

Министерство образования Республики Беларусь
Учреждение образования «Институт информационных технологий
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

Введение.....	7
1 Анализ предметной области	9
1.1 Обзор области применения программного средства	9
1.2 Обзор конкурирующих систем.....	10
1.3 Постановка целей и задач на дипломное проектирование	15
1.4 Входные данные.....	15
1.5 Выходные данные	16
2 Анализ требований к программному средству	17
2.1 Разработка функциональной модели	17
2.2 Диаграмма вариантов использования	19
2.3 Схема работы программы	25
3 Техническое проектирование программного средства.....	31
3.1 Разработка диаграммы развёртывания	31
3.2 Выбор решений и инструментов для разработки	32
3.3 Разработка модели данных	34
4 Тестирование программного средства.....	46
4.1 Описание тестируемого стенда	46
4.2 Функциональное тестирование	46
4.3 Тестирование производительности.....	51
4.4 Примеры ошибок	52
5 Руководство пользователя.....	57
6 Определение экономической эффективности.....	63
6.1 Характеристики программного продукта.....	63
6.2 Расчёт стоимостной оценки затрат программного продукта	63
Заключение	68
Список использованной литературы	69
Приложение А Исходный код приложения	70

[ГЛАВНАЯ](#) / [КАБИНЕТ](#) /

Оригинальность 94,4%

Заминтования 5,6%

Цитирования 0%

Самоцитирования 0%

Полный отчет

Краткий отчет

История отчетов

 **РАСПЕЧАТАТЬ** ▾

 **ВЫГРУЗИТЬ** ▾

 **СОЗДАТЬ ССЫЛКУ** ▾

Свойства документа

Имя исходного файла

ДП 15.01.pdf

Авторы документа

Артём Malofeevskiy

Не указано

Название документа

ДП 15.01

Тип документа

Не указано

Параметры проверки

Текстовые метрики

Статистика по документу

[РЕДАКТИРОВАТЬ СВОЙСТВА](#)

РЕФЕРАТ

ПРОГРАММНОЕ СРЕДСТВО «ОНЛАЙН ГЕНЕРАТОР API ДЛЯ ПРИЛОЖЕНИЙ» НА ПЛАТФОРМЕ NODE JS: дипломный проект / А. Д. Малофеевский. – Минск : БГУИР, 2020, – п.з. – с., чертежей (плакатов) – 6 л. формата А1.

Объектом исследования является клиент-серверное приложение для работы с предустановленными и индивидуальными модулями с возможностью использования API, которое позволит использовать контент на сторонних клиентских приложениях.

Цель работы – разработка клиент-серверного приложения для упрощения развертывания серверного приложения основанного на микро-сервисах работающих на REST API, также в приложение позволит работать с пред установленными модулями и создавать индивидуальные модули.

Разработка данного программного средства позволит упростить развертывание серверного приложения основанного на микро-сервисах и административное веб-приложение.

В ходе проектирования программного средства была разработана функциональная и инфологическая модель программного средства, схема алгоритмов основных процедур, схема программного средства, также было разработано руководство пользователя и были проанализированы аналоги разрабатываемого приложения. Клиент-серверное приложение было полностью протестировано. Определена экономическая эффективность приложения.

После тестирования и исправления всех ошибок, разработанное программное средство было использовано для реализации двух проектов, одном из проектов использовалось сгенерированное API для веб и мобильного приложения.

СОДЕРЖАНИЕ

Введение.....	7
1 Анализ предметной области	9
1.1 Обзор области применения программного средства	9
1.2 Обзор конкурирующих систем.....	10
1.3 Постановка целей и задач на дипломное проектирование	15
1.4 Входные данные.....	15
1.5 Выходные данные	16
2 Анализ требований к программному средству	17
2.1 Разработка функциональной модели	17
2.2 Диаграмма вариантов использования	19
2.3 Схема работы программы	25
3 Техническое проектирование программного средства.....	31
3.1 Разработка диаграммы развёртывания	31
3.2 Выбор решений и инструментов для разработки	32
3.3 Разработка модели данных	34
4 Тестирование программного средства.....	46
4.1 Описание тестируемого стенда	46
4.2 Функциональное тестирование	46
4.3 Тестирование производительности.....	51
4.4 Примеры ошибок	52
5 Руководство пользователя.....	57
6 Определение экономической эффективности.....	63
6.1 Характеристики программного продукта.....	63
6.2 Расчёт стоимостной оценки затрат программного продукта	63
Заключение	68
Список использованной литературы	69
Приложение А Исходный код приложения	70

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения:

БД – база данных.

СУБД – система управления базами данных.

ПС – программное средство.

ПО – программное обеспечение.

ОС – операционная система.

ПК – персональный компьютер.

ЦП – центральный процессор персонального компьютера.

ОЗУ – оперативное запоминающее устройство.

ПЗУ – постоянное запоминающее устройство.

SSD – solid-state drive – твердотельный накопитель.

REST – representational state transfer – передача состояния представления

API – application programming interface – программный интерфейс приложения.

JS – javascript – мультипарадигменный язык программирования.

JSON – javascript object notation – простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером.

JWT – json web token – это открытый стандарт (RFC 7519) для создания ключа доступа, основанный на формате JSON.

SQL – structured query language – Структурированный язык запросов.

HTML – стандартизированный язык разметки документов в сети Интернет.

XML – расширяемый язык разметки.

XHTML – расширяемый язык гипертекстовой разметки.

DOM – document object model – объектная модель документа

URL – uniform resource locator – унифицированный указатель ресурса.

Codemods – скрипт разработанный компанией Facebook для обновления кода React JS из старой версии к новой.

Apache Bench – Используется для тестирования нагрузки на веб-сервер.

Apache2 – свободный веб-сервер. Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Framework – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

UNIX-платформа – семейство переносимых, многозадачных и многопользовательских операционных систем, которые основаны на идеях оригинального проекта AT&T Unix.

ВВЕДЕНИЕ

Каждое современное приложение сегодня дает пользователям сети интернет возможность взаимодействовать с другими пользователями или приложением. Важно, чтобы приложение могло легко отправлять и получать данные между клиентами через ЛВС сеть или сеть 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 приложения предоставляют данные, которые не предоставляет никакой другой сервис или приложение. Поскольку API может легко содержать множество модульных 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. Самым известным шаблон API из множества является REST. REST расшифровывается как передача состояния представления [1].

REST – это архитектурный стиль, на котором основано все современное программное обеспечение и веб-сервисы. Однако, несмотря на популярность REST, у него есть некоторые явные недостатки, которые нуждались в исправлении.

Взаимодействие различных сервисов с использованием API, из новаторства превращается в рядовую операцию. Количество бесплатных и платных API уже исчисляется тысячами, и с каждым днем их число активно растет. Продажа удаленных запросов к своему новаторскому сервису может принести больше прибыли, чем распространение услуг через свою площадку [2].

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

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

- back-end 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 используется как комплекс из нескольких микро-сервисных приложений. Зачастую встречаются такие приложения в которых используется от 4 микро-

сервисов которые обледенены одним сервисом который контролирует работу остальных микро-сервисов.

На данный момент существует множество реализованных серверных приложений, работающих на API, но у подавляющего большинства есть проблемы. В основном проблемы заключаются в том, что приложение довольно старое, в них используются не актуальные архитектуры или устаревшие библиотеки, которые в свою очередь имеют уязвимости в безопасности. Также большая часть приложений являются платными или с закрытым исходным кодом что не дает возможность модернизировать приложение. За частую встречаются сервисы, которые работают не так как описано в документации или содержат критические ошибки в безопасности сессий или доступа к контенту.

Основными причинами, по которым стоит реализовать клиент-серверное приложение для генерации модулей API являются:

- простая и быстрая развёртка клиент-серверное приложение;
- масштабируемая архитектура, позволяющая модернизировать приложения;
- уменьшение нагрузки на серверное приложение за счёт микро-сервисной архитектуры.

База данных должна иметь чёткую структуру, позволяющую добавлять новые таблицы и вносить несущественные правки в старых, должна обладать свойством достаточности и полнотой индексов. Предпочтение следует отдать подсистеме СУБД PostgreSQL.

Темой данного дипломного проекта является программное средство онлайн генератор API для приложений на платформе Node JS.

Дипломный проект выполнен самостоятельно и проверен в системе «Антиплагиат» [3]. Процент оригинальности соответствует норме, установленной кафедрой.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор области применения программного средства

В сети интернет есть множество доступных серверных приложений, часть из них платная с закрытым исходным кодом, а также существует большое количество готовых серверных приложений, основанных на микро-сервисах, на ресурсе github.com. В сети интернет есть ресурсы, которые предоставляют веб-приложение, в котором можно настроить архитектуру приложения для работы интернет магазина, которая будет использовать технологию API для взаимодействия с пользователями. Или приложение, которое позволяет настроить работу CRM в веб-приложении также с применением технологии API для создания новых лидов или проводить полный цикл продажи через любое другое клиентское приложение. Но у всех этих приложений есть одна большая проблема, а именно данные приложения невозможно расширять, весь функционал, который предоставлен изначально невозможно расширить новыми микро-сервисами. По этому компании, которые разрабатывают собственное микро-сервисное приложение часто прибегают к таким ресурсам как github.com.

Проанализировав популярные решения на ресурсе github.com, можно прийти к следующему выводу, из тех приложений, которые были рассмотрены примерно 63% используют старые библиотеки в которых имеются дыры в безопасности или самописные библиотеки которые очень сложно поддерживать сторонним разработчикам, а также давно никем не обновлялись. 21% приложений на основе микро-сервисов не соответствуют описанию или документации представленной на ресурсе. Остальная часть приложений не доделанные, которые сложно взять за основу нового разрабатываемого приложения или приложения, которые использовались для создания видео уроков для начинающих программистов [4].

За счёт перечисленных проблем было решено разрабатывать собственное серверное приложения работающее на основе микро-сервисов API. Разработка собственного приложения решит сразу ряд проблем, таких как:

- масштабируемая архитектура приложения, она позволит легко разворачивать малые микро-сервисы или большие приложения позволяющее решать различные проблемы;
- использование необходимых модулей для решения современных задач;
- возможность дублирования или создания индивидуальных модулей;
- возможность модернизировать приложение;
- уменьшение нагрузки на приложение за счёт разделения модулей на отдельные микро-сервисы.

1.2 Обзор конкурирующих систем

Для создания программного средства, необходимо изучить аналоги и выделить их основные недостатки и преимущества, определить ведущие тенденции в данном направлении.

В качестве исследуемых аналогов были выбраны программные продукты, связанные с работой API, добавление индивидуальных модулей, работа с контентом, работа с аналитикой. Основным критерием для выбора служила актуальность данных программных средств, частота их использования.

Отличия разрабатываемого программного средства от выявленных аналогов.

Airship – это framework для Node JS, который помогает разрабатывать большие, масштабируемые и обслуживаемые API-серверы.

Основная идея проста, у каждого запроса есть своя модель у каждого ответа тоже есть модель. Важно, что на данный момент вся система даже ничего не знает о сети. Из-за этого система абстрактна, она просто обрабатывает указанные запросы и возвращает указанные ответы. Это дает возможность изменить сетевой протокол или даже прекратить использование системы в качестве веб-сервера и использовать его как часть локального приложения пользовательского интерфейса [5].

Преимущества системы:

- простая архитектура;
- автоматическую сериализацию / десериализацию моделей;
- возможность генерации схемы API;
- генерация документации на основе схемы;
- генерация простого клиента.

Недостатки системы:

- используется не реляционная БД;
- отсутствует какое-либо административное приложение;
- вся работа с приложением ведётся в коде.

Airship это оболочка для разработки серверного программного средства, оно обладает простой архитектурой. Данное программное средство позволяет быстро разработать серверное приложение на схемах API, но на данной платформе довольно сложно разработать сложную бизнес логику.

Вся не посредственная работа с framework ведётся в коде что для многих компаний является трудностью, а именно найме программиста, который должен будет разобраться с ПС или нанять сотрудника, который уже работал с ним.

На рисунке 1.1 представлена схема работы Airship framework.

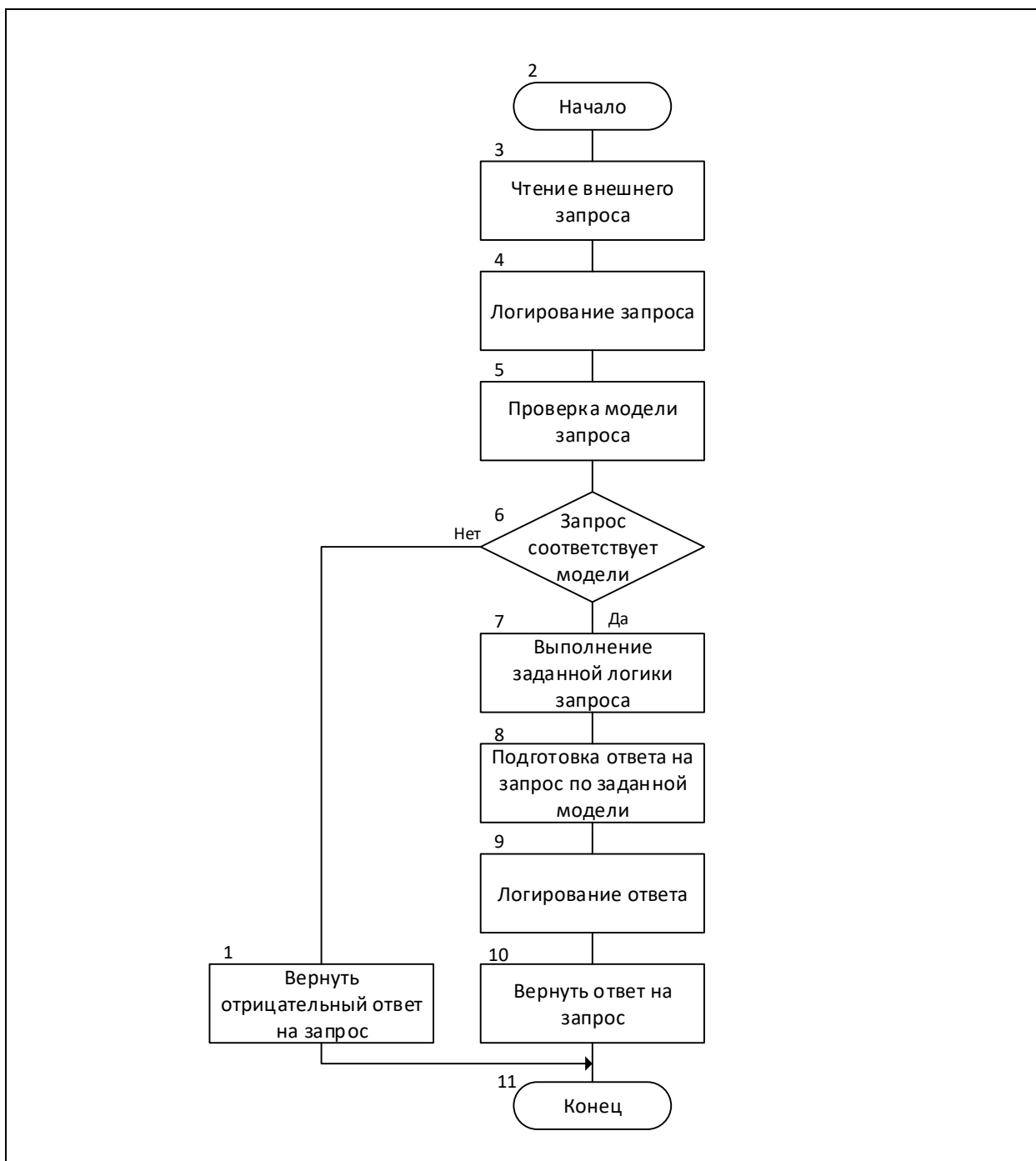


Рисунок 1.1 – Схема работы Airship framework

Hiboutik это веб-POS программное обеспечение, позволяет организовать работу интернет-магазина. Данное веб-приложение предоставляет разработчикам мощный API-интерфейс для интеграции данных в существующие системы [6].

Преимущества системы:

- быстрое и простое развёртывание;
- подходит для работы с интернет магазинами;
- подробная документация;

- не требует собственных или арендованных серверных машин.

Недостатки системы:

- узко направленная система;
- нет возможности модернизации приложения;
- приложение является платным.

Данное программное средство позволяет вести учет товаров, список клиентов и вести список продаж. С товарами предусмотрена базовые функции, такие как: прибытие товара на склад, наличие товара на складе, импорт и экспорт. На рисунке 1.2 представлена страница со списком товаров.

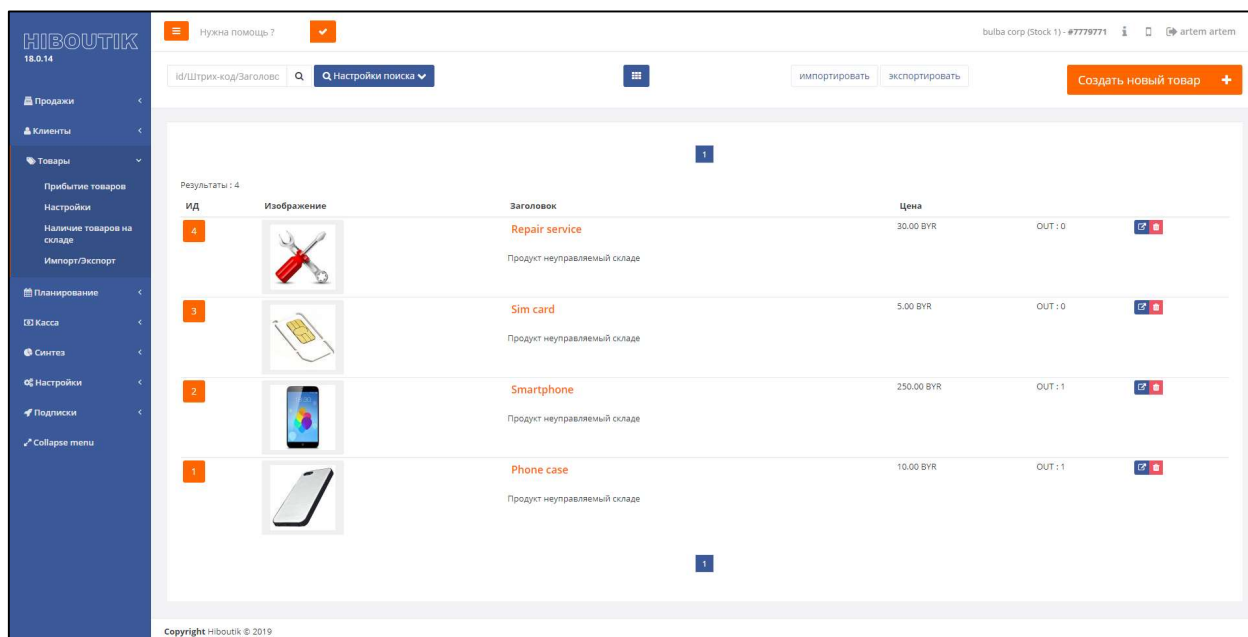


Рисунок 1.2 – Страница списка товаров

На странице продажи можно создать новую продажу или посмотреть продажи за определённый период или продажи по необходимому клиенту, на рисунке 1.3 представлена страница продаж.

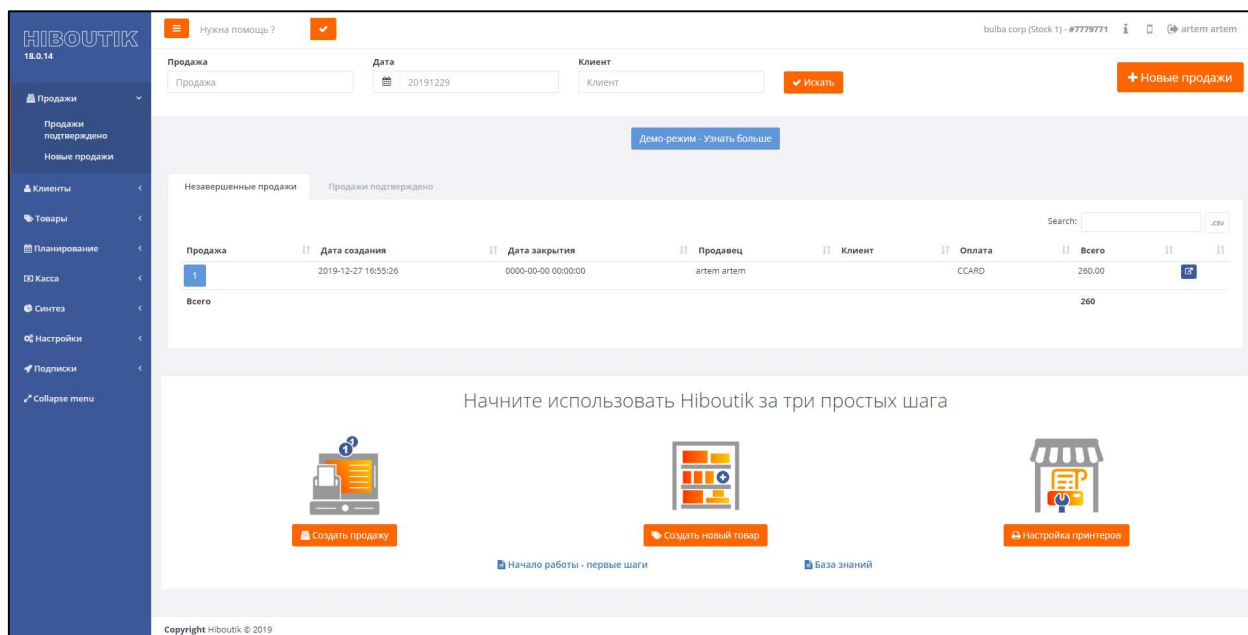


Рисунок 1.3 – Страница списка продаж

По итогу анализа программного средства Hiboutik было выяснено что данный продукт подходит только для ведения малого бизнеса и не годится для использования в крупных компаниях где процесс продаж или иная деятельность с клиентами не является стандартной или шаблонной по отношению к данным продуктам где работа с клиентами ведётся потоково.

AmoCRM - система управления взаимоотношениями с клиентами, позволяющая контролировать ход продаж и доступная в режиме online из любой точки мира [7].

Преимущества системы:

- быстрое и простое развёртывание;
- реализовано множество решений для работы с клиентами и продажами;
- простая в пользовании и не требует длительного обучения пользователя;
- присутствует пробный период;
- возможность управлять продажами и клиентами с помощью API;
- присутствует аналитика.

Недостатки системы:

- узко направленная система;
- нет возможности модернизации приложения;
- приложение является платным.

В данном программном продукте сделан упор на проведение сделок, в ПС ведется учёт клиентов, сделок, а также проводится аналитика по сделкам. Аналитику можно проводить по сделкам, по сотрудникам, звонкам. На рисунке 1.4 представлена страница анализа продаж.

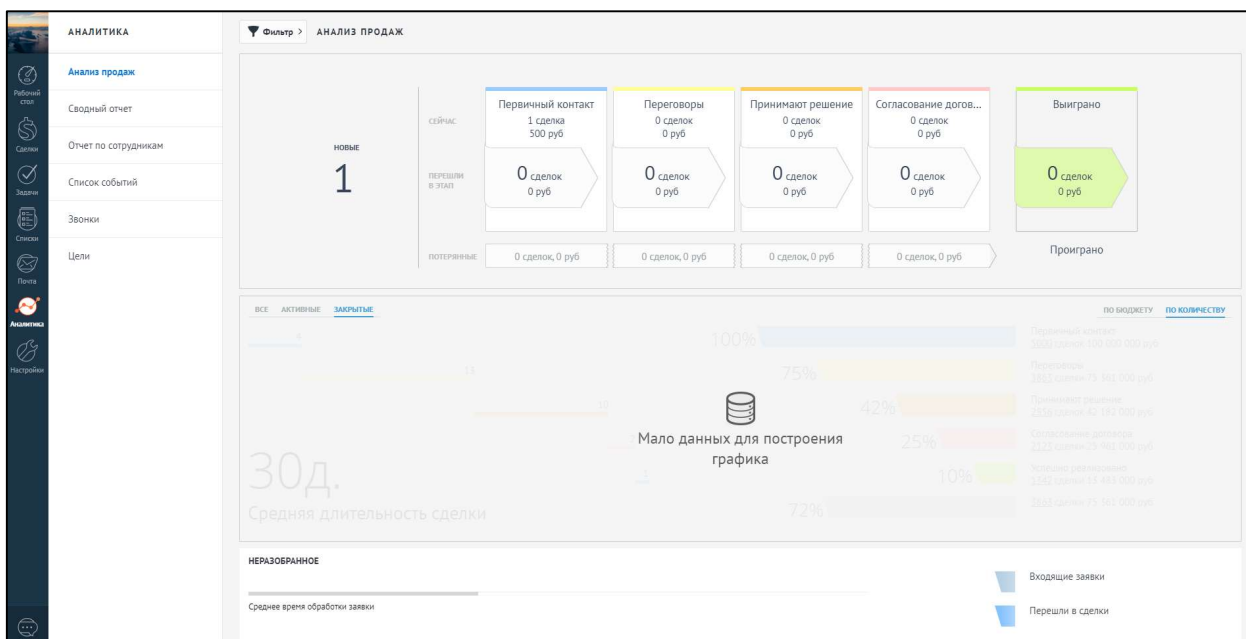


Рисунок 1.4 – Страница анализа продаж

У каждого добавленного сотрудника в программное средство индивидуальная страница сделок где он их проводит на данной странице он может отслужить на какой из стадий находится сделка, также пользователь может создать новую сделку или завершить существующую. На рисунке 1.5 представлена страница сделок.

