

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»
Магистерская программа: «Вычислительные машины, комплексы, системы и сети»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

**АРХИВИРОВАНИЕ ВИРТУАЛЬНЫХ
МАШИН В КЛАСТЕРЕ**

Студент Посадский Андрей Вадимович (ФИО полностью)
(подпись, дата) 29.05.2023
Руководитель К.В. Попов (фамилия, инициалы)
(подпись, дата) 29.05.2023
Нормоконтролёр А.С. Бычков (фамилия, инициалы)
(подпись, дата) 31.05.2023
Рецензент С.В. Петерский (фамилия, инициалы)
(должность, место работы) доцент ПензГТУ (подпись, дата) 02.06.2023

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 г. № 14)

Заведующий кафедрой М.А. Митрохин (фамилия, инициалы)
(подпись)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

9.06.2023 № 7)

Секретарь ГЭК Е.И. Гурин

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

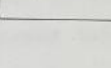
Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 29 » 11 2021г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

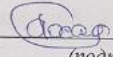
1. Студент Посадский Андрей Вадимович гр. 21BВВМ1 факультета вычислительной техники.
2. Тема работы «Архивирование виртуальных машин в кластере»
Тема утверждена приказом ПГУ №924/о от "29" ноября 2021 г.
3. Руководитель работы к.т.н., доцент кафедры ВТ ПГУ Попов К.В.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Архиватор должен представлять собой распределённые приложения, управляющая часть которого располагается на управляющей виртуальной и физической машине, работающей под ОС Linux, исполнительная часть на мастере кластера, работающего под ОС хер-ng. В работе должен быть проведён анализ существующих систем архивирования виртуальных машин, рассмотрены принципы построения подобных систем, проведён анализ и выбор инструментальных средств для разработки распределённого ПО. Определены принципы взаимодействия серверной и клиентской части приложения.
5. Перечень подлежащих разработке вопросов
Анализ команд ОС хер-ng, выполняющих копирование виртуальных машин, получение snapshot, формирование template из snapshot, установка соединения с хранилищем для хранения template и восстановления виртуальной машины из template. Разработка базы данных для хранения планов и результатов работы приложения при архивировании и восстановлении виртуальных машин. Разработка интерфейса для управления заданиями и просмотра результатов.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ существующих систем архивирования	5%	20.12.2021	
2.	Анализ средств разработки web-приложений	5%	20.12.2021	
3.	Анализ существующих СУБД и средств разработки БД	5%	20.06.2022	
4.	Анализ команд ОС x86-ng	5%	20.06.2022	
5.	Разработка серверной части системы	10%	20.12.2022	
6.	Разработка БД	20%	20.12.2022	
7.	Разработка web-приложения	30%	20.02.2023	
8.	Комплексная отладка системы архивирования	10%	20.05.2023	
9.	Оформление документации	10%	31.05.2023	

Дата выдачи задания "29" ноября 2021 г.

Руководитель работы 
(подпись, дата)

Задание к исполнению принял студент 
(подпись, дата)

Работу к защите допустить
Декан факультета  Д. Р. Ревина 05.06.2023
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 154 листов, 54 рисунка, 24 таблицы, 9 источников, 6 приложений.

БЕКАП, XILINX, VUE.JS, Laravel, СУБД PostgreSQL, PHPMYADMIN, РЕЗЕРВНОЕ КОПИРОВАНИЕ, SQL, БАЗА ДАННЫХ, UBUNTU, CRON.

Цель выпускной квалификационной работы - разработка автоматизированной системы для создания и учета резервных копий в кластере виртуальных машин.

Разработанная автоматизированная система предназначена для создания резервных копий виртуальных машин в автоматическом и ручном режимах, она позволяет настраивать параметры резервного копирования и имеет клиент-серверную архитектуру. Для хранения данных в автоматизированной системе используется база данных.

Разработанная автоматизированная система была реализована на языках JavaScript, PHP, языке гипертекстовой разметки HTML, каскадных таблиц стилей CSS, а также с использованием веб-фреймворка Vue.js и серверного фреймворка Laravel. Для хранения данных использовалась СУБД PostgreSQL.

Для инфологического проектирования было использовано CASE-средство StarUML, для проектирования базы данных использовался инструмент OpenModelSphere.

