (CSS3) – еще более масштабное расширение, находится в разработке до сих пор, поддерживает сглаживание, градиенты, тени и анимацию, для этого не приходится использовать JavaScript;
* четвёртый (CSS4) – находится в разработке, новые модули пока доступны как черновики.

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. Среди особенности стоит выделить:

* 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 [7].
  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 к HTML применяются сторонние фреймворки и библиотеки, что позволит 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 - 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 [8].
  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.

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.

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 [9].

* 1. **Git / GitHub**

Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день его поддерживает Джунио Хамано.

Среди проектов, использующих Git — ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивов Linux.

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

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и дает возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранятся операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы.

Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создает в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создается рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

Нижний уровень git является так называемой контентно-адресуемой файловой системой. Инструмент командной строки git содержит ряд команд по непосредственной манипуляции этим репозиторием на низком уровне. Эти команды не нужны при нормальной работе с git как с системой контроля версий, но нужны для реализации сложных операций (ремонт повреждённого репозитория и так далее), а также дают возможность создать на базе репозитория git свое приложение.

Для каждого объекта в репозитории вычисляется SHA-1-хеш, и именно он становится именем файла, содержащего данный объект в каталоге .git/objects. Для оптимизации работы с файловыми системами, не использующими деревья для каталогов, первый байт хеша становится именем подкаталога, а остальные — именем файла в нём, что снижает количество файлов в одном каталоге (ограничивающий фактор производительности на таких устаревших файловых системах).

В классическом обычном сценарии в репозитории git есть три типа объектов — файл, дерево и «коммит». Файл есть какая-то версия какого-то пользовательского файла, дерево — совокупность файлов из разных подкаталогов, «коммит» — дерево и некая дополнительная информация (например, родительские коммиты, а также комментарий).

Репозиторий Git бывает локальный и удаленный. Локальный репозиторий — это подкаталог .git, создаётся (в пустом виде) командой git init и (в непустом виде с немедленным копированием содержимого родительского удалённого репозитория и простановкой ссылки на родителя) командой git clone.

Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием. Удалённый репозиторий можно только синхронизировать с локальным как «вверх» (push), так и «вниз» (pull).

Наличие полностью всего репозитория проекта локально у каждого разработчика даёт Git ряд преимуществ перед SVN. Так, например, все операции, кроме push и pull, можно осуществлять без наличия интернет-соединения.

Очень мощной возможностью git являются ветви, реализованные куда более полно, чем в SVN: по сути, ветвь git есть не более чем именованная ссылка, указывающая на некий коммит в репозитории (используется подкаталог refs). Коммит без создания новой ветви всего лишь передвигает эту ссылку на себя, а коммит с созданием ветви — оставляет старую ссылку на месте, но создает новую на новый коммит, и объявляет её текущей. Заменить локальные девелоперские файлы на набор файлов из иной ветви, тем самым перейдя к работе с ней — так же тривиально.

Команда push передает все новые данные (те, которых еще нет в удалённом репозитории) из локального репозитория в репозиторий удаленный. Для исполнения этой команды необходимо, чтобы удалённый репозиторий не имел новых коммитов в себя от других клиентов, иначе push завершается ошибкой, и придётся делать pull и слияние.

Команда pull — обратна команде push. В случае, если одна и та же ветвь имеет независимую историю в локальной и в удаленной копии, pull немедленно переходит к слиянию.

Слияние в пределах разных файлов осуществляется автоматически (всё это поведение настраивается), а в пределах одного файла — стандартным двухпанельным сравнением файлов. После слияния нужно объявить конфликты как разрешенные.

Результатом всего этого является новое состояние в локальных файлах у того разработчика, что осуществил слияние. Ему нужно немедленно сделать коммит, при этом в данном объекте коммита в репозитории окажется информация о том, что коммит есть результат слияния двух ветвей и имеет два родительских коммита.

Также Git имеет временный локальный индекс файлов. Это — промежуточное хранилище между собственно файлами и очередным коммитом (коммит делается только из этого индекса). С помощью этого индекса осуществляется добавление новых файлов (git add добавляет их в индекс, они попадут в следующий коммит), а также коммит не всех измененных файлов (коммит делается только тем файлам, которым был сделан git add). После git add можно редактировать файл далее, получатся три копии одного и того же файла — последняя, в индексе (та, что была на момент git add), и в последнем коммите.

