Министерство образования Республики Беларусь

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

Факультет компьютерных технологий

Кафедра программного обеспечения информационных технологий

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

**ОТЧЕТ**

**по преддипломной практике**

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

Сроки прохождения практики: с 28.10.2019 по 23.11.2019

|  |  |  |
| --- | --- | --- |
| Руководитель практики от предприятия:  \_\_\_\_\_\_\_\_\_\_\_\_А.Ф. Моисеенко  Подпись руководителя М.П. |  | Студент группы 681061  А. Д. Малофеевский  Руководитель практики от БГУИР  И. Л. Калитеня |

Минск 2019

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ

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

REST был правящим правителем более десяти лет. REST – это архитектурный стиль, на котором основано все современное программное обеспечение и веб-сервисы. Многие используеют службы или приложения, созданные на основе API REST, сотни раз на своем телефоне, компьютере или любом устройстве с подключением к Интернету. Однако, несмотря на популярность REST, у него есть некоторые явные недостатки, которые нуждались в исправлении[<https://medium.com/javascript-in-plain-english/the-state-of-rest-in-2019-75005eaf05b9>].

Основные проблемы с REST:

* Нет контроля над данными, которые мы получаем;
* Слишком много HTTP-запросов.

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

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

* обмен данными между различными приложениями вне зависимости от платформы на которой они разработаны;
* бэкенд сервер для реализации веб/мобильных-приложений с бизнес-логикой;
* микро сервисы работающие в реальном времени(блоки из соц. сетей, игры);
* микро сервисы для глубокой аналитики.

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

1. Анализ предметной области
   1. Обзор области применения программного средства

Областью применения, разрабатываемого микро сервисного приложения может являться абсолютно любой, так как основной реализуемые сервисы используются в большинстве [десктопных](https://www.google.com/search?sxsrf=ACYBGNQ5ToJIaFGnEfcyAEJ_b3E2PjhqFg:1574970626593&q=%D0%B4%D0%B5%D1%81%D0%BA%D1%82%D0%BE%D0%BF%D0%BD%D1%8B%D1%85&spell=1&sa=X&ved=2ahUKEwib5erl1o3mAhWFyKYKHdVBD80QkeECKAB6BAgMECk), веб и мобильных приложениях. Пользователями микро сервисного приложения будут являться разработчики, администраторы, менеджеры программных продуктов.

Основные проблемы в реализации данного продукта будут:

* Налаживание без перебойной связи и работы микро-сервисов приложения;
* Контроль доступа к API из вне;
* Защита пользовательских данных;
* Аналитика нотификации для пользователей;
* Организация защиты сессии;
* Масштабируемые уровни доступа;
* Организация оповещения пользователей по почте и соц. сетей;
* Организовать связную работу нескольких микро-сервисов на платформе Node JS;
* ПРИВЕТ
  1. Обзор конкурирующих систем
  2. **Обзор решений для создания проекта**

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

Для реализации микро-сервисов будут использоваться следующие модули:

* Express – минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений;
* bodyParser – Анализирует входящие данные запросов в промежуточном программном обеспечении перед обработчиками;
* cookieParser –
* path – Модуль предоставляет утилиты для работы с файлами и каталогами путями;
* session – модуль для работы с сесиями;
* passport – является проверка подлинности промежуточного слоя для [Node.js](https://nodejs.org/). Гибкий и модульный Passport может быть незаметно вставлен в любое веб-приложение на основе [Express](https://expressjs.com/).
* JWTStrategy – Этот модуль позволяет аутентифицировать конечные точки с помощью веб-токена JSON. Он предназначен для защиты конечных точек RESTful без сеансов.
* pg – Неблокирующий клиент PostgreSQL для Node.js.
* joi – позволяет описывать данные для волидации, используя простой, интуитивно понятный и читаемый язык.
* crypto – Модуль обеспечивает криптографическую функциональность, которая включает в себя набор оберток для хэша OpenSSL, HMAC, шифра, расшифровки, знака, и проверить функции.
* ПРИВЕТ
  1. **Входные данные**

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

Входной информацией будут является:

* информация о пользователе:
* ФИО пользователя;
* фото пользователя;
* дополнительные данные о пользщователе
* ПРИВЕТ
* данные вносимые пользователем администраторами:
* настройка сервисов;
* создание кастомных сервисов;
* ПРИВЕТ
* Данные вносимые рядовым пользователем:
* заполненные формы (их множество проще описать POST запросы на сервер приложения);
* ПРИВЕТ
  1. **Выходные данные**

В качестве выходных данных будет выступать:

* JWT ключе сессии пользователя;
* права доступа пользователя;
* нотификация оповещения пользователя;
* аналитические данные;
* данные в формате картинок или видео;
* данные из сервисов, работающих в реальном времени;
* данные из реализованных и кастомных модулей приложения;
* логи доступа как административной части проекта, так и к пользовательской части;
* ПРИВЕТ
  1. **Постановка задачи на дипломное проектирование**