ПГУ1.09.04.01.09.001 ПЗ

Изм.	Лист	№ докум.	Подпись	Дата	Лит.	Лист	Листов
Разраб.		Посадский А.В.				2	154
Провер.		Попов К.В.			Архивирование виртуальных машин в кластере Пояснительная записка		
Н. Контр.		Бычков А.С.					
Реценз.		Петерский					
Утверд.		Митрохин М.А.			ФВТ, каф. ВТ гр. 21БВВм1		

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ	5
ВВЕДЕНИЕ	6
1 Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин	7
1.1 Анализ предметной области	7
1.1.1 Анализ существующих решений в России	8
1.1.2 Постановка задачи	11
1.1.3 Выбор инструментов разработки	11
1.2 Инфологическое проектирование	15
1.2.1 Разработка диаграммы пакетов АС	15
1.2.2 Разработка диаграммы вариантов использования АС	16
1.2.3 Разработка диаграммы последовательности АС	16
1.2.4 Разработка диаграмм деятельности АС	17
1.2.5 Разработка диаграммы классов	21
1.2.6 Разработка диаграммы развертывания	22
1.3 Проектирование базы данных	22
1.3.1 Разработка концептуальной модели данных	23
1.3.2 Разработка логической модели данных	27
1.3.3 Нормализация базы данных	31
1.3.4 Разработка физической модели данных	35
1.4 Разработка приложения	38
1.4.1 Анализ функций приложения	38
1.4.2 Разработка логической структуры приложения	38
1.4.3 Описание модулей клиентской части приложения	40
1.4.4 Описание модулей серверной части приложения	48
1.5 Описание применения	56
1.6 Тестирование приложения	64
ЗАКЛЮЧЕНИЕ	66

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	67
ПРИЛОЖЕНИЕ А Листинг кода БД.....	68
ПРИЛОЖЕНИЕ Б Листинг программы	82
ПРИЛОЖЕНИЕ В Диаграмма классов	135
ПРИЛОЖЕНИЕ Г Схема данных	136
ПРИЛОЖЕНИЕ Д Схема функции handle.....	137
ПРИЛОЖЕНИЕ Е Схема функции processingCluserPlans	138
ПРИЛОЖЕНИЕ Ж Схема функции parseCurrentMachinesStates	139
ПРИЛОЖЕНИЕ З Презентация	140

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

В настоящей работе применяются следующие термины с соответствующими определениями

Снепшот (snapshot) – снимок виртуальной машины

Гипервизор – система, обеспечивающая параллельное выполнение нескольких операционных систем на одном хост-компьютере

Безагентное резервное копирование – способ резервного копирования без установки клиента внутри гостевой системы

Кластер виртуальных машин – набор виртуальных машин, объединенных в виртуальную сеть

КТС – комплекс технических средств

СУБД - система управления базами данных

РК - (англ. Primary Key)-первичный ключ

FK - (англ. Foreign Key)-внешний ключ

ВВЕДЕНИЕ

Все больше компаний в последнее время применяют технологии виртуализации для повышения эффективности выполнения различного рода задач. Если раньше виртуальные машины в основном использовались энтузиастами, то теперь качество платформ виртуализации рабочих столов и серверов позволило использовать их в больших масштабах профессионалам.

Возможность запуска нескольких виртуальных систем на одном физическом компьютере дает множество преимуществ, включая экономию оборудования, упрощение обслуживания и снижение затрат на электроэнергию в крупных центрах обработки данных. Кроме того, важным преимуществом виртуальных машин является их легкий перенос на другую физическую платформу и простая процедура резервного копирования.

Но при использовании виртуальных сред, также, как и при использовании обычных операционных систем необходимо позаботиться о различных аспектах безопасности таких систем. Одним из важнейших вопросов безопасности является резервное копирования критически важных данных.

В условиях цифровой экономики резервное копирование данных, жизненно важно для выживания организации [1]. Данные могут быть подвержены краже, утере или порче. Резервное копирование данных — это практика, сочетающая методы и решения для эффективного и экономичного резервного копирования, для защиты критически важных данных в случае бедствия, аварии или злонамеренного действия [1].

Целью данной работы является разработка автоматизированной системы для создания и учета резервных копий в кластере виртуальных машин. Автоматизированная система должна иметь клиент-серверную архитектуру. Клиентская часть должна отвечать за настройку параметров резервного копирования, а также отображение информации о виртуальных машинах, хранилищах и резервных копиях. Серверная часть должна отвечать за создание резервных копий и сохранение настроек резервного копирования.

1 Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин

1.1 Анализ предметной области

Резервное копирование как правило выполняется с определенной периодичностью и в определенном объеме для последующего восстановления системы в случае какого-либо сбоя или вмешательства со стороны, для этого используются различные программные или программно-аппаратные комплексы. Данные комплексы используют два метода резервного копирования: копирование изнутри и копирование виртуальной машины целиком [2].

Первый способ подразумевает использование агента внутри виртуальной машины, который осуществляет копирование посредством самой операционной системы машины. Данный способ привлекателен тем, что одно и то же программное обеспечение можно использовать для копирования физических и виртуальных машин, и нет необходимости обретать его дополнительно. Однако для того, чтобы восстановить систему, необходимо сперва создавать новую виртуальную систему с теми же самыми параметрами и настройками, что значительно увеличивает время восстановления.

