Министерство образования Республики Беларусь

Учреждение образования «[Институт информационных технологий](https://www.google.by/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&sqi=2&ved=0CDcQFjAC&url=http%3A%2F%2Fwww.iit-bsuir.by%2F&ei=unOvUpDmGemB4gSZ3IGQDQ&usg=AFQjCNHEr4the3QhkSSjmxbzcNJBZi5-Tg&sig2=GxV3hq7_8t34Csduk0HElg&bvm=bv.57967247,d.bGE&cad=rja) Белорусского государственного университета информатики и радиоэлектроники»

Факультет компьютерных технологий

Кафедра программного обеспечения информационных технологий

|  |
| --- |
|  |
|  |
|  |

**ОТЧЕТ**

**по преддипломной практике**

Место прохождения практики: ООО «Стоматологическое образование», г.Минск

Сроки прохождения практики: с 28.10.2019 по 23.11.2019

|  |  |  |
| --- | --- | --- |
|  |  | Студент группы 681061  А. Д. Малофеевский  Руководитель практики от БГУИР  И. Л. Калитеня |

Минск 2019

**СОДЕРЖАНИЕ**

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

БД – база данных

ПС – программное средство

ПО – программное обеспечение

ОС – операционная система

ПК – персональный компьютер

REST – representational state transfer

API – application programming interface

JSON – javascript object notation

JWT – json web token

SQL – structured query language

URL – uniform resource locator

ВВЕДЕНИЕ

Каждое современное приложение сегодня дает возможность для внешнего мира (интернета) соединяться и взаимодействовать с ним. Важно, чтобы приложение могло легко отправлять и получать данные между клиентами через Интернет. Это также можно назвать способом обслуживания потребителей. Часть приложения, которая позволяет это сделать, называется API. Все основные приложения используют какой-то API для облегчения связи, API может быть внутренним или общедоступным или также может быть коммерческим. Продажа доступа к API приложений - это большой бизнес, особенно если ваши приложения предоставляют данные, которые не предоставляет никакой другой сервис или приложение. Это всего лишь введение в API, и о них можно многое узнать, но это не то, для чего предназначена эта статья. Поскольку API может легко содержать множество движущихся частей, перед его созданием должен соблюдаться архитектурный стиль. Существует множество шаблонов для одного и того же, таких как Peer-to-Peer (P2P), REST, сервис-ориентированный, ориентированный на данные, управляемый событиями и т.д. Самым известным из множества является REST. REST расшифровывается как представительский государственный трансферт[<https://medium.com/javascript-in-plain-english/the-state-of-rest-in-2019-75005eaf05b9>].

REST был правящим правителем более десяти лет. REST - это архитектурный стиль, на котором основано все современное программное обеспечение и веб-сервисы. Скорее всего, вы используете службы или приложения, созданные на основе API REST, сотни раз на своем телефоне, компьютере или любом устройстве с подключением к Интернету. Возможно, вы даже работали над REST API или написали сами! Однако, несмотря на популярность REST, у него есть некоторые явные недостатки, которые нуждались в исправлении. Вы можете спросить, что даже после десятилетия времени, почему они не были устранены? Когда вы будете читать дальше и узнавать больше о REST, вы узнаете, что недостатки на самом деле не являются недостатками, и именно так REST работает, что мешает его исправить[<https://medium.com/javascript-in-plain-english/the-state-of-rest-in-2019-75005eaf05b9>].

Взаимодействие различных сервисов с использованием АPI, из новаторства превращается в обыденность. Количество бесплатных и платных API уже исчисляется тысячами, и с каждым днем их число активно растет. А почему бы и нет? Продажа удаленных запросов к своему новаторскому сервису может принести больше прибыли, чем распространение услуг через свою площадку.

Веб-приложение, а именно генератор API предоставит возможность пользователем реализовывать микро сервисы в веб-интерфейсе позволяющие:

- Обмен данными между различными приложениями вне зависимости от платформы на которой они разработаны;

- бэкенд сервер для реализации веб/мобильных-приложений с бизнес-логикой;

- микро сервисы работающие в реальном времени(блоки из соц. сетей, игры);

- микро сервисы для глубокой аналитики

Микро сервисная архитектура приложений в последние несколько лет как описание способа дизайна приложений в виде набора независимо развертываемых сервисов. В то время как нет точного описания этого архитектурного стиля, существует некий общий набор характеристик: организация сервисов вокруг бизнес-потребностей, автоматическое развертывание, перенос логики от шины сообщений к приемникам и децентрализованный контроль над языками и данными.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Обзор области применения программного средства
   2. Обзор конкурирующих систем
   3. Постановка целей и задач на дипломное проектирование

Назначение программного средства является автоматизация и упрощение развёртывание серверного приложения основанном на микро-сервисах, работающих на RESTful API.

Программа должна обеспечивать выполнение перечисленных ниже функций:

* простое развертывание серверного приложения;
* масштабирование микро-сервисов;
* регистрация пользователей;
* надёжное шифрование паролей;
* создание сессии для пользователей с использованием JWT;
* защита сессии от не санкционированного доступа из вне;
* пользовательская настройка прав доступа к админской и внешней части приложения;
* выдача прав доступа пользователям;
* защита CROS от не санкционированного доступа к API проекта;
* аналитика просмотра контента уникальными пользователями;
* хранение и рассылка почтовых шаблонов(оповещений) по электронной почте;
* аналитика по рассылке;
* создание индивидуальных модулей;
* ограничение доступа к пользовательскому контенту;
* обработка и хранение файлов на сервере;
* сервисы, работающие в реальном времени (лента новостей, комментарии, чаты).
  1. Входные данные

Для входа в систему необходимо ввести e-mail и пароль, после чего пользователь сможет выполнять свои задачи.

Входной информацией будут является:

* данные для регистрации и авторизации;
* данные о пользователе;
* пользовательские настройки;
* загрузка файлов;
* добавление, редактирование, удаление и просмотр предустановленных и индивидуальных модулей;
* добавление, редактирование, удаление и просмотр записей в предустановленных и индивидуальных модулях;
* пользовательские запросы: POST, GET, PUT, DELETE.
  1. Выходные данные

В качестве выходных данных будет выступать:

* сессия и JWT;
* пользовательская информация;
* оповещения пользователя;
* данные предустановленных и индивидуальных модулей;
* файлы;
* аналитика.

1. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ
   1. Разработка диаграммы развёртывания

Программное средство имеет клиент-серверную архитектуру – система состоит из пяти компонентов – клиента и микро-сервисов. Клиент – это совокупное название потребительского (пользовательского) приложения, а микро-сервисы – это служебная часть, скрытая от пользователя. Структура системы представлена на диаграмме развертывания, рисунок 3.1.