Функциональным назначением программного средства является автоматизация и упрощенного развертывания серверного приложения основанном на микро сервисах и рабочим сгенерированным API.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* регистрация пользователей;
* надёжное шифрование пароля пользователей;
* создание сессии для пользователей защищённым JWT токеном;
* защита сессии он не санкционированного доступа из вне;
* пользовательская настройка прав доступ к админской части приложения и внешней части приложения;
* выдача прав доступа пользователям;
* парковка домена для защиты от не санкционированного доступа к API проекта;
* защита от парсинга данных с сервера;
* аналитика о просмотре контента уникальными пользователями;
* хранение и рассылка почтовых шаблонов (оповещений) по почте;
* аналитика по рассылкам;
* использование готовых модулей, сервисов;
* создание костомных модулей;
* ограничения доступа к пользовательскому контенту;
* хранение файлов (картинки, видео);
* хранение информации из модулей;
* сервисы работающие в реальном времени;
* ПРИВЕТ

Надежное функционирование программы должно быть обеспечено выполнением совокупности организационно–технических мероприятий, перечень которых приведен ниже:

* организация бесперебойного питания серверного оборудования;
* установка антивирусного программного обеспечения.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения всех условий эксплуатации технических и программных средств.

1. Анализ требований к программному средству
   1. Разработка модели данных

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

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

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

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

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

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

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

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

Связи между сущностями:

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

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

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

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

Диаграмма сущность–связь будет представлена на рисунке 2.1.

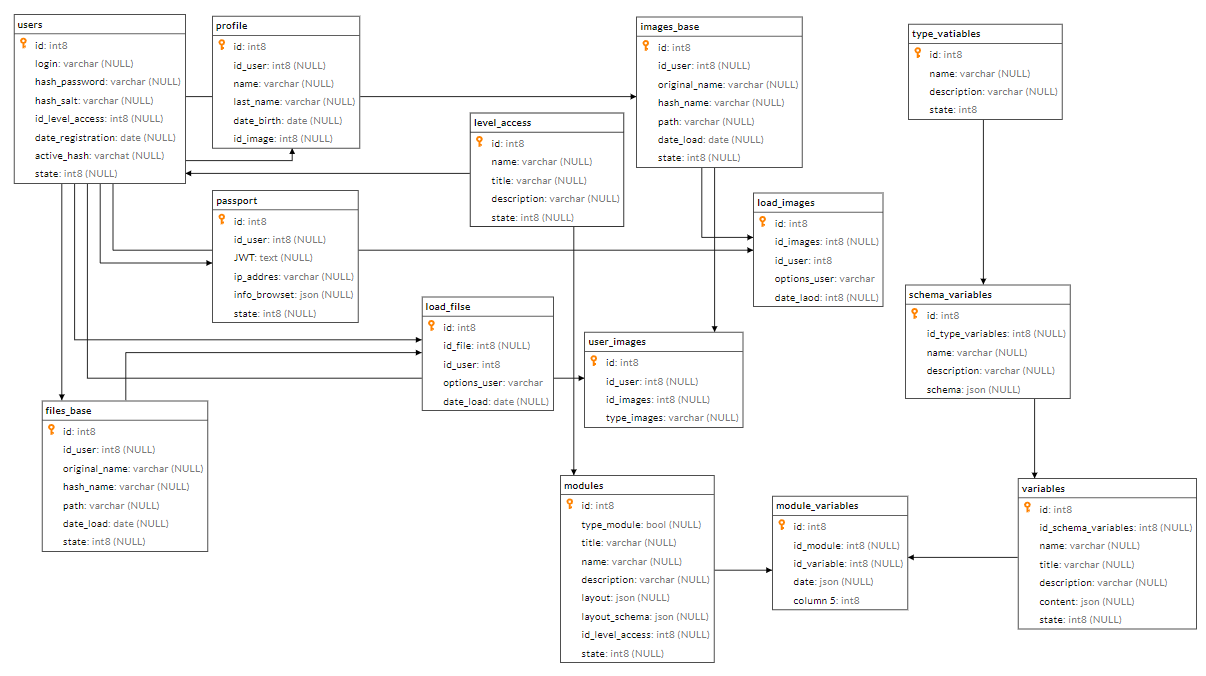


Рисунок 2.1 – Диаграмма сущность–связь

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

* 1. Разработка функциональной модели

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

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

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

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

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

Диаграммы А–0 представлена на рисунке 2.2.