Второй способ выполняется средствами монитора виртуальных машин на более низком уровне, что увеличивает скорость таких операций. Поскольку гостевая система представляет собой набор файлов, находящихся в памяти, система резервного копирования должна получить доступ к этим файлам и скопировать их в другое место. В большинстве случаев для полноценного и безопасного копирования используют временный файл, который называется снэпшот - снимок виртуальной машины, который позволяет вернуть машину в рабочее состояние в случае сбоя. Процесс копирования с использованием снимка виртуальной машины представлен на рисунке 1.1, состоит из трех этапов: создание снэпшота, копирование виртуальной машины и удаление (слияние) снимка с машиной.

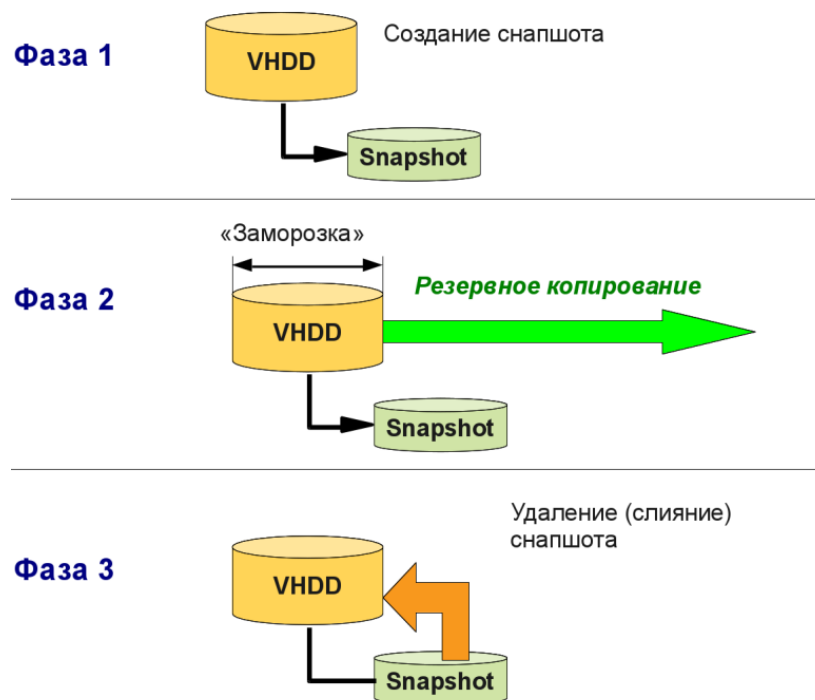


Рисунок 1.1 – Создание копии с использованием снимка

1.1.1 Анализ существующих решений в России

Потребность в использовании инструментов резервного копирования данных описана в нормативных документах Федеральной службы по техническому и экспортному контролю России, а именно приказах № 17, 21 и 31. Для того, чтобы не нарушать требования данных приказов компании должны использовать сертифицированные инструменты резервного копирования и восстановления данных. В настоящее время доступны следующие современные системы защиты, сертифицированные для виртуальных сред в России:

- Acronis Backup & Recovery Advanced Server 11.7,
- Veeam Backup & Replication v9.5,
- Vinchin Backup & Recover.

К плюсам системы Acronis Backup можно отнести широкий спектр поддерживаемых операционных систем и платформ виртуализации, в том числе большинство версий Hyper-V, VMware, а также других гипервизоров в том числе Citrix XenServer. В данной системе присутствует встроенная репликация виртуальных машин, возможность автоматического резервного копирования

виртуальных машин и удаления в соответствии с настроенной политикой удаления, а также возможность масштабирования путем установки агентов на хосты по мере необходимости.

Минусами данной системы является необходимость установки отдельных агентов для каждого хоста в сети для подключения их к системе, что при большом количестве виртуальных машин может быть затруднительным, а также зависимость стоимости подписки на программное обеспечение от количества физических или виртуальных машин в сети.

Система Veeam Backup поддерживает системы виртуализации Hyper-V и VMware. Плюсами данного ПО является поддержка облачных платформ Microsoft и Amazon, поддержка зашифрованных виртуальных машин в vSphere 6.5, возможность масштабирования в распределенных системах и объединения нескольких серверов резервного копирования с централизованным управлением на главном сервере, а также возможность управления нагрузкой на системы хранения данных в процессе резервного копирования.

Поскольку целевой системой виртуализации для разрабатываемой системы резервного копирования является Citrix XenServer, главным минусом Veeam Backup является отсутствие поддержки данного гипервизора.

Vinchin Backup & Recovery - это удобное, надежное и экономичное решение нового поколения для резервного копирования виртуальных машин в разных виртуальных средах, включая VMware, Hyper-V, XenServer, RHV/oVirt, OLVM, XCPng, OpenStack, Sangfor, Huawei FusionCompute, H3C CAS/UIS и ZStack.