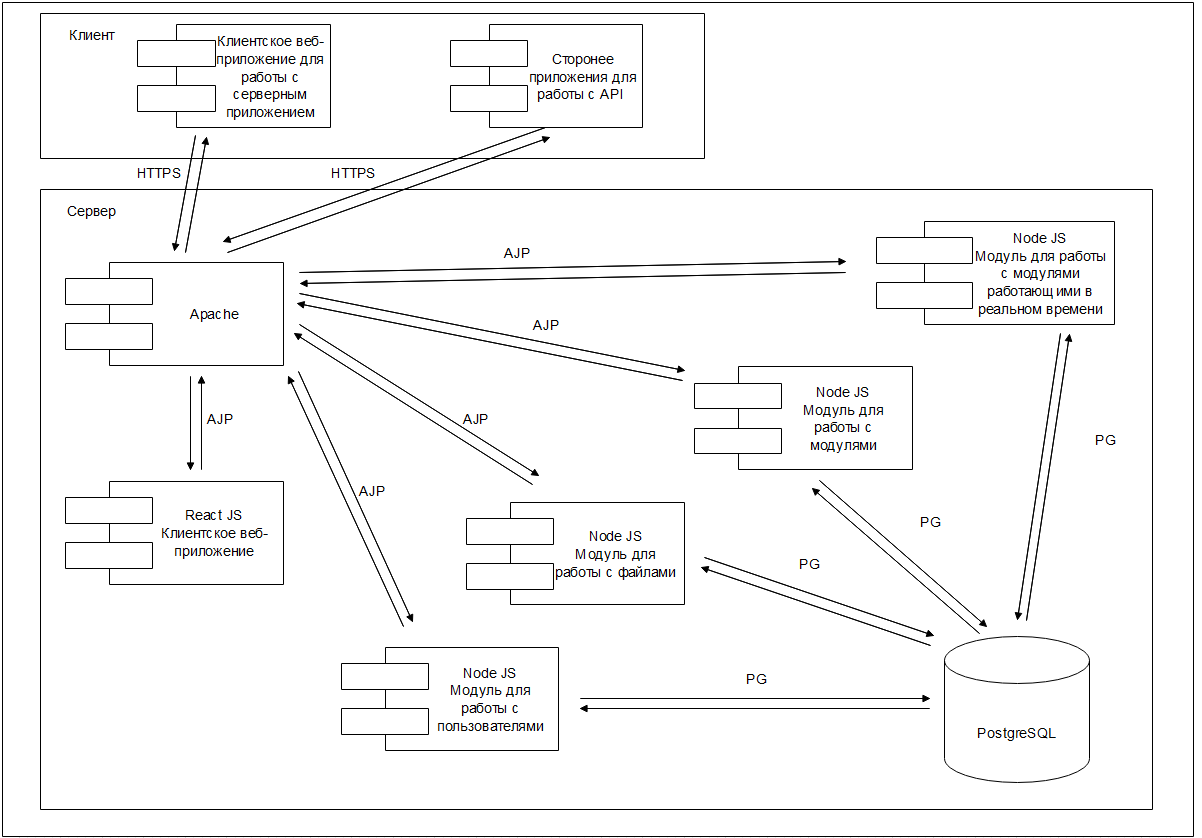


Рисунок 2.1 – Диаграмма развёртывания

Диаграммы развертывания отображают физическое развертывание артефактов (программных компонентов) на узлах (аппаратных компонентах), поэтому возможно увидеть, какие части программного обеспечения живут на каких аппаратных компонентах и как эти артефакты на узлах взаимосвязаны.

На диаграмме видно, что основная часть информации системы будет обрабатываться на сервере в микро-сервисных приложениях. Клиентское-приложения будет отвечать за визуальную часть.

Серверу передается запрос, который в свою очередь после обработки передается на СУБД. Далее из базы данных на сервер поступают данные, после чего данные передаются клиенту.

* 1. Разработка функциональной модели

Функциональная модель программного средства представлена диаграммами А-0 и А0.

Методология IDEF0 используется благодаря простой и понятной для понимания графической нотации. Главное место в методологии отводится диаграммам. На диаграммах отображают функции системы посредством геометрических прямоугольников, а также имеющиеся связи между функциями и внешней средой.

IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес–процессов. Отличительной особенностью IDEF0 является акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

Стандарт IDEF0 представляет организацию как набор модулей, существует правило – наиболее важная функция находится в верхнем левом углу, кроме того есть правило стороны:

* стрелка входа всегда приходит в левую кромку активности;
* стрелка управления – в верхнюю кромку;
* стрелка механизма – нижняя кромка;
* стрелка выхода – правая кромка.

Диаграммы А-0 представлена на рисунке 2.2.

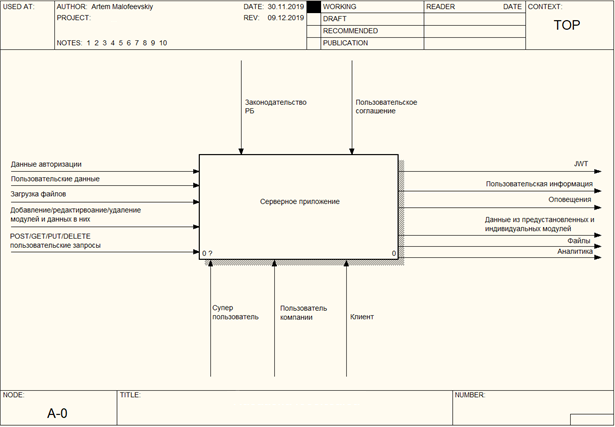


Рисунок 2.2 – Диаграмма А-0

На диаграмме А–0 (IDEF0) изображены выходные данные – JWT, пользовательская информация, оповещения, данные из предустановленных и индивидуальных модулей, файлы, аналитика. Входные данные – данные авторизации, пользовательские данные, загрузка файлов, добавление / редактирование / удаление индивидуальных модулей и данных в них, POST / GET / PUT / DELETE пользовательские запросы.

В верхней части диаграммы отображаются основный законы, которыми необходимо руководствоваться при создании программного средства.

Диаграмма А0 представлена на рисунке 2.3.