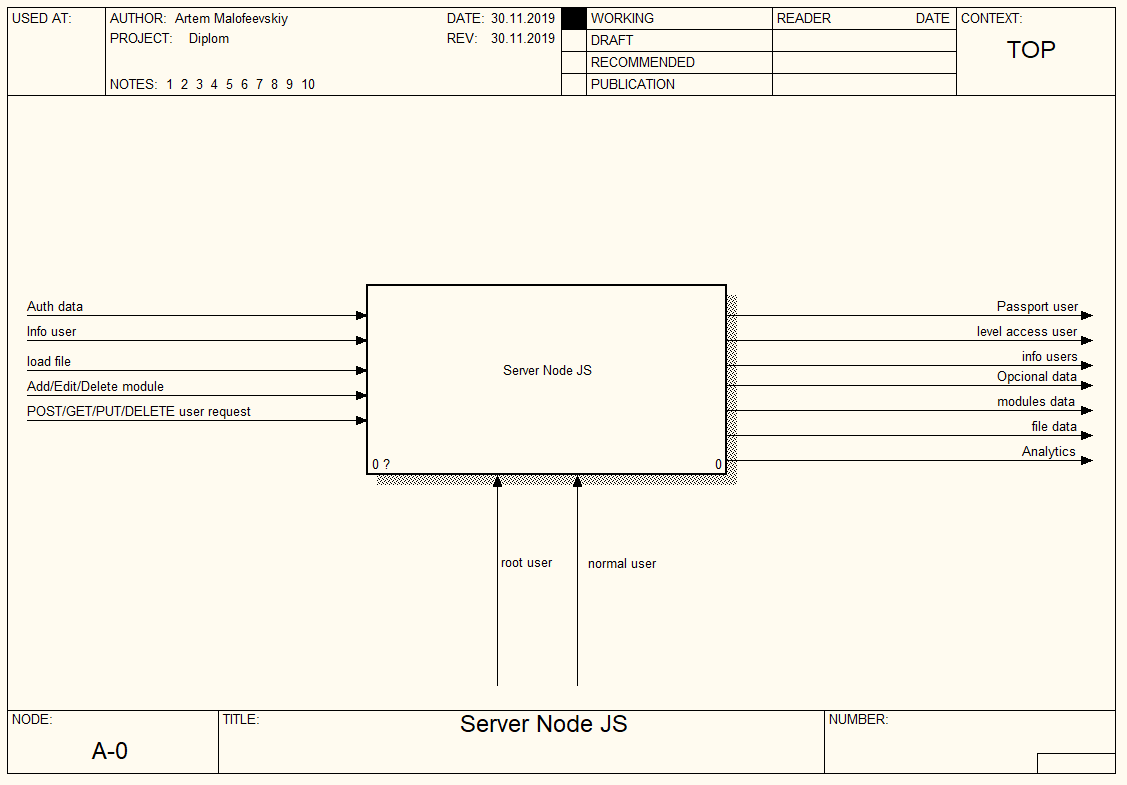


Рисунок 2.2 – Диаграмма А–0

На диаграмме А–0 (IDEF0) изображены выходные данные – отчетность по тренировке и питанию, а также измененная база данных, входные данные – сведения о дневных тренировках, программе тренировок, питании, пользователе.

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

Диаграмма А0 представлена на рисунке 2.3.

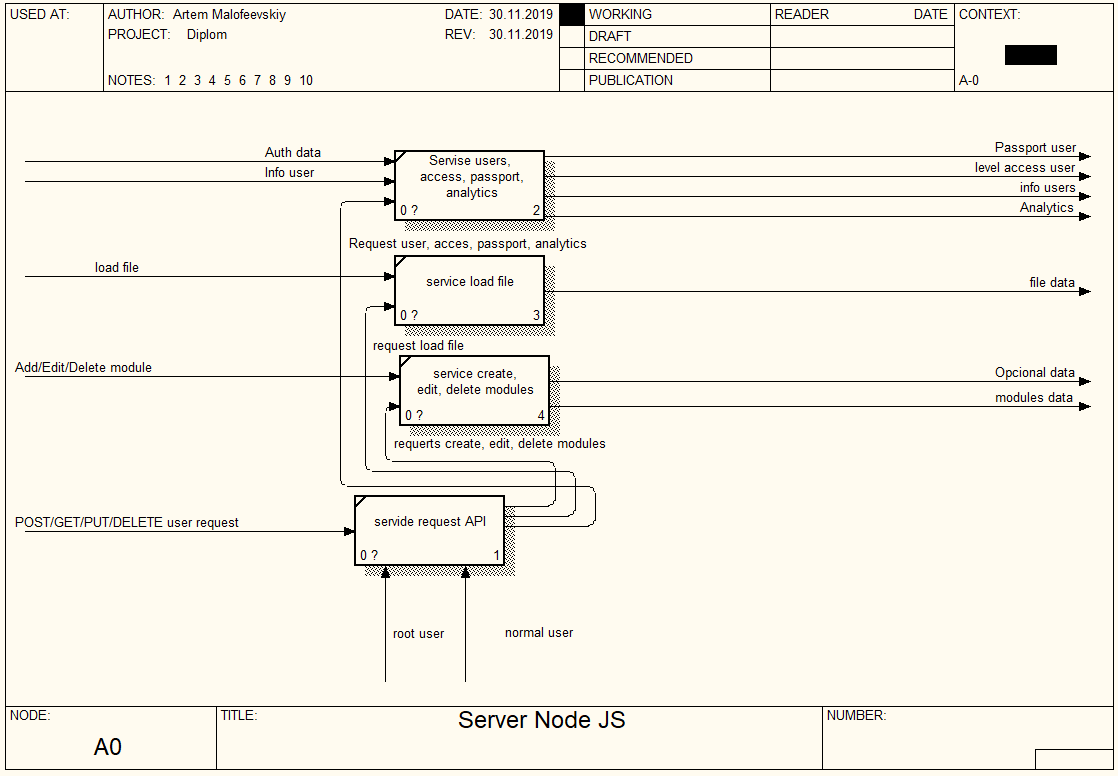


Рисунок 2.3 – Диаграмма А0

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

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

* 1. Диаграмма вариантов использования

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

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

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

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

В программном средстве присутствует два актера пользователь и администратор. Пользователю предоставлены права на выполнение следующих функций:

ПРИВЕТ Как разграничить костомные уровни доступа???