Имя ветви по умолчанию: master. Имя удалённого репозитория по умолчанию, создаваемое git clone во время типичной операции «взять имеющийся проект с сервера себе на машину»: origin.

Таким образом, в локальном репозитории всегда есть ветвь master, которая есть последний локальный коммит, и ветвь origin/master, которая есть последнее состояние удаленного репозитория на момент завершения и исполнения последней команды pull или push.

Команда fetch (частичный pull) — берёт с удалённого сервера все изменения в origin/master, и переписывает их в локальный репозиторий, продвигая метку origin/master.

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

* прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования;
* можно создавать приватные репозитории, которые будут видны только вам и выбранным вами людям. Раньше возможность создавать приватные репозитории была платной;
* есть возможность прямого добавления новых файлов в свой репозиторий через веб-интерфейс сервиса;
* код проектов можно не только скопировать через Git, но и скачать в виде обычных архивов с сайта;
* кроме Git, сервис поддерживает получение и редактирование кода через SVN и Mercurial [10].

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

# 4 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

# 5 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

# 6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

## 6.1 Описание функций, назначения и потенциальных пользователей программного обеспечения

Разрабатываемый в дипломном проекте программный модуль предназначен для отслеживания доходов и расходов по категориям и периодам отдельного пользователя, для создания заметок, а также для просмотра ленты новостей с указанного сайта.

Разрабатываемый продукт представляет собой веб-сайт, позволяющий:

* отображать визуализацию общих доходов и расходов в периоде, а также доходов и расходов по категориям;
* отображать, создавать, редактировать и удалять заметки;
* хранить данные в облачной базе данных;
* отображать ленту новостей с указанного сайта;
* работать адаптивно в веб-браузере как десктопных, так и мобильных устройств.

Среди потенциальной аудитории программного продукта можно выделить частных лиц, заинтересованных в удобном отслеживании своего бюджета, личных заметок и просмотра ленты новостей. Модуль «Финансовый менеджер» позволит пользователю анализировать свои траты по заданным категориям, а также научит ограничивать и оптимизировать свой бюджет. Модуль «Заметки» позволит лёгким и доступным образом создавать себе напоминания о каких-либо грядущих событиях. Модуль «Лента новостей» запрашивает у пользователя URL-адрес сайта с новостями, откуда будут считываться последние на данный момент статьи и в удобном формате преподноситься пользователю приложения.

Программное средство разрабатывается для свободной реализации на рынке информационных технологий и использования широким кругом потребителей.

Экономическое обоснование разработки и реализации ПО будет осуществляться в соответствии с п. 3.3. методического пособия [11].

## 6.2 Расчёт затрат на разработку программного обеспечения

Длительность разработки составит три месяца или 504 нормо-часов. Для разработки проекта «Персональный менеджер» необходимо привлечь трёх специалистов:

* менеджера проекта на 252 часа рабочего времени, должностной оклад которого равен 5208 руб./мес. (31 руб./ч.);
* программиста – на весь срок (504 часа), должностной оклад которого равен 4368 руб./мес. (26 руб./ч.);
* тестировщика на 120 часов, должностной оклад которого равен 2790 руб./мес. (15 руб./ч.);
* дизайнера на 120 часов, должностной оклад которого равен 3531 руб./мес. (19 руб./ч.).

Расчет затрат на основную заработную плату команды разработчиков осуществляется по формуле:

где *n* – количество исполнителей, занятых разработкой программного продукта;

– часовая тарифная ставка *i*-го исполнителей, *руб*.;

– трудоёмкость работ, выполняемых *i*-ым исполнителем, *ч*.

Расчёт затрат на основную заработную плату осуществляется в табличной форме (табл. 6.2.1.).