Решение Vinchin Backup & Recovery получило высокие оценки от ведущих мировых поставщиков решений в области виртуализации, а также различные сертификаты, включая VMware Ready, Citrix Ready, Redhat certified Technology и др. Компании могут воспользоваться всеми преимуществами сертифицированного ПО для резервного копирования от компании Vinchin для защиты своей среды виртуализации и снижения проектных рисков.

В связи с быстрым ростом объема данных в современном мире и высокими требованиями, которые бизнес закладывает в SLA, защита данных становится все более важной для большинства отраслей, особенно для ИТ-компаний.

Vinchin Backup & Recovery обеспечивает комплексную защиту данных для инфраструктуры виртуализации и даже в прикладных системах. Развертывание Vinchin Backup & Recovery в среде виртуализации позволяет не только защитить все виртуальные машины, но и сэкономить бюджет по сравнению с другими решениями для резервного копирования.

Основные преимущества VINCHIN:

- Vinchin Backup & Recovery проверено и сертифицировано ведущими поставщиками решений по виртуализации, то есть это на 100% стабильное и надежное ПО для резервного копирования, которому можно доверять,
- Защитите свои рабочие нагрузки в локальных, удаленных и даже облачных средах с помощью быстрого резервного копирования и аварийного восстановления, сведя риск перебоев в работе бизнеса практически к нулю,
- Реагирование на любой сервисный запрос в течение 1 часа (в Северной и Латинской Америке - в течение 8 часов) в режиме 24/7. Минимальное время ожидания и максимально качественная поддержка пользователей,
- Бесплатный пробный период для корпоративной полнофункциональной версии составляет 60 дней. Заказчику предоставляется бесплатная поддержка по установке и настройке на период пилотного и внутреннего оценочного тестирования. Так пользователи могут в полной мере оценить продукт перед покупкой.

Ввиду анализа существующих систем резервного копирования, является актуальной задача разработки системы для резервного копирования без необходимости установки агента на виртуальные машины для системы виртуализации Citrix XenServer.

1.1.2 Постановка задачи

Требуется разработать автоматизированную систему для создания и учета резервных копий в кластере виртуальных машин. Автоматизированная система должна иметь клиент-серверную архитектуру и обеспечивать:

- загрузку в АСУ информации о машинах и хранилищах, находящихся в кластере,
- безагентное резервное копирование виртуальных машин, находящихся в кластере,
- периодическое резервное копирование виртуальных машин с заданной периодичностью и сохранением в указанное хранилище, а также возможность немедленного начала резервирования машины в ручном режиме,
- хранение копий виртуальных машин на хранилище и удаление копий в ручном и автоматическом режимах,
- управление настройками резервного копирования, с помощью веб-формы,
- авторизацию в системе.

В системе должна быть реализована веб-форма, которая должна обеспечивать:

- вывод информации о машинах в кластере,
- вывод информации о хранилищах,
- вывод информации о резервных копиях,
- настройка периодичности резервного копирования и количества хранимых копий на хранилище,
- загрузка учетных данных для удаленных хранилищ.

1.1.3 Выбор инструментов разработки

1.1.3.1 Выбор инструментов для разработки серверной части системы

Серверная часть приложения может быть написана на множестве языков, самые популярные серверные языки программирования - это Ruby, PHP, NodeJS (JavaScript), Python и JAVA.

Ruby - интерпретируемый язык высокого уровня, чаще всего используется для написания различных скриптов. Для разработки веб-приложений используется фреймворк Ruby on Rails. Однако ввиду ограниченного набора библиотек расширения, решение многих повседневных задач может отнимать много времени.

Фреймворк - это программная платформа, определяющая внутреннюю структуру будущего проекта и позволяющая значительно сократить время разработки путем разбиения приложения на отдельные составляющие – компоненты, которые выполняют свою узкую задачу, что также увеличивает читаемость кода.

JavaScript - гибкий язык, позволяющий разрабатывать как скрипты для клиентской части сайта так и серверной с помощью программной платформы Node.js. Развита экосистема npm позволяет не тратить время на большинство типовых задач, а брать готовые решения других разработчиков.

Java - один из самых популярных языков программирования. Он универсален и используется уже более 20 лет. Универсальность обеспечивается виртуальной машиной Java (JVM). Во многих языках во время компиляции программа преобразуется в код, который может по-разному работать на разных устройствах или платформах. У Java нет этой проблемы. JVM действует как промежуточный уровень - она создает код из Java-программы, которую можно выполнить на любом компьютере, независимо от того, где код был скомпилирован. Python - последнее время набирает популярность за счет простоты синтаксиса и низкого порога входа, что делает его доступным для новичков. На данном языке возможно разрабатывать как настольные, так и веб-приложения.

PHP – используется на около 80% всех сайтов в интернете. Имеет огромное количество средств для разработки веб-интерфейсов и серверных частей веб-приложений.