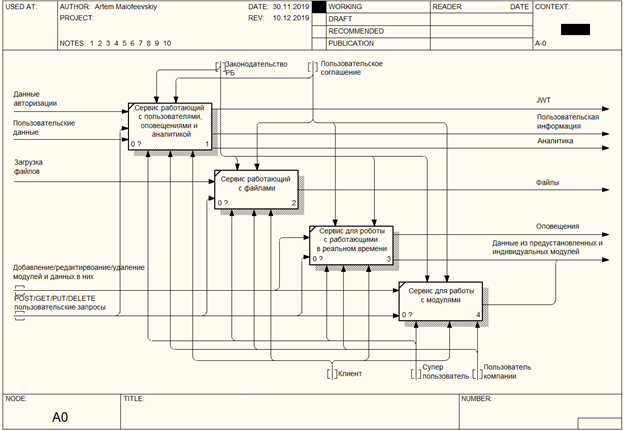


Рисунок 2.3 – Диаграмма А0

На диаграмме А0 (декомпозиция IDEF0) изображено подробное описание логики функционирования модулей программного средства и их взаимодействие с входными и выходными данными.

Модель IDEF0 используется при организации бизнес-процессов и проектов, основанных на моделировании всех процессах как административных, так и организационных.

* 1. Схема работы программы

При проектировании программного средства была разработана функциональная модель. На основе этой модели были построены алгоритмы. Общая схема работы программного средства представлена на рисунке 2.4.

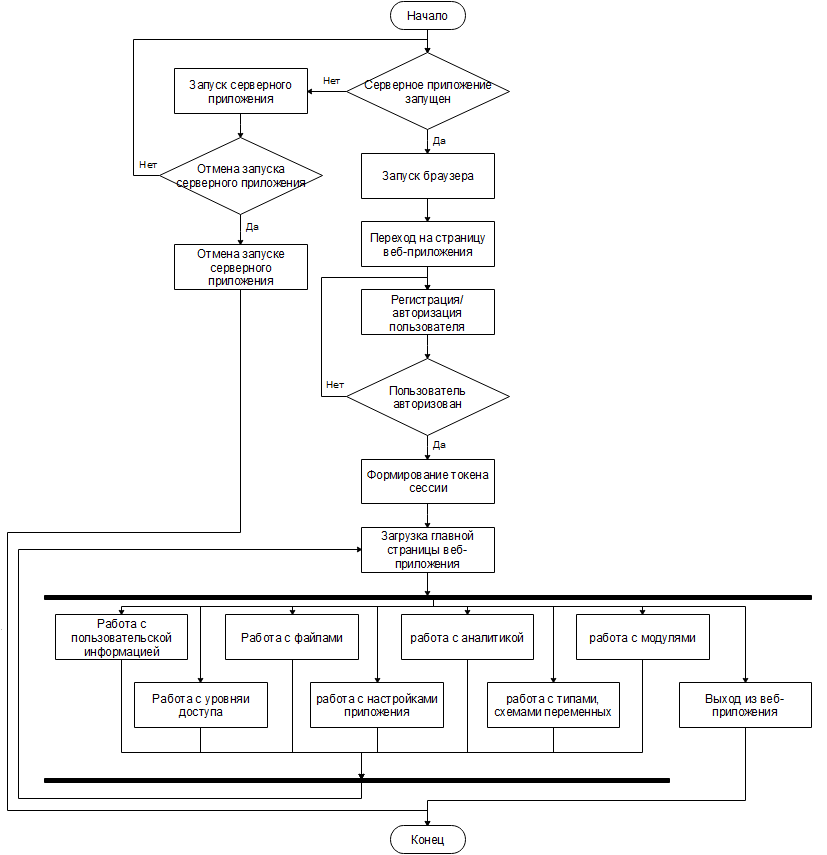


Рисунок 2.4 – Общая блок-схема работы программного средства

* + 1. Алгоритм регистрации

Схема работы алгоритма регистрации представлена на рисунке 2.5.

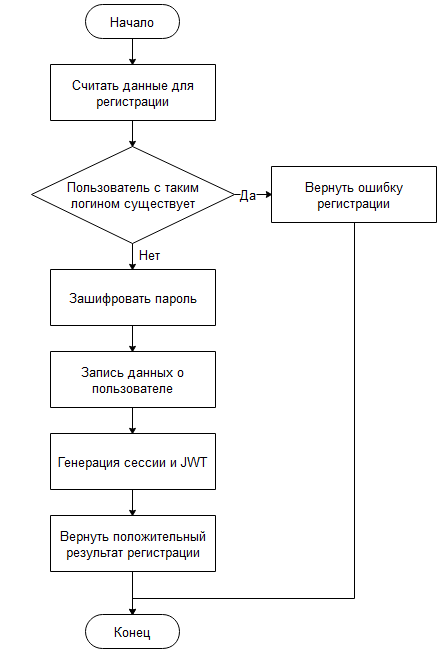


Рисунок 2.5 – Блок-схема работы алгоритма регистрации

* + 1. Алгоритм проверки сессии и JWT

Схема работы алгоритма проверки сессии и JWT представлена на рисунке 2.6.

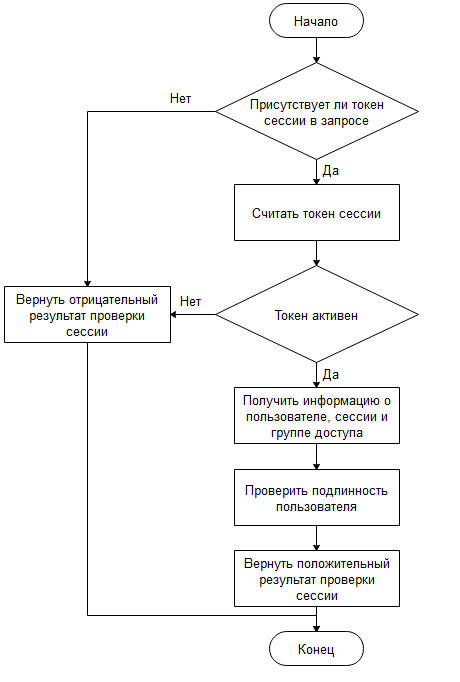


Рисунок 2.6 – Блок-схема проверки алгоритма генерации сессии и JWT

* + 1. Алгоритм роботы с модулями

Схема работы алгоритма работы с предустановленными и индивидуальными модулями представлена на рисунке 2.7.