Таблица 6.2.1. – Расчет затрат на основную заработную плату разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Участник команды | Должностной оклад, *руб./ч.* | Трудоемкость работ, *ч.* | Зарплата по тарифу, *руб.* |
| 1 | Менеджер проекта | 31 | 252 | 7 812 |
| 2 | Программист (разработчик веб-приложения и базы данных) | 26 | 504 | 13 104 |
| 3 | Тестировщик | 15 | 120 | 1 800 |
| 4 | Дизайнер | 19 | 120 | 2 280 |
| Итого затраты на основную заработную плату | | | | 24996 | |

Затраты на дополнительную заработную плату команды разработчиков и определяется по формуле:

где ‒ норматив дополнительной заработной платы (25 %),

–затраты на основную заработную плату, руб.;

Затраты на дополнительную заработную плату составят:

= 6 249 руб.

Отчисления на социальные нужды определяются в соответствии с действующими законодательными актами по формуле:

где ‒ норматив отчислений в ФСЗН и Белгосстрах (35 %).

Затраты на социальные нужды составят:

(24996 + 6 249) 0,35 = 10 935,75 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:

8 748,6 руб.

Полная информация о формировании затрат на разработку программного средства приведена в таблице 6.2.2.

Таблица 6.2.2. – Затраты на разработку программного обеспечения

|  |  |
| --- | --- |
| Статья затрат | Сумма, *руб*. |
| Основная заработная плата команды разработчиков, | 24 996 |
| Дополнительная заработная плата команды разработчиков, | 6 249 |
| Отчисления на социальные нужды, | 10 935,75 |
| Прочие затраты, | 8 748,6 |
| Всего | 50 929,35 |

Таким образом, общая сумма затрат на разработку программного средства «Персональный менеджер» составит 50 929,35 руб.

## 6.3 Оценка эффекта от продажи программного обеспечения

Экономический эффект организации-разработчика программного средства представляет собой чистую прибыль от его продажи на рынке потребителям, величина которой зависит от объема продаж, цены реализации и затрат на разработку программного средства.

Организация-разработчик является налогоплательщиком (налог на прибыль), следовательно, экономический эффект можно рассчитать по формуле:

где П – прибыль за использование программного продукта;

согласно законодательству, равная 18%.

Для оценки стоимости разработанного ПО уровень рентабельности определяется по формуле:

Цена рассчитывается по формуле:

Из формулы 6.3.2. следует, что Прибыль, включаемая в цену, рассчитывается по формуле:

где – затраты на разработку и реализацию ПО

– запланированный норматив рентабельности, (по согласованию сторон равен 40%).

Налог на добавленную стоимость определяется по формуле:

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

Прибыль равна:

руб*.*

Налог на добавленную стоимость равен:

руб.

В таком случае, цена для заказчика равна:

= 85 561,31 руб.

Экономический эффект для разработчика равен:

## Расчёт показателей эффективности инвестиций в разработку программного обеспечения

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.

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 (рисунок 6.4.5-6.4.6).