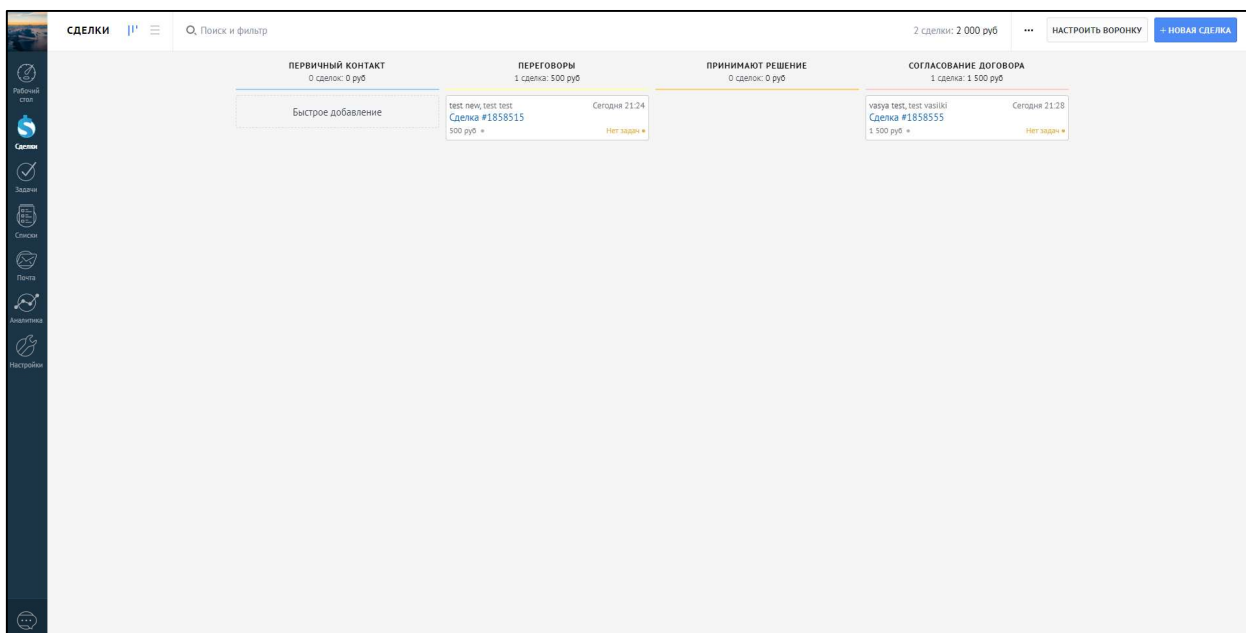


Рисунок 1.5 – Страница сделок

По итогу анализа данного ПС было выяснено что оно больше подходит для отдела продаж. AmoCRM является не плохим решение для проведения

продаж, поддержки клиентов. Также реализовано множество не плохих решений, таких как встроенная почтовая служба для общения с клиентами, телефония, аналитика и постановка задачи для напоминания о намеченном действии с клиентом или поставить задачу на другого сотрудника. Также данное программное средство имеет не плохое встроенное API что позволяет начать работать с клиентом с первого его входа на сайт и не вносить в ручную сотрудникам данные клиента.

1.3 Постановка целей и задач на дипломное проектирование

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

Программа должна обеспечивать выполнение перечисленных ниже функций:

- простое развёртывание 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).

1.4 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;
- методы запросов: GET, POST, PUT, DELETE.

1.5 Выходные данные

В качестве выходных данных будет выступать:

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

В результате проведенного анализа предметной области, был проведён обзор области применения программного средства. Исходя из обзора были проанализированы ресурсы с готовыми решениями программных средств позволяющие работать с API, в результате чего представлена статистка по приложениям, по которой можно сделал вывод что реализация собственного приложения, работающего на микро-сервисах правильное решение которое позволит решить множество проблем готовых решений. Также были рассмотрены конкурирующие системы по итогу которых были описаны положительные и отрицательные стороны систем. По итогам анализа была поставлена цель и задачи, которые необходимо выполнить в дипломном проекте. Также описаны входные и выходные данные.

2 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ

2.1 Разработка функциональной модели

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

Для формализации и описания процесса разработки ПО используется методология функционального моделирования и графическая нотация IDEF0. Отличительной особенностью данной нотации является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность.

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 [8].

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 [9].

Каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер. Блок выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня. Также для того чтобы быть правильно понятым, существуют словари описания активностей и стрелок. В этих словарях можно дать описания того, какой смысл вы вкладываете в данную активность либо стрелку.

Диаграммы А-0 представлена на рисунке 2.1.

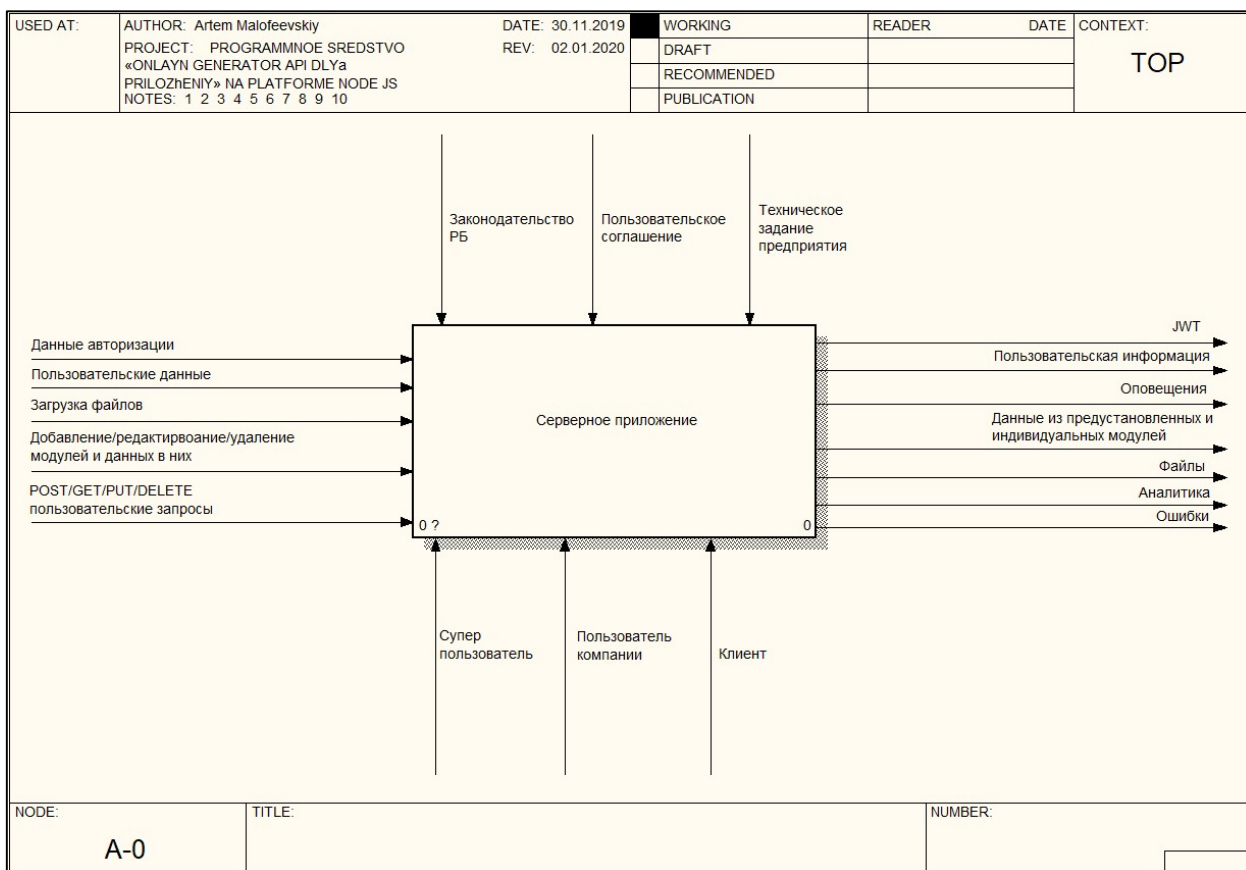


Рисунок 2.1 – Диаграмма A-0

На диаграмме A–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 методы запросов.

Используемые методы запросов:

- метод POST запроса – предназначен для запроса, при котором веб-сервер принимает данные;
- метод GET запроса – предназначен для получения информации от веб-сервер;
- метод PUT запроса – предназначен для запроса, при котором веб-сервер принимает данные и заменяет их в существующей записи;
- метод DELETE запроса – предназначен для запроса, при котором веб-сервер удаляет выбранную запись.

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

Диаграмма A0 представлена на рисунке 2.2.

- 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.

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).

The diagram illustrates the functional requirements of a project management system, showing the interactions between different user roles and the system's use cases.

Actors (Users):

- Не авторизованный пользователь (Not authorized user)
- Супер пользователь (Super user)
- Пользователь компании (Company user)
- Клиент (Client)

Use Cases:

- Авторизация (Authorization)
- Работа с правами супер пользователя (Super user rights management)
- Работа с правами пользователя (User rights management)
- Работа с правами пользователя компании (Company user rights management)
- Работа с правами клиента (Client rights management)
- Работа с настройками приложения (Application settings management)
- Работа с правами доступа (Access rights management)
- Работа с пользователями (User management)
- Работа с модулями (Module management)
- Просмотр записей модулей (Module records viewing)
- Загрузка и просмотр файлов (File upload and viewing)
- Работа с пользовательской информацией (User information management)

Relationships:

- The **Не авторизованный пользователь** actor is associated with the **Авторизация** use case.
- The **Супер пользователь** actor is associated with the **Авторизация**, **Работа с правами супер пользователя**, **Работа с правами пользователя**, **Работа с правами пользователя компании**, and **Работа с правами клиента** use cases.
- The **Пользователь компании** actor is associated with the **Авторизация**, **Работа с правами супер пользователя**, **Работа с правами пользователя**, **Работа с правами пользователя компании**, and **Работа с правами клиента** use cases.
- The **Клиент** actor is associated with the **Работа с правами супер пользователя**, **Работа с правами пользователя**, **Работа с правами пользователя компании**, and **Работа с правами клиента** use cases.
- The **Работа с правами супер пользователя** use case is associated with the **Работа с настройками приложения** use case.
- The **Работа с правами супер пользователя** use case is associated with the **Работа с правами доступа**, **Работа с пользователями**, and **Работа с модулями** use cases.
- The **Работа с правами супер пользователя** use case is associated with the **Просмотр записей модулей**, **Загрузка и просмотр файлов**, and **Работа с пользовательской информацией** use cases.

20

Диаграмма вариантов использования описывает взаимоотношения и зависимости между группами вариантов использования и действующими лицами, участвующими в процессе.

2.2.1 Описание вариантов использования

Описанные варианты использования «Регистрации», «Добавление пользовательской информации», «Загрузка файлов», «Создание индивидуальных модулей», «Просмотр записей модуля», «Просмотр аналитики». Для удобства рассмотрения представлены в табличном формате.

В таблице 2.1 представлено описание варианта «Регистрации».

Таблица 2.1 – Описание варианта «Регистрации»

Наименование	Описание	Актёры	Предусловия	Постусловия
Регистрация	Описывает регистрацию в системе. Используется, когда пользователь хочет зарегистрироваться в системе.	Применимо ко всем актёрам	Отсутствуют	При успешном выполнении происходит вход в систему. При неудачном – состояние системы не изменится

В таблице 2.2 представлено описание варианта «Добавление пользовательской информации».

Таблица 2.2 – Описание варианта «Добавление пользовательской информации»

Наименование	Описание	Актёры	Предусловия	Постусловия
Добавление пользовательской информации	Описывает добавление пользовательской информации в систему. Используется, когда пользователь хочет добавить пользовательскую информацию в систему.	Применимо ко всем актёрам	Отсутствуют	При успешном выполнении происходит сохранение данных в системе. При неудачном – состояние системы не изменится

В таблице 2.3 представлено описание варианта «Загрузка файлов».

Таблица 2.3 – Описание варианта «Загрузка файлов»

Наименование	Описание	Актёры	Предусловия	Постусловия
Загрузка файлов	Описывает загрузку файлов в систему. Используется, когда пользователь хочет загрузить новый файл в систему.	Применимо ко всем актёрам	Проверка наличия прав доступа	При успешном выполнении происходит сохранение файла или файлов в систему. При неудачном – состояние системы не изменится

В таблице 2.4 представлено описание варианта «Создание индивидуальных модулей».

Таблица 2.4 – Описание варианта «Создание индивидуальных модулей»

Наименование	Описание	Актёры	Предуслови я	Постуслови я
Создание индивидуальных модулей	Описывает создание индивидуальных модулей в системе. Используется, когда пользователь хочет создать индивидуальный модуль в системе.	Супер пользователь, Пользователь компании	Проверка наличия прав доступа	При успешном выполнении происходит сохранение модуля в систему. При неудачном – состояние системы не изменится

В таблице 2.5 представлено описание варианта «Просмотр записей модуля».

Таблица 2.5 – Описание варианта «Просмотр записей модуля»

Наименование	Описание	Актёры	Предуслови я	Постуслови я
Просмотр записей модуля	Описывает просмотр записей модуля в системе. Используется, когда пользователь хочет просмотреть информацию из модуля в системе.	Применимо ко всем актёрам	Проверка наличия прав доступа	При успешном выполнении происходит вывод информации в приложение. При неудачном – состояние системы не изменится

В таблице 2.6 представлено описание варианта «Просмотр аналитики».

Таблица 2.6 – Описание варианта «Просмотр аналитики»

Наименование	Описание	Актёры	Предусловия	Постусловия
Просмотр аналитики	Описывает просмотр аналитики в системе. Используется, когда пользователь хочет просмотреть аналитику в системе.	Супер пользователь, Пользователь компании	Проверка наличия прав доступа	При успешном выполнении происходит вывод информации в приложение. При неудачном – состояние системы не изменится

В таблице 2.7 представлено описание варианта «просмотр, редактирование и добавление меток в предустановленном модуле «Метки»».

Таблица 2.7 – Описание варианта «просмотр, редактирование и добавления меток в предустановленном модуле «Метки»»

Наименование	Описание	Актёры	Предуслови я	Постуслови я
Просмотр и редактирование меток в предустановленном модуле «Метки»	Описывает просмотр, редактирование и добавление меток в систему. Используется, когда пользователь хочет работать с метками.	Супер пользователь, Пользователь компании	Проверка наличия прав доступа	При успешном выполнении происходит вывод существующих записей из модуля в приложение. При неудачном – состояние системы не изменится

В таблице 2.8 представлено описание варианта «добавление записи в модуль «Записи»».

Таблица 2.8 – Описание варианта «добавление записи в модуль «Записи»»

Наименование	Описание	Актёры	Предусловия	Постусловия
добавление записи в модуль «Записи»	Описывает добавление записи в модуль «Записи». Используется, когда пользователь хочет добавить новую запись в модуль «записи».	Супер пользователь, Пользователь компании	Проверка наличия прав доступа	При успешном выполнении происходит запись в базу данных вывод информирующего уведомления и добавляется запись в приложение. При неудачном – состояние системы не изменится

2.3 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.

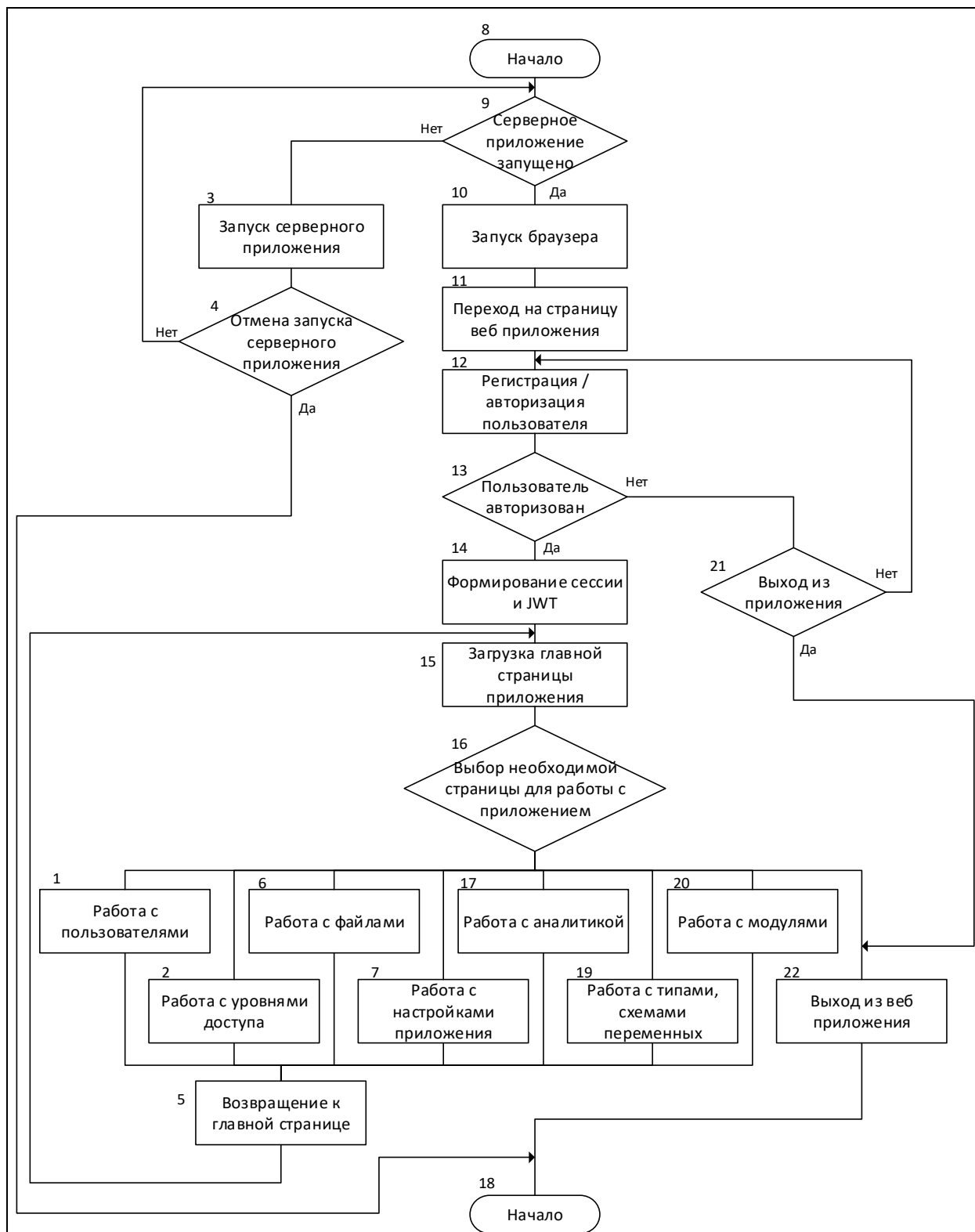


Рисунок 2.4 – Общая блок-схема работы программного средства

Представленный алгоритм представляет общую схему работы приложения. Перед запуском приложения необходимо запустить сервер, после чего будет доступно веб-приложение в котором ведётся всё основная работа. Чтобы выполнять работу в приложении необходимо авторизоваться, после чего загрузиться главная страница, на главной странице будет отображены разделы, к которым у пользователя есть доступ.

2.3.1 Алгоритм регистрации на стороне сервера

Схема работы алгоритма регистрации на стороне сервера представлена на рисунке 2.5.

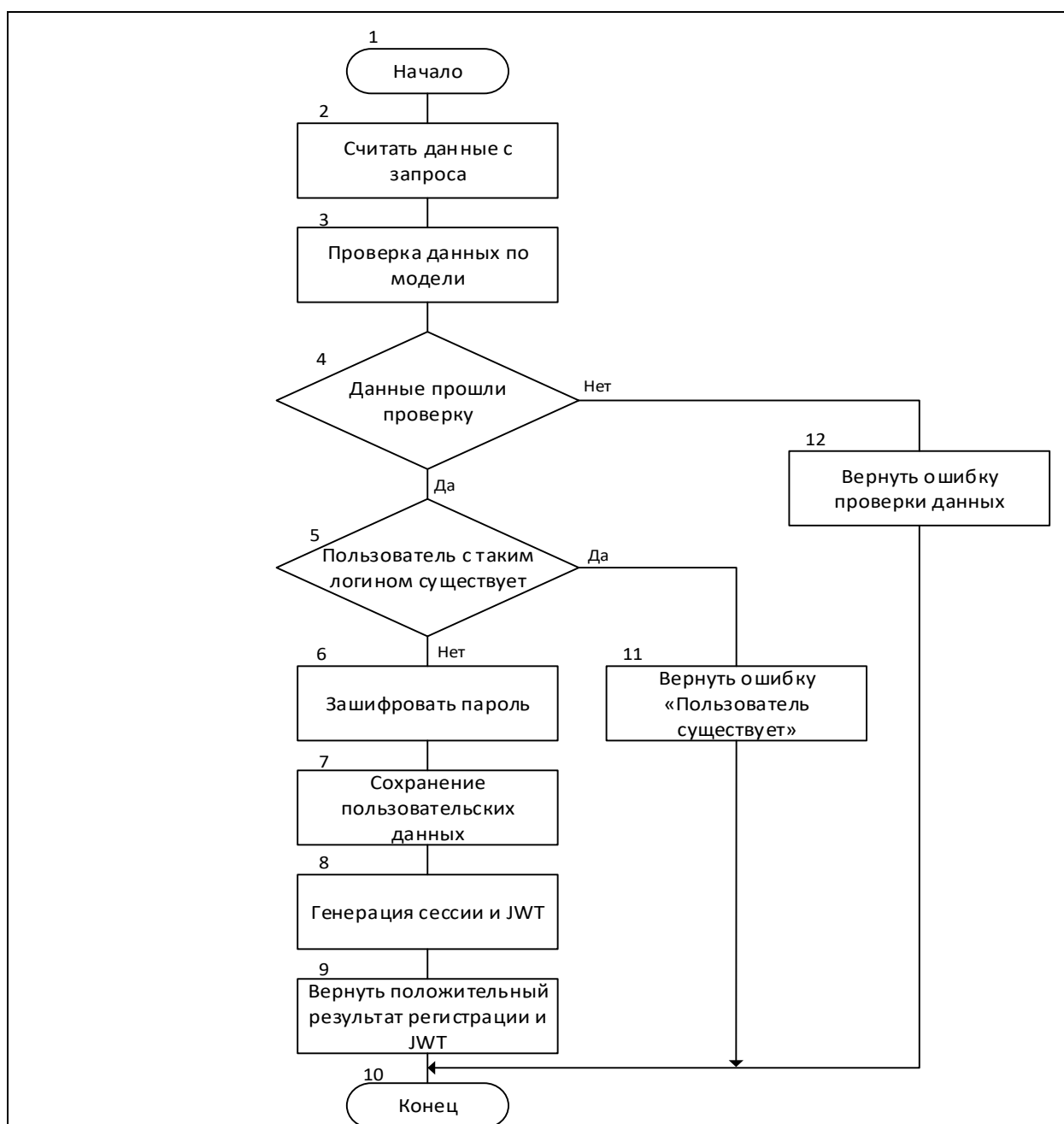


Рисунок 2.5 – Блок-схема работы алгоритма регистрации на стороне сервера

В представленном алгоритме отображена схема работы регистрации на серверном приложении, веб-приложение отправляет запрос на сервер после чего приложение считывает данные, проводит проверку данных, проверяет существует ли пользователь с таким же логином, если пользователь не существует, то приложение шифрует пароль, записывает данные в БД, генерирует сессию и JWT и возвращает данные пользователю.

2.3.2 Алгоритм проверки сессии и JWT на стороне сервера

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

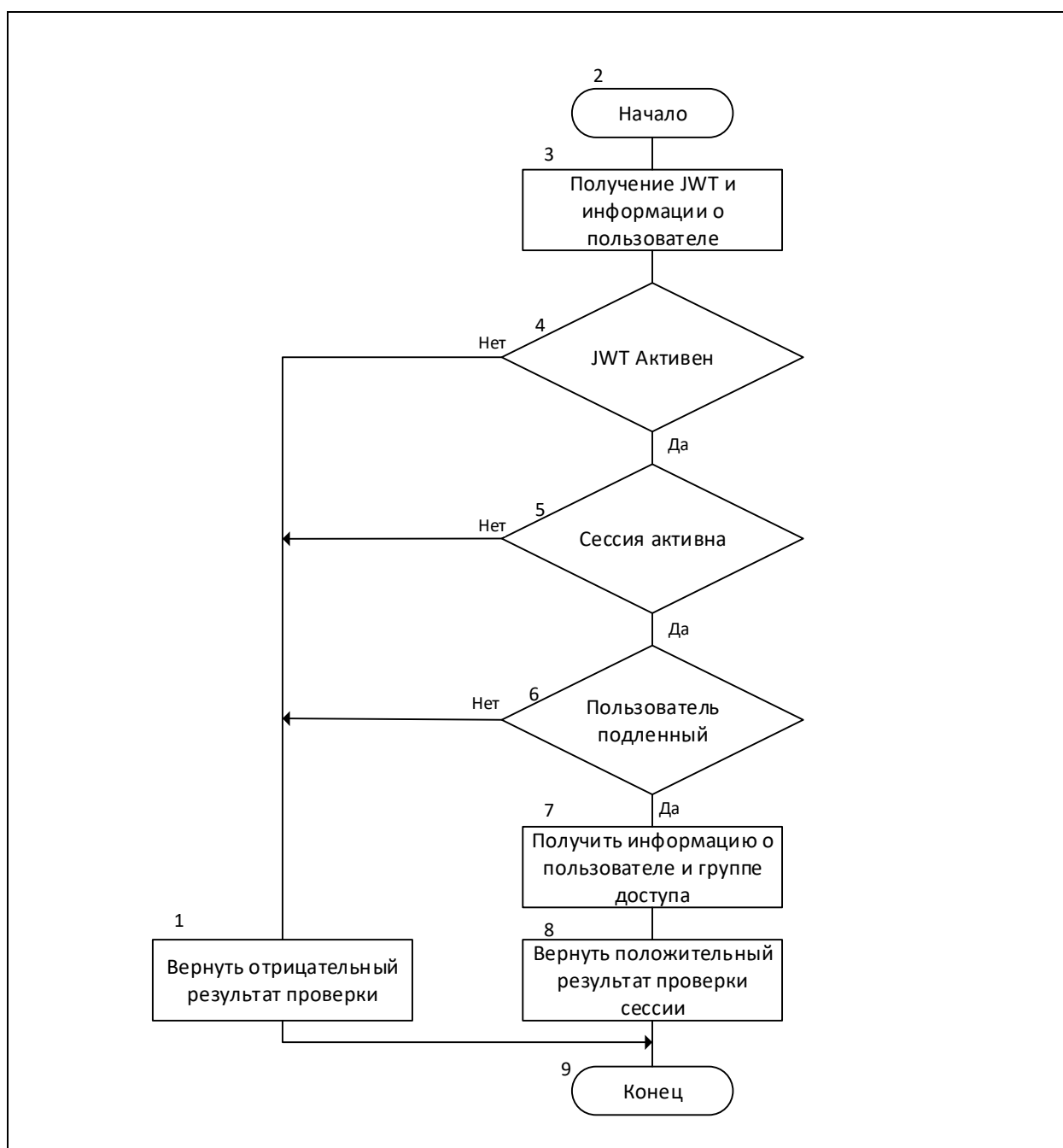


Рисунок 2.6 – Блок-схема проверки алгоритма генерации сессии и JWT

Представленная схема представляет работу модуля проверки сессии и JWT. При каждом запросе на сервер модуль автоматически проверяет есть ли у в теле запроса JWT, если есть считываем JWT и проверяем активен ли он, если активен, то получаем информацию о пользователе и правах доступа, после чего проверяется подлинный ли пользователь использует JWT, если да, то пользователю или в следующий модуль поступает информация о пользователе и о группе прав.