Диаграмма вариантов использования для клиент–серверного приложения представлена на рисунке 2.4.

РИСУНОК

Рисунок 2.4 – Диаграмма вариантов использования

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

* + 1. Описание вариантов использования

Опишем варианты использования «Регистрация», «Авторизация», «Управление пользовательскими данными», «Управление уровнями доступа», «Загрузка файлов», «Управление настройками приложения», «Управление модулями приложения». Для удобства рассмотрения сделаем это в таблице.

В таблице 2.1 представлено описание варианта «Регистрация».

Таблица 2.1 – Описание варианта «Регистрации»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Регистрация | Описывает регистрацию пользователя в системе. Используется, когда пользователь хочет зарегистрирова–ться в системе | Применимо ко всем актёрам | Отсутствуют | При успешной регистрации происходит запись в базу данных, а также шифрование пароля. При неудачном –состояние система выдаёт информацию пользователю |

В таблице 2.1 представлено описание варианта «Авторизации».

Таблица 2.2 – Описание варианта «Авторизации»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Актёры | Предусловия | Постусловия |
| Авторизация | Описывает вход в систему. Используется, когда пользователь хочет войти в систему | Применимо ко всем актёрам | Отсутствуют | При успешном выполнении происходит вход в систему. При неудачном –состояние системы выдаёт соответствующую информацию. |

Варианты использования разрабатываемой системы будут также использованы на этапе тестирования программного средства. И лягут в основу функциональных тестов, которые определят, насколько программа соответствует функциональным требованиям.

1. Техническое проектирование программного средства
   1. **Выбор инструментов разработки**

Инструментами разработки будут являться:

* JavaScript [мультипарадигменный](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Поддерживает [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [императивный](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [функциональный](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) стили;
* Node JS программная платформа, основанная на движке [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) (транслирующем [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения;
* React JS [JavaScript](https://ru.wikipedia.org/wiki/JavaScript)-[библиотека](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript) с [открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для разработки [пользовательских интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F);
* декларативный язык программирования SQL;
* свободная [объектно-реляционная](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94) система управления базами данных «PostgreSQL»;
* интегрированная среда разработки программного обеспечения JetBrains PhpStorm IDEA;
* серверное оборудование с операционной системой Ubuntu server 18.04.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам [приложений](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). Наиболее широкое применение находит в [браузерах](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) как язык сценариев для придания [интерактивности](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [веб-страницам](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0).

Основные архитектурные черты: [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [слабая типизация](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B0%D0%B1%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [прототипное программирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), функции как [объекты первого класса](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0).

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования [непрограммистами](https://ru.wiktionary.org/wiki/%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C-%D0%BD%D0%B5%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82). Языком JavaScript не владеет какая–либо компания или организация, что отличает его от ряда языков программирования, используемых в веб–разработке.

Node.js добавляет возможность [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) взаимодействовать с устройствами [ввода–вывода](https://ru.wikipedia.org/wiki/%D0%92%D0%B2%D0%BE%D0%B4-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) через свой [API](https://ru.wikipedia.org/wiki/API) (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript–кода. Node.js применяется преимущественно на сервере, выполняя роль [веб–сервера](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи [NW.js](https://ru.wikipedia.org/wiki/NW.js), AppJS или [Electron](https://ru.wikipedia.org/wiki/Electron) для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel и espruino). В основе Node.js лежит [событийно–ориентированное](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [асинхронное (или реактивное)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) программирование с [неблокирующим вводом/выводом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B0%D1%8F_%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F).

React разрабатывается и поддерживается [Facebook](https://ru.wikipedia.org/wiki/Facebook), [Instagram](https://ru.wikipedia.org/wiki/Instagram) и сообществом отдельных разработчиков и корпораций.

React может использоваться для разработки [одностраничных](https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и мобильных приложений. Его цель – предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как [Redux](https://ru.wikipedia.org/wiki/Redux).

Декларативный язык программирования «SQL». SQL Structured Query Language – язык структурированных запросов – универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL–запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую.

Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка.

С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать решает СУБД непосредственно при обработке SQL запроса.

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 ПК.

PostgreSQL является одной из наиболее популярных систем управления базами данных. PostgreSQL cуществует в реализациях для множества [UNIX–подобных](https://ru.wikipedia.org/wiki/UNIX-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) платформ, включая [AIX](https://ru.wikipedia.org/wiki/AIX), различные [BSD–системы](https://ru.wikipedia.org/wiki/BSD), [HP–UX](https://ru.wikipedia.org/wiki/HP-UX), [IRIX](https://ru.wikipedia.org/wiki/IRIX), [Linux](https://ru.wikipedia.org/wiki/Linux), [macOS](https://ru.wikipedia.org/wiki/MacOS), [Solaris](https://ru.wikipedia.org/wiki/Solaris)/[OpenSolaris](https://ru.wikipedia.org/wiki/OpenSolaris" \o "OpenSolaris), [Tru64](https://ru.wikipedia.org/wiki/Tru64), [QNX](https://ru.wikipedia.org/wiki/QNX), а также для [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows).