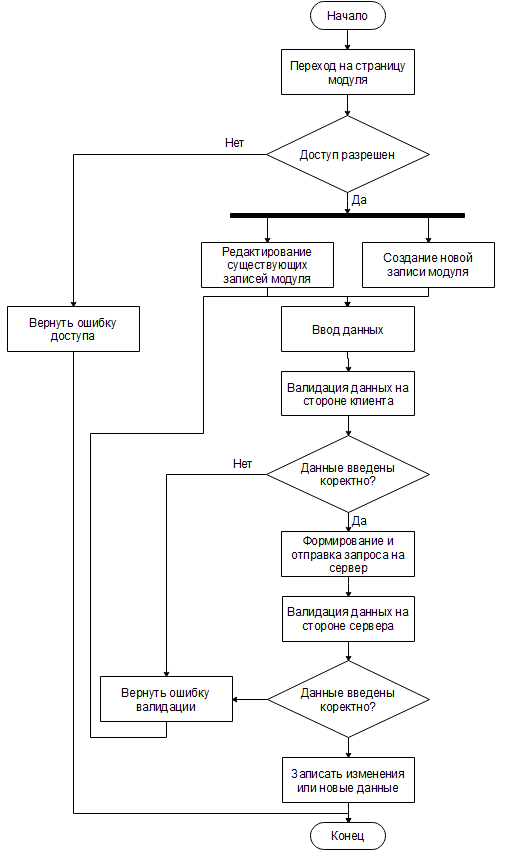


Рисунок 2.7 – Блок-схема работы с предустановленными и индивидуальными модулями

* 1. Диаграмма вариантов использования

Диаграмма вариантов использования (use case) – это исходное концептуальное представление или концептуальная модель системы в процессе ее проектирования и разработки.

Диаграмма вариантов использования – диаграмма, на которой изображаются отношения между актерами и вариантами использования.

Вариант использования – внешняя спецификация последовательности действий, которые система или другая сущность могут выполнять в процессе взаимодействия с актерами .

Актер – согласованное множество ролей, которые играют внешние сущности по отношению к вариантам использования при взаимодействии с ними.

В программном средстве присутствует два актера супер пользователь, пользователь компании, клиент. Супер пользователю предоставлены права на выполнение всех доступных функций:

* просмотр, добавление, редактирование, удаление пользователей;
* просмотр, добавление, редактирование, удаление пользовательской информации;
* просмотр, добавление, редактирование, удаление модулей;
* просмотр, добавление, редактирование, удаление записей модуля;
* просмотр, добавление, удаление файлов.

Пользователю компании обладает почти всеми вышеперечисленные функции за исключением тех, на которые у пользователя нет прав доступа.

Клиенту предоставлены права на выполнение следующих функций:

* авторизация и регистрация;
* просмотр, добавление, редактирование, удаление пользовательской информации;
* просмотр записей модуля;
* добавление записей в модуль (если есть права доступа).

Диаграмма вариантов использования для клиент-серверного приложения представлена на рисунке 2.8.

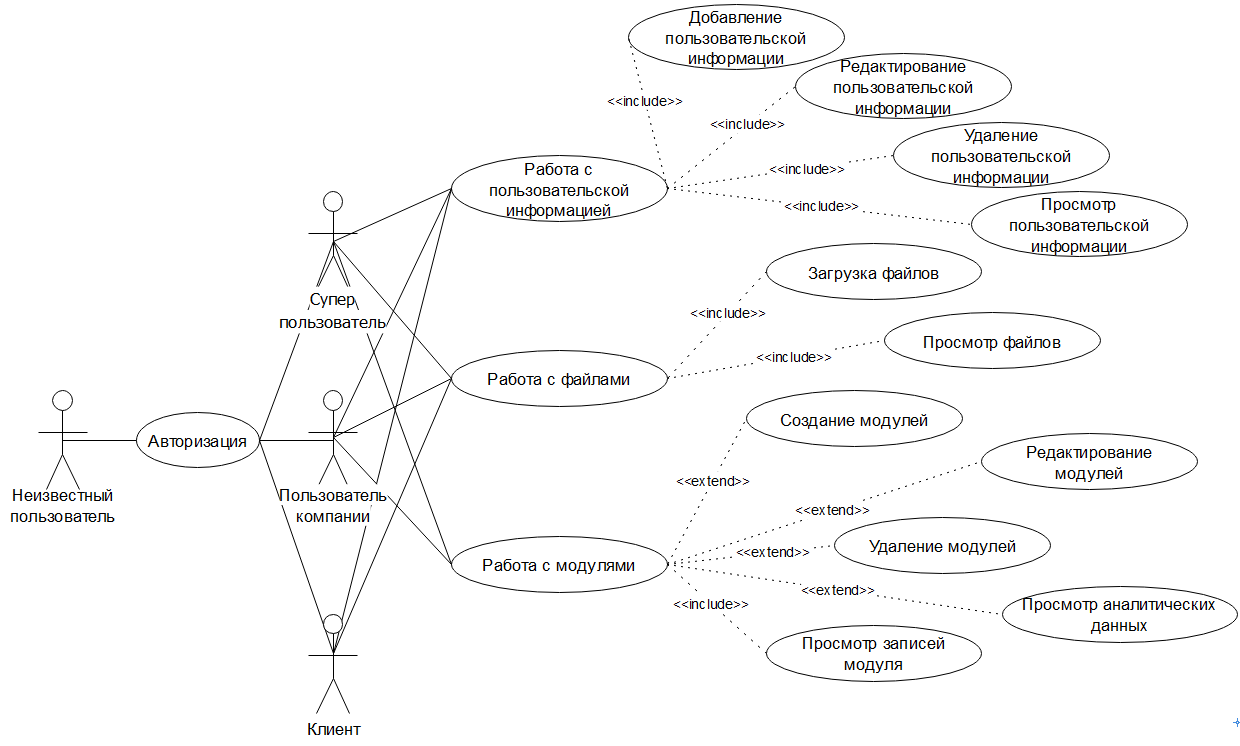


Рисунок 2.4 – Диаграмма вариантов использования

* + 1. Описание вариантов использования

1. ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА
   1. Выбор решений и инструментов для разработки

Для реализации данного микро-сервисного приложения было решено использовать платформу Node JS, а для реализации веб-приложения будет использовать React JS.

Node JS – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node JS проста и понятна интуитивно даже для начинающих кодеров. При этом Node.JS позволяет работать с серверными технологиями, реализовывать интерактивную работу с использованием компьютерных мощностей пользователей. В числе прочего, эта платформа позволяет запускать код из командной строки любой из распространенных ОС.

Преимущества использования Node JS:

* простой и широко известный язык разработки JavaScript;
* богатая стандартная библиотека;
* огромное кол-во внешних библиотек и готовых модулей;
* движок V8.