2.3.3 Алгоритм проверки уровня доступа на стороне сервера

Схема работы алгоритма работы проверки уровня доступа пользователя на серверном приложении представлена на рисунке 2.7.

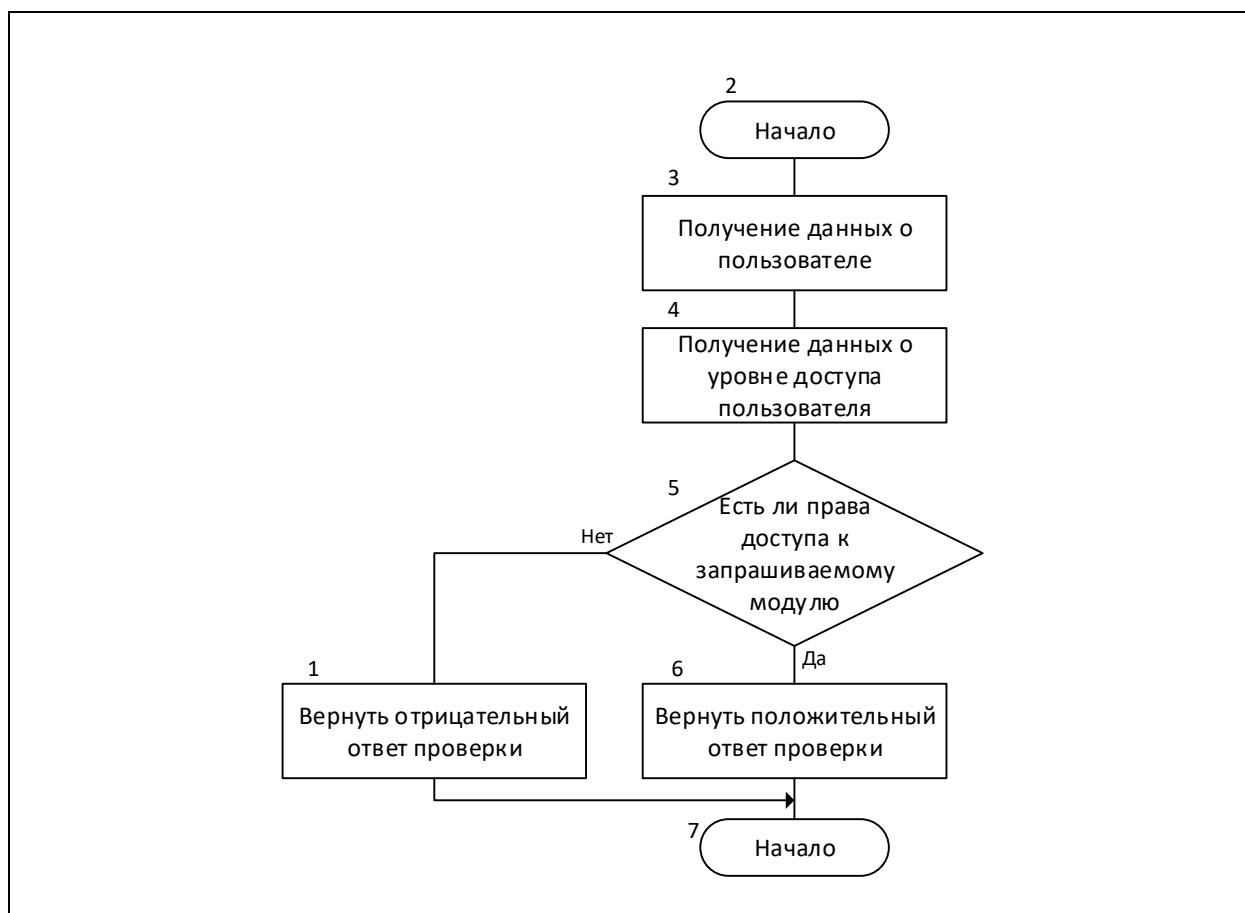


Рисунок 2.7 – Блок-схема работы проверки уровня доступа пользователя на серверном приложении

2.3.4 Алгоритм работы с модулями

Схема работы алгоритма работы с предустановленными и индивидуальными модулями представлена на рисунке 2.8.

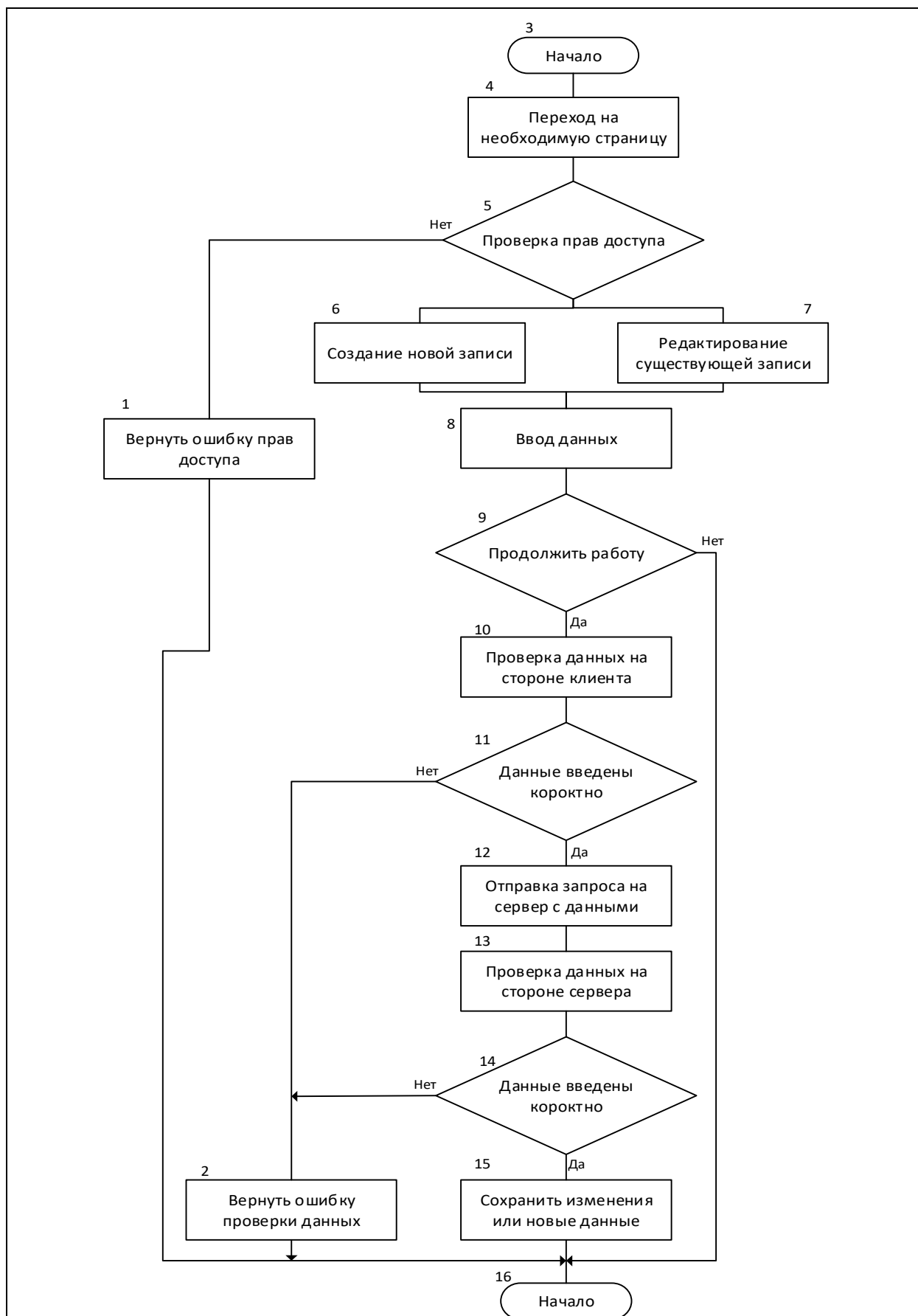


Рисунок 2.8 – Блок-схема работы с предустановленными и индивидуальными модулями

В представленном алгоритме отображена работа модулей, изначально нужно перейти на нужную страницу в веб-приложении, после чего проверятся уровень доступа и если доступ есть, то страница модуля загрузится, иначе выводится ошибка уровня доступа. После чего пользователь может добавлять, редактировать и удалять записи из модуля, после заполнения формы для записи она проходит проверку данных на стороне клиента в положительном результате данные отправятся на сервер, сервер также проведёт проверку данных и в случае успеха данные будут записаны в БД, в отрицательном результате пользователю выведет соответствующую ошибку.

В результате анализа требований к программному средству была разработана и описана функциональная модель, которая представлена в виде диаграмм А-0 и А0. Также была разработана диаграмма вариантов использования (use case) в которой описаны алгоритмы использования программного средства пользователями с разными уровнями доступа, также были описаны отдельно некоторые варианты использования. Были также реализованы и описаны схемы работы клиентского веб-приложения и алгоритмы, которые выполняются на стороне сервера. Все схемы были описаны и проиллюстрированы на рисунках.

3 ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 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 развёртывания, рисунок 3.1.

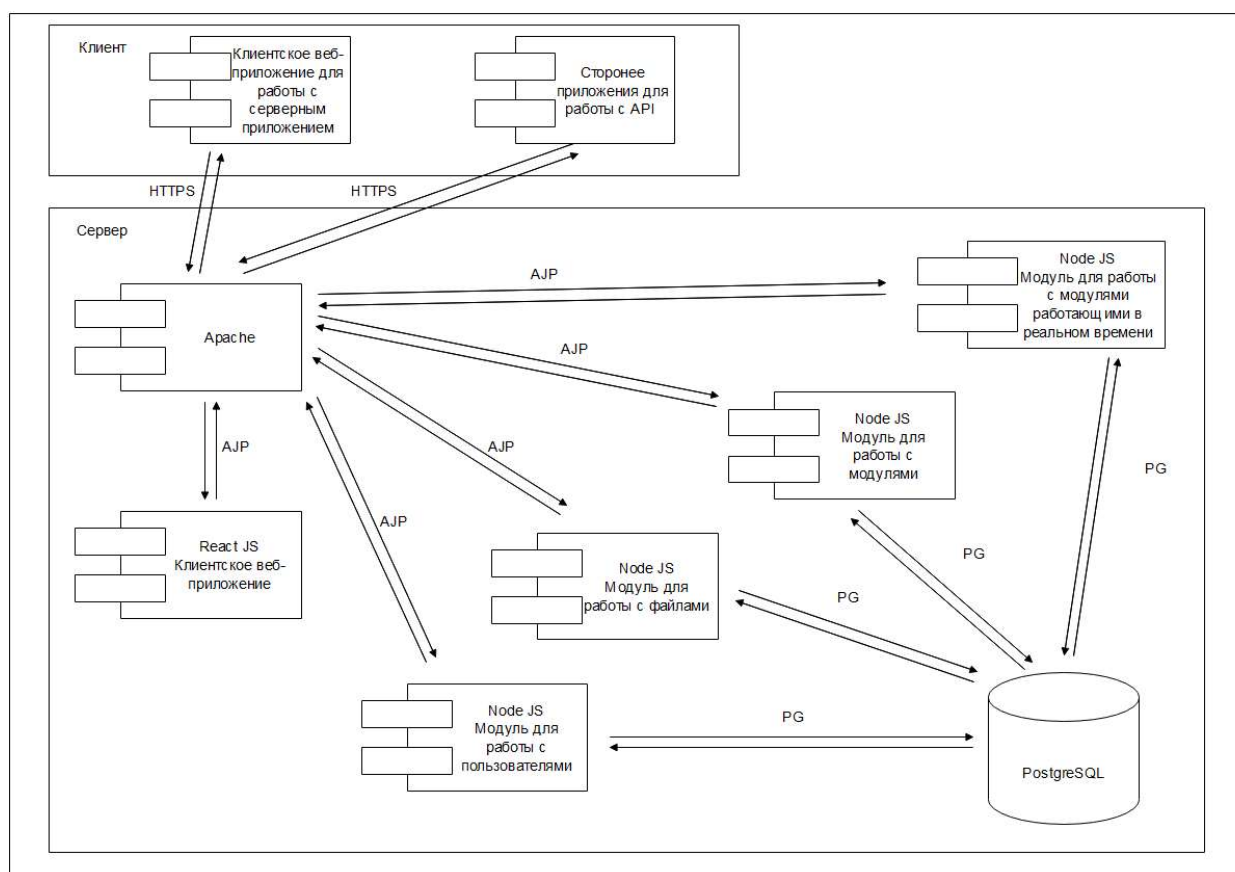


Рисунок 3.1 – Диаграмма развёртывания

Диаграммы развёртывания отображают физическое развёртывание 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 приложение 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.

3.2 Выбор решений и инструментов для разработки

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

Node JS – программная платформа, основанна на движке V8 (транслирующем JavaScript в машинный код), преобразовывает JavaScript из узко направленного языка в язык общего назначения. Node JS проста и интуитивно понятна даже для начинающих разработчиков. 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 [11].

Преимущества использования Node JS:

- простой и известный язык разработки JavaScript;
- богатая стандартная библиотека;
- большое количество 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 ответы на запросы.
- 3) cross-env – служит для определения переменных в запуске приложения;
- 4) body-parser – считывает и получает входящие данные в теле запросов в промежуточном программном обеспечении перед обработчиками;
- 5) cookie-parser – считывает и получает 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 framework подходит для создания веб-приложений, где данные могут меняться на регулярной основе [12].

Преимущества использования React JS:

- легок в изучении, ввиду простоты его синтаксиса;
- высокий уровень гибкости;
- 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;
- простая миграция между версиями, Facebook предоставляет codemods для автоматизации большей части этого процесса.

PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности. Она работает практически на всех UNIX-платформах. Ее можно применять на Windows NT Server и Windows Server, а для разработки годятся такие системы Microsoft для рабочих станций, как ME. Кроме того, PostgreSQL свободно распространяется и имеет открытый исходный код [13].

PostgreSQL отличается от многих других СУБД. Помимо того, что она обладает всеми возможностями других СУБД, а также обладает дополнительными возможностями.

Перечень функциональных возможностей PostgreSQL:

- транзакции;
- вложенные запросы;
- представления;
- ссылочная целостность - внешние ключи;
- сложные блокировки;
- типы, определяемые пользователем;
- наследственность;
- правила;
- проверка совместимости версий.

SQL – язык, который дает возможность создавать и работать в реляционных базах данных, являющихся наборами связанной информации, сохраняемой в таблицах.

Информационное пространство становится более унифицированным. Это привело к необходимости создания стандартного языка, который мог бы использоваться в большом количестве различных видов компьютерных сред. Стандартный язык позволит пользователям, знающим один набор команд, использовать их для создания, нахождения, изменения и передачи

информации - независимо от того, работают ли они на персональном компьютере, сетевой рабочей станции, или на универсальной ЭВМ [14].

3.3 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 [15].

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:M);
- многие–ко–многим (M:M).

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 пользователя», «логин», «хеш пароля», «хеш соли», «дата регистрации», «активный хеш ключ», «текст статуса», «статус»;
- таблица «profiles»: «код профиля», «код пользователя», «имя», «фамилия», «дата рождения», «код картинки пользователя»;
- таблица «level_access»: «код уровня доступа», «название», «заголовок», «описание», «статус»;
- таблица «users_access»: «код уровня доступа пользователя», «код пользователя», «код уровня доступа»;
- таблица «module_access»: «код уровня доступа модуля», «код

пользователя», «код модуля»;

- таблица «image_base»: «код картинки», «код пользователя», «оригинальное название», «хеш названия», «путь к картинке», «дата загрузки», «статус»;

- таблица «load_images»: «код загрузки картинки», «код картинки», «код пользователя», «опциональные настройки»;

- таблица «users_images»: «код пользователя картинки», «код пользователя», «код картинки», «опция»;

- таблица «files_base»: «код файла», «код пользователя», «оригинальное название», «хеш названия», «путь к файлу», «дата загрузки», «статус»;

- таблица «load_files»: «код загрузки файла», «код файла», «код пользователя», «опциональные настройки»;

- таблица «users_files»: «код пользователя файла», «код пользователя», «код файла», «опция»;

- таблица «type_variables»: «код типа переменной», «название», «заголовок», «описание», «статус»;

- таблица «schema_variables»: «код схемы переменной», «название», «заголовок», «описание», «схема», «статус»;

- таблица «variables»: «код переменной», «код схемы переменной», «название», «заголовок», «описание», «контент», «статус»;

- таблица «type_modules»: «код типа модуля», «название», «заголовок», «описание», «статус»;

- таблица «modules»: «код модуля», «код типа модуля», «название», «заголовок», «описание», «шаблон модуля», «схема шаблона модуля», «контент», «настройки», «статус»;

- таблица «rows_modules»: «код строки модуля», «код модуля», «дата создания», «дата обновления», «код пользователя», «статус»;

- таблица «content_rows_module»: «код контента строки модуля», «код строки модуля», «код переменной», «контент».

Диаграмма сущность–связь будет представлена на рисунке 3.2.

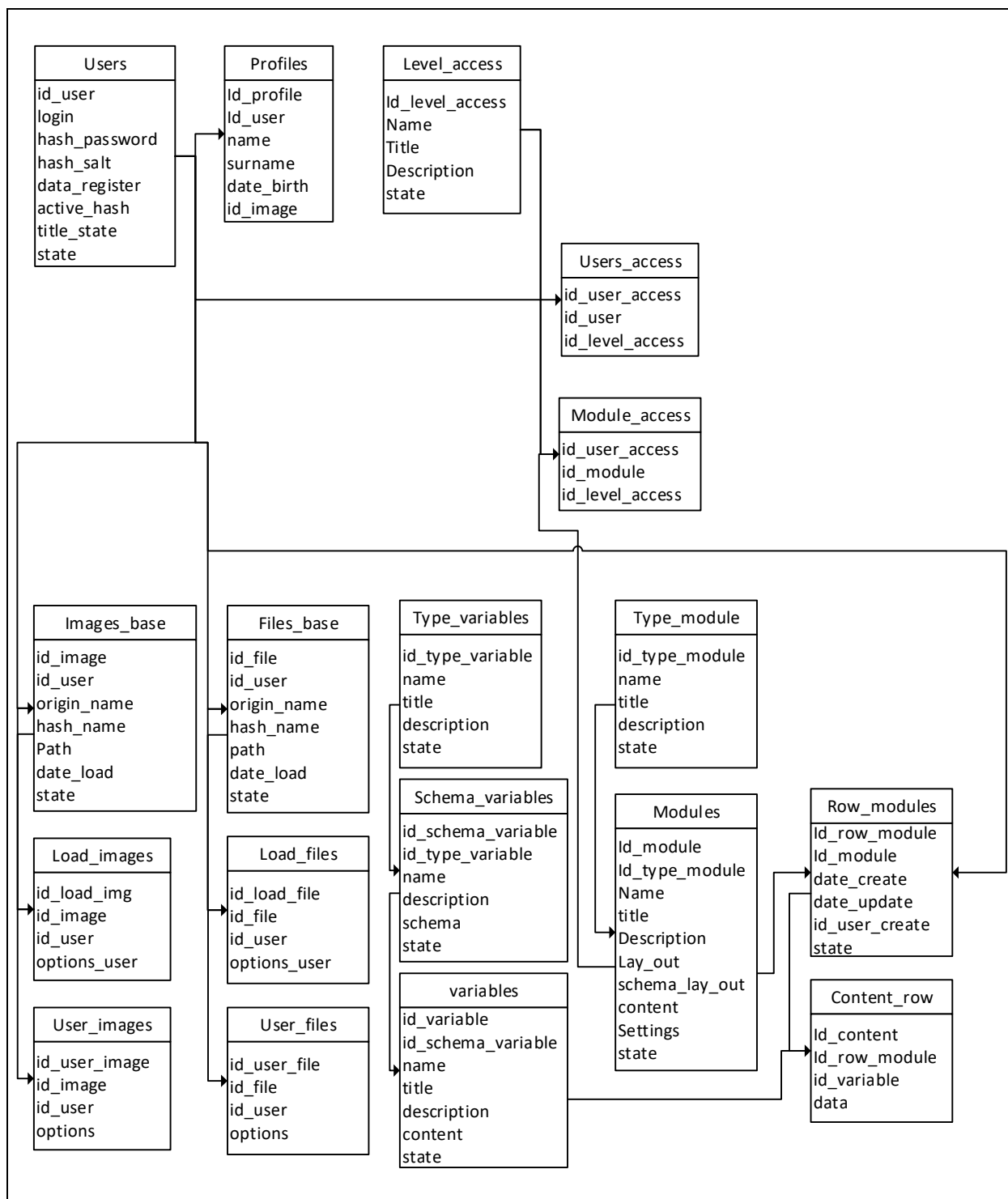


Рисунок 3.2 – Диаграмма сущность–связь

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.

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- 3.18.

Таблица «users» содержит информацию о пользователях.

Таблица 3.1 – Структура таблицы «users»

Поле	Тип данных	Источник данных	Описание
id_user	INT(8)	Заполняется автоматически	Ключевое поле
login	VARCHAR(255)	Заполняется пользователем	Логин пользователя
hash_password	TEXT	Генерируется алгоритмом шифрования	Зашифрованный пароль пользователя
hash_salt	TEXT	Генерируется алгоритмом шифрования	Индивидуальный ключ для пароля пользователя
date_registration	DATE	Заполняется автоматически	Дата регистрации пользователя
active_hash	TEXT	Заполняется автоматически	Активный ключ хеш, служит для восстановления пароля
title_state	TEXT	Заполняется пользователем или автоматически	Причина блокировки пользователя
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_profile	INT(8)	Заполняется автоматически	Ключевое поле
id_user	INT(8)	Заполняется автоматически	Код пользователя
name	VARCHAR(255)	Заполняется пользователем	Имя пользователя
surname	VARCHAR(255)	Заполняется пользователем	Фамилия пользователя
date_birth	DATE	Заполняется пользователем	Дата рождения пользователя
id_image	INT(8)	Заполняется автоматически	Код картинки пользователя

Таблица «level_access» содержит информацию об уровнях доступа.

Таблица 3.3 – Структура таблицы «level_access»

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

Таблица «users_level_access» содержит информацию об уровнях доступа.

Таблица 3.4 – Структура таблицы «users_level_access»

Поле	Тип данных	Источник данных	Описание
id_user_access	INT(8)	Заполняется автоматически	Ключевое поле
id_user	INT(8)	Заполняется пользователем	Код пользователя
id_level_access	INT(8)	Заполняется пользователем	Код уровня доступа

Таблица «modules_level_access» содержит информацию об уровнях доступа.

Таблица 3.5 – Структура таблицы «modules_level_access»

Поле	Тип данных	Источник данных	Описание
id_module_accesses	INT(8)	Заполняется автоматически	Ключевое поле
id_module	INT(8)	Заполняется пользователем	Код модуля
id_level_access	INT(8)	Заполняется пользователем	Код уровня доступа

Таблица «images_base» содержит информацию о картинках.

Таблица 3.6 – 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	TEXT	Заполняется автоматически	Оригинальное название картинки
hash_name	TEXT	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 картинок.

Таблица 3.7 – 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	JSON	Заполняется автоматически	Опциональные настройки пользователя
date_load	DATE	Заполняется автоматически	Дата просмотра

Таблица «user_images» содержит информацию о картинках который задействованы в профиле пользователя.

Таблица 3.8 – 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
options	JSON	Заполняется пользователем	опции используемой картинки

Таблица «files_base» содержит информацию о файлах.