PostgreSQL базируется на языке [SQL](https://ru.wikipedia.org/wiki/SQL) и поддерживает многие из возможностей стандарта SQL:2011.

В PostgreSQL версии 9.5.3 есть следующие ограничения:

* Максимальный размер базы данных не ограничен;
* Максимальный размер таблицы 32 [Тбайт](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%B0%D0%B1%D0%B0%D0%B9%D1%82);
* Максимальный размер записи 1,6 Тбайт;
* Максимальный размер поля 1 Гбайт;
* Максимальное кол–во записей в таблице не ограничено;
* Максимум полей в записи 250 – 1600, в зависимости от типов полей;
* Максимальное кол–во индексов в таблице не ограничено.

Сильными сторонами PostgreSQL считаются:

* высокопроизводительные и надёжные механизмы [транзакций](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [репликации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0));
* расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются [PL/pgSQL](https://ru.wikipedia.org/wiki/PL/pgSQL), [PL/Perl](https://ru.wikipedia.org/wiki/PL/Perl), [PL/Python](https://ru.wikipedia.org/wiki/PL/Python) и [PL/Tcl](https://ru.wikipedia.org/wiki/PL/Tcl); дополнительно можно использовать [PL/Java](https://ru.wikipedia.org/w/index.php?title=PL/Java&action=edit&redlink=1), [PL/PHP](https://ru.wikipedia.org/w/index.php?title=PL/PHP&action=edit&redlink=1), [PL/Py](https://ru.wikipedia.org/w/index.php?title=PL/Py&action=edit&redlink=1), [PL/R](https://ru.wikipedia.org/w/index.php?title=PL/R&action=edit&redlink=1), [PL/Ruby](https://ru.wikipedia.org/w/index.php?title=PL/Ruby&action=edit&redlink=1), [PL/Scheme](https://ru.wikipedia.org/wiki/PL/Scheme), [PL/sh](https://ru.wikipedia.org/w/index.php?title=PL/sh&action=edit&redlink=1) и [PL/V8](https://ru.wikipedia.org/w/index.php?title=PL/V8&action=edit&redlink=1), а также имеется поддержка загрузки модулей расширения на языке[C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F));
* [наследование](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5));
* возможность индексирования геометрических объектов и наличие базирующегося на ней расширения [PostGIS](https://ru.wikipedia.org/wiki/PostGIS);
* встроенная поддержка [слабоструктурированных данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B0%D0%B1%D0%BE%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5) в формате [JSON](https://ru.wikipedia.org/wiki/JSON) с возможностью их индексации;
* расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

PhpStorm — коммерческая кросс–платформенная [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для PHP. Разрабатывается компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains) на основе платформы [IntelliJ IDEA](https://ru.wikipedia.org/wiki/IntelliJ_IDEA).

PhpStorm представляет собой интеллектуальный редактор для [PHP](https://ru.wikipedia.org/wiki/PHP), [HTML](https://ru.wikipedia.org/wiki/HTML) и [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) с возможностями анализа кода на лету, предотвращения ошибок в коде и автоматизированными средствами [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) для PHP и JavaScript. [Автодополнение](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5) кода в PhpStorm поддерживает спецификацию PHP 5.3, 5.4, 5.5, 5.6, 7.0, 7.1 и 7.2 (современные и традиционные проекты), включая генераторы, сопрограммы, пространства имен, замыкания, типажи и синтаксис коротких массивов. Имеется полноценный [SQL](https://ru.wikipedia.org/wiki/SQL)–редактор с возможностью редактирования полученных результатов запросов.

PhpStorm разработан на основе платформы [IntelliJ IDEA](https://ru.wikipedia.org/wiki/IntelliJ_IDEA), написанной на [Java](https://ru.wikipedia.org/wiki/Java). Пользователи могут расширить функциональность среды разработки за счет установки плагинов, разработанных для платформы IntelliJ, или написав собственные плагины.

Ubuntu [операционная система](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), основанная на [Debian GNU/Linux](https://ru.wikipedia.org/wiki/Debian_GNU/Linux). Основным разработчиком и спонсором является компания [Canonical](https://ru.wikipedia.org/wiki/Canonical). В настоящее время проект активно развивается и поддерживается [свободным сообществом](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%BE).

По утверждениям Canonical, Ubuntu используется примерно 20 миллионами пользователей. Он является 1–м в списке самых популярных дистрибутивов Linux для веб–серверов. По количеству пользователей, посетивших сайт [DistroWatch.com](https://ru.wikipedia.org/wiki/DistroWatch.com) (на 2017 год), занимает 4–е место.

Обычно новые версии дистрибутива выходят каждые полгода и поддерживаются обновлениями безопасности в течение 9 месяцев (начиная с версии 13.04, до этого поддержка осуществлялась в течение полутора лет).