В данном проекте для серверной части был выбран язык PHP и фреймворк Laravel по ряду причин:

- развитое сообщество и огромное количество дополнительных средств расширения,
- подход Model-View-Controller, который позволяет разделять на логические части приложение и легче его поддерживать,
- концепция Object-relational mapping, которая упрощает работу с базой данных.

В процессе разработки автоматизированной системы необходимо позаботиться о том, где хранить данные.

В стандартной сборке Laravel позволяет использовать следующие реляционные СУБД:

- MySQL,
- PostgreSQL,
- SQLite,
- SQL Server.

Для организации хранения данных приложения была выбрана СУБД PostgreSQL как самая популярная из всех крупных серверных систем управления базами данных, наиболее легко устанавливаемая и имеющая большую часть функционала SQL.

1.1.3.2 Выбор инструментов для разработки клиентской части

Выбор инструмента для разработки клиентской части осуществлялся по следующим параметрам:

- скорость разработки,
- скорость работы сайта,
- размер итоговой сборки проекта.

Наиболее популярным языком для разработки веб-интерфейсов является JavaScript, было решено выбрать его, так как благодаря большому объему

готовых инструментов для решения типовых задач и популярным фреймворкам возможно значительно сократить время на разработку.

Среди самых популярных можно выделить React.js, Angular и Vue.js.

React – наиболее популярная JavaScript библиотека с открытым кодом, созданная программистами Facebook, позволяющая разрабатывать полноценные динамические сайты. Используется синтаксис, объединяющий в себе JavaScript и язык гипертекстовой разметки HTML. Обладает большим сообществом разработчиков, что помогает при поиске ответов на вопросы.

Angular – веб-фреймворк, позволяющий разрабатывать динамические веб-сайты. Данный фреймворк используется более чем на 400 тысячах сайтов, в том числе на крупных площадках, таких как Google, YouTube и Microsoft.

Vue.js – прогрессивный фреймворк, набирающий популярность в последнее время благодаря низкому порогу входа для начала работы, легкой работы с потоками данных и мощному интерфейсу для командной строки, позволяющему значительно ускорить разработку проекта, легко применяя к проекту нужные настройки.

Сравнение размеров сборок пустых проектов в данных библиотеках и фреймворках приведено в таблице 3.1, можно увидеть, что фреймворк Vue.js является наиболее оптимизированным для сборки, а от итогового размера файлов сайта напрямую зависит скорость его загрузки и удобство работы с ним.

Таблица 1.1 – Сравнение размеров библиотек и фреймворков

Фреймворк/библиотека	Размер, кб
React (v.16)	97.5
Vue (v.2)	58.8
Angular (v.2)	566

Для данного проекта был выбран веб-фреймворк Vue.js как наиболее простой в изучении, помогающий разработать сайт наиболее быстро и имеющий наименьший размер итоговых файлов.

1.2 Инфологическое проектирование

UML расшифровывается как Unified Modeling Language, и является стандартом, предназначенным для построения объектно-ориентированных модели программных продуктов. Этот стандарт позволяет составлять объектно-ориентированные модели для описания структуры программно-аппаратных комплексов и средств любой сложности.

Для проектирования UML – диаграмм был выбран программный инструмент – StarUML. Данное программное средство ориентирован на UML версии 1.4, а также поддерживает подход модельно-управляемой архитектуры. StarUML строго следует спецификации UML для моделирования программ, разработанной международным консорциумом, занимающимся разработкой и продвижением объектно-ориентированных стандартов и технологий.

1.2.1 Разработка диаграммы пакетов АС

Диаграмма пакетов – структурная диаграмма UML, на которой изображены пакеты и зависимости между ними. Разработанная диаграмма пакетов АС представлена на рисунке 1.2.

Пакет «Автоматизированная система создания и учета резервных копий в кластере виртуальных машин» содержит три основных пакета: пакет «Клиентский интерфейс» содержит основной функционал для взаимодействия администратора с автоматизированной системой, пакет «Сервер» содержит основные функции для работы с резервными копиями и машинами, пакет «СУБД PostgreSQL» предназначен для хранения всех необходимых данных для АС.



Рисунок 1.2 – Диаграмма пакетов автоматизированной системы

1.2.2 Разработка диаграммы вариантов использования АС

В данной автоматизированной системе был выделен 1 актер: Администратор, поскольку он неразделим с автоматизированной системой и ведет постоянное взаимодействие с ней.

Администратор осуществляет авторизацию в системе, инициализацию загрузки в АСУ информации о машинах и хранилищах, находящихся в кластере, ведет мониторинг состояния виртуальных машин, производит настройку резервного копирования, осуществляет запуск резервного копирования в ручном режиме, выполняет удаление резервных копий с хранилища, а также производит загрузку в систему учетных данных для удаленных хранилищ.