Таблица 3.9 – 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

Продолжение таблицы 3.9

Поле	Тип данных	Источник данных	Описание
origin_name	TEXT	Заполняется автоматически	Оригинальное название картинки
hash_name	TEXT	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 3.10 – 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	JSON	Заполняется автоматически	Опциональные настройки пользователя
date_load	DATE	Заполняется автоматически	Дата просмотра

Таблица «user_files» содержит информацию о файлах который задействованы в профиле пользователя.

Таблица 3.11 – Структура таблицы «user_images»

Поле	Тип данных	Источник данных	Описание
id	INT(8)	Заполняется автоматически	Ключевое поле
id_file	INT(8)	Заполняется автоматически	Код файла

Продолжение таблицы 3.11

Поле	Тип данных	Источник данных	Описание
id_user	INT(8)	Заполняется автоматически	Код пользователя
options	JSON	Заполняется пользователем	опции используемой файла

Таблица «type_variables» содержит информацию о типах переменных.

Таблица 3.12 – 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	TEXT	Заполняется пользователем	Описание типа переменной
state	BOOL	Заполняется пользователем	Статус типа переменной

Таблица «schema_variables» содержит информацию о схемах переменных.

Таблица 3.13 – 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	TEXT	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 3.14 – 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	TEXT	Заполняется пользователем	Заголовок переменной
description	TEXT	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 3.15 – 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	TEXT	Заполняется пользователем	Заголовок типа модуля
description	TEXT	Заполняется пользователем	Описание типа модуля
state	BOOL	Заполняется пользователем	Статус типа модуля

Таблица «modules» содержит информацию о модулях.

Таблица 3.16 – Структура таблицы «modules»

Поле	Тип данных	Источник данных	Описание
id_module	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
settings	JSON	Заполняется пользователем	Настройки модуля
state	BOOL	Заполняется пользователем	Статус модуля

Таблица «rows_modules» содержит информацию о строках в модулях.

Таблица 3.17 – Структура таблицы «rows_modules»

Поле	Тип данных	Источник данных	Описание
id_row_module	INT(8)	Заполняется автоматически	Ключевое поле
id_module	INT(8)	Заполняется автоматически	Код модуля
date_create	DATE	Заполняется автоматически	Код переменной
date_update	DATE	Заполняется автоматически	Содержание переменной
id_user	INT(8)	Заполняется автоматически	Код пользователя
state	BOOL	Заполняется пользователем	Статус переменной

Таблица «content_rows» содержит информацию данных в строке модуля.

Таблица 3.18 – Структура таблицы «content_rows»

Поле	Тип данных	Источник данных	Описание
id	INT(8)	Заполняется автоматически	Ключевое поле
id_row_module	INT(8)	Заполняется автоматически	Код строки модуля
id_variable	INT(8)	Заполняется автоматически	Код переменной
data	JSON	Заполняется пользователем	Содержание переменной

В ходе выполнения технического проектирования ПС было разработана диаграмма развёртывания, которая описывает архитектуру развёрнутого программного средства на сервере. Также выбраны и обоснованы решения для реализации программного средства. Разработана модель данных которая описывает структуру используемой базы данных, описаны все таблицы с полями, каждое поле в описании имеет следующие характеристики: название поля, тип данных, источник данных, описании. Также реализована диаграмма сущность-связь.

4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1 Описание тестируемого стенда

Для проведения тестирования использовался сервер на ОС Ubuntu 18.04.3 LTS x86_64, характеристики сервера:

- ЦП: Intel(R) Core(TM) i3-4130 CPU @ 3.40GHz;
- ОЗУ: 8 GB;
- ПЗУ: 60 GB SSD;
- подключение к сети Интернет с пропускной способностью: 25/12,5 Мбит/с

4.2 Функциональное тестирование

Для того, чтобы удостовериться в правильности работы программного средства и реализации функциональных требований, было проведено тестирование модулей, которые были разработаны.

В процессе разработки проводилось модульное тестирование, которое помогало выявлять ошибки на ранних этапах, а также позволяли максимально полно проверить все основные модули программного средства.

По мере разработки программного средства, одного модульного тестирования мало и был выбран метод функционального тестирования.

Функциональное тестирование является важным аспектом при тестировании программного средства, так-как каждая функция программы тестируется исходя из поведения реальных пользователей, а также полный проход по всему бизнес сценарию. Поскольку функциональное тестирование не позволяет полностью проверить функцию по всей области её определения, поэтому каждая функция проверяется по основному сценарию работы программного средства [16].

К плюсам данного метода можно отнести:

- обнаружение ошибок интерфейса;
- обнаружение не работающих функций;
- обнаружение некорректного поведения функций;
- имитация поведения реального пользователя;
- корректность внешней структуры данных и отсутствие ошибок.

Из минусов является следующие:

- избыточное тестирование;
- вероятность пропуска логических ошибок.

Функциональное тестирование было реализовано при помощи набора тест-кейсов, собранных в тестовый сценарий, представленный в таблице 4.1

Таблица 4.1 – Результаты функционального тестирования

№	Тестовый случай	Ожидаемый результат	Фактический результат
1	1) Первое открытие клиентского приложения 2) Регистрация супер-пользователя 3) Успешный вход в аккаунт супер-пользователя	Форма регистрации открыта, пользователь успешно зарегистрирован, открыта главная страница клиентского приложения.	Совпадает с ожидаемым
2	1) Открытие клиентского приложения 2) Открытие формы регистрации 3) Ввод данных 4) Нажать «Регистрация»	Клиентское приложение запущено, форма регистрации открыта, пользователь зарегистрирован, открыта главная страница клиентского приложения.	Совпадает с ожидаемым
3	1) Открытие клиентского приложения 2) Открытие формы авторизации 3) Ввод данных 4) Нажать «Вход»	Клиентское приложение запущено, форма авторизации открыта, пользователь авторизован, открыта главная страница клиентского приложения.	Совпадает с ожидаемым
4	1) Переход на страницу настроек приложения 2) Редактирование существующей переменной 3) Ввод данных 4) Нажать «сохранить»	Открытие страницы настроек приложения, редактирование существующих переменных, сохранение изменений, переход на страницу настроек.	Совпадает с ожидаемым

Продолжение таблицы 4.1

№	Тестовый случай	Ожидаемый результат	Фактический результат
5	1) Переход на страницу пользователя 2) Выбор необходимого пользователя 3) Блокировка пользователя 4) Ввод причины блокировки 5) Авторизация данными заблокированного пользователя 6) Нажать «Вход»	Открытие страницы списка пользователей, открытие необходимого пользователя, блокировка пользователя, ввод причины блокировки, открыть клиентское приложение в другом браузере, авторизоваться под заблокированным пользователем, вывод ошибки пользователь заблокирован и причину блокировки.	Совпадает с ожидаемым
6	1) Переход на страницу профиля 2) Редактирование информации 3) Нажать «сохранить» 4) Просмотр информации пользователя другим пользователем	Открытие страницы профиля, редактирование пользовательской информации, данные сохранены, переход на страницу пользователя.	Совпадает с ожидаемым
7	1) Переход на страницу модуля 2) Добавление записи модуля 3) Нажать сохранить 4) Просмотреть запись другим пользователем	Переход на страницу модуля, добавление новой записи модуля, запись данных, переход на страницу записи модуля.	Совпадает с ожидаемым
8	1) Переход на страницу пользователя 2) Выбрать необходимого пользователя 3) Поднять уровень доступа пользователя 4) Нажать «сохранить» 5) Авторизовать под данным пользователей 6) Проверить уровень доступа	Открыть страницу списка пользователей, выбор необходимого пользователя, поднятие уровня доступа, сохранение изменений, авторизация под данным пользователем, проверка уровня доступа.	Совпадает с ожидаемым

Продолжение таблицы 4.1

№	Тестовый случай	Ожидаемый результат	Фактический результат
9	1) Переход на страницу файлов 2) Добавление нового файла 3) Нажать «Сохранить» 4) Проверить доступность файла	Открытие страницу файлов, загрузить файл, проверка доступности файла.	Совпадает с ожидаемым
10	1) Переход на не существующую страницу 2) Вывод ошибки «404: страница не найдена»	Ввести в адрес клиентского приложения не существующую ссылку на страницу, вывод ошибки на экран	Совпадает с ожидаемым
11	1) Переход на страницу авторизации 2) Ввести не правильны пароль 3) Нажать «войти» 4) Повторить 2 и 3 пункт 3 раза 5) Вывод ошибки пользователь заблокирован	Открыть страницу авторизации, провести 3 попытки авторизации с не правильным паролем, вывод ошибки на экран «пользователь заблокирован»	Совпадает с ожидаемым
12	1) Переход на страницу метки 2) Добавить новую метку 3) Заполнить все поля кроме названия 4) Нажать «добавить» 5) Вывод ошибки проверки данных	Открыть страницу метки, заполнить все поля кроме поля название, нажать «добавит», вывод ошибки проверки данных	Совпадает с ожидаемым
13	1) Переход на страницу к которой нет прав доступа 2) Вывод ошибки «нет прав доступа»	Открыть страницу к которой у данного пользователя нет прав доступа, вывод на экран ошибки «У вас нет доступа к данной странице»	Совпадает с ожидаемым

Продолжение таблицы 4.1

№	Тестовый случай	Ожидаемый результат	Фактический результат
14	1) Переход на несуществующую страницу за счёт ввода в адрес клиентского приложения несуществующей страницы 2) Вывод ошибки «Страница не найдена»	Открыть клиентское приложение, ввести в адресную строку несуществующую страницу, вывод на экран ошибки «страница не найдена»	Совпадает с ожидаемым
15	1) Переход на страницу иконки 2) Выбор необходимой категории иконок 3) Ввести в поиск название искомой иконки 4) Выбрать нужную иконку 5) Скопировать кодовое название иконки и использовать для настроек модуля 6) Переход на страницу настроек модулей 7) Выбор необходимого модуля 8) Изменение иконки модуля 9) Нажать «Сохранить»	Открыть страницу иконок, выбор необходимой категории, ввести в поиск ключевое слово, нажать на необходимую иконку, скопировать кодовое название, перейти на страницу настроек необходимого модуля, в настройках модуля вставить кодовое название иконки в соответствующее поле, сохранить, изменить иконку модуля в панели меню.	Совпадает с ожидаемым
16	1) Переход на страницу компонентов 2) Выбор необходимого типа переменных 3) Просмотр как работает переменная	Переход на страницу компонентов, выбор необходимого типа переменной, просмотреть как работает переменная.	Совпадает с ожидаемым
17	1) Переход на страницу профиля 2) Нажать «Редактировать» 3) Изменить пользовательские данные 4) Нажать «Сохранить» 5) Проверить изменились ли данные пользователя	Переход на страницу профиля, переход в режим редактирования, редактирование данных профиля пользователя, сохранение данных, проверка изменились ли данные на странице пользователя.	Совпадает с ожидаемым

Продолжение таблицы 4.1

№	Тестовый случай	Ожидаемый результат	Фактический результат
18	1) Переход на страницу настроек 2) Открыть настройки внешнего клиентского приложения 3) Изменить настройки прокси сервера для доступа к API 4) Нажать «Сохранить» 5) Проверить доступность API из внешнего приложения	Переход на страницу настроек, открыть страницу настроек внешнего клиентского приложения, изменить настройки прокси сервера, проверка доступа к API из внешнего клиентского приложения	Совпадает с ожидаемым

4.3 Тестирование производительности

Тестирование производительности – это комплекс типов тестирования, целью которого является определение работоспособности, стабильности, потребления ресурсов и других атрибутов качества приложения в условиях различных сценариев использования и нагрузок. Тестирование производительности позволяет находить возможные уязвимости и недостатки в системе с целью предотвратить их пагубное влияние на работу программы в условиях использования. Необходимые параметры работы системы в определенной среде можно тестировать с помощью:

- определения рабочего количества пользователей приложения;
- измерение времени выполнения различных операций системы;
- определения производительности приложения при различных степенях нагрузки;
- определения допустимых границ производительности программы при разных уровнях нагрузки [17].

Нагрузочное тестирование – тестирование времени отклика приложения на запросы различных типов, с целью удостовериться, что приложение работает в соответствии с требованиями при обычной пользовательской нагрузке [18].

Для проведения тестирования реализован тестовый чат, к которому будут подключены виртуальные пользователи. С помощью утилиты Apache Benchmark с другой машины подключаются несколько Постоянных HTTP-соединений клиентов (время тестирования 20 секунд). В это время сторонней формой начинается запись в БД сообщения. Для каждого числа Постоянное HTTP-соединение соединений настроено по 5 итераций, полученные данные усредняем и полученный результат внесен в таблицу 4.2.

Таблица 4.2 - Результаты нагрузочного тестирования

Количество запросов	Запросов в секунду	Среднее время на запрос в ms (по всем одновременным запросам)	Минимальное время выполнения запроса в ms	Максимальное время выполнения запроса в ms
10 пользователей				
1285	64	15.618	16	9064
100 пользователей				
1284	64.12	15.765	17	13347
500 пользователей				
1236	64.71	16.611	23	16078
1000 пользователей				
1528	75.02	13.354	17	16785

4.4 Примеры ошибок

В результате тестирования программного средства было выявлено несколько ошибок в работе API, все ошибки на стороне клиентского приложения обработаны и выводятся соответствующие уведомления.

Ниже представлены часто возникающие ошибки на стороне клиента:

Ошибка, возникающая при не правильном вводе данных или не заполнении обязательных полей, ошибка представлена на рисунке 4.1

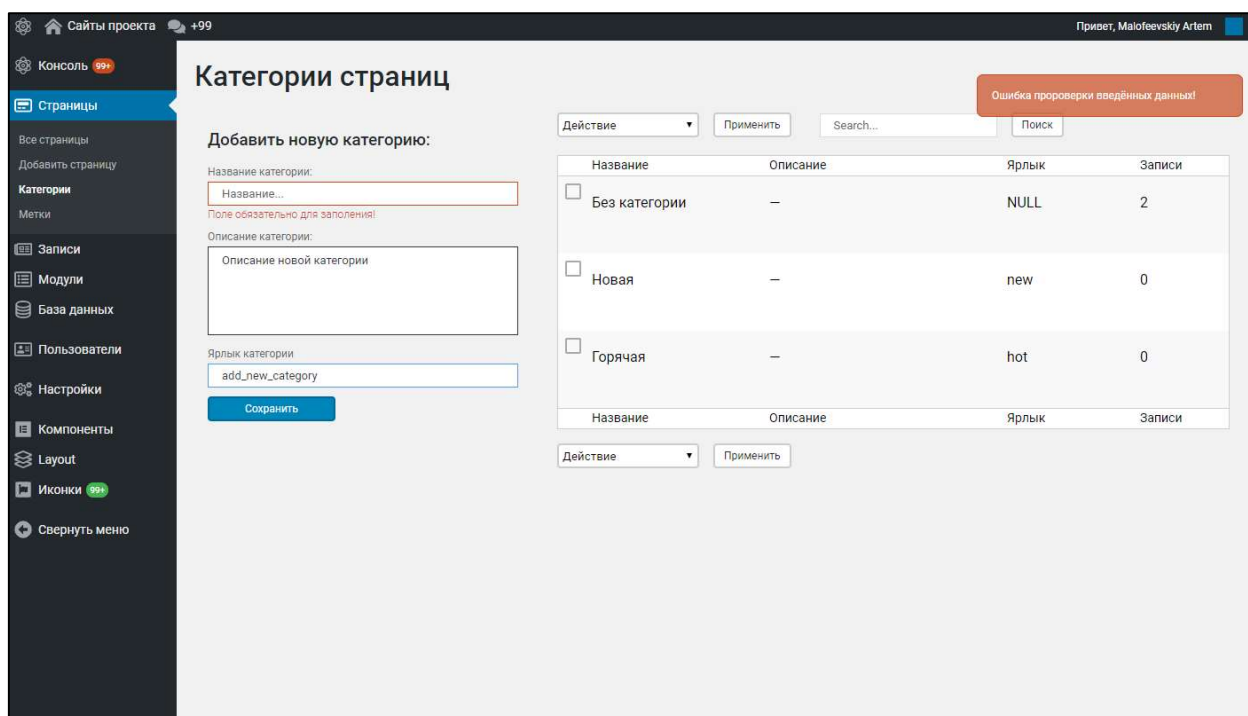


Рисунок 4.1 – Ошибка проверки введенных данных

Ошибка, возникающая если пользователь пытается открыть раздел, к которому у его нет доступа, ошибка представлена на рисунке 4.2.

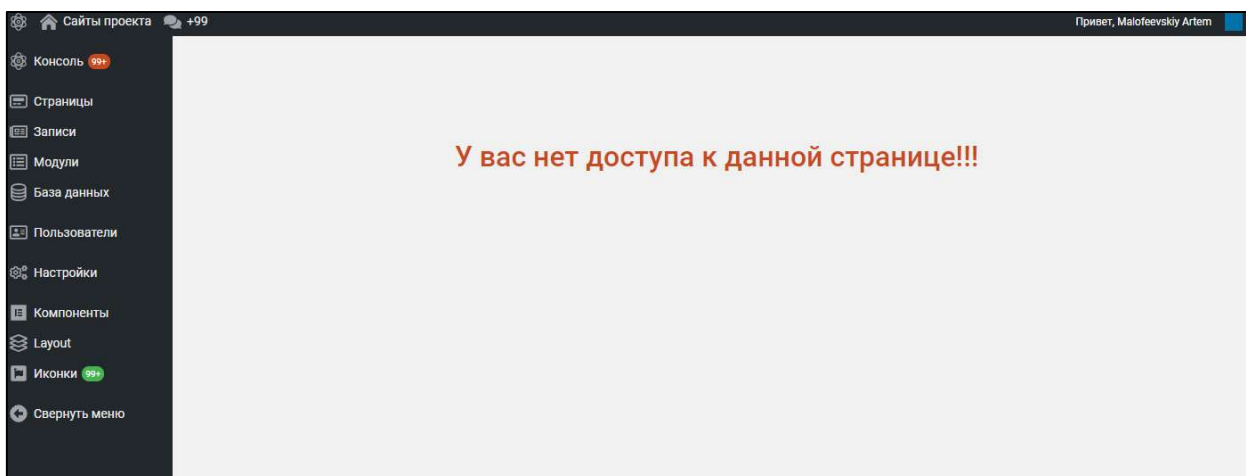


Рисунок 4.2 – Ошибка прав доступа к странице

Ошибка, возникающая если пользователь заблокирован, пользователь может быть заблокирован в 2х случаях если его заблокирует пользователь с уровнем выше данного пользователя или заблокирует система если будет введено более 3х попыток ввода пароля, ошибка представлена на рисунке 4.3.

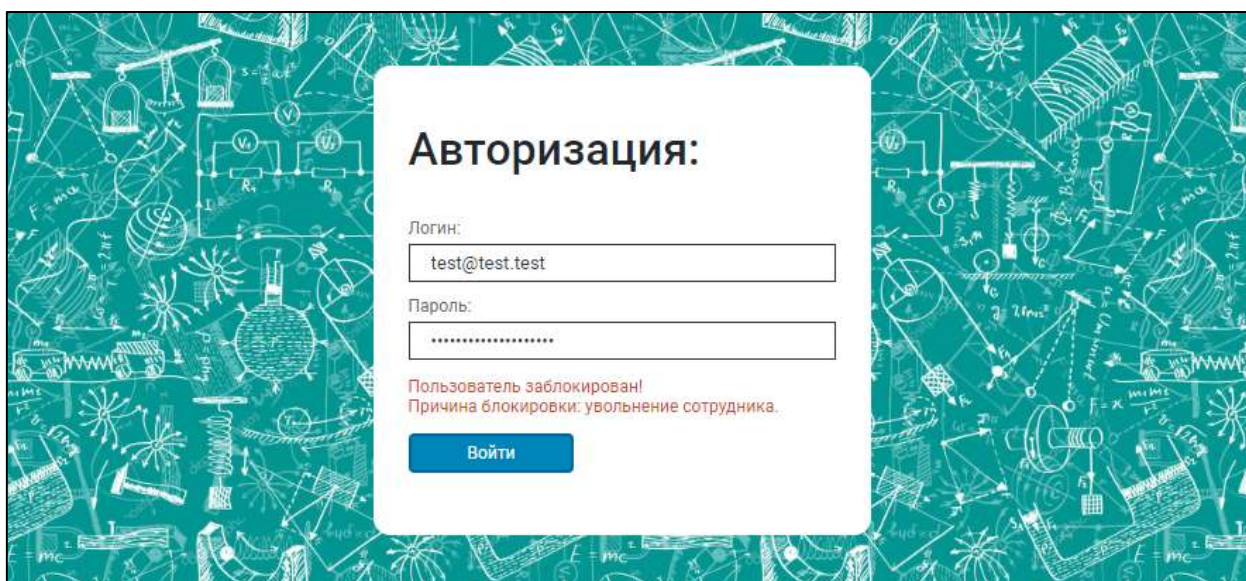


Рисунок 4.3 – Ошибка пользователь заблокирован

Ошибка, возникающая если пользователь пытается перейти на не существующую страницу, текст ошибки «404: страница не найдена!», ошибка представлена на рисунке 4.4. Если страница ранее существовала, но ее удалили, то в тексте ошибки будет «Страница удалена!», данная ошибка представлена на рисунке 4.5.

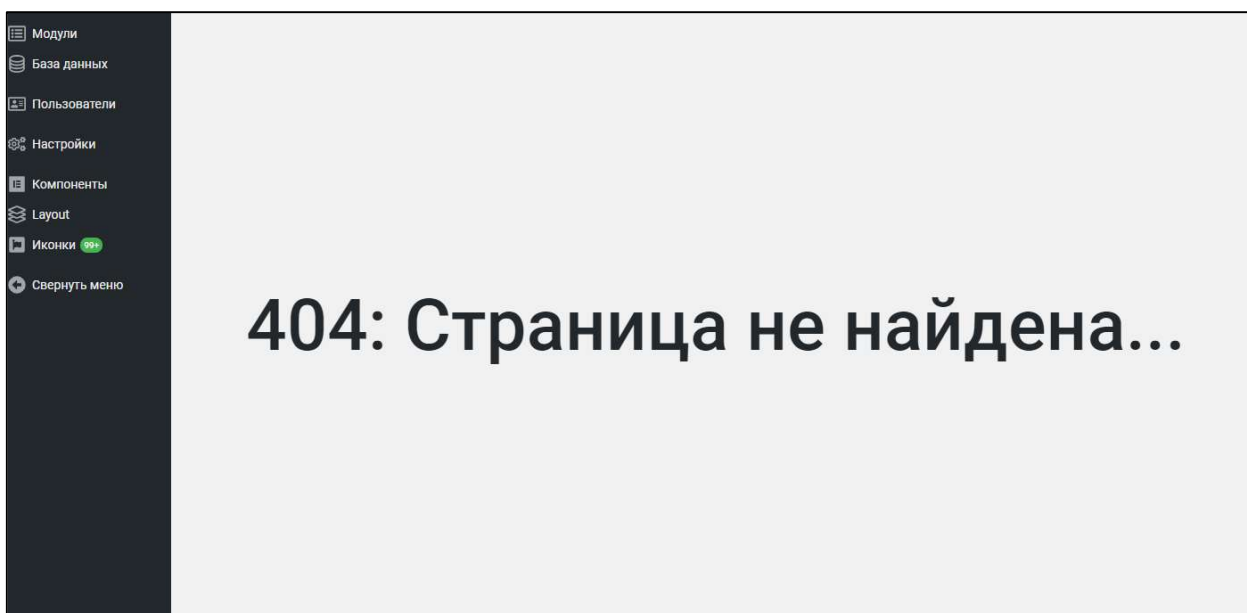


Рисунок 4.4 – Ошибка страница не найдена

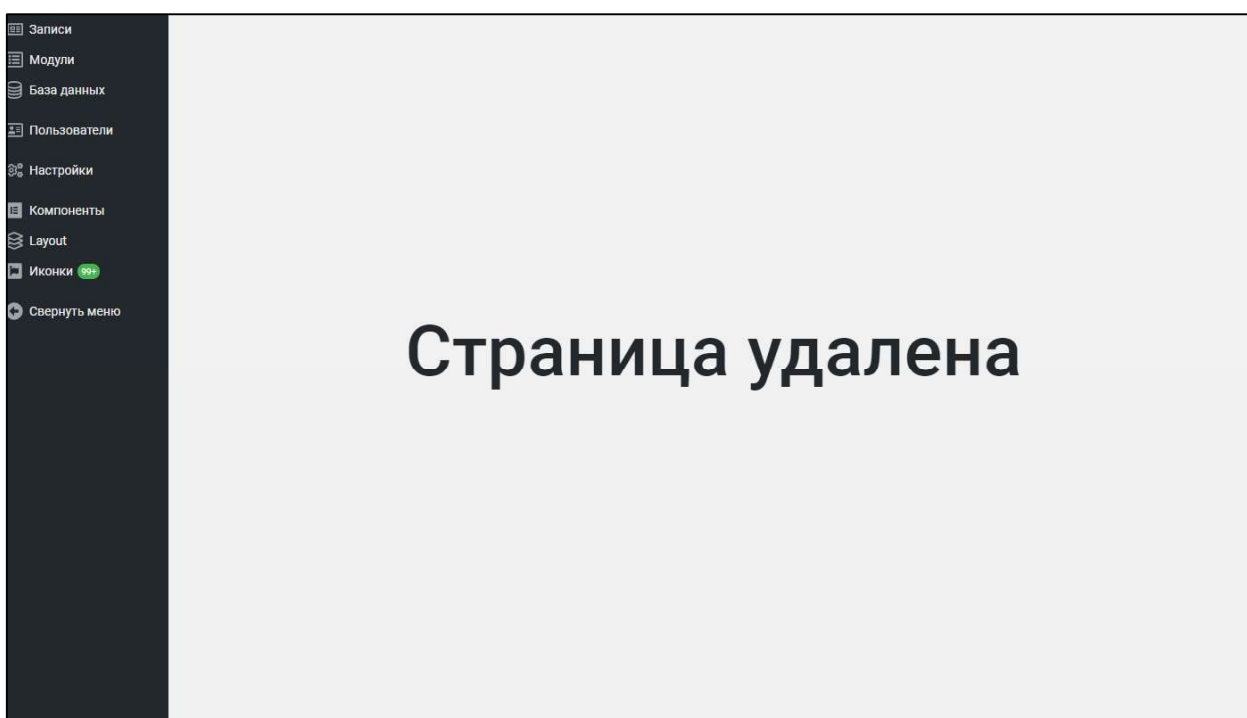


Рисунок 4.5 – Ошибка страница удалена

Ошибка, возникающая если сервер, не обработал запрос, превышен интервал ожидания, или другая иная проблема будет выведена ошибка с текстом «Проблемы на стороне сервера», ошибка представлена на рисунке 4.6.