Версии LTS, выпускаемые раз в 2 года, поддерживаются в течение 5 лет — как [серверные](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)), так и десктопные варианты. (До версии 12.04 LTS срок поддержки для десктопных LTS–версий составлял 3 года.) На другие дистрибутивы LTS семейства Ubuntu действует полная поддержка в 3 года, а для основы системы ([ядро](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux), [Xorg](https://ru.wikipedia.org/wiki/X_Window_System) и прочие компоненты) — 5 лет.

Ubuntu поставляется с подборкой программного обеспечения для [серверов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) и [рабочих станций](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B1%D0%BE%D1%87%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D0%BD%D1%86%D0%B8%D1%8F). Она устанавливается на [настольные персональные компьютеры](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%82%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) c помощью [Live CD](https://ru.wikipedia.org/wiki/Live_CD) (версия Desktop), [Live USB](https://ru.wikipedia.org/wiki/Live_USB) или текстового установщика (версия Alternate, предоставлялась до версии Ubuntu 12.04.2). В версии [Live DVD](https://ru.wikipedia.org/wiki/Live_DVD) присутствуют несколько бóльшие возможности — начиная от установки не только в графическом, но и в [текстовом](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%BE%D0%B9_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8) режимах, загрузки в режиме восстановления системы и заканчивая полной [локализацией](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) и бóльшим количеством [пакетов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D1%8B%D1%85_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC) на диске. Есть версии для официально поддерживаемых архитектур, таких как [i386](https://ru.wikipedia.org/wiki/I386), [AMD64](https://ru.wikipedia.org/wiki/AMD64), [ARM](https://ru.wikipedia.org/wiki/ARM_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)).

Существует редакция Ubuntu Core, которая может работать на [IoT](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D0%B2%D0%B5%D1%89%D0%B5%D0%B9)–устройствах, и на [роботах](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B1%D0%BE%D1%82).

Ubuntu Server – это бесплатная серверная операционная система на базе ядра Linux. Ubuntu Server можно использовать в качестве платформы для Web–серверов, серверов баз данных, DNS–серверов, файловых серверов и других типов серверов. Ubuntu очень популярный дистрибутив Linux, в том числе и серверный вариант, который активно используется организациями разных размеров, за счет того, что главной особенностью Ubuntu Server, да и всех серверных операционных систем на базе Linux, является надежность, производительность и безопасность.

* 1. Разработка диаграммы развертывания

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

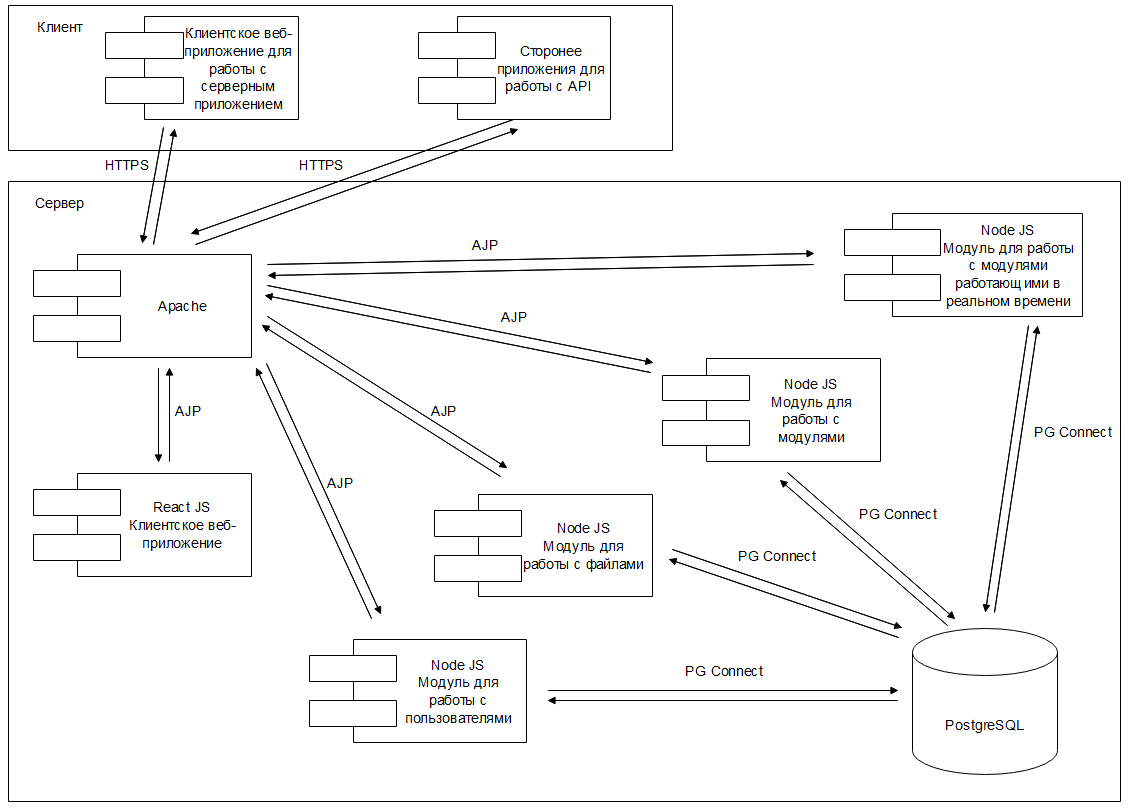


Рисунок 3.1 ­ Диаграмма развертывания

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

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

Серверу передается запрос, который в свою очередь после обработки передается на СУБД. Далее из базы данных на сервер поступают данные, после чего данные передаются клиенту. Специфика конкретной СУБД может включать в себя ограничения наименование объектов базы данных, ограничения на поддерживаемые типы данных.