Для реализации микро-сервисов на платформе решено использовать следующие модули:

1. express – веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений;
2. express-promise-router – простая оболочка для маршрутизатора Express 4, которая позволяет промежуточному ПО возвращать обещания.
3. cross-env – служит для определения переменных в запуске приложения;
4. body-parser – считывает и парсит входящие данные в теле запросов в промежуточном программном обеспечении перед обработчиками;
5. cookie-parser – считывает и парсит входящие данные запросов в теле cookie в промежуточном программном обеспечении перед обработчиками;
6. path – предоставляет утилиты для работы с путями файлов и каталогов;
7. mime-types – служит для работы с типами файлов;
8. multer – необходим для сохранения файлов;
9. session-express – требуется для работы с сессиями;
10. passport – библиотека для работы с сессиями и JWT;
11. passport-jwt –дополнительная библиотека для работы модуля passport с JWT;
12. passport-local – служит для хранения сессии
13. pg – необходим для подключения к базе данных PostgreSQL и заботы с ней;
14. crypto – библиотека обеспечивает криптографическую функциональность;
15. jsonwebtoken – служит для генерации JWT;
16. joi – необходим для валидации запросов;

React JS – это библиотека JavaScript, исходный код которой был открыт Facebook. Этот фреймворк отлично подходит для создания огромных веб-приложений, где данные могут меняться на регулярной основе.

Преимущества использования React JS:

* легок в изучении. React JS гораздо легче учится ввиду простоты его синтаксиса;
* высокий уровень гибкости и максимальная отзывчивость;
* виртуальная DOM, которая позволяет упорядочивать документы форматов HTML, XHTML или XML в дерево, которое подходит веб-браузерам для анализа различных элементов веб-приложения;
* в сочетании с ES6/7 React JS может работать при высоких нагрузках;
* связывание данных от больших к меньшим. Поток данных, при котором дочерние элементы не могут влиять на родительские данные;
* миграция между версиями, как правило, очень проста. Также Facebook предоставляет «codemods» для автоматизации большей части этого процесса.
  1. Разработка модели данных

Модель данных представлена инфологической моделью. Инфологическое моделирование выполняется с целью обеспечения естественных для человека способов представления и сбора информации, которая будет храниться в создаваемой БД.

Поэтому инфологическая модель данных строится в соответствии с естественным языком, который невозможно использовать в чистом виде в виду сложности обработки текстов с помощью компьютера и неоднозначности естественного языка.

Инфологическая модель – это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области.

Сущность – это любой объект, отличающийся от другого, информацию о котором необходимо сохранить.

Связь – это ассоциирование нескольких сущностей с целью отыскания одних из них по значениям других.

База данных может содержать неограниченное количество сущностей и такое же количество связей между ними, что определяет сложность инфологических моделей.

Атрибут – это характеристика сущности. Его наименование должно быть уникальным.

Ключ представляет собой минимальное количество атрибутов, с помощью которого можно отыскать необходимый экземпляр сущности.

Связи между сущностями:

* один–к–одному (1:1);
* один–ко–многим (1:М);
* многие–ко–многим (М:М).

Цель инфологического моделирования – обеспечить оптимальные способы сбора и представления информации, хранимой в базе данных.

При работе над проектом были созданы таблицы, которые содержат следующие поля:

* таблица «users»: «Код», «Логин», «Хеш пароля», «Соль», «Дата регистрации», «Группа доступа», «Активный ключ доступа», «Статус»;
* таблица «profiles»: «Код», «Код пользователя», «Имя», «Фамилия» «Дата рождения», «Ключ Фото»;
* таблица «passport»: «Ключ», «Ключ пользователя», «JWT», «ip адрес», «Информация о браузере»;
* таблица «images\_base»: «Ключ», «Оригинальное название», «Название», «Путь», «Дата загрузки», «Ключ пользователя», «Статус»;
* таблица «users\_images»: «Ключ», «Ключ пользователя», «Ключ картинки»;
* таблица «load\_images»: «ключ», «Ключ картинки», «Ключ пользователя», «Дата просмотра», «Опции просмотра»;
* таблица «files\_base»: «Ключ», «Название», «Оригинальное название», «Путь», «Ключ пользователя», «Дата загрузки»;
* таблица «load\_files»: «ключ», «Ключ картинки», «Ключ пользователя», «Дата просмотра», «Опции просмотра»;
* таблица «types\_variables»: «Код», «Название», «Описание», «Статус»;
* таблица «schema\_variables»: «Код», «Код Типа», «Схема переменной»;
* таблица «variables»: «Код», «Код схемы переменной», «Название», «Опции», «Контент»;
* таблица «modules»: «Код», «Название», «Заголовок», «Описание», «Тип модуля (опции, список)», «Шаблон», «Общая схема переменных»;
* таблица «modules\_variavles»: «Код», «Код модуля», «Код переменной», «Данные»;

Диаграмма сущность–связь будет представлена на рисунке 2.1.

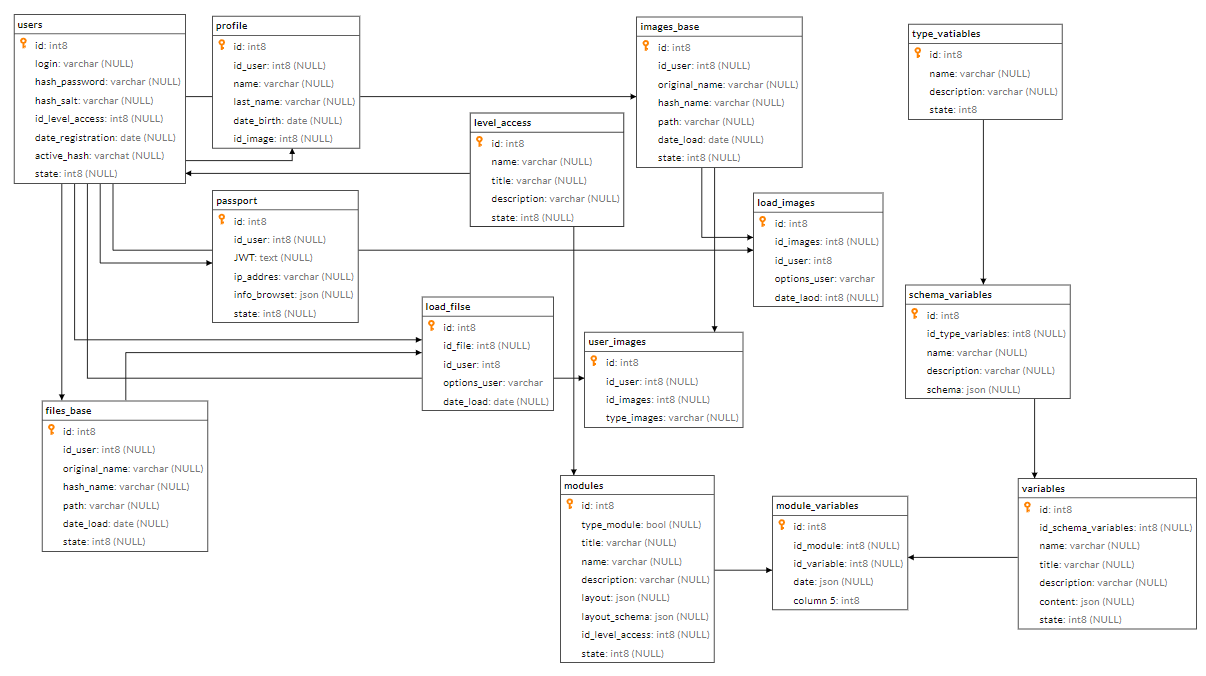


Рисунок 2.1 – Диаграмма сущность–связь

Диаграмма сущность связь предоставляет полную информацию о структуре базы данных программного средства в графическом виде.