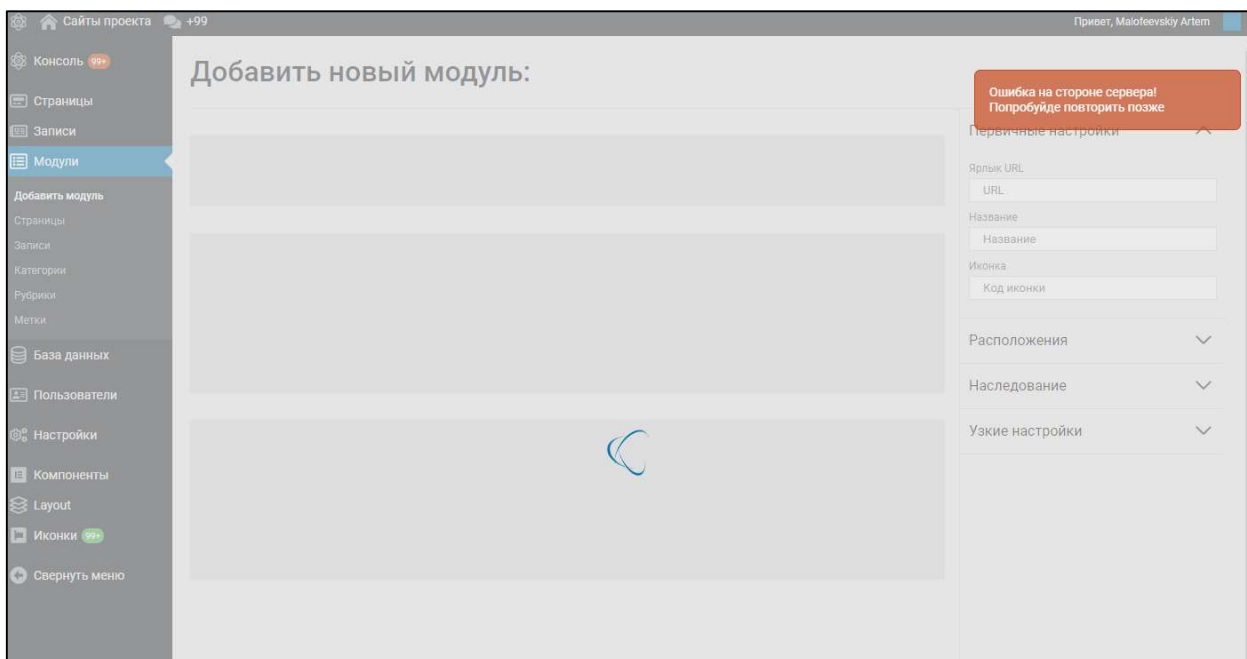


Рисунок 4.6 – Ошибка на стороне сервера

Ошибка, возникающая если пользователь не авторизован, но пытается открыть страницу по ссылке, ошибка отображена на рисунке 4.7.

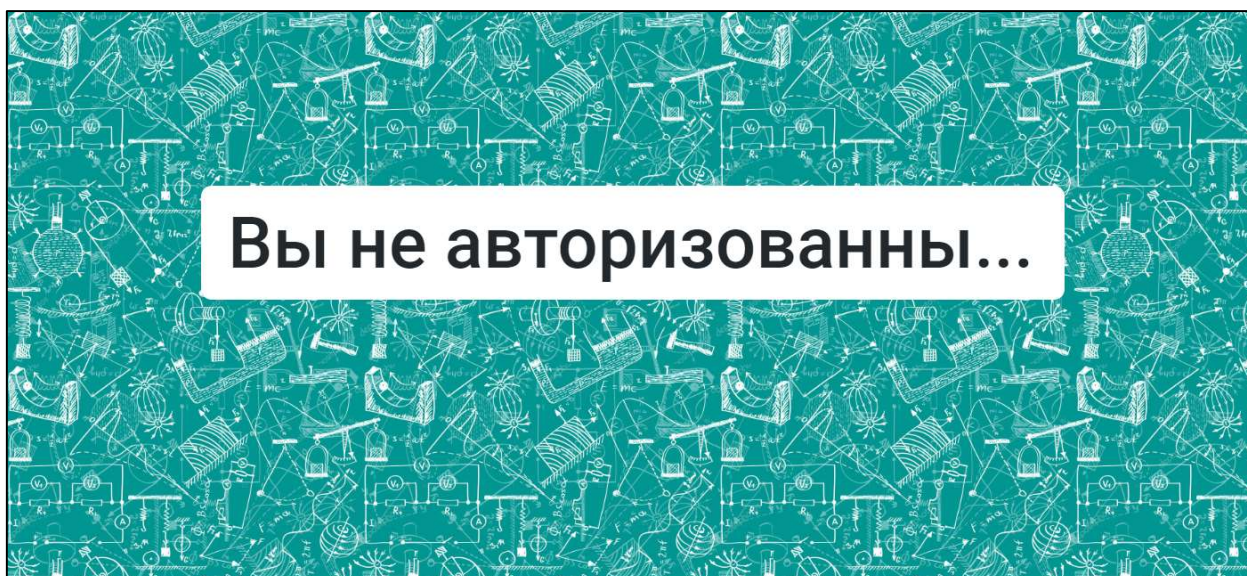


Рисунок 4.7 – Ошибка, не авторизованный пользователь

Ошибка, возникающая в случае неправильной компиляции динамических страниц, ошибка представлена на рисунке 4.8.



Рисунок 4.8 – Ошибка компиляции динамических страниц

Ошибка, возникающая при не верном API запросе с внешнего клиента к серверу, ошибка представлена на рисунке 4.9.

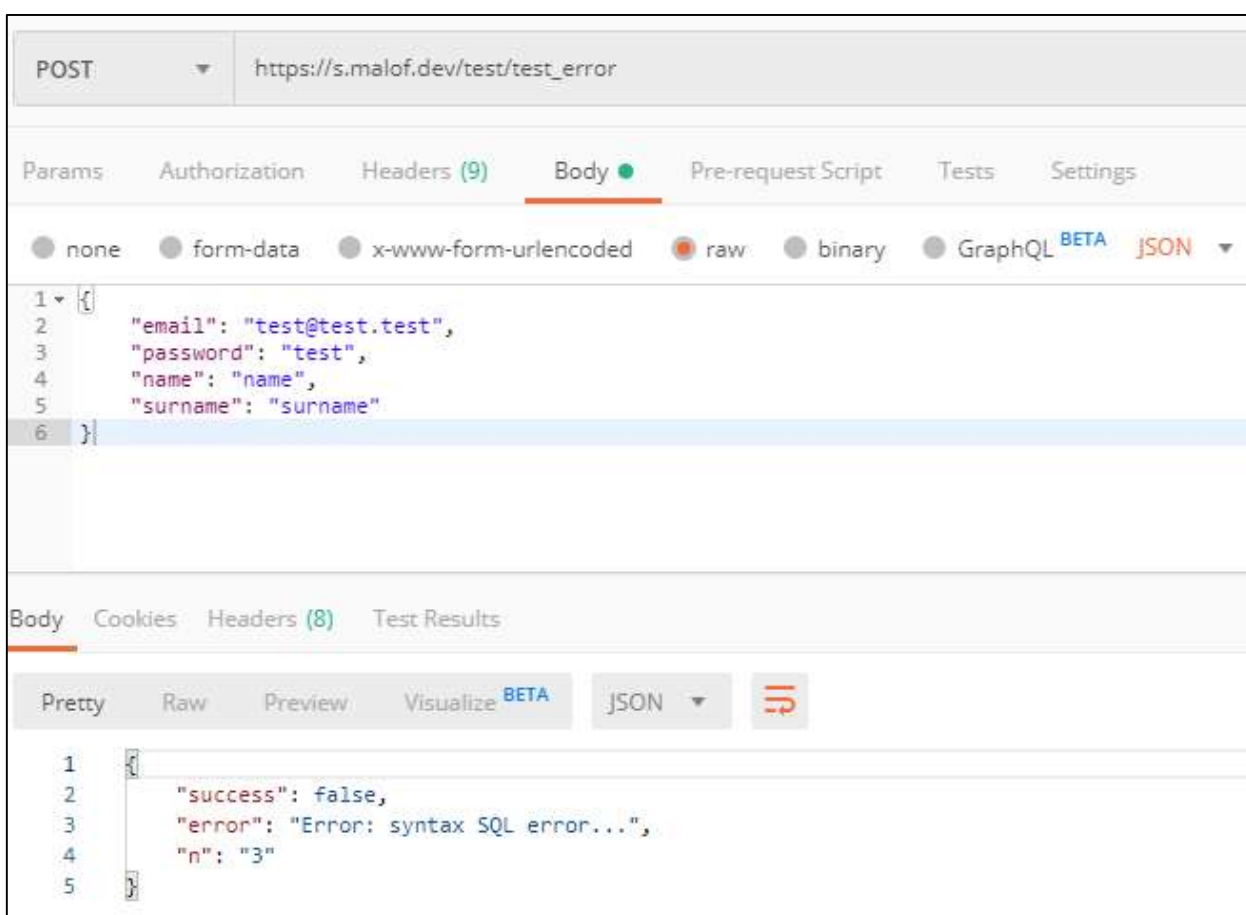


Рисунок 4.9 – Ошибка API из внешнего клиентского приложения

В результате проведённого тестирования программного средства были использованы методы функционального тестирования и тестирование производительности. Были описаны все используемые тест-кейсы в функциональном тестировании, также представлена таблица с результатами нагрузочного тестирования. Было описано большинство возникающих ошибок при работе с клиентским приложением.

5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для использования программного средства необходимо использовать выделенный или виртуальный сервер, операционная система может использоваться любая, но рекомендуется Linux Ubuntu 18.04.3 LTS x86_64.

Рекомендованные системные параметры:

- ЦП: Intel(R) Core(TM) i3-4130 CPU @ 3.40GHz;
- ОЗУ: 8 GB;
- ПЗУ: 60 GB SSD;
- подключение к сети Интернет с пропускной способностью: 25/12,5

Мбит/с.

На сервере должен быть настроен apache2 или nginx, также установлен Node JS/

Перед запуском приложения необходимо настроить конфигурационный файл клиент-серверного приложения:

- JWT_SECRET – секретный ключ для хеширования JWT;
- EXP_JWT_HOURS – время которое будет активен JWT в часах;
- SALT_SEVRET – секретный ключ хеширования соли;
- URL_ADDRES_API – адрес по которому будет доступно серверное приложение;

– URL_ADDRES_ADMIN – адрес по которому будет доступно клиентское приложение;

- DATA_BASE – данные для подключения к БД;

– PATH_FILE_BASE – путь куда будут сохраняться загруженные файлы на сервере.

Поле чего можно приступить к запуску приложения, для запуска можно выбрать несколько конфигураций:

1) Запуск приложение с единым сервисом, команда для запуска `npm run one`.

2) Запуск приложения с 2 микро-сервисами:

- Первый микро-сервис работает с пользователями и файлами;
- Второй микро-сервис работает с модулями

Команда для запуска `npm run double`.

3) Запуск приложения с 4 микро-сервисами

- Первый микро-сервис работает с пользователями;
- Второй микро-сервис работает с файлами;
- Третий микро-сервис работает предустановленными модулями;
- Четверной микро-сервис работает с индивидуальными модулями.

Команда для запуска `npm run full`.

После запуска серверного приложения необходимо запустить клиентское приложения для администрирования, приложения запускается с помощью команды `npm run client`.

После выполнения данных команд в сети будет доступен клиент, с помощью браузера на его можно будет перейти, ссылка на клиентское приложение устанавливается в настройках сервера.

При первом входе в приложение необходимо зарегистрировать супер-пользователя, страница регистрации супер пользователя представлена на рисунке 5.1.

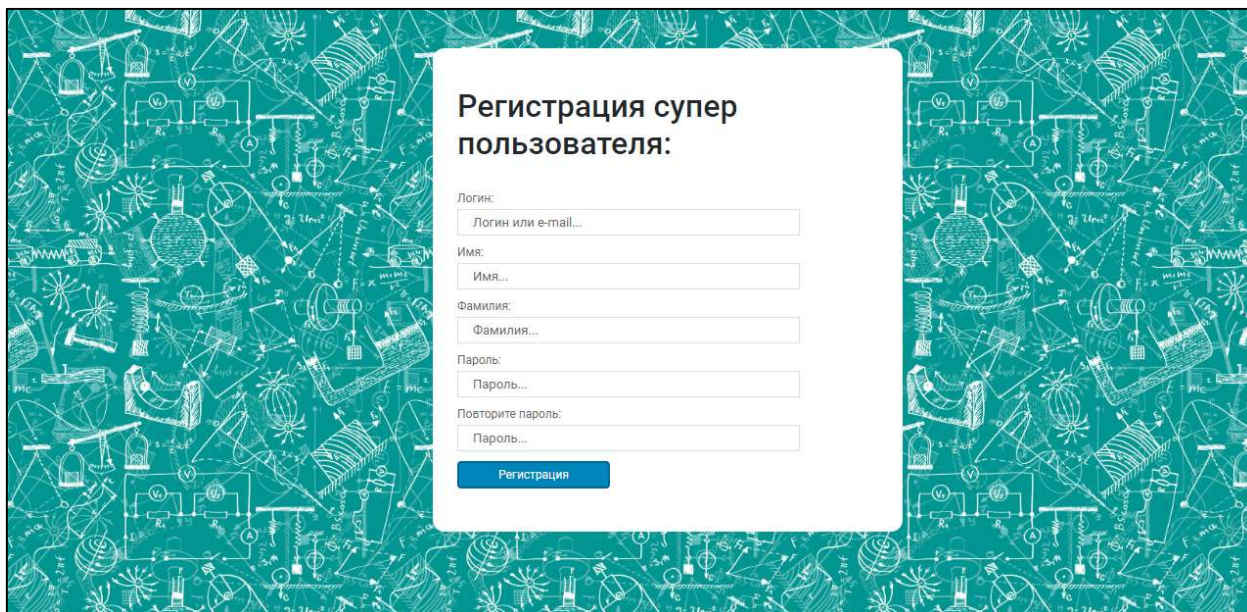


Рисунок 5.1 – Страница регистрации супер пользователя

После того как супер пользователь будет успешно зарегистрирован, ссылка на регистрацию будет закрыта, ее можно будет открыть в настройках клиентского приложения и настроить уровень доступа который будет присвоен пользователю прошедшему регистрацию.

После регистрации супер пользователя можно начать работать с приложением, первое что можно сделать это в настройках приложения настроить доступ к API с внешнего клиента, также можно добавлять новых пользователей, работать с группами доступа, загружать файлы, работать с предустановленными модулями или создавать свои.

Изначально в системе присутствуют предустановленные модули:

- страницы;
- категории страниц;
- записи;
- рубрики записей;
- метки для страниц и записей;
- файлы.

Модуль страниц представлен из 6 переменных которые можно изменять или добавлять новые, на рисунке 5.2 представлен модуль страниц со списком переменных.

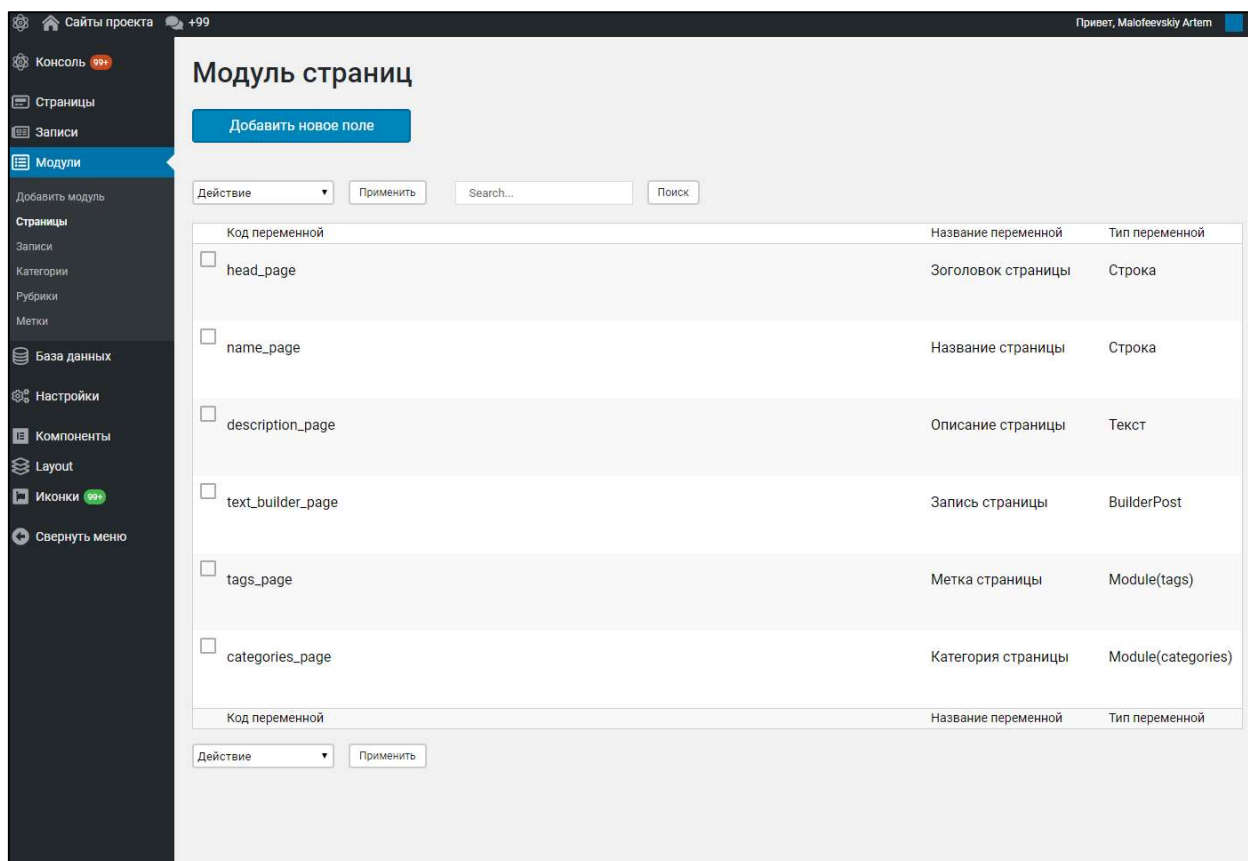


Рисунок 5.2 – Список переменных модуля страниц

Первые 3 переменные являются стандартными текстовыми, четвёртая переменная является конструктором записи с возможность вставки HTML кода. Также используются две не стандартные переменные, тип данных переменных является модулем, а именно теги и категории. Для данных переменных отдельно предусмотрены модули тегов и категорий. Данные модули содержат в себе 4 переменные: название, описание, ярлык и записи. Поле ярлык используется для фильтрации записей, а поле записи используется для автоматического подсчёта сколько раз было использована категория. На рисунке 5.3 представлен модуль категорий.

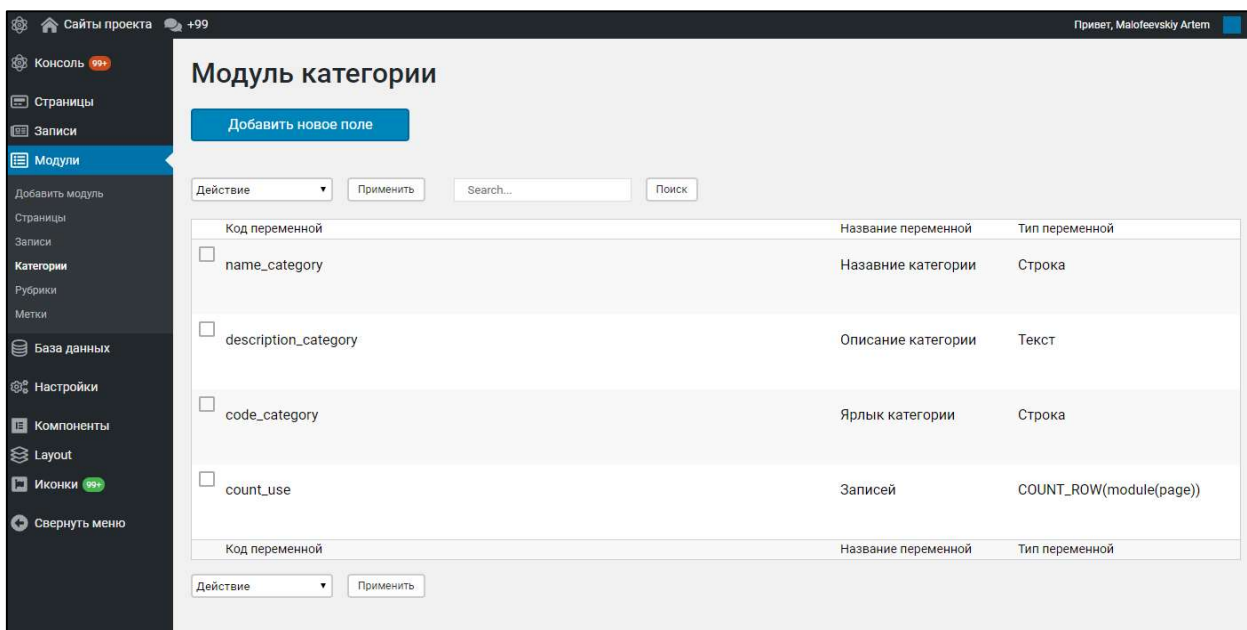


Рисунок 5.3 – Список переменных модуля категорий

Перед созданием индивидуального модуля желательно просмотреть существующую в приложении компоненты переменных, для того чтобы понимать, как будут работать поля и компоненты в индивидуальных модулях, страница компонентов кнопок представлена на рисунке 5.4.

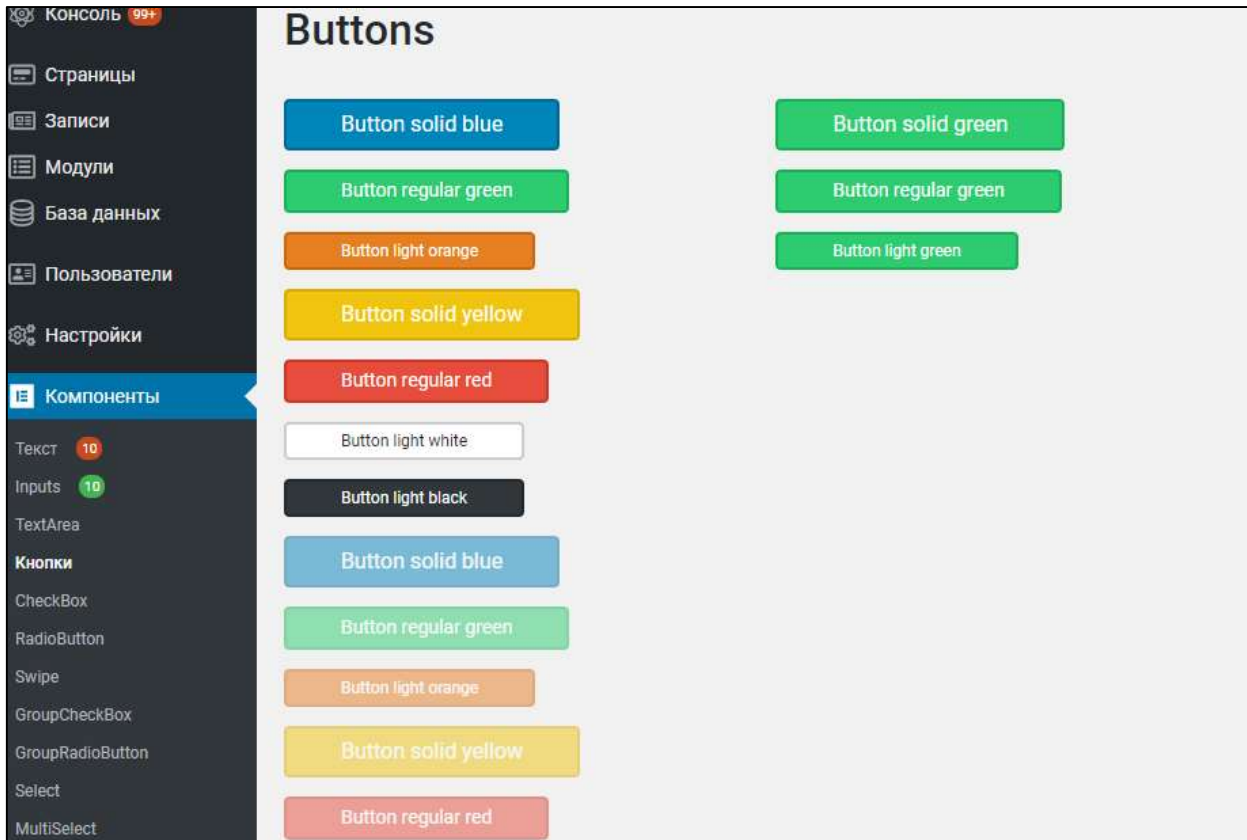


Рисунок 5.4 – Страница компонента кнопок

Также перед созданием индивидуального модуля можно просмотреть предустановленные иконки в клиентском приложении для стилизации модуля, также данные иконки могут быть доступны для внешнего клиентского приложения, чтобы сделать иконки доступными для внешнего клиентского приложения необходимо перейти в настройки внешних приложений выбрать необходимое приложение и на странице настроек установить галочку сделать доступными иконки, в приложении на данный момент доступно более пяти тысяч различных иконок которые разбиты на 4 категории: бедны, тонкие, обычные и жирные иконки, на рисунке 5.5 представлена страница брендовых иконок.