Разработанная диаграмма вариантов использования АС представлена на рисунке 1.3.

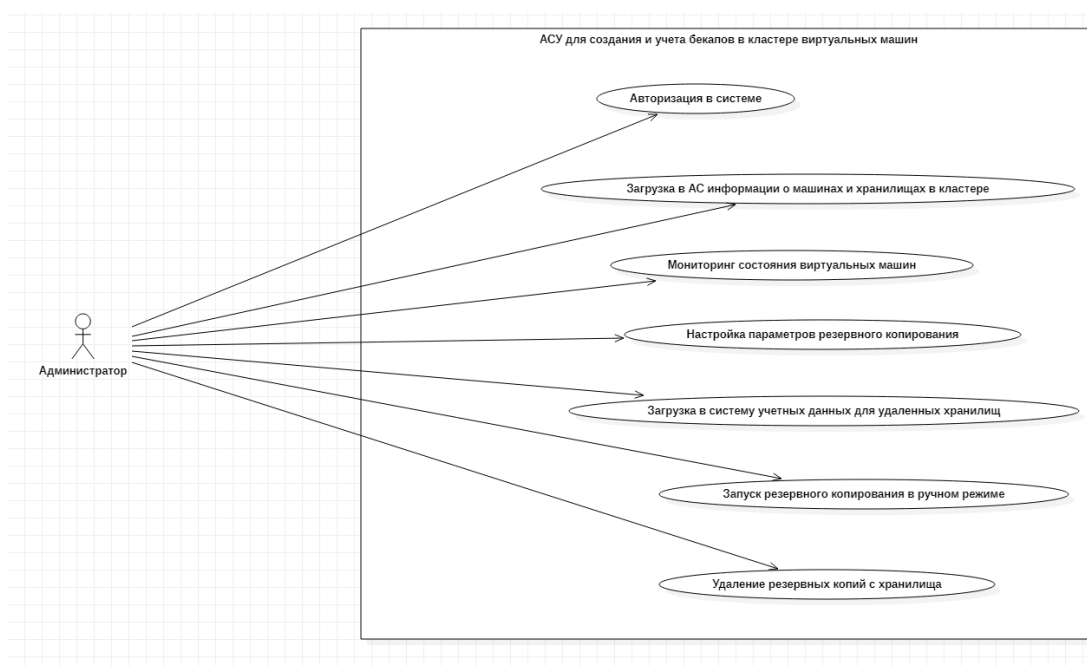


Рисунок 1.3 – Диаграмма вариантов использования

1.2.3 Разработка диаграммы последовательности АС

Диаграммы последовательности показывают коммуникацию между объектами системы. Они иллюстрируют, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного варианта использования.

Диаграмма последовательности для данной АС представлена на рисунке 1.4.