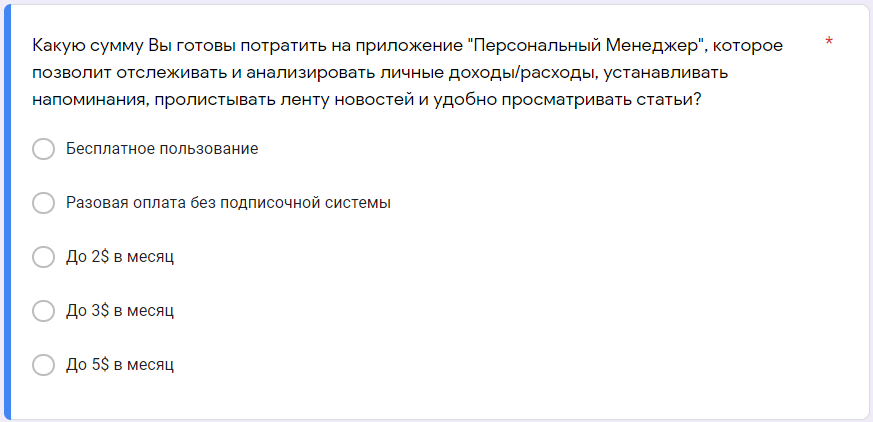


Рисунок 6.4.5. - Условие социологического опроса

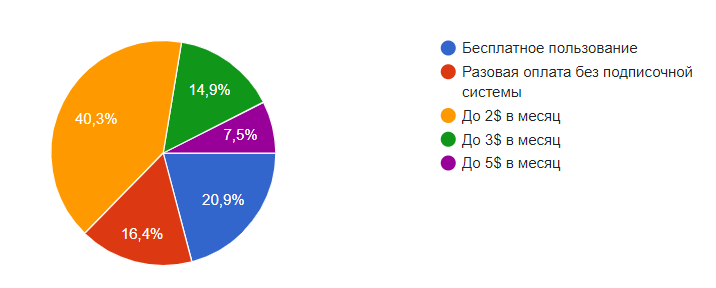


Рисунок 6.4.6. - Результат социологического опроса

Из социологического опроса следует, что оптимальная стоимость месячной подписки за пользование приложением - 2$ (5 руб.).

Из пункта 2 следует, что разработка длилась 3 месяца – с января 2021 года до апреля 2021 года.

Для привлечения пользователей и дальнейшего увеличения количества подписчиков был произведён анализ рынка на предмет стоимости контекстной рекламы, администрирования баз данных, а также на продвижение веб-приложения в поисковых системах Google и Yandex. Затраты в течении девяти месяцев на привлечение пользователей и администрирование приложения:

* затраты на контекстную рекламу в социальных сетях и прочих веб-сайтах [12]: 700 руб. \* 9 месяцев = 6 300 руб;
* затраты на продвижение в поисковых системах Google, Yandex [13]: (600 руб. + 350 руб.) \* 9 месяцев = 8 550 руб;
* затраты на администрирование [14]: 400 руб. \* 9 месяцев = 3 600 руб.

Общая сумма затрат на поддержку и продвижение программного средства в течение 9 месяцев будет равна:

*руб.*

Можно спрогнозировать рост числа пользователей разрабатываемого веб-приложения в течение года с момента начала его разработки, с помощью условий, гарантий и возможностей, которые предоставляют услуги рекламы, продвижения и администрирования программного продукта:

* 1-ый месяц: около 500 пользователей;
* 2-ой месяц: около 1 000 пользователей;
* 3-ий месяц: около 1 700 пользователей;
* 4-ый месяц: около 2 400 пользователей;
* 5-ый месяц: около 3 400 пользователей;
* 6-ой месяц: около 4 400 пользователей;
* 7-ой месяц: около 5 700 пользователей;
* 8-ой месяц: около 7 000 пользователей;
* 9-ый месяц: около 8 500 пользователей.

Из этого следует, что годовая прибыль равна:

руб*.*

Тогда, чистая прибыль равна:

руб*.*

Экономический эффект равен:

руб.

Оценка экономической эффективности инвестиций в разработку программного средства осуществляется с помощью расчета 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%, следовательно, программное средство целесообразно разрабатывать и реализовывать по установленной цене, т.к. рентабельность инвестиций превышает 115%.

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.

# ЗАКЛЮЧЕНИЕ

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

***Список используемых источников***

1 RSS: что это такое, как работает и как им пользоваться [Электронный ресурс]. Режим доступа:  
<https://www.calltouch.ru/glossary/rss-sayta/> Дата доступа: 13.05.21

2 Как работает облачное хранилище данных, его достоинства и недостатки [Электронный ресурс]. Режим доступа:  
<https://itelon.ru/blog/oblachnye-sistemy-khraneniya/> Дата доступа: 13.05.21

3 Облачные хранилища данных: для чего они нужны и какие типы существуют – База знаний Timeweb Community [Электронный ресурс]. Режим доступа: <https://timeweb.com/ru/community/articles/tipy-oblachnih-hranilishch-dannyh> Дата доступа: 13.05.21

4 Топ-7 приложений для контроля расходов | Компьютерра [Электронный ресурс]. Режим доступа:  
<https://www.computerra.ru/263304/top-7-prilozhenij-dlya-kontrolya-rashodov/> Дата доступа: 13.05.21

5 Что такое HTML? Основы языка разметки гипертекста [Электронный ресурс]. Режим доступа: <https://www.hostinger.ru/rukovodstva/shto-takoje-html/> Дата доступа: 13.05.21