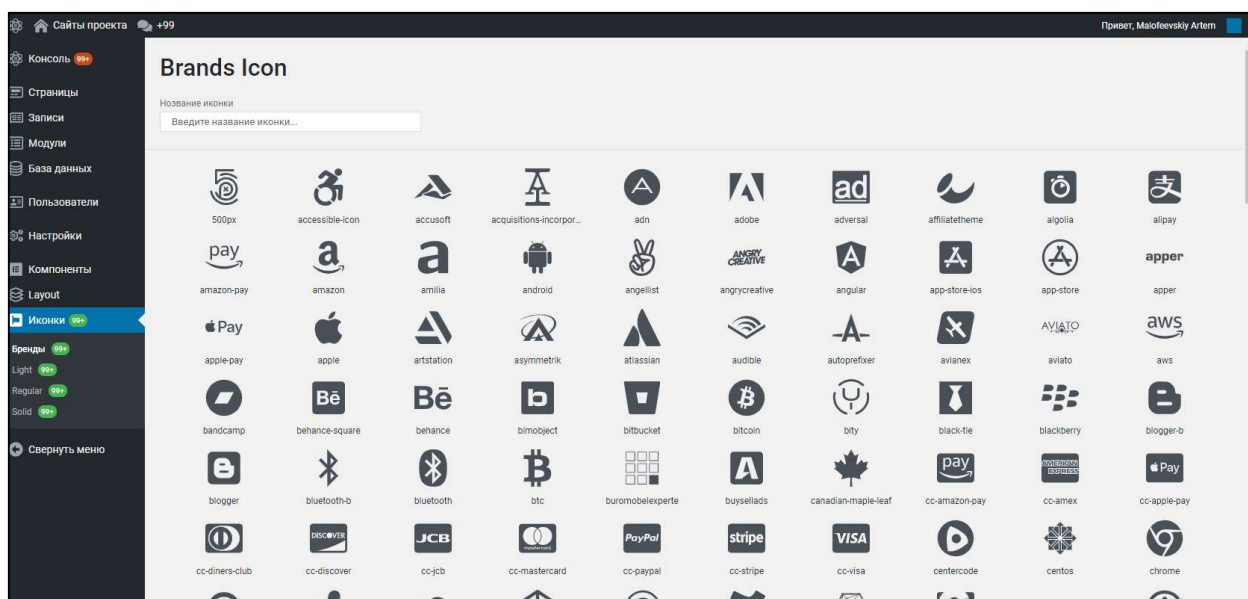


Рисунок 5.5 – Страница компонента кнопок

Для добавления нового индивидуально модуля необходимо перейти на страницу модули, добавить модуль. После чего необходимо ввести обязательные поля в правой панели:

- ярлык URL – служит для определения ссылки к модулю;
- название модуля – название поля на латинском языке или можно на русском, в таком случае название будет автоматически переведено на латинский язык;
- заголовок модуля – заголовок будет отображён в левой панели меню;

Также есть поля описания модуля, иконка модуля в которое можно вставить код иконки и уровень доступа, который будет необходим для добавления новых записей пользователем.

На рисунке 5.6 представлена страница добавления нового индивидуального модуля.

Добавить новый модуль:

Поля модуля:

Название	Заголовок	Тип поля	Компонент
ID_new_module	Ключ новый модуль	SERIAL_BIG_INT	TEXT
<input type="checkbox"/> title	Заголовок	TEXT	INPUT_CONTROL

Действие:

Новое поле:

Название поля: ☐ NOT NULL
 Заголовок поля:
 Тип поля:
 Компонент:
 Модуль:
 Заголовок из модуля:
 Расположение поля:
 Левая панель управления:
 Название новой панели:
 Заголовок новой панели:

Первичные настройки

Язык URL:
 Название:
 Заголовок:
 Описание:
 Иконка:
 Код иконки:
 Уровень доступа:
 Расположения:
 Наследование:
 Узкие настройки:

Рисунок 5.6 – Страница добавления нового индивидуального модуля

Также на рисунке 5.6 можно увидеть процесс добавления поля в индивидуальный модуль. При добавлении нового поля в модуль изначально доступны следующие поля:

- название поля – служит для ввода названия нового поля, ввод на латинском языке или на русском, в случае ввода на русском название будет автоматически переведено на латинский;
- заголовок поля – заголовок будет отображаться в индивидуальном модуле как название поля;
- not null – определяет обязательное ли поля для заполнения;
- тип поля – служит для определения типа поля;
- компонент – служит для выбора компонента к данному полю;
- Расположение поля – служит для определения где поле будет находится в шаблоне страницы модуля, либо в центральной области, либо в правой панели управления;

В случае выбора типа поля «Модуль» то добавятся дополнительные поля:

- модуль – список доступных модулей данному пользователю;
- заголовок для списка из модуля – служит для выбора какое поле будет выводится в роли заголовка для списка выбора в переменную.

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:

$$З_0 = \sum_{i=1}^n T_{qi} \cdot T_q \cdot \Phi_3 \cdot П_{пр} \quad (6.1)$$

где n – количество исполнителей, $n = 1$;

T_q – количество часов работы в день, $T_q = 8$ ч.;

T_{qi} – часовая тарифная ставка i -го исполнителя, $T_{qi} = 5,2$ руб.;

Φ_3 – эффективный фонд рабочего времени i -го исполнителя, $\Phi_3 = 96$ дней;

$К_{пр}$ – коэффициент премирования, $К_{пр} = 2,48$.

В модернизации будет участвовать один исполнитель (таблица 6.1).

$$З_0 = 5,2 \cdot 8 \cdot 96 \cdot 2,48 = 9904,13 \text{ руб.}$$

Таблица 6.1 – Разряды, ставки и тарифные коэффициенты работников

Наименование должности	Разряд	Тарифный коэффициент	Часовая тарифная ставка $T_{чi}$, руб.
Инженер-программист	10	2,48	5,2

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

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (6.2)$$

где $З_д$ – дополнительная заработная плата исполнителя ПС, $З_д = 1980,83$ руб.;

$Н_д$ – норматив дополнительной заработной платы, $Н_д = 20 \%$;

$З_о$ – сумма основной заработной платы исполнителя ПС, $З_о = 9904,13$ руб.

$$З_д = \frac{9904,13 \cdot 20}{100} = 1980,83 \text{ руб.}$$

3) 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:

$$З_{соц} = (З_о + З_д) \cdot Н_{соц}, \quad (6.3)$$

где $З_о$ – сумма основной заработной платы исполнителя ПС, $З_о = 9904,13$ руб.;

$З_д$ – дополнительная заработная плата исполнителя ПС, $З_д = 1980,83$ руб.;

$Н_{соц}$ – норматив отчислений в фонд социальной защиты населения и обязательное страхование, $Н_{соц} = 35 \%$.

$$З_{соц} = (9904,13 + 1980,83) \cdot 0,35 = 4159,74 \text{ руб.}$$

4) Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для модернизации и отладки ПС, которое определяется по нормативам на 100 строк ЛОС ($Н_{мз}$) машинного времени в зависимости от характера решаемых задач и типа ПК. Рассчитывается по формуле:

$$P_M = C_M \cdot V_o \cdot H_{M3}, \quad (6.4)$$

где C_M – цена одного машино-часа, $C_M = 0,098$ руб.;
 V_o – общий объём, $V_o = 25822$ LOC;
 H_{M3} – норматив расхода машинного времени на отладку 100 строк LOC, $H_{M3} = 3\%$.

$$P_M = 0,098 \cdot 25822 \cdot 0,03 = 75,92 \text{ руб.}$$

5) 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.

$$P_{пз} = Z_o \cdot H_{пз}, \quad (6.5)$$

где Z_o – сумма основной заработной платы исполнителя ПС, $Z_o = 9904,13$ руб.;
 $H_{пз}$ – норматив прочих затрат, $H_{пз} = 20\%$.

$$P_{пз} = 9904,13 \cdot 0,2 = 1980,83 \text{ руб.}$$

6) 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 ($H_{нр}$) в процентном отношении к основной заработной плате исполнителя. Рассчитывается по формуле:

$$P_{нр} = Z_o \cdot H_{нр}, \quad (6.6)$$

где Z_o – сумма основной заработной платы исполнителя ПС, $Z_o = 9904,13$ руб.;
 $H_{нр}$ – норматив накладных расходов в целом по научной организации, $H_{нр} = 58\%$.

$$P_{нр} = 9904,13 \cdot 0,58 = 5744,40 \text{ руб.}$$

7) Общая сумма производственной себестоимости ($C_{пр}$) на ПС рассчитывается по формуле:

$$C_{пр} = Z_o + Z_d + Z_{соц} + P_{пз} + P_M + P_{нр}, \quad (6.7)$$

где Z_o – сумма основной заработной платы исполнителя ПС, $Z_o = 9904,13$ руб.;

Z_d – дополнительная заработная плата исполнителя ПС, $Z_d = 1980,83$ руб.;

$Z_{соц}$ – отчисления в фонд социальной защиты населения, $Z_{соц} = 4159,74$ руб.;

$P_{пз}$ – расходы по статье «Прочие затраты», $P_{пз} = 1980,83$ руб.;

P_M – расходы по статье «Машинное время», $P_M = 75,92$ руб.;

$P_{нр}$ – расходы по статье «Накладные расходы», $P_{нр} = 5844,40$ руб.

$$C_{пр} = 9904,13 + 1980,83 + 4159,74 + 1980,83 + 75,92 + 5844,40 \\ = 23945,85 \text{ руб.}$$

8) Расходы пользователя на оплату услуг по сопровождению и адаптацию ПС, которые определяются по нормативу (H_c) и по формуле:

$$P_c = C_{пр} \cdot H_c, \quad (6.8)$$

где H_c – норматив расходов на сопровождение и адаптацию, $H_c = 20\%$;

$C_{пр}$ – общая сумма производственной себестоимости на ПС, $C_{пр} = 23945,85$ руб.

$$P_c = 23945,85 \cdot 0,2 = 4789,17 \text{ руб.}$$

9) Полная себестоимость ПС определяется по формуле:

$$C_{п} = C_{пр} + P_c, \quad (6.9)$$

где $C_{пр}$ – общая сумма производственной себестоимости на ПС, $C_{пр} = 23945,85$ руб.;

P_c – расходы пользователя на оплату услуг по сопровождению и адаптацию ПС, $P_c = 4789,17$ руб.

$$C_{п} = 23945,85 + 4789,17 = 28735,02 \text{ руб.}$$

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

$$\Delta_1 = K_{пр} \cdot \sum_{i=1}^n \Delta Ч \cdot 3 \cdot (1 + H_d/100) \cdot (1 + H_{но}/100), \quad (6.10)$$

где $K_{\text{пр}}$ – коэффициент премий за выполнение плановых заданий, $K_{\text{пр}} = 2,48$;
 $\Delta\text{Ч}$ – абсолютное высвобождение работников, $\Delta\text{Ч} = 1$;
 Z – заработная плата высвобождаемых работников i -ой категории, $Z = 915,2$ руб.;
 $H_{\text{д}}$ – процент дополнительной заработной платы, $H_{\text{д}} = 20\%$;
 $H_{\text{соц}}$ – ставка отчислений от заработной платы, включаемых в себестоимость продукции, $H_{\text{соц}} = 35\%$;
 n – количество категорий, которым принадлежат высвобожденные работники.

$$\text{Э}_i = 2,48 \cdot (1 \cdot 915,2 \cdot 12) \cdot (1 + 0,2) \cdot (1 + 0,35) = 44122,89 \text{ руб.}$$

В результате полученных расчётов можно посчитать экономическую эффективность ПС. За счёт прироста прибыли с экономии расходов мы получаем за год 15387,87 чистой прибыли.

Таким образом, при определении экономической эффективности использования программного средства для генерации индивидуальных модулей и API:

– полная себестоимость ПС в реализации проекта составляет $\text{СП} = 28735,02$ руб.;

– чистая прибыль $\text{Пч} = 15203,65$ руб.;

Таким образом, процент рентабельности разработки программного средства для генерации индивидуальных модулей и API является 52.91% что является экономически выгодным.

ЗАКЛЮЧЕНИЕ

За время прохождения преддипломной практики было разработано программное средство генератор API на платформе Node JS, позволяющие быстро развёртывать клиент-серверное приложение, использовать предустановленные модули, создавать индивидуальные модули, масштабировать приложение.

При реализации проекта была изучена методика проектирования и разработки веб-сервисов используя Spring Boot. Проведено исследование аналогов программного средства и были выявлены их достоинства и недостатки. На основе полученной информации были сформулированы требования к собственному программному средству.

Выполнен анализ предметной области. В процессе работы и проектирования выполнены следующие исследования и разработки:

- созданы модели IDEF0;
- построена и проработана модель вариантов использования;
- построена схема базы данных и модель «Сущность-связь»;

Проведено функциональное проектирование, выявлены все связи и особенности взаимодействия.

В ходе технического проектирования на основе функциональных моделей была разработана диаграмма развёртывания, построены алгоритмы.

Проведено тестирование основных модулей программного средства, проверены функции расчета. Разработана методика работы с программным средством для упрощения работы пользователя с ней.

Был проведён расчёт экономического эффекта от внедрения программного средства, который показал его целесообразность.

Разработанное программное средство обеспечивает решение поставленных перед ним функциональных задач.

Таким образом, цель дипломного проекта достигнута.

Данное программное приложение может быть дополнено и модернизировано.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

- [1] The State of REST in 2019 [Электронный ресурс] – 2019 – Режим доступа: <https://medium.com/javascript-in-plain-english/the-state-of-rest-in-2019-75005eaf05b9> – Дата доступа: 17.11.2019.
- [2] Современный мир держится на API [Электронный ресурс] – 2019 – Режим доступа: <https://habr.com/ru/company/softwareag/blog/449544/> – Дата доступа: 17.11.2019.
- [3] Антиплагиат [Электронный ресурс]. – 2019. – Режим доступа: <https://users.antiplagiat.ru/> – Дата доступа: 27.12.2019.
- [4] приложения, реализованные с использованием микро-сервисов и API [Электронный ресурс] – 2019 – Режим доступа: <https://github.com/search?q=microservices+api> – Дата доступа: 18.12.2019.
- [5] Airship is a framework for Node.JS [Электронный ресурс] – 2019 – Режим доступа: <https://github.com/Naltox/airship> – Дата доступа: 18.12.2019.
- [6] Управление хранить бесплатно с программным обеспечением Hiboutik POS [Электронный ресурс] – 2019 – Режим доступа <https://www.hiboutik.com/ru/> – Дата доступа: 18.12.2019.
- [7] amoCRM – "Хотите увеличить продажи?" [Электронный ресурс]. – 2019 – Режим доступа: <https://www.amocrm.ru/> – Дата доступа: 18.12.2019.
- [8] Методология IDEF0 [Электронный ресурс] – 2019 – Режим доступа: <https://itteach.ru/bpwin/metodologiya-idef0> – Дата доступа: 19.12.2019.
- [9] Бахтизин, В.В. Структурный анализ и моделирование в среде CASE-средства BPWin: Учеб. пособие по курсу “Технология проектирования программ” для студ. спец. 40 01 01. / Глухова Л.А. – Мн.: БГУИР, 2002. – 44с.\
- [10] Диаграмма вариантов использования (use case diagram) [Электронный ресурс] – 2019 – Режим доступа: <http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html> – Дата доступа: 25.12.2019.
- [11] Node JS [Электронный ресурс] – 2019 – Режим доступа: <https://nodejs.org/ru/> – Дата доступа: 04.12.2019.
- [12] React JS [Электронный ресурс] – 2019 – Режим доступа: <https://ru.reactjs.org/> – Дата доступа: 04.12.2019.
- [13] PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс] – 2019 – Режим доступа: <https://www.postgresql.org/> – Дата доступа: 04.12.2019.
- [14] Форта, Б. SQL за 10 минут. Б. Форта. – М. : Вильямс, 2019. – 288с.
- [15] Базы данных и модели данных. Основы проектирования реляционных баз данных [Электронный ресурс] – 2019 – Режим доступа: http://edu.tltsu.ru/sites/sites_content/site216/html/media67139/theor_bd.pdf – Дата доступа: 20.12.2019.
- [16] Тестирование. Фундаментальная теория [Электронный ресурс] – 2016 – Режим доступа: <https://habr.com/ru/post/279535/> – Дата доступа: 20.12.2019.
- [17] Блэк, Р. Ключевые процессы тестирования программного обеспечения. Р. Блэк. – М. : Символ, 2012. – 772с.
- [18] Тестирование производительности [Электронный ресурс] – 2019 – Режим доступа: <https://qalight.com.ua/baza-znaniy/testirovanie-proizvoditelnosti/> – Дата доступа: 20.12.2019.

ПРИЛОЖЕНИЕ А
(обязательное)
Текст программного средства

```
const dev = {
  VER:'dev',
  NAME_PROJECT:'API DIPLOM',
  JWT_SECRET: '[REDACTED]',
  EXP_JWT_HOURS: 1,
  SALT_SECRET: '[REDACTED]',
  PORT: '3300',
  DATA_BASE: {
    user: 'postgres',
    host: 'localhost',
    database: 'diplom_v1',
    password: "",
    port: 5432,
  }
};

const test = {
  VER:'test',
  NAME_PROJECT:'API DIPLOM',
  JWT_SECRET: '[REDACTED]',
  EXP_JWT_HOURS: 1,
  SALT_SECRET: '[REDACTED]',
  PORT: '3301',
  DATA_BASE: {
    user: 'postgres',
    host: 'localhost',
    database: 'diplom_v1',
    password: "",
    port: 5432,
  }
};

const prod = {
  VER:'prod',
  NAME_PROJECT:'API DIPLOM',
  JWT_SECRET: '[REDACTED]',
  EXP_JWT_HOURS: 1,
  SALT_SECRET: '[REDACTED]',
  PORT: '2201',
  DATA_BASE: {
    user: 'user_diplom',
    host: 'localhost',
```

```

    database: 'diplom',
    password: '████████████████████',
    port: 5432,
  }
};
switch (process.env.NODE_ENV) {
  case 'dev':
    return module.exports = dev;
  case 'test':
    return module.exports = test;
  case 'prod':
    return module.exports = prod;
  default:
    return module.exports = prod;
}
const express = require("express");
const bodyParser = require('body-parser');
const cookieParser = require('cookie-parser');
const path = require('path');
const session = require('express-session');
const cors = require('cors');
const {PORT, VER} = require('./config/config');
let app = express();
const allowCrossDomain = function (req, res, next) {
  res.header('Access-Control-Allow-Origin', '*');
  res.header('Access-Control-Allow-Headers', '*');
  next();
};
app.use(bodyParser.json());
app.use(cookieParser());
app.use(allowCrossDomain);
app.use(cors());
app.use(session({secret: 'passport-tutorial', cookie: {maxAge: 60000}, resave: false, saveUninitialized: false}));
app.use('/public', express.static('public'));
app.get('/', (req, res) => {
  res.send('Hello World!')
});
app.use('/users', require('./app/routers/users'));
app.use('/file', require('./app/routers/file'));
app.use('/module', require('./app/routers/module'));
app.use('/settings', require('./app/routers/settings'));
app.use('/analytic', require('./app/routers/analytic'));
app.use('/notification', require('./app/routers/notification'));

```

```

let now = new Date();
app.listen(PORT, () => {
  console.log(`${now.toString()}: Server running on port: ${PORT}, ver:
${VER}`);
});
const {Pool} = require('pg');
const {DATA_BASE} = require('./config/config');
const pool = new Pool({
  user: DATA_BASE.user,
  host: DATA_BASE.host,
  database: DATA_BASE.database,
  password: DATA_BASE.password,
  port: DATA_BASE.port,
});
module.exports = pool;
const express = require('express');
const router = require('express-promise-router')();
const {validateBody, schemas} = require('../validates/users')
const usersController = require('../controller/users');
const passport = require('passport');
const passportConf = require('.././config/passport');
const passportSignIn = passport.authenticate('local', {session: false});
const passportJWT = passport.authenticate('jwt', {session: false});
const crypto = require('crypto');
const mime = require('mime-types');
let multer = require('multer');
let upload = multer({
const storage_user_avatar = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'public/user_avatar/')
  },
  filename: function (req, file, cb) {
    crypto.pseudoRandomBytes(16, function (err, raw) {
      cb(null, raw.toString('hex') + Date.now() + '!' +
mime.extension(file.mimetype));
    });
  }
});
const user_avatar = multer({
  storage: storage_user_avatar
});
router.route('/signUp')
  .post(validateBody(schemas.sign_up), usersController.sign_up);
router.route('/signIn')

```



```

    .post(validateBody(schemas.sign_in),passportSignIn, usersController.sign_in);
router.route('/signout')
    .get(passportJWT, usersController.sign_out);
module.exports = router;
const Joi = require('joi');
module.exports = {
  validateBody: (schema) => {
    return (req, res, next) => {
      const result = Joi.validate(req.body, schema);
      if (result.error) {
        return res.status(400).json(result.error);
      }
      if (!req.value) {
        req.value = {};
      }
      req.value['body'] = result.value;
      next();
    }
  },
  schemas: {
    sign_up: Joi.object().keys({
      email: Joi.string().email().required(),
      password: Joi.string().min(6).max(20).required(),
      name: Joi.string().required(),
      last_name: Joi.string().required(),
    }),
    sign_in: Joi.object().keys({
      email: Joi.string().email().required(),
      password: Joi.string().required(),
    })
  }
}
const crypto = require('crypto');
const pool = require('../bd');
const pass = require('../middleware/password');
module.exports = {
  sign_up: async (req, res, next) => {
    const {
      email,
      password,
      name,
      last_name
    } = req.value.body;

```

```

    const pas = pass.set_password(password);
    pool.query(`SELECT login FROM users WHERE login = '${email}'`, (err,
user_list) => {
      if (err) {
        return res.status(200).json({success: false, error: err, n: 1})
      }
      if (user_list.rows.length) {
        return res.status(200).json({success: false, error: 'Email is already in
use'})
      } else {
        pool.query(`INSERT INTO users (login, hash, salt) VALUES
('${email}', '${pas.hash}', '${pas.salt}'), (err, insert_user) => {
          if (err) {
            return res.status(200).json({success: false, error: err, n: 2})
          }
          pool.query(`SELECT id_user FROM users WHERE login =
'${email}', (err, select_user) => {
            if (err) {
              return res.status(200).json({success: false, error: err, n: 3})
            }
            pool.query(`INSERT INTO profile (id_user, name_user,
first_name_user) VALUES (${select_user.rows[0]['id_user']}, '${name}',
'${last_name}'), (err, insert_profile) => {
              if (err) {
                return res.status(200).json({success: false, error: err, n: 4})
              }
              const user = {
                id: email,
                email: email,
              };
              const token = pass.generateJWT(user);
              res.cookie('token', token, {
                httpOnly: true
              });
              return res.status(200).json({success: true})
            }
          });
        });
      }
    });
  },
  sign_in: async (req, res, next) => {
    const user = {id: req.user['email'], ...req.user}
    const token = pass.generateJWT(user);

```

```

    res.cookie('token', token, {
      httpOnly: true
    });
    res.status(200).json({
      success: true,
      data: {
        token: token,
        nameUser: req.user.name,
        lastNameUser: req.user.last_name
      }
    });
  },
  sign_out: async (req, res, next) => {
    res.clearCookie('token');
    res.json({success: true});
  },
  yes: async (req, res, next) => {
    res.status(200).json({success: true, access: true})
  },
  no: async (req, res, next) => {
    res.status(200).json({success: true, access: false})
  },
  select_all_users: async (req, res, next) => {
    pool.query('SELECT * FROM users', (err, q_res) => {
      if (err) {
        throw err
      }
      res.status(200).json(q_res.rows)
    })
  },
  load_avatar: async (req, res, next) => {
    console.log(req.files[0].filename)
    res.status(200).json({success: true});
  }
}

const crypto = require('crypto');
const JWT = require('jsonwebtoken');
const {SALT_SECRET, JWT_SECRET, EXP_JWT_HOURS} =
require('../config/config');
let exports_pass = module.exports = {};
exports_pass.set_password = (password) => {
  const salt = crypto.randomBytes(16).toString('hex');
  const sec_salt = crypto.pbkdf2Sync(salt, SALT_SECRET, 1224, 32,
'sha512').toString('hex')