* 1. Схема работы программы

При проектировании программного средства была разработана функциональная модель. На основе этой модели были построены алгоритмы. Общая схема работы программного средства представлена на рисунке 3.2.

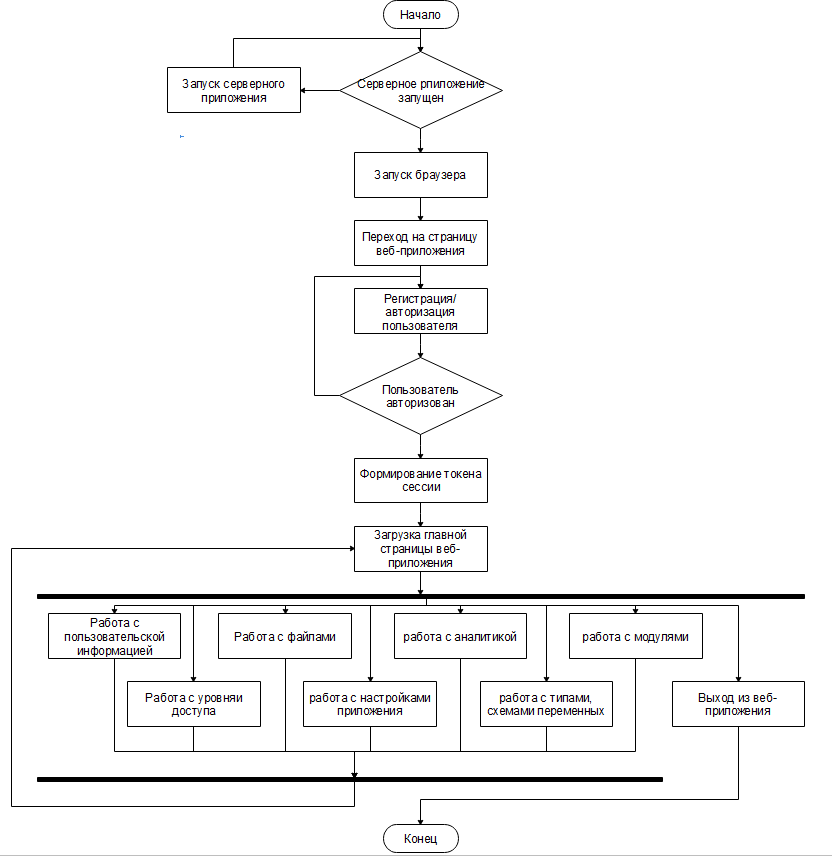


Рисунок 3.2 – Общая блок–схема работы программного средства

Блок–схема описывает полный цикл работы программы от запуска да закрытия, то есть также все функции, которые предоставляет программное средство.

* + 1. Алгоритм регистрации

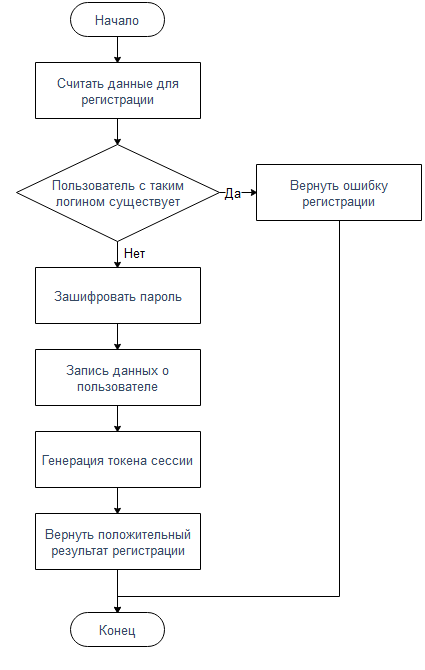


Рисунок 3.3 – Блок–сема алгоритма регистрации

* + 1. Алгоритм авторизации и права доступа

В начале работы с программным средством необходимо выполнить авторизацию и проверку прав пользователя. Блок–сема алгоритма авторизации представлена на рисунке на рисунке 3.3.

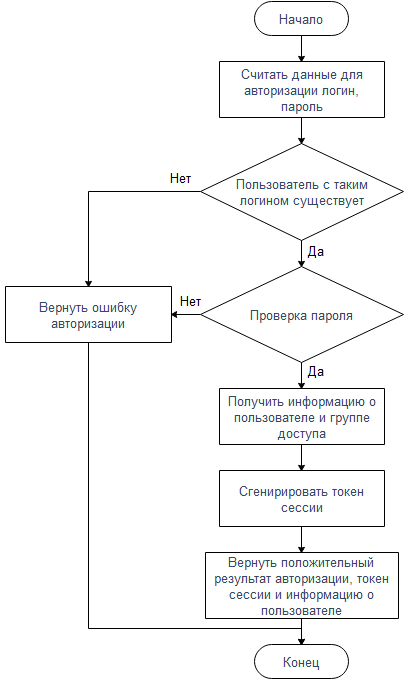


Рисунок 3.4 – Блок–сема алгоритма авторизации

Перед авторизацией необходимо проверить если ли доступная сесия для клиента, елси сессия доступна – авторизация непонадобится, если сессия не доступна – будет произведен ввод логина и пароле после проверики осуществится будет осуществлен вход в программу.