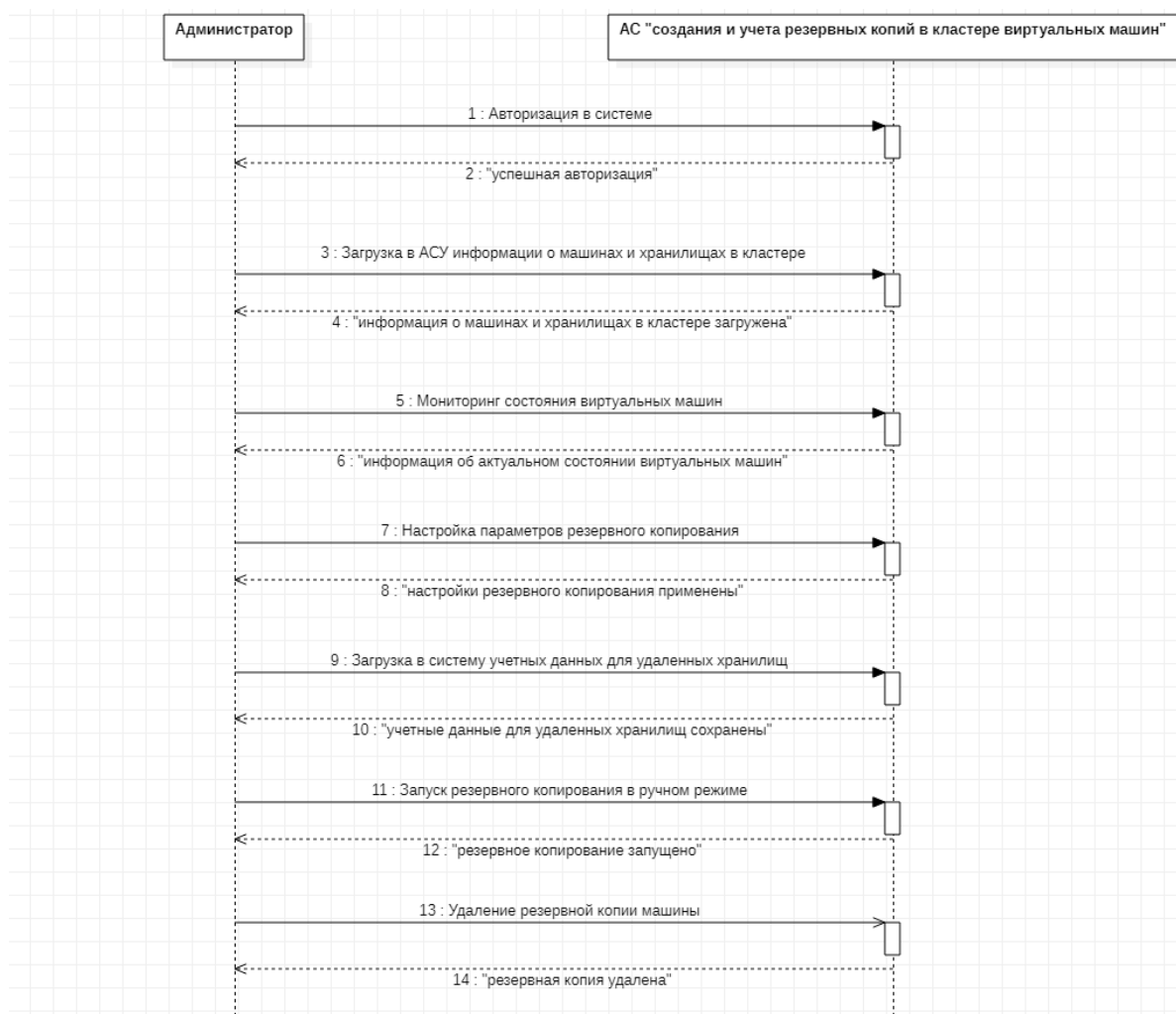


Рисунок 1.4 – Диаграмма последовательности АС

1.2.4 Разработка диаграмм деятельности АС

Диаграммы деятельности иллюстрируют пошагово рабочие процессы с поддержкой выбора, итерации и параллелизма. Разработанные диаграммы деятельности «Загрузка в АС информации о машинах и хранилищах в кластере», «Мониторинг состояния виртуальных машин», «Настройка параметров резервного копирования», «Загрузка в систему учетных данных для удаленных хранилищ», «Запуск резервного копирования в ручном режиме», «Удаление резервной копии с хранилища» представлены на рисунках 1.5-1.10.

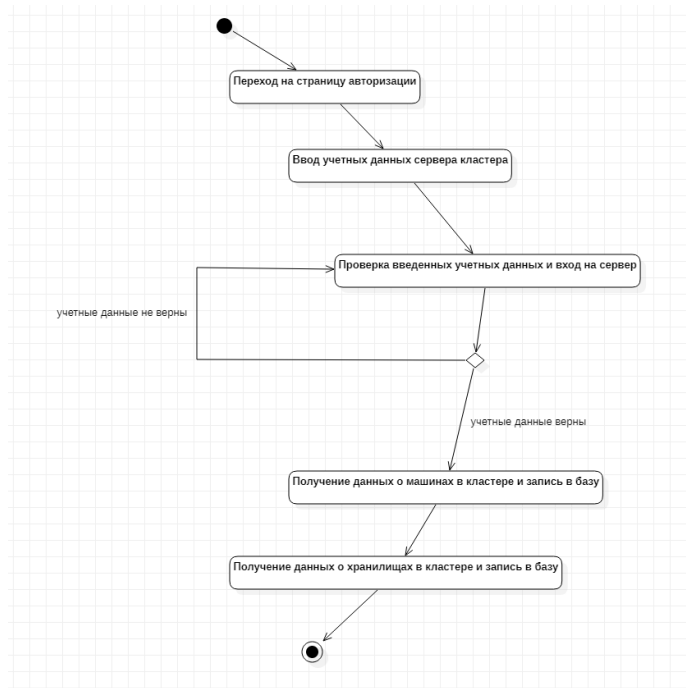


Рисунок 1.5 - Диаграмма деятельности «Загрузка в АС информации о машинах и хранилищах в кластере»