6 Основы HTML - Изучение веб-разработки | MDN [Электронный ресурс]. Режим доступа: <https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics> Дата доступа: 13.05.21

7 Что такое CSS, для чего нужны каскадные таблицы стилей CSS [Электронный ресурс]. Режим доступа: <https://blog.ingate.ru/seo-wikipedia/css/> Дата доступа: 13.05.21

8 Основы JavaScript - Изучение веб-разработки | MDN [Электронный ресурс]. Режим доступа: <https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics> Дата доступа: 13.05.21

9 Что такое Firebase? [Электронный ресурс]. Режим доступа: <https://avada-media.ua/services/firebase/> Дата доступа: 13.05.21

10 GitHub — Википедия [Электронный ресурс]. Режим доступа: <https://ru.wikipedia.org/wiki/GitHub> Дата доступа: 13.05.21

11 Технико-экономическое обоснование дипломных проектов (работ). Методические указания для студентов 1 ступени высшего образования специальностей, закрепленных за УМО/ А.А. Горюшкин, А.В. Грицай, В. Г. Горовой. –Минск.: БГУИР, 2020–100с.

12 Продвижение в социальных сетях: раскрутка бизнеса ВК, ФБ, Инстаграм [Электронный ресурс]. Режим доступа: <https://1ps.ru/cost/smm/> Дата доступа: 13.05.2021

13 Продвижение сайтов в Яндексе и Google [Электронный ресурс]. Режим доступа: <https://1ps.ru/blog/dirs/stoimost-prodvizheniya-sajta-v-yandekse-i-google/> Дата доступа: 13.05.2021

14 Администрирование веб-сервера | Настройка и техническое обслуживание веб-сервера [Электронный ресурс]. Режим доступа: <https://hoster.by/service/solutions/administration-web/> Дата доступа: 13.05.2021.

# ПРИЛОЖЕНИЕ А

# ПРИЛОЖЕНИЕ Б

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | | *Дополнительные сведения* | |
|  | | | | | Текстовые документы | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР ДП 1–40 01 01 054 ПЗ | | | | | Пояснительная записка | | | | 42 с. | |
|  | | | | | Отзыв руководителя | | | |  | |
|  | | | | | Рецензия | | | |  | |
|  | | | | | Акт о внедрении | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | | Графические документы | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.753503-01  СА | | | | | Порядок разработки программных | | | | Формат А4 | |
|  | | | | | модулей. | | | |  | |
|  | | | | | Схема алгоритма | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.753503-01 СП | | | | | Программа обработки данных. | | | | Формат А4 | |
|  | | | | | Схема программы | | | |  | |
|  | | | | |  | | | |  | |
| ГУИР.753503-01 СД | | | | | Схема базы данных персонального | | | | Формат А4 | |
|  | | | | | менеджера. | | | |  | |
|  | | | | | Схема данных | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР. 753503.001 ПЛ | | | | | Диаграмма классов. | | | | Формат А1 | |
|  | | | | | Плакат | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР. 753503.002  ПЛ | | | | | Результаты проверки работоспо- | | | | Формат А1 | |
|  | | | | | собности. | | | |  | |
|  | | | | | Плакат | | | |  | |
|  | | | | |  | | | |  | |
| БГУИР. 753503.003  ПЛ | | | | | Экранные формы программы. | | | | Формат А1 | |
|  | | | | | Плакат | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  | | | | |  | | | |  | |
|  |  |  |  |  | *БГУИР ДП 1-40 04 01 054 Д1* | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Из* | *Лист* | *№ докум.* | *Подп.* | *Дата* | *Приложение персональный менеджер финансов.*  *Ведомость дипломного  проекта* | *Литера* | | | *Лист* | *Листов* |
| *Разраб.* | | *Студент* |  | *25.05* |  | *Т* |  | *46* | *46* |
| *Пров.* | | *Руководитель* |  | *29.05* | *Кафедра информатики гр. 753503* | | | | |
| *Т.контр.* | | *Консультант* |  | *29.05* |
| *Н.контр.* | | *Контролевич* |  | *30.05* |
| *Утв.* | |  |  | *10.06* |