Данные в таблицах хранятся в определённом формате, который называется типом данных. Типы данных могут быть числовыми, для работы с датой и временем, составные, бинарные и символьными.

Числовые данные – это все числа от целых до чисел двойной точности с плавающей точкой, а символьные данные содержат строки текста.

Размер поля определяется для текстовых полей. Он показывает максимальное количество символов в поле. База данных соответствует реляционной модели данных, где каждый выделенный в ходе проектировании сущности соответствует таблица. Структура базы данных разрабатываемого программного средства включает двадцать две таблицы.

Формализованное описание объектов предметной области представлено в таблицах 3.1- 3.14.

Таблица «users» содержит информацию о пользователях.

Таблица 3.1 – Структура таблицы «users»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| login | VARCHAR(255) | Заполняется пользователем | Логин пользователя |
| hash\_password | VARCHAR(255) | Генерируется алгоритмом шифрования | Зашифрованный пароль пользователя |
| hash\_salt | VARCHAR(255) | Генерируется алгоритмом шифрования | Индивидуальный ключ для пароля пользователя |
| id\_level\_acces | INT(8) | Заполняется автоматически | Уровень доступа пользователя |
| date\_registration | DATE | Заполняется автоматически | Дата регистрации пользователя |
| active\_hash | VARCHAR(255) | NULL | Активный ключ хеш, служит для восстановления пароля |
| state | BOOL | Заполняется автоматически | Статус пользователя, активен или заблокирован |

Таблица «profiles» содержит подробную информацию о пользователях.

Таблица 3.2 – Структура таблицы «profiles»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| name | VARCHAR(255) | Заполняется пользователем | Имя пользователя |
| last\_name | VARCHAR(255) | Заполняется пользователем | Фамилия пользователя |
| date\_birth | DATE | Заполняется пользователем | Дата рождения пользователя |
| id\_image | INT(8) | Заполняется автоматически | Код картинки позьзователя |

Таблица «passport» содержит информацию о закрытых и открытых сессиях.

Таблица 3.3 – Структура таблицы «passport»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| jwt | TEXT | Заполняется автоматически | Зашифрованный ключ сессии пользователя |
| ip\_addres | VARCHAR(255) | Заполняется автоматически | IP адрес пользователя |
| info\_browser | JSON | Заполняется автоматически | Информация о браузере пользователя |
| state | BOOL | Заполняется автоматически | Статус сессии |

Таблица «level\_access» содержит информацию об уровнях доступа.

Таблица 3.4 – Структура таблицы «level\_access»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| name | VARCHAR(255) | Заполняется пользователем | Название уровня доступа |
| title | VARCHAR(255) | Заполняется пользователем | Заголовок уровня доступа |
| description | VARCHAR(255) | Заполняется пользователем | Описание уровня доступа |
| state | BOOL | Заполняется пользователем | Статус уровня доступа |

Таблица «images\_base» содержит информацию о картинках.

Таблица 3.5 – Структура таблицы «images\_base»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| origin\_name | VARCHAR(255) | Заполняется автоматически | Оригинальное название картинки |
| hash\_name | VARCHAR(255) | Заполняется автоматически | Хеш названия картинки |
| path | VARCHAR(255) | Заполняется автоматически | Ссылка на картинку |
| date\_load | DATE | Заполняется автоматически | Дата загрузки картинки |
| state | BOOL | Заполняется пользователем | Статус |

Таблица «load\_images» содержит информацию о загрузках картинкок.

Таблица 3.6 – Структура таблицы «load\_images»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_image | INT(8) | Заполняется автоматически | Код картинки |
| id\_user | INT(8) | Заполняется автоматически | Код зарегистрированного пользователя |
| options\_user | VARCHAR(255) | Заполняется автоматически | Информация о не зарегистрированном пользователе |
| date\_load | DATE | Заполняется автоматически | Дата просмотра |

Таблица «user\_images» содержит информацию о картинках который задействованы в профиле пользователя.

Таблица 3.7 – Структура таблицы «user\_images»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_image | INT(8) | Заполняется автоматически | Код картинки |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| type\_images | VARCHAR(255) | Заполняется пользователем | Тип используемой картинки |

Таблица «files\_base» содержит информацию о файлах.

Таблица 3.8 – Структура таблицы «files\_base»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_user | INT(8) | Заполняется автоматически | Код пользователя |
| origin\_name | VARCHAR(255) | Заполняется автоматически | Оригинальное название картинки |
| hash\_name | VARCHAR(255) | Заполняется автоматически | Хеш названия картинки |
| path | VARCHAR(255) | Заполняется автоматически | Ссылка на картинку |
| date\_load | DATE | Заполняется автоматически | Дата загрузки картинки |
| state | BOOL | Заполняется пользователем | Статус |

Таблица «load\_files» содержит информацию о загрузках файлов.

Таблица 3.9 – Структура таблицы «load\_files»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_file | INT(8) | Заполняется автоматически | Код картинки |
| id\_user | INT(8) | Заполняется автоматически | Код зарегистрированного пользователя |
| options\_user | VARCHAR(255) | Заполняется автоматически | Информация о не зарегистрированном пользователе |
| date\_load | DATE | Заполняется автоматически | Дата просмотра |

Таблица «type\_variables» содержит информацию о типах переменных.

Таблица 3.10 – Структура таблицы «type\_variables»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| name | VARCHAR(255) | Заполняется пользователем | Название типа переменной |
| title | VARCHAR(255) | Заполняется пользователем | Заголовок типа переменной |
| description | VARCHAR(255) | Заполняется пользователем | Описание типа переменной |
| state | BOOL | Заполняется пользователем | Статус типа переменной |

Таблица «schema\_variables» содержит информацию о схемах переменных.

Таблица 3.11 – Структура таблицы «schema\_variables»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_type\_variable | INT(8) | Заполняется автоматически | Код типа переменной |
| name | VARCHAR(255) | Заполняется пользователем | Название схемы переменной |
| description | VARCHAR(255) | Заполняется пользователем | Описание схемы переменной |
| schema | JSON | Заполняется пользователем | Схема переменной |
| state | BOOL | Заполняется пользователем | Статус типа переменной |

Таблица «variables» содержит информацию о переменных.