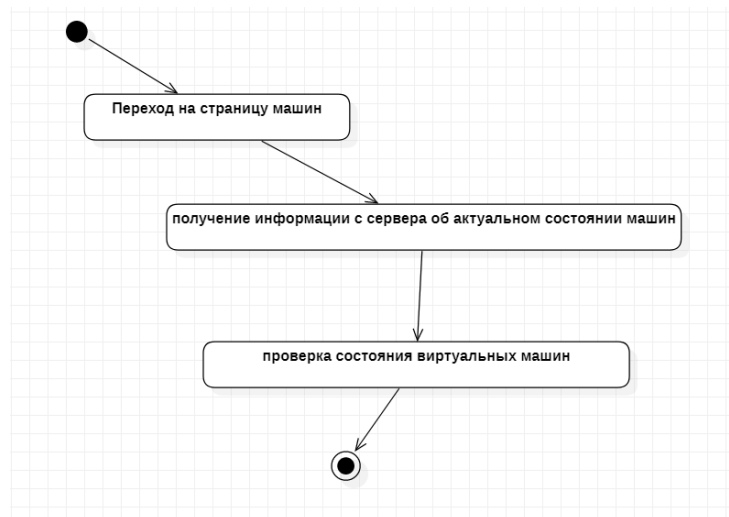


Рисунок 1.6 - Диаграмма деятельности «Мониторинг состояния виртуальных машин»

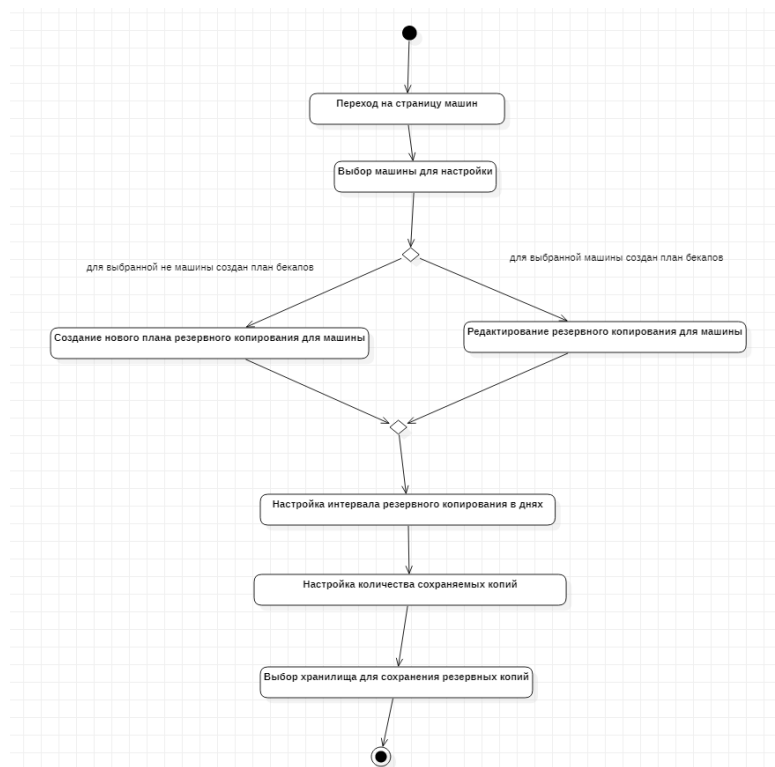


Рисунок 1.7 - Диаграмма деятельности «Настройка параметров резервного копирования»

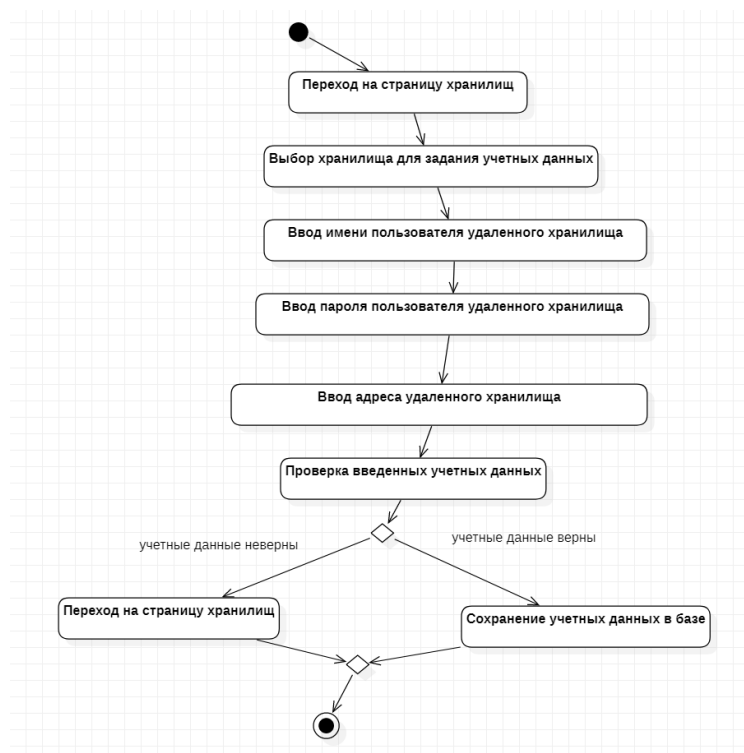


Рисунок 1.8 - Диаграмма деятельности «Загрузка в систему учетных данных для удаленных хранилищ»