```

```

    const hash = crypto.pbkdf2Sync(password, sec_salt, 5540, 128,
'sha512').toString('hex');
    const res = {
      salt: salt,
      hash: hash
    };
    return res;
  };
  exports_pass.validate_password = (password, hash_bd, salt_bd) => {
    const sec_salt = crypto.pbkdf2Sync(salt_bd, SALT_SECRET, 1224, 32,
'sha512').toString('hex')
    const hash = crypto.pbkdf2Sync(password, sec_salt, 5540, 128,
'sha512').toString('hex');
    return hash_bd === hash;
  };
  exports_pass.generateJWT = user => {
    const today = new Date();
    const expirationDate = new Date(today);
    expirationDate.setDate(today.getDate() + 60);
    return JWT.sign({
      sub: user.id,
      email: user.email,
      iss: 'CodeWorker',
      exp: Math.floor(Date.now() / 1000) + (EXP_JWT_HOURS * 60 * 60),
    }, JWT_SECRET);
  };
  const passport = require('passport');
  const JwtStrategy = require('passport-jwt').Strategy;
  const LocalStrategy = require('passport-local').Strategy;
  const {JWT_SECRET} = require('./config');
  const pool = require('./bd');
  const pass = require('./app/middleware/password');
  const cookieExtractor = req => {
    let token = null;
    if (req && req.cookies) {
      token = req.cookies['token'];
    }
    return token;
  }
  // JSON WEB TOKENS STRATEGY
  passport.use(new JwtStrategy({
    jwtFromRequest: cookieExtractor,
    secretOrKey: JWT_SECRET,
    passReqToCallback: true

```

```

    }, async (req, payload, done) => {
      try {
        pool.query(`SELECT id_user, login, name, last_name FROM users WHERE
login = '${payload.sub}'`, (err, q_res) => {
          if (err) {
            return done(null, false);
          }
          if (q_res.rows.length) {
            req.user = q_res.rows[0];
            done(null, q_res.rows[0]);
          } else {
            return done(null, false);
          }
        });
      } catch (error) {
        done(error, false);
      }
    });
  // LOCAL STRATEGY
  passport.use(new LocalStrategy({
    usernameField: 'email'
  }, async (email, password, done) => {
    try {
      pool.query(`SELECT login, hash, salt, name, last_name FROM users
WHERE login = '${email}'`, (err, q_res) => {
        if (err) {
          return done(null, false);
        }
        if (q_res.rows.length) {
          if (pass.validate_password(password, q_res.rows[0]['hash'],
q_res.rows[0]['salt'])) {
            done(null, q_res.rows[0]);
          } else {
            return done(null, false);
          }
        } else {
          return done(null, false);
        }
      });
    } catch (error) {
      done(error, false);
    }
  }));

```

```

import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from "react-router-dom";
import App from './App'
import $ from 'jquery';
window.$ = $;
ReactDOM.render(
  <BrowserRouter>
    <App/>
  </BrowserRouter>,
  document.getElementById('root'));
import React, { Component } from 'react';
import Routers from './routers'
import './style.css'
import './libs/bootstrap-grid.css'
import './font/stylesheet.css'
class App extends Component {
  render() {
    return (
      <div className="App">
        <Routers/>
      </div>
    );
  }
}
export default App;
import React from 'react'
import {Route, Switch} from 'react-router-dom'
import Home from './components/testPages/home';
import Test from './components/testPages'
import Login from './components/testPages/auth/login'
import Reg from './components/testPages/auth/reg'
import NoAuth from './components/pages/no_auth'
import Error404 from './components/pages/404'
import config from './config/config.json'
const Routes = () => (
  <Switch>
    <Route exact path="/" component={Home}/>
    <Route path={config.test.homeLink} component={Test}/>
    <Route path={'/reg'} component={Reg}/>
    <Route path={'/login'} component={Login}/>

    <Route path={'/no_auth'} component={NoAuth}/>
  </Switch>
)

```

```

        <Route component={Error404}/>
    </Switch>
);

export default Routes
import React from 'react';
import config from '../config/config.json'

import Container from '../layout/container'
import iconSolid from '../config/iconSolid.json'
import iconRegular from '../config/iconRegular.json'
import iconLight from '../config/iconLight.json'
import iconBrands from '../config/iconBrands.json'

import Route from './routes'

export default class index extends React.Component {
    constructor(props) {
        super(props);
        this.state = {}
    }

    render() {
        let Regular = Object.keys(iconRegular).length;
        let Solid = Object.keys(iconSolid).length;
        let Light = Object.keys(iconLight).length;
        let Brands = Object.keys(iconBrands).length;

        return (
            <div>
                <Container
                    dataMenuTop={this.props.dataMenuTop}
                    dataMenuLeft={this.props.dataMenuLeft}
                    location={this.props.location}
                >
                    { /*test {this.state.id} */ }

                    <Route/>

                </Container>
            </div>
        )
    }
}

```

```

        </div>
    );
}
}
import React from 'react';
import './style.css'

export default class button extends React.Component {
  constructor(props) {
    super(props);
    this.state = {}
  }

  render() {
    let {
      title,
      onPress,
      typeSize,//solid,regular,light
      typeColor,//green,blue,red,yellow,orange,white
      padding,
      disabled
    } = this.props;
    let styleButton = {};
    if (padding) {
      if (padding > 0) {
        styleButton = {
          display: 'inline-block',
          paddingLeft: padding + 'px',
          paddingRight: padding + 'px'
        }
      }
    }
    else {
      styleButton = {display: 'inline-block', width: '100%'}
    }
    } else {
      styleButton = {display: 'inline-block', width: '100%'}
    }
    }

    return (
      <div className={`block-button ${typeSize} ${typeColor} ${disabled} &&
'disabled'}` style={styleButton} onClick={disabled?null:onPress}>
        <p className="title-button">

```



```

        {title}
      </p>
    </div>
  );
}
}
import React from 'react';
import Icon from '../icon'
import './style.css'
import iconSolid from '../../config/iconSolid.json'
import iconRegular from '../../config/iconRegular.json'
import iconLight from '../../config/iconLight.json'
import iconBrands from '../../config/iconBrands.json'
import {Link} from "react-router-dom";

export default class buttonLeftMenu extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      active: false
    }
    this.LickActive = this.LickActive.bind(this);
  }

  LickActive = (path, sub) => {

  }

  render() {
    let {
      title,
      subLinkList,
      icon,
      typeIcon,
      fullMenu,
      countRound,
      colorRound, /*green, red, blue, orange, null(gray)*/
      path,
      onPress,
      separator
    } = this.props;
    let active = false;

    let url = this.props.location['pathname'];

```

```

    if (url === path) {
      active = true;
    }
    if (url[url.length - 1] === '/') {
      if (url.substring(0, url.length - 1) === path) {
        active = true;
      }
    }
  }
}

let subItemLick = subLinkList ? subLinkList.map((item, key) => {
  let activeSub = false
  let url = this.props.location['pathname'];

  if (url === item.path) {
    activeSub = true;
    active = true;
  }
  if (url[url.length - 1] === '/') {
    if (url.substring(0, url.length - 1) === item.path) {
      activeSub = true;
      active = true;
    }
  }
}
return (
  <div key={key}>
    {item.path} &&
    <Link to={item.path}>
      <div
        className={`block-item-sub-link-left-menu ${activeSub} &&
'active'}` `}
        onClick={item.onPress ? item.onPress : null}>
        <p className="title-item-sub-link-left-menu">
          {item.title}
          {item.countRound} &&
          <span className={`round-info ${item.colorRound}`} `}
            title={item.countRound}>
              {item.countRound > 99 ? '99+' : item.countRound}
            </span>
          </p>
        </div>
      </Link>
    </div>
  )
}

```

```

        {!item.path &&
        <div
            className={`block-item-sub-link-left-menu ${activeSub &&
'active'}}` }
            onClick={item.onPress ? item.onPress : null}>
            <p className="title-item-sub-link-left-menu">
                {item.title}
                {item.countRound &&
                <span className={`round-info ${item.colorRound}`} `}
title={item.countRound}>
                    {item.countRound > 99 ? '99+' : item.countRound}
                </span>
            }
            </p>
        </div>
        }
    </div>

    )
    }): null;

    return (
        <div style={separator? {margin:'0 0 11px 0'}: {margin:'0'}}>
            {!path &&
            <div
                className={`block-button-left-menu ${active && 'active'}
${fullMenu?'max':'min'} ${subLinkList?'sub':'no-sub'}}` }
                onClick={onPress ? onPress : null}>
                <div className="block-top-button-left-menu">
                    <div className="block-icon-button-left-menu">
                        <Icon
                            icon={typeIcon === 'Brands' ? iconBrands[icon] :
                                typeIcon === 'Light' ? iconLight[icon] :
                                typeIcon === 'Regular' ? iconRegular[icon] :
                                typeIcon === 'Solid' ? iconSolid[icon] :
iconBrands.dev}/>
                        </div>
                    <div className="block-title-button-left-menu">
                        <p className="title-button-left-menu">
                            {title}
                            {countRound &&

```

```

        <span className={`round-info ${colorRound}`}
title={countRound}>
        {countRound > 99 ? '99+' : countRound}
    </span>
    }
    </p>
</div>
<div className="block-active-button-left-menu">
</div>

    {subLinkList &&
    <div
        className={`block-list-sub-link-left-menu ${active && 'active'}`}
        style={active ? fullMenu ? {height: 16 + (subLinkList.length * 28)
+ 'px'} : {height: '0',padding:'0'} : {height: '0',padding:'0'}}>
        <div className={`block-top-title ${active && 'active'}`}>
            <p className="title-top-link">{title}</p>
        </div>
        {subItemLick}
    </div>
    }

</div>
}
{!!path &&

<div
    className={`block-button-left-menu ${active && 'active'}
${fullMenu?'max':'min'} ${subLinkList?'sub':'no-sub'}`}
    onClick={onPress ? onPress : null}>
    <div className="block-top-button-left-menu">
        <Link to={path}>
            <div className="block-icon-button-left-menu">
                <Icon
                    icon={typeIcon === 'Brands' ? iconBrands[icon] :
                    typeIcon === 'Light' ? iconLight[icon] :
                    typeIcon === 'Regular' ? iconRegular[icon] :
                    typeIcon === 'Solid' ? iconSolid[icon] :
iconBrands.dev}/>
                </div>
            <div className="block-title-button-left-menu">
                <p className="title-button-left-menu">
                    {title}
                    {countRound &&

```

```

        <span className={`round-info ${colorRound}`}
title={countRound}>
        {countRound > 99 ? '99+' : countRound}
    </span>
    }
    </p>
    </div>
    <div className="block-active-button-left-menu">
    </Link>
    </div>
    {subLinkList &&
    <div
        className={`block-list-sub-link-left-menu ${active && 'active'}`}
        style={active ? fullMenu ? {height: 16 + (subLinkList.length * 28)
+ 'px'} : {height: '0',padding:'0'} : {height: '0',padding:'0'}}>
        <div className={`block-top-title ${active && 'active'}`}>
        <p className="title-top-link">{title}</p>
        </div>
        {subItemLick}
        </div>
    }

    </div>

    }
    </div>
    );
    }
}

import React from 'react';
import './style.css'
import Icon from './icon'
import iconBrands from "../../config/iconBrands";
import iconLight from "../../config/iconLight";
import iconRegular from "../../config/iconRegular";
import iconSolid from "../../config/iconSolid";
import {Link} from "react-router-dom";

export default class buttonTopMenu extends React.Component {
    constructor(props) {
        super(props);
        this.state = {}
    }
}

```

```

render() {
  let {
    title,
    subLinkList,
    icon,
    typeIcon,
    //location,
    path,
    hrefLink,
    onPress,
    separator,
  } = this.props;
  return (

    <div className="block-button-top-menu" style={separator ? {margin: '0
10px 0 0'} : {margin: '0'}}>

      {path &&
      <div className="block-top-button-top-menu">
        <Link to={path}>
          {icon &&
          <div className="block-icon-top-button-top-menu">
            <Icon
              icon={typeIcon === 'Brands' ? iconBrands[icon] :
                typeIcon === 'Light' ? iconLight[icon] :
                typeIcon === 'Regular' ? iconRegular[icon] :
                typeIcon === 'Solid' ? iconSolid[icon] :
iconBrands.dev}
            />
          </div>
          }
          {title &&
          <div className="block-title-top-button-top-menu">
            <p className="title-top-button-top-menu">
              {title}
            </p>
          </div>
          }
          </Link>
        </div>
        }
        {!path && hrefLink &&
        <div className="block-top-button-top-menu">

```

```

<a href={hrefLink} target={'_blank'}>
  {icon &&
    <div className="block-icon-top-button-top-menu">
      <Icon
        icon={typeIcon === 'Brands' ? iconBrands[icon] :
          typeIcon === 'Light' ? iconLight[icon] :
          typeIcon === 'Regular' ? iconRegular[icon] :
          typeIcon === 'Solid' ? iconSolid[icon] :
iconBrands.dev}
      />
    </div>
  }
  {title &&
    <div className="block-title-top-button-top-menu">
      <p className="title-top-button-top-menu">
        {title}
      </p>
    </div>
  }
</a>
</div>
}
{!path && !hrefLink &&
  <div className="block-top-button-top-menu" onClick={onPress ?
onPress : null}>

  {icon &&
    <div className="block-icon-top-button-top-menu">
      <Icon
        icon={typeIcon === 'Brands' ? iconBrands[icon] :
          typeIcon === 'Light' ? iconLight[icon] :
          typeIcon === 'Regular' ? iconRegular[icon] :
          typeIcon === 'Solid' ? iconSolid[icon] : iconBrands.dev}
      />
    </div>
  }
  {title &&
    <div className="block-title-top-button-top-menu">
      <p className="title-top-button-top-menu">
        {title}
      </p>
    </div>
  }
}

```

```

</div>
}

{subLinkList &&
<div className="block-body-button-top-menu">
  {
    subLinkList.map((item, key) => {
      return (
        <div className="block-item-sub-link-top-menu" key={key}>
          {item.path &&
            <Link to={item.path}>
              <p className="title-item-sub-link-top-menu">
                {item.title}
              </p>
            </Link>
          }
          {!item.path && item.hrefLink &&
            <a href={item.hrefLink} target={'_blank'}>
              <p className="title-item-sub-link-top-menu">
                {item.title}
              </p>
            </a>
          }
          {!item.path && !item.hrefLink &&
            <p className="title-item-sub-link-top-menu"
              onClick={item.onPress ? item.onPress : null}>
              {item.title}
            </p>
          }
        </div>
      )
    })
  }
</div>
}

</div>

);
}
}

```



```

import React from 'react';
import './style.css'

export default class index extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      check: this.props.check,
      onChange: this.props.onChange
    }
  }

  onChange = () => {
    this.setState({check: !this.state.check})
    this.state.onChange(!this.state.check)
  }

  render() {
    let {
      title,
      //check,
      //onChange,
      active,
    } = this.props;
    return (
      <div className={`block-check-box ${this.state.check ? 'check' : ''}
${active ? '' : 'disabled'} `}
        onClick={() => active ? this.onChange() : null}>
        <span className="block-check"/>
        <p className="title-check-box">{title}</p>
      </div>
    );
  }
}

import React from 'react';

export default class index extends React.Component {
  constructor(props) {
    super(props);
    this.state = {}
  }

  render() {

```

```

    let {icon} = this.props;
    return (
      <svg
        viewBox={icon.viewbox}
      >
        <path
          d={icon.d}
        />
      </svg>
    );
  }
}
import React from 'react';
import './style.css'

export default class index extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: this.props.value ? this.props.value : "",
      onChange: this.props.onChange
    }
  }

  onChange = (e) => {
    this.setState({value: e.target.value})
    this.state.onChange(e.target.value)
  }

  render() {
    let {
      type,
      name,
      title,
      // value,
      placeholder,
      // onChange,
      error
    } = this.props;
    let height = 27;
    if (title){
      height += 18;
    }
    if (error){

```

```

        height += 18;
    }
    return (
      <div className={`block-input ${error && 'error'}`}
style={{height:height+'px'}}>
        {title &&
        <p className="title-input">{title}</p>
        }
        <input type={type} name={name}
          className={this.state.value ? 'active' : ''}
          value={this.state.value}
          placeholder={placeholder}
          onChange={(e) => this.onChange(e)}>
          {error &&
          <p className="title-error">{error}</p>
          }
        </div>
    );
  }
}
import React from 'react';
import './style.css'

```

```

export default class index extends React.Component {
  constructor(props) {
    super(props);
    this.state = {}
  }

```

```

  render() {
    let {
      position, // fixed, absolute, relative

    } = this.props
    return (
      <div
        className={`container-orbit-spinner ${position === 'fixed' ? 'fix' :
position === 'absolute' ? 'absolute' : 'relative'}`}>
        <div className="orbit-spinner">
          <div className="orbit"/>
          <div className="orbit"/>
          <div className="orbit"/>

```

```

        </div>
      </div>
    );
  }
}
import React from 'react';
import './style.css'
import ButtonLeftMenu from '../buttonLeftMenu/buttonLeftMenu'

export default class menuLeft extends React.Component {
  constructor(props) {
    super(props);
    this.state = {}
  }

  update = (path) => {
    const url = window.location.pathname.split('/');
    url[url.length - 1] = path;
    console.log('test' + url.join('/'));
    // window.location.href = url.join('/') === '/' ? 'test' + url.join('/') : " +
    url.join('/')
  };

  render() {
    let {
      dataMenuLeft,
      onSizeMenu,
      fullMenu,
      location
    } = this.props;
    return (
      <div className="block-menu-left">
        {dataMenuLeft.map((item, key) => {
          return (
            <ButtonLeftMenu
              key={key}
              title={item.title}
              subLinkList={item.subLinkList}
              icon={item.icon}
              typeIcon={item.typeIcon}
              fullMenu={fullMenu}
              location={location}
              countRound={item.countRound}

```

```

        colorRound={item.colorRound}
        path={item.path}
        onPress={null}
        separator={item.separator}
      />
    )
  )))

  <ButtonLeftMenu
    title={'Свернуть меню'}
    icon={'arrow-alt-circle-left'}
    typeIcon={'Solid'}
    fullMenu={fullMenu}
    onPress={() => onSizeMenu()}
    location={location}
    path={null}
  />
</div>
);
}
}
import React from 'react';
import './style.css'
import {Link} from "react-router-dom";
import ButtonTopMenu from "../buttonTopMenu/buttonTopMenu";

export default class menuTop extends React.Component {
  constructor(props) {
    super(props);
    this.state = {}
  }

  render() {
    let {
      dataMenuTop,
      location,
    } = this.props;
    return (
      <div className="block-menu-top">
        <div className="block-left-menu-top">
          {dataMenuTop.menuTop.map((item, key) => {
            return (
              <ButtonTopMenu

```

```

        key={key}
        title={item.title}
        subLinkList={item.subLinkList}
        icon={item.icon}
        typeIcon={item.typeIcon}
        location={location}
        path={item.path}
        hrefLink={item.hrefLink}
        onPress={null}
        separator={item.separator}
    />
)
}}}
</div>
<div className="block-right-menu-top">
    <div className="block-user-button">
        <div className="block-top-user-button">
            {dataMenuTop.userData.path &&
            <div className="block-top-user-button">
                <Link to={dataMenuTop.userData.path}>
                <p className="title-user-button">
                    Привет, {dataMenuTop.userData.userName}
                </p>
                <div className="block-img-user-button"/>
            </Link>
            </div>
            }
            {!dataMenuTop.userData.path &&
            <div className="block-top-user-button"
onClick={dataMenuTop.userData.onPress?dataMenuTop.userData.onPress:null}>
                <p className="title-user-button">
                    Привет, {dataMenuTop.userData.userName}
                </p>
                <div className="block-img-user-button"/>
            </div>
            }
        </div>
        <div className="block-body-user-button">
            <div className="block-left-body-user-button">
                <div className="block-img-user-button"/>
            </div>
            <div className="block-right-body-user-button">
                {
                    dataMenuTop.userData.LinkList &&

```

```

dataMenuTop.userData.LinkList.map((item, key) => {
  if (item.path) {
    return (
      <div className="block-item-button-body-user"
key={key}
style={item.separator ? {margin: '0 0 8px 0'} :
{margin: '0'}}>
      <Link to={item.path}>
        <p className="title-item-button-body-
user">{item.title}</p>
      </Link>
    </div>
    )
  } else {
    if (item.onPress) {
      return (
        <div className="block-item-button-body-user"
key={key} onClick={item.onPress}
style={item.separator ? {margin: '0 0 8px 0'} :
{margin: '0'}}>
        <p className="title-item-button-body-
user">{item.title}</p>
        </div>
      )
    } else {
      return (
        <div className="block-item-button-body-user"
key={key}
style={item.separator ? {margin: '0 0 8px 0'} :
{margin: '0'}}>
        <p className="title-item-button-body-
user">{item.title}</p>
        </div>
      )
    }
  }
})
}
</div>
</div>
</div>
</div>
</div>

```

```

    );
  }
}
import React from 'react';
import './style.css';

export default class index extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      open: this.props.open
    }
  }

  changePanel = () => {
    this.setState({open: !this.state.open})
  }

  render() {
    let {
      title,
      //open
    } = this.props;
    return (
      <div className={`block-panel ${this.state.open ? 'open' : ''}`>
        <div className="block-panel-top" onClick={() => this.changePanel()}>
          <p className="title-panel-top">
            {title}
          </p>
          <span className="arrow-panel-top"/>
        </div>
        <div className="block-panel-body">
          {this.props.children}
        </div>
      </div>
    );
  }
}
import React from 'react';
import './style.css'

export default class index extends React.Component {
  constructor(props) {

```



```

    super(props);
    this.state = {
      check: this.props.check,
      onChange: this.props.onChange
    }
  }

  onChange = () => {
    this.setState({check: !this.state.check})
    this.state.onChange(!this.state.check)
  }

  render() {
    let {
      title,
      check,
      onChange,
      onClick,
      active,
    } = this.props;
    return (
      <div
        className={`block-radio-button ${onChange ? this.state.check ? 'check'
: " : check ? 'check' : ""} ${active ? " : 'disabled'}`}
        onClick={() => onChange ? active ? this.onChange() : null : active ?
onClick() : null}>
        <p className="title-radio-button">{title}</p>
      </div>
    );
  }
}
import React from 'react';
import './style.scss';

export default class index extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      check: this.props.check,
      onChange: this.props.onChange
    }
  }

  onChange = () => {

```

```

    this.setState({check: !this.state.check})
    this.state.onChange(!this.state.check)
  }

  render() {
    let {
      title,
      //check,
      //onChange,
      active,
    } = this.props;
    return (
      <div className={`shell-swipe ${this.state.check ? 'check' : ''} ${active ? '' :
'disabled'} `}
        onClick={() => active ? this.onChange() : null}>
        <div className="block-swipe">
          <span className="swipe"/>
        </div>
        <p className="title-swipe">{title}</p>
      </div>
    );
  }
}
import React from 'react';
import './style.css'

export default class index extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: this.props.value ? this.props.value : "",
      onChange: this.props.onChange
    }
  }

  onChange = (e) => {
    this.setState({value: e.target.value})
    this.state.onChange(e.target.value)
  }

  render() {
    let {
      title,
      // value,

```

```

        placeholder,
        // onChange,
        error,
    } = this.props;
    return (
        <div className="block-textarea">
            <p className="title-textarea">{title}</p>
            <textarea
                className={this.state.value ? 'active' : ''}
                value={this.state.value}
                placeholder={placeholder}
                onChange={(e) => this.onChange(e)} />
            {error &&
                <p className="title-error">{error}</p>
            }
        </div>
    );
}
}
import axios from 'axios'

export const getNotification = (period) => {
    return axios.get(`/v2/news?period=${period}`);
}
export const userLoggedIn = (token) => {
    return {
        type: 'USER_LOGGED_IN',
        token
    }
}

export const getUser = (user) => {
    return {
        type: 'GET_USER_INFO',
        user
    }
}

export const updateUserInfo = (user) => {
    return {
        type: 'UPDATE_USER_INFO',
        user
    }
}

```

```

export const updateFundInfo = (fund) => {
  return {
    type: 'UPDATE_FUND_INFO',
    fund: {...fund}
  }
}

export const updateAngelInfo = (angel) => {
  return {
    type: 'UPDATE_ANGEL_INFO',
    angel: {...angel}
  }
}

export const updateExpertInfo = (expert) => {
  return {
    type: 'UPDATE_EXPERT_INFO',
    expert: {...expert}
  }
}

export const userLoggedOut = () => {
  return {
    type: 'USER_LOGGED_OUT'
  }
}

import axios from 'axios'

export const getComments = (type, id) => {
  return axios.get(`/v2/comments/${type}/${id}`);
}

export const likeUp = (id, data) => {
  return axios.post(`/v2/comments/${id}/like`, data)
}

export const sendComment = (type, id, data) => {
  return axios.post(`/v2/comments/${type}/${id}`, data)
}

export const deleteComment = (id) => {
  return axios.delete(`/v2/comments/${id}`)
}

```

```

export const LOADER_SPIN = "LOADER_SPIN";

export function spin(data) {
  return { type: LOADER_SPIN, payload: data };
}

import axios from 'axios'
import packageJSON from '../package.json'

export const sendFile = (file) => {
  let data = new FormData();
  data.set('file', file);
  return axios.post('/file', data);
}
export const sendFile2 = (blob) => {
  let data = new FormData();
  data.append("file", blob, "imageFilename.png");
  return axios.post('/file', data);
}

export const getFileById = (id) => {
  return dispatch => {
    return axios.get('/file/' + id)
      .then(res => {
        // console.log(res.data)
      })
  }
}

export const getInvestise = (id) => {
  return axios.get('/users/' + id + '/investise')
}
export const getExpertise = (id) => {
  return axios.get('/users/' + id + '/expertise')
}

//User private document
export const getDocs = (id) => {
  return axios.get('/users/' + id + '/documents')
}