* + 1. Алгоритм работы создание токена сессии

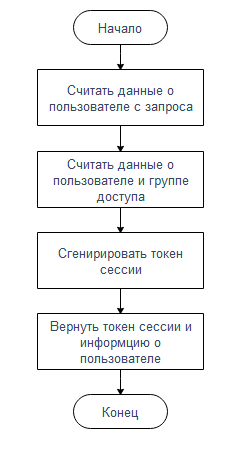


Рисунок 3.5 – Блок–сема алгоритма создание токена сессии

* + 1. Алгоритм работы проверки токена сессии

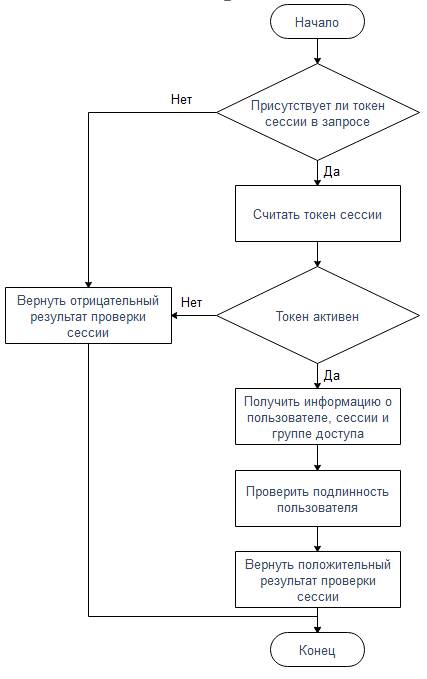


Рисунок 3.6 – Блок–сема алгоритма проверка токена сессии

* + 1. Алгоритм работы с пользовательской информации
    2. Алгоритм работы с настройками приложения

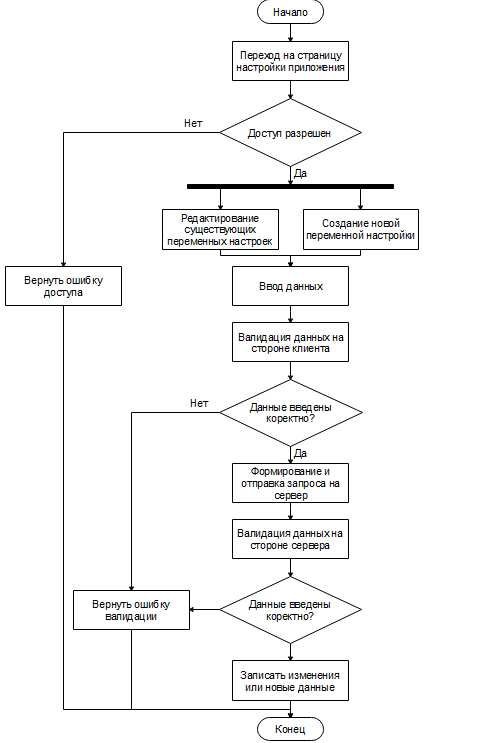


Рисунок 3.7 – Блок–сема алгоритма работы с настройками приложения

* + 1. Алгоритм работы с файлами
    2. Алгоритм работы с правами доступа
    3. Алгоритм работы аналитикой
    4. Алгоритм работы с переменными
    5. Алгоритм работы с модулями
  1. Разработка физической модели данных

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

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

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

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

Таблица «users» содержит информацию о пользователях.

Таблица 3.1 – Структура таблицы «users»

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

Таблица «profiles» содержит подробную информацию о пользователях.

Таблица 3.2 – Структура таблицы «profiles»

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

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

Таблица 3.3 – Структура таблицы «passport»

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

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

Таблица 3.4 – Структура таблицы «level\_access»

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

Таблица «images\_base» содержит информацию о картинках.

Таблица 3.5 – Структура таблицы «images\_base»

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

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

Таблица 3.6 – Структура таблицы «load\_images»

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

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

Таблица 3.7 – Структура таблицы «user\_images»

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

Таблица «files\_base» содержит информацию о файлах.

Таблица 3.8 – Структура таблицы «files\_base»

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

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

Таблица 3.9 – Структура таблицы «load\_files»

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

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

Таблица 3.10 – Структура таблицы «type\_variables»

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

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

Таблица 3.11 – Структура таблицы «schema\_variables»

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

Таблица «variables» содержит информацию о переменных.

Таблица 3.12 – Структура таблицы «variables»

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

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

Таблица 3.13 – Структура таблицы «type\_modules»

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

Таблица «modules» содержит информацию о модулях.

Таблица 3.14 – Структура таблицы «modules»

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

Таблица «module\_variables» содержит информацию о модулях.

Таблица 3.12 – Структура таблицы «modules\_variables»

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

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