Таблица 3.12 – Структура таблицы «variables»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_schema\_variable | INT(8) | Заполняется автоматически | Код схемы переменной |
| name | VARCHAR(255) | Заполняется пользователем | Название переменной |
| title | VARCHAR(255) | Заполняется пользователем | Заголовок переменной |
| description | VARCHAR(255) | Заполняется пользователем | Описание переменной |
| content | JSON | Заполняется пользователем | Содержимое переменной |
| state | BOOL | Заполняется пользователем | Статус переменной |

Таблица «type\_modules» содержит информацию о типах переменных.

Таблица 3.13 – Структура таблицы «type\_modules»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| name | VARCHAR(255) | Заполняется пользователем | Название типа модуля |
| title | VARCHAR(255) | Заполняется пользователем | Заголовок типа модуля |
| description | VARCHAR(255) | Заполняется пользователем | Описание типа модуля |
| state | BOOL | Заполняется пользователем | Статус типа модуля |

Таблица «modules» содержит информацию о модулях.

Таблица 3.14 – Структура таблицы «modules»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_type\_module | INT(8) | Заполняется пользователем | Код типа модуля |
| name | VARCHAR(255) | Заполняется пользователем | Название модуля |
| title | VARCHAR(255) | Заполняется пользователем | Заголовок модуля |
| description | VARCHAR(255) | Заполняется пользователем | Описание модуля |
| lay\_out | JSON | Заполняется пользователем | Шаблон модуля |
| schema\_lay\_out | JSON | Заполняется пользователем | Схема шаблона модуля |
| id\_level\_access | JSON | Заполняется пользователем | Коды уровня доступа к модулю |
| content | JSON | Заполняется пользователем | Содержимое переменной |
| state | BOOL | Заполняется пользователем | Статус переменной |

Таблица «module\_variables» содержит информацию о модулях.

Таблица 3.12 – Структура таблицы «modules\_variables»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Источник данных | Описание |
| id | INT(8) | Заполняется автоматически | Ключевое поле |
| id\_module | INT(8) | Заполняется пользователем | Код модуля |
| id\_variable | INT(8) | Заполняется пользователем | Код переменной |
| data | JSON | Заполняется пользователем | Содержание переменной |

6. ОПРЕДЕЛЕНИЕ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ

# 6.1. Характеристики программного продукта

Программное средство для генерации индивидуальных модулей и API предназначено для упрощенного и быстрого развёртывания микро-сервисного приложения на стороне сервера, приложение обладает следующими базовыми функциями:

* регистрация и авторизация пользователей;
* надёжное шифрование паролей;
* создание сессии для пользователей с использованием JWT;
* защита сессии от не санкционированного доступа из вне;
* пользовательская настройка прав доступа к админской и внешней части приложения;
* выдача прав доступа пользователям;
* защита CROS от не санкционированного доступа к API проекта;
* аналитика просмотра контента уникальными пользователями;
* хранение и рассылка почтовых шаблонов(оповещений) по электронной почте;
* аналитика по рассылке;
* создание индивидуальных модулей;
* ограничение доступа к пользовательскому контенту;
* обработка и хранение файлов на сервере;
* сервисы, работающие в реальном времени.

# 6.2. Расчёт стоимостной оценки затрат программного продукта

Рассчитаем основную заработную плату исполнителя модернизируемого программного средства.

1. Основная заработная плата рассчитывается по формуле:

 (6.1)

где n – количество исполнителей, n = 1;

– количество часов работы в день, 8 ч.;

– часовая тарифная ставка i-го исполнителя, руб. [1];

– эффективный фонд рабочего времени i-го исполнителя,

96 дней;

– коэффициент премирования, 1,4 [2].

В модернизации будет участвовать один исполнитель (таблица 6.1).

5,2·8·96·2,48 = 9904,13 руб.

Таблица 6.1 – Разряды, ставки и тарифные коэффициенты работников

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование должности | Разряд | Тарифный коэффициент | Часовая тарифная ставка, руб. |
| Инженер-программист | 10 | 2,48 | 5,2 |

1. Дополнительная заработная плата определяется в зависимости от норматива прибавки к заработной плате в процентах по формуле:

, (6.2)

где – дополнительная заработная плата исполнителя ПС,

1980,83 руб.;

– норматив дополнительной заработной платы,

20 %;

– сумма основной заработной платы исполнителя ПС,

9904,13 руб.

1. Отчисления в фонд социальной защиты населения и обязательное страхование определяются в соответствии с действующими законодательными актами по нормативу в процентном соотношении к фонду основной и дополнительной зарплаты исполнителей, определённой по нормативу, установленному в целом по организации. Вычисляется по формуле:

, (6.3)

где – сумма основной заработной платы исполнителя ПС,

;

– дополнительная заработная плата исполнителя ПС,

.;

– норматив отчислений в фонд социальной защиты населения и обязательное страхование, 34 %.

1. Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для модернизации и отладки ПС, которое определяется по нормативам на 100 строк LOC (Нмз) машинного времени в зависимости от характера решаемых задач и типа ПК. Рассчитывается по формуле:

, (6.4)

где – цена одного машино-часа, 0,098 руб.;

– общий объём, 25822 LOC;

– норматив расхода машинного времени на отладку 100 строк LOC,3%.

1. Расходы по статье «Прочие затраты» включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по научной организации, в процентах к основной заработной плате.

, (6.5)

где – сумма основной заработной платы исполнителя ПС,

9904,13 руб.;

– норматив прочих затрат,  20 %.

руб.

1. Расходы по статье «Накладные расходы», связанные с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных (экспериментальных) производств, а также с расходами на общехозяйственные нужды, относятся по нормативу (Ннр) в процентном отношении к основной заработной плате исполнителя. Рассчитывается по формуле:

, (6.6)

где – сумма основной заработной платы исполнителя ПС,

;

– норматив накладных расходов в целом по научной организации, 58 %.

1. Общая сумма производственной себестоимости (Спр) на ПС рассчитывается по формуле:

, (6.7)

где – сумма основной заработной платы исполнителя ПС,

;

– дополнительная заработная плата исполнителя ПС,;

Зсоц – отчисления в фонд социальной защиты населения, Зсоц = 4040,89 руб.;

Рпз – расходы по статье «Прочие затраты», Рпз = 1980,83 руб.;

расходы по статье «Машинное время», ;

– расходы по статье «Накладные расходы»,

1. Расходы пользователя на оплату услуг по сопровождению и адаптацию ПС, которые определяются по нормативу (Нс) и по формуле:

, (6.8)

где – норматив расходов на сопровождение и адаптацию, 20%;

– общая сумма производственной себестоимости на ПС,

1. Полная себестоимость ПС определяется по формуле:

, (6.9)

где Спр – общая сумма производственной себестоимости на ПС,

;

Рс – расходы пользователя на оплату услуг по сопровождению и адаптацию ПС, Рс = 4765,40 руб.

1. Прирост прибыли за счёт экономии расходов, связанный с высвобождением работника с повременной оплатой труда, определяется по формуле:

 (6.10)

где – коэффициент премий за выполнение плановых заданий, ;

– абсолютное высвобождение работников, = 1;

– заработная плата высвобождаемых работников i-ой категории, З = 915,2 руб.;

– процент дополнительной заработной платы, 20%;

– ставка отчислений от заработной платы, включаемых в себестоимость продукции, 34%;

– количество категорий, которым принадлежат высвобожденные работники.

1. Чистая прибыль ПС за пять лет реализации проекта, оставшаяся в распоряжении организации-разработчика, определяется по формуле:

, (6.11)

где Эi – прирост прибыли за счёт экономии расходов, Эi = 43796,05 руб.;

– норматив налога на прибыль высокотехнологичных товаров, работ и услуг,12 %.

# 6.3 Расчёт экономической эффективности программного продукта

Затраты на освоение ПС рассчитываются по формуле:

(6.12)

где – среднемесячная заработная плата сотрудника компании,

= 915,2 руб.;

– коэффициент начислений на зарплату, = 0,5;

– численность сотрудников, занятых освоением ПС, = 3 чел.;

– продолжительность освоения, = 1 мес.;

– цена одного машино-часа работы ПК, = 0,098 руб.;

– расход машинного времени на освоение ПС, = 12 машино-часов.

Всего затрат на приобретение и использование модернизации ПС, рассчитываются по формуле:

Зt = Кпр + Ко + Рс + Ктс + Коб, (6.13)

где Кпр – затраты на приобретение ПС у разработчика с учётом стоимости услуг по эксплуатации, примем в размере Кпр = 23827 руб.;

Ко – затраты пользователя на освоение ПС, Ко = 1373,98 руб.;

Рс – расходы пользователя на оплату услуг по сопровождению и адаптацию ПС, Рс = 4765,40 руб.;

Ктс – затраты на доукомплектование ВТ техническими средствами в связи с внедрением ПС, Ктс = 0 руб.;

Коб – затраты на пополнение оборотных средств, Коб = 0 руб.

Зt = 23827 + 1373,98 + 4765,40 + 0 + 0 = 29966,38 руб.

В процессе использования модернизируемого программного средства чистая прибыль в конечном итоге возмещает все затраты. Однако, полученные при этом суммы чистой прибыли и всего затрат по годам приводят к единому времени – расчётному году (за расчётный период принят 2019 год) путём умножения чистой прибыли и всего затрат за каждый год на коэффициент дисконтирования (), который рассчитывается по формуле:

(6.14)

где – требуемая норма дисконта,= 15%;

– расчётный год, 2019 год;

*–* номер года, чистой прибыли и затрат которого приводятся к расчётному периоду.

Следовательно, при решении данной задачи коэффициентам дисконтирования () по годам будут соответствовать следующие значения:

– 2020 год

– 2021 год

– 2022 год

– 2023 год

– 2024 год

Экономический эффект (Эинт.) за пять лет, рассчитывается как разность между дисконтированной суммой чистой прибыли (P*t*) и дисконтированной суммой всего затрат (З*t*) за расчётный период по формуле:

 (6.15)

Эинт. = 77355,79 – 36696,34 = 40659,45 руб.

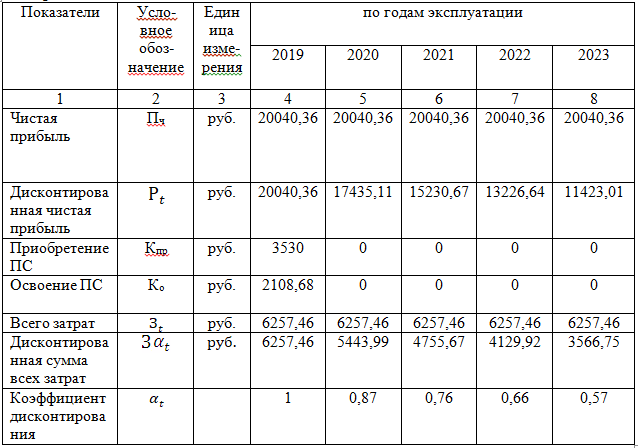
где – расчётный период, лет;

– дисконтированная чистая прибыль, полученный в году t, руб.;

 – дисконтированная сумма всех затрат в году t, руб.;

– коэффициент дисконтирования

Результаты расчёта показателей экономического эффекта приведены в таблице 6.3.

Таблица 6.3 – Расчёт экономического эффекта от использования и внедрения модернизации ПС

Ниже представлена общая диаграмма основных показателей в определении экономической эффективности:

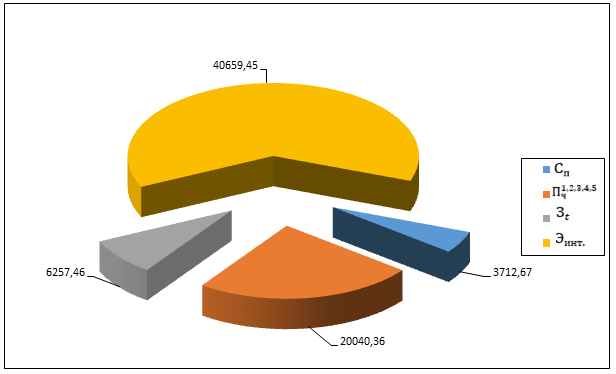


Рисунок 6.1 – Основные показатели в определении экономической эффективности

Таким образом, при определении экономической эффективности использования модернизации программного средства управления персоналом на платформе Bpm’online:

– полная себестоимость ПС в реализации проекта составляет СП =

= 3712,67 руб.;

– чистая прибыль ПС за пять лет реализации проекта, оставшаяся в распоряжении организации-разработчика составляет Пч 1,2,3,4,5 = 20040,36 руб.;

– всего затрат на приобретение и использование модернизации ПС в реализации проекта составляет Зt = 6257,46 руб.;

– экономический эффект ПС за пять лет имеет положительное значение и составил Эинт. = 40659,45 руб.

Таким образом, использование модернизации программного средства управления персоналом на платформе bpm’online является перспективным для коммерческого успеха.