```

```

export const sendDoc = (id, fileObj) => {
  return axios.post('/users/' + id + '/documents' , fileObj)
}
export const deleteUserDocument = (id, docId) => {
  return axios.delete('/users/' + id + '/documents/' + docId)
}
export const editUserDocument = (userId, documentId, obj) => {
  return axios.put('/users/' + userId + '/documents/' + documentId, obj)
}

// Project documents
export const getDocsProject = (id) => {
  return axios.get('/projects/' + id + '/documents')
}
export const sendDocProject = (id, fileObj) => {
  return axios.post('/projects/' + id + '/documents' , fileObj)
}
export const deleteDocProject = (projectId, id) => {
  return axios.delete('/projects/' + projectId + '/documents/' + id )
}

export const sendInvestise = (id, fileObj) => {
  return axios.post('/users/' + id + '/investise' , fileObj)
}
export const putInvestise = (id, investiseId, fileObj) => {
  return axios.put('/users/' + id + '/investise/' + investiseId , fileObj)
}
export const deleteInvestise = (id, investiseId) => {
  return axios.delete('/users/' + id + '/investise/' + investiseId)
}

export const sendExpertise = (id, fileObj) => {
  return axios.post('/users/' + id + '/expertise' , fileObj)
}
export const putExpertise = (id, expertiseId, fileObj) => {
  return axios.put('/users/' + id + '/expertise/' + expertiseId , fileObj)
}
export const deleteExpertise = (id, investiseId) => {
  return axios.delete('/users/' + id + '/expertise/' + investiseId)
}

export const uploadAvatar = (userId, file) => {

```

```

var binary = atob(file.split(',')[1]);
var array = [];
for(var i = 0; i < binary.length; i++) {
  array.push(binary.charCodeAt(i));
}
var img = new Blob([new Uint8Array(array)], {type: 'image/jpeg'});

let data = new FormData();
data.set('file', img);
data.set('userId', userId);

return axios.post('/file', data)
}

```

```

export const downloadFileById = (id) => {
  if(id) {
    var a = document.createElement("a");
    document.body.appendChild(a);
    a.setAttribute("target", "_blank");
    a.style = "display: none";
    a.href = packageJSON.proxy + '/file/' + id + '/download';
    a.click();
  }
}

```

```

import axios from "axios";
import {
  userLoggedIn,
  getUser,
  userLoggedOut,
  updateUserInfo,
  updateFundInfo, updateAngelInfo, updateExpertInfo
} from "../AuthActions";
import _ from "lodash";
import { setAuthToken } from "../utils/setAuthToken";
import { goToStep, getQuery } from "../utils/regNavigation";
import { showMessage } from "../utils/showMessage";
import { updateProjectInfo } from "../ProjectActions";
import history from "../history";

```

```

//send data for sign in of user
export const login = (data, ref_b, team_r, team_f, team_a) => {

```

```

    return
    axios.post(`/users/auth?&ref_b=${ref_b}&team_r=${team_r}&team_f=${team_f}&team_a=${team_a}`, data);
  };
  //send data for create account
  export const register = (data, role, ref_b, team_r, team_f, team_a) => {
    return
    axios.post(`/users?role=${role}&ref_b=${ref_b}&team_r=${team_r}&team_f=${team_f}&team_a=${team_a}`, data);
  };
  //send token for sign in via facebook
  export const signUpFacebook = (token, role, ref_b, team_r, team_f, team_a) => {
    return
    axios.get(`/v1/auth/facebook/token?access_token=${token}&role=${role}&ref_b=${ref_b}&team_r=${team_r}&team_f=${team_f}&team_a=${team_a}`);
  };
  //send token for sign in via google
  export const signUpGoogle = (token, role, ref_b, team_r, team_f, team_a) => {
    return
    axios.get(`/v1/auth/google/token?access_token=${token}&role=${role}&ref_b=${ref_b}&team_r=${team_r}&team_f=${team_f}&team_a=${team_a}`);
  };
  //send token for sign in via telegram
  export const signUpTelegram = data => {
    return axios.post("/auth/telegram", data);
  };
  //send email for resetting of password
  export const forgot = email => {
    return axios.post("/users/generate-reset-password-token", { email: email });
  };

  export const resendMail = (email, type) => {
    return axios.post(`/users/activation/resend?email=${email}&type=${type}`);
  };

  //send new password
  export const setNewPassword = (password, token) => {
    return axios.put("/users/reset-password/" + token, { password: password });
  };

  export const tokenLogin = token => {
    //Add header 'Authorization' to every request to the server.
    setAuthToken(token);
    //get request

```



```

return dispatch => {
  return axios
    .get("/users/auth")
    .then(res => {
      const user = res.data;
      localStorage.setItem("userId", user.id);
      if(user.isAdmin) {
        localStorage.setItem('isAdmin', true)
      }
      if (!_isNil(res.data.error)) {
        dispatch(getUser(user));
      }
    })
    .catch(error => {
      localStorage.removeItem("RocketToken");
      setAuthToken(false);
      dispatch(userLoggedOut());
    });
};

export const createRole = (data) => {
  return axios.post(`/v2/experts`, data);
};

export const updateInvestor = (investor = {}, onlyStep, nextStep, goToUrl = null)
=> {
  let obj = investor;
  obj.registrationStep = nextStep
  if(onlyStep) obj = {registrationStep: nextStep, event_type: investor.event_type,
event_data: investor.event_data};
  obj.role = 'investor'

  return dispatch => {
    return axios
      .put("/v2/investor", obj)
      .then(res => {
        dispatch(updateUserInfo(res.data));

        if(goToUrl){
          history.push(`/registration/investor/${nextStep}`)
        } else {

```

```

        goToStep(investor.registrationStep || nextStep)
      }
      // if(onlyStep){
      //   goToStep(expert.registrationStep)
      // } else {
      //   return res
      // }
    })
    .catch(err => {
      goToStep(nextStep);
      showMessage({type: 'error', data: err})
      throw err
    })
  };
};

```

```

export const updateFund = (fund = {}, id, onlyStep, nextStep, goToUrl = true) =>
{
  let obj = fund;
  obj.registrationStep = nextStep;
  if(onlyStep) obj = {registrationStep: nextStep, event_type: fund.event_type,
event_data: fund.event_data};

  return dispatch => {
    return axios
      .put(`/v2/funds/${id}`, obj)
      .then(res => {
        dispatch(updateFundInfo(res.data));
        if(goToUrl){
          history.push(`/create/fund/${nextStep}?id=${id}`)
        } else {
          goToStep(fund.registrationStep || nextStep)
        }
      })
      .catch(err => {
        showMessage({type: 'error', data: err})
        throw err
      })
  };
};

```

```

export const updateAngel = (fund = {}, id, onlyStep, nextStep, goToUrl = true) =>

```

```

{
  let obj = fund;
  obj.registrationStep = nextStep;
  if(onlyStep) obj = {registrationStep: nextStep, event_type: fund.event_type,
event_data: fund.event_data};

  return dispatch => {
    return axios
      .put(`/v2/angels/${id}`, obj)
      .then(res => {
        dispatch(updateAngelInfo(res.data));
        if(goToUrl){
          history.push(`/create/angel_network/${nextStep}?id=${id}`)
        } else {
          goToStep(fund.registrationStep || nextStep)
        }
      })
      .catch(err => {
        showMessage({type: 'error', data: err})
        throw err
      })
  };
};

export const getLastFund = () => {
  return dispatch => {
    const urlParams = new URLSearchParams(window.location.search);
    const id = urlParams.get('id');

    let url = '/v2/funds/last';
    if (id) {
      url = `/v2/funds/my/${id}`;
    }

    return axios
      .get(url)
      .then(res => {
        if(res.data.registrationStep){
          dispatch(updateFundInfo(res.data));
          //dispatch(updateAngelInfo(res.data));
          goToStep(res.data.registrationStep)
          //dispatch(updateFundInfo(res.data));
        } else {
          goToStep(1)
        }
      })
  };
};

```

```

        }
        // dispatch(updateFundInfo(res.data));
        // goToStep(res.data.registrationStep)
    })
    .catch(err => {
        showMessage( {type: 'error', data: err})
        throw err
    })
}

};

};

export const getLastAngel = () => {
    return dispatch => {

        const urlParams = new URLSearchParams(window.location.search);
        const id = urlParams.get('id');

        let url = '/v2/angels/last';
        if (id) {
            url = `/v2/angels/my/${id}`;
        }

        return axios
            .get(url)
            .then(res => {
                if(res.data.registrationStep){
                    dispatch(updateAngelInfo(res.data));
                    goToStep(res.data.registrationStep)
                } else {
                    goToStep(1)
                }
            })
            .catch(err => {
                showMessage( {type: 'error', data: err})
                throw err
            })
    }

};

};

// export const getExpert = (id) => {

```

```

//   return dispatch => {
//     return axios
//       .get(`/v2/experts/${id}`)
//       .then(res => {
//         dispatch(updateExpertInfo(res.data));
//       })
//       .catch(err => {
//         showMessage({type: 'error', data: err})
//         throw err
//       })
//   };
// };

export const createFund = (fund = {}, onlyStep, nextStep) => {
  let obj = fund;
  obj.registrationStep = nextStep;
  if(onlyStep) obj = {registrationStep: nextStep};
  obj.role = 'fund';

  return dispatch => {
    return axios
      .post("/v2/funds", obj)
      .then(res => {
        dispatch(updateFundInfo(res.data));
        goToStep(fund.registrationStep || nextStep)
      })
      .catch(err => {
        showMessage({type: 'error', data: err})
        throw err
      })
  };
};

export const createAngel = (fund = {}, onlyStep, nextStep) => {
  let obj = fund;
  obj.registrationStep = nextStep;
  if(onlyStep) obj = {registrationStep: nextStep};
  obj.role = 'fund';

  return dispatch => {
    return axios
      .post("/v2/angels", obj)
      .then(res => {
        if(res.data){

```

```

        dispatch(updateAngelInfo(res.data));
        goToStep(fund.registrationStep || nextStep)
      }
    })
  ).catch(err => {
    showMessage({type: 'error', data: err})
    throw err
  })
};

export const createStartup = (startup = {}, onlyStep, nextStep) => {
  let obj = startup;
  obj.registrationStep = nextStep;
  if(onlyStep) obj = {registrationStep: nextStep};
  obj.role = 'startup';

  return dispatch => {
    return axios
      .post("/v2/projects", obj)
      .then(res => {
        dispatch(updateUserInfo(res.data));
        goToStep(startup.registrationStep || nextStep)
      })
      .catch(err => {
        showMessage({type: 'error', data: err})
        throw err
      })
  };
};

export const createFromEvent = (data, type) => {
  if(type === 'angel'){
    return axios
      .post("/v2/angels", data)
      .then(res => {
        if(res.data){
          return res.data;
        }
      })
      .catch(err => {
        showMessage({type: 'error', data: err})
        throw err
      })
  }
};

```

```

    })
  }

  if(type === 'fund'){
    return axios
      .post("/v2/funds", data)
      .then(res => {
        return res.data;
      })
      .catch(err => {
        showMessage({type: 'error', data: err})
        throw err
      })
  }

  if(type === 'startup'){
    return axios
      .post("/v2/projects", data)
      .then(res => {
        return res.data;
      })
      .catch(err => {
        throw err
      })
  }

  if(type === 'expert'){
    return axios
      .put("/v2/experts", data)
      .then(res => {
        return res.data;
      })
      .catch(err => {
        showMessage({type: 'error', data: err})
        throw err
      })
  }
};

export const updateExpert = (expert = {}, onlyStep, nextStep, goToUrl = null) =>
{
  // typeof (expert.append) === 'function'
  let obj = expert;
  obj.registrationStep = nextStep

```

```

    if(onlyStep) obj = {registrationStep: nextStep, isFinished: !!expert.isFinished,
event_type: expert.event_type, event_data: expert.event_data};
    obj.role = 'expert'

    return dispatch => {
      return axios
        .put("/v2/experts", obj)
        .then(res => {
          dispatch(updateUserInfo(res.data));
          if(goToUrl){
            history.push(`/registration/expert/${nextStep}`)
          } else {
            goToStep(expert.registrationStep || nextStep)
          }
        })
        .catch(err => {
          showMessage({type: 'error', data: err})
          throw err
        })
    };
  };

export const updateStartup = (startup = {}, onlyStep, nextStep, goToUrl = true) =>
{
  let obj = startup;
  const id = obj.id;

  obj.registrationStep = nextStep;

  if(onlyStep) obj = {registrationStep: nextStep, event_type: startup.event_type,
event_data: startup.event_data};

  obj.role = 'startup';

  return dispatch => {
    return axios
      .put(`/v2/projects/${id}`, obj)
      .then(res => {
        dispatch(updateProjectInfo(res.data));
        if(goToUrl){
          history.push({pathname: `/registration/startup/${nextStep}`, search:
getQuery(id)})
        } else {
          goToStep(obj.registrationStep);
        }
      })
  };
}

```



```

    }

    return res
  })
  .catch(err => {
    //console.log(err)
    //showMessage({type: 'error', data: err})
    throw err
  })
};

};

export const sendInvitation = (emails) => {
  return axios.post("/v2/experts/invite", { emails: emails });
};

export const createAchievementInvestor = (investorId, parameterId, data) => {
  return axios.post(`/v2/investor/${investorId}/achievements/${parameterId}`,
    data);
};

export const editAchievementInvestor = (investorId, parameterId, achId, data) => {
  return
  axios.put(`/v2/investor/${investorId}/achievements/${parameterId}/${achId}`,
    data);
}

export const deleteAchievementInvestor = (investorId, parameterId, achId) => {
  return
  axios.delete(`/v2/investor/${investorId}/achievements/${parameterId}/${achId}`);
}

export const createAchievementInvestorFund = (fundId, parameterId, data) => {
  return axios.post(`/v2/funds/${fundId}/achievements/${parameterId}`, data);
};

export const editAchievementInvestorFund = (investorId, parameterId, achId, data)
=> {
  return
  axios.put(`/v2/funds/${investorId}/achievements/${parameterId}/${achId}`, data);
}

export const deleteAchievementInvestorFund = (investorId, parameterId, achId)
=> {
  return
  axios.delete(`/v2/funds/${investorId}/achievements/${parameterId}/${achId}`);
}

```

```
export const createAchievementInvestorAngel = (fundId, parameterId, data) => {
  return axios.post(`/v2/angels/${fundId}/achievements/${parameterId}`, data);
};
```

```
export const editAchievementInvestorAngel = (investorId, parameterId, achId,
data) => {
  return
  axios.put(`/v2/angels/${investorId}/achievements/${parameterId}/${achId}`,
data);
}
export const deleteAchievementInvestorAngel = (investorId, parameterId, achId)
=> {
  return
  axios.delete(`/v2/angels/${investorId}/achievements/${parameterId}/${achId}`);
}
```

```
export const getAchievement = (role, id, fundId) => {
  switch (role){
    case 'fund':{
      return axios.get(`/v2/funds/${fundId}/achievements/${id}`);
    }
    case 'angel':{
      return axios.get(`/v2/angels/${fundId}/achievements/${id}`);
    }
    default: {
      return axios.get(`/v2/achievements/${role}/${id}`);
    }
  }
};
```

```
export const getAchievementExpert = (role, id, userId) => {
  return axios.get(`/v2/achievements/${role}/${id}/${userId}`);
};
export const getAchievementInvestor = (role, id, userId) => {
  return axios.get(`/v2/achievements/${role}/${id}/${userId}`);
};
```

```
export const createAchievement = (data, parameterId) => {
  return axios.post(`/v2/achievements/expert/${parameterId}`, data);
};
```

```

};

export const createExpertParameter = (data, parameterId) => {
  return axios.post(`/v2/achievements/expert/create/parameter/${parameterId}`,
    data);
};

export const editAchievement = (data, parameterId, achId) => {
  return axios.put(`/v2/achievements/expert/${parameterId}/${achId}`, data);
}
export const deleteAchievement = (parameterId, achId) => {
  return axios.delete(`/v2/achievements/expert/${parameterId}/${achId}`);
}

export const createAchievementStartup = (data, parameterId) => {
  return axios.post(`/v2/achievements/startup/${parameterId}`, data);
};
export const editAchievementStartup = (data, parameterId, achId) => {
  return axios.put(`/v2/achievements/startup/${parameterId}/${achId}`, data);
}
export const deleteAchievementStartup = (parameterId, achId) => {
  return axios.delete(`/v2/achievements/startup/${parameterId}/${achId}`);
}

export const updateUser = (user = {}, id = null) => dispatch => {
  return axios.put(`/users/${id ? id : user.id || 0}`, user)
    .then(res => {
      dispatch(updateUserInfo(res.data));
      return res
    })
    .catch(err => {
      //console.log(err)
      //showMessage({type: 'error', data: err})
      return err
    })
}

export const updateUser2 = (user = {}) => {
  return axios.put(`/users/${user.id || 0}`, user)
}

export const getUserByToken = (token) => {
  return axios.get(`/users/unsubscribe/${token}`)
    .then(res => {

```

```

        return res.data
      }
    ).catch(err => {
      throw err;
    })
  });

export const updateUserByToken = (token, data) => {
  return axios.put(`/users/unsubscribe/${token}`, data)
    .then(res => {
      return res.data
    })
    .catch(err => {
      throw err;
    })
  };

// export const updateUser = (user = {}) => {
//   //request
//   return dispatch => {
//     return axios.put("/users/" + user.id, user)
//       .then(res => {
//         dispatch(updateUserInfo(res.data));
//       })
//       .catch(error => {
//
//       });
//   };
// };

export const logout = () => {
  return dispatch => {
    localStorage.removeItem("RocketToken");
    setAuthToken(false);
    dispatch(userLoggedOut());
  };
};

export const sendEmailToSubscribe = email => {
  return axios.post("/v1/users/subscribe/email", {email: email});
};

```

```

export const share = type => {
  return axios.post('/v1/users/reward', {rewardType: type})
    .then(res => {
      return res
    })
    .catch(err => {
      showMessage({type: 'error', data: err})
    })
};

export const shareLinkedin = data => {
  return axios({
    method: 'post',
    url: 'https://api.linkedin.com/v1/people/~shares?format=json',
    data: data,
    headers: {
      'content-type': 'application/json',
      'x-li-format': 'json'
    }
  });
};

export const twitterAuth = data => {
  return axios.post("/v1/data/twitter", data);
};

export const editInvestor = (investor = {}) => {
  let obj = investor;
  obj.role = 'investor'

  return dispatch => {
    return axios
      .put("/v2/investor", obj)
      .then(res => {
        dispatch(updateUserInfo(res.data));
        return res;
      })
      .catch(err => {
        throw err
      })
  };
};

```

```

export const editExpert = (expert = {}) => {
  let obj = expert;
  obj.role = 'expert';

  return dispatch => {
    return axios
      .put("/v2/experts", obj)
      .then(res => {
        dispatch(updateUserInfo(res.data));
        return res;
      })
      .catch(err => {
        throw err
      })
  };
};

export const checkResetToken = (token) => {
  return axios.get(`/v1/users/check_reset/${token}`)
}
import {updateMenuAction, updateExpertAchMenuAction} from './menuActions';

export const updateMenu = (object) => {
  return dispatch => {
    dispatch(updateMenuAction(object));
  }
};

export const updateExpertAchMenu = (object) => {
  return dispatch => {
    dispatch(updateExpertAchMenuAction(object));
  }
};
export const updateMenuAction = object => {
  return {
    type: "REFRESH_MENU",
    object
  };
};

export const updateExpertAchMenuAction = object => {
  return {
    type: "REFRESH_EXPERT_MENU",

```

```

    object
  };
};
export const newMessage = message => {
  return {
    type: 'NEW_MESSAGE',
    message
  }
}

export const sendMessage = message => {
  return {
    type: 'SEND_MESSAGE',
    message
  }
}

import axios from 'axios'
import {updateUserInfo} from "../AuthActions";
import {showMessage} from "../utils/showMessage";

export const getUserProfile = (id) => {
  return axios.get('/users/' + id)
}

export const updateProfile = (id, data) => dispatch => {
  return axios.put(`/users/${id}`, data)
    .then(res => {
      dispatch(updateUserInfo(res.data));
      return res
    })
}

export const putProfile = (id, data) => {
  return axios.put('/users/' + id, data)
}

export const changePassword = (id, data) => {
  return axios.put('/users/' + id + '/change-password', data)
}

export const getMyPools = (id) => {
  return axios.get('/funds/?userId=' + id)
}

```

```

export const sendKyc = (id, data) => {
  return axios.post('/users/' + id + '/kyc', data)
}

export const getWalletList = (id) => {
  return axios.get('/users/balances')
}

export const getTokens = (contractAddress) => {
  return axios.get('/users/get-tokens?contractAddress=' + contractAddress)
}
export const getRefund = (contractAddress) => {
  return axios.get('/users/refund?contractAddress=' + contractAddress)
}
export const getTransaction = (contractAddress) => {
  return axios.get(`/tx?contractAddress=${contractAddress}`)
}

export const checkCountConfirmation = (hash) => {
  return axios.get(`/tx/check/${hash}`)
}

export const getMyProjects = (id) => {
  return axios.get('/projects?my=1')
}

export const sendExpertInfo = (id, data) => {
  return axios.post('/experts', data)
}

export const getListSkills = (isRecommended) => {
  return axios.get(`/v2/experts/skills?isRecommended=${isRecommended}`)
}
export const getListSkillsByString = (str) => {
  return axios.get(`/v2/experts/skills?search=${str}`)
}

export const getListSkillsInvestor = (isRecommended) => {
  return axios.get(`/v2/investor/skills?isRecommended=${isRecommended}`)
}

export const getListSkillsByStringInvestor = (str) => {
  return axios.get(`/v2/investor/skills?search=${str}`)
}

```



```

export const getListFunds = (str) => {
  return
  axios.get(`/v2/funds?search=${str}&type=simple,angels&isFinished=true`)
}

export const getListParametersInvestor = () => {
  return axios.get(`/v2/investor/parameters`)
}

export const getListParametersFunds = () => {
  return axios.get(`/v2/funds/parameters`)
}
export const getListParametersAngels = () => {
  return axios.get(`/v2/angels/parameters`)
}

export const getListParameters = () => {
  return axios.get(`/v2/experts/parameters`)
}
export const getListParametersStartup = () => {
  return axios.get(`/v2/startup/parameters`)
}

export const getListExperts = (string, skip, take) => {
  if(!string) string = "";
  if(!skip) skip = 0;
  if(!take) take = "";
  return axios.get(`/experts?skip=${skip} + '&take=${take} + '&search=${string}`)
}

export const putExpert = (id, obj) => {
  return axios.put(`/experts/${id}, obj`)
}

export const getKnowledge = () => {
  return axios.get(`/knowledge`)
}
export const getKnowledgeFilter = (obj) => {
  let tags = obj.tags || "",
  search = obj.search || "";
  return axios.get(`/knowledge/?search=${search}&tag=${tags}&newApi=true`)
}

```

```

export const createThing = (obj) => {
  let data = new FormData();
  for (let key in obj) {
    if(obj[key]){
      data.append(key, obj[key]);
    }
  }
  return axios.post('/knowledge', data )
}

export const getKnowledgeOne = (id) => {
  return axios.get(`/knowledge/${id}`)
}

export const deleteKnowledgeOne = (id) => {
  return axios.delete(`/knowledge/${id}`)
}
export const editKnowledgeOne = (data) => {
  // let obj = {
  //   title: data.title
  // }
  return axios.put(`/knowledge/${data.id}`, data)
}

export const senMessage = (data) => {
  return axios.post('/messages', data)
}

export const getPrivateChats = (skip, take) => {
  if(!skip) skip = 0;
  if(!take) take = 20;
  return axios.get(`/messages/chats?skip=${skip}&take=${take}`)
}
export const getPrivateMessages = (id, skip, take) => {
  if(!skip) skip = 0;
  if(!take) take = 20;
  return axios.get(`/messages/user/${id}`)
}

```

```

export const getNotifications = () => {
  return axios.get(`/users/notifications`)
}

export const getBGI = () => {
  return axios.get(`/default/background`)
}

export const getTagsSearch = (input) => {
  return axios.get(`/users/search_tags?search=${input}`)
}

export const sendInfoToServer = (data) => {
  return axios.post(`/default/front-logs`, data)
}

export const tokensale = (data) => {
  return axios.post(`/icos/emails`, data)
}

export const createWorkExp = (userId, data) => {
  return axios.post(`/users/${userId}/experience`, data)
}

export const updateWorkExp = (userId, data) => {
  return axios.put(`/users/${userId}/experience/${data.id}`, data)
}

export const deleteWorkExp = (userId, id) => {
  return axios.delete(`/users/${userId}/experience/${id}`)
}

export const getListWorkExp = (userId) => {
  return axios.get(`/users/${userId}/experience`)
    .catch(err => {
      showMessage({type: 'error', data: err})
    })
}

export const getProfileSettings = (userId) => {
  return axios.get(`/v1/users/settings`)
    .catch(err => {
      showMessage({type: 'error', data: err})
    })
}

export const putProfileSettings = (data) => {
  return axios.put(`/v1/users/settings`, data)
}

```

```

}

export const getCitySearch = (str) => {
  return axios.get(`/v2/default/cities?search=${str}`)
}

export const getCapitals = () => {
  return axios.get(`/v2/default/capitals`)
}

export const getCountrySearch = (str) => {
  return axios.get(`/v2/default/countries?search=${str}`)
}

export const createPosition = (title) => {
  return axios.post(`/v2/positions`, {
    title
  })
}

export const USER_LOGGED_IN = 'USER_LOGGED_IN'
export const USER_LOGGED_OUT = 'USER_LOGGED_OUT'
export const USER_LOGIN_FAILED = 'USER_LOGIN_FAILED'
import axios from 'axios'
import { updateUserInfo } from './AuthActions'

export const signup = (userData) => {
  return axios.post('/users', userData)
}

//send email for resetting of password
export const forgot = (email) => {
  return axios.post('/users/generate-reset-password-token', {email: email})
}

//get list of questions
export const getQuestions = () => {
  return axios.get('/users/questions')
}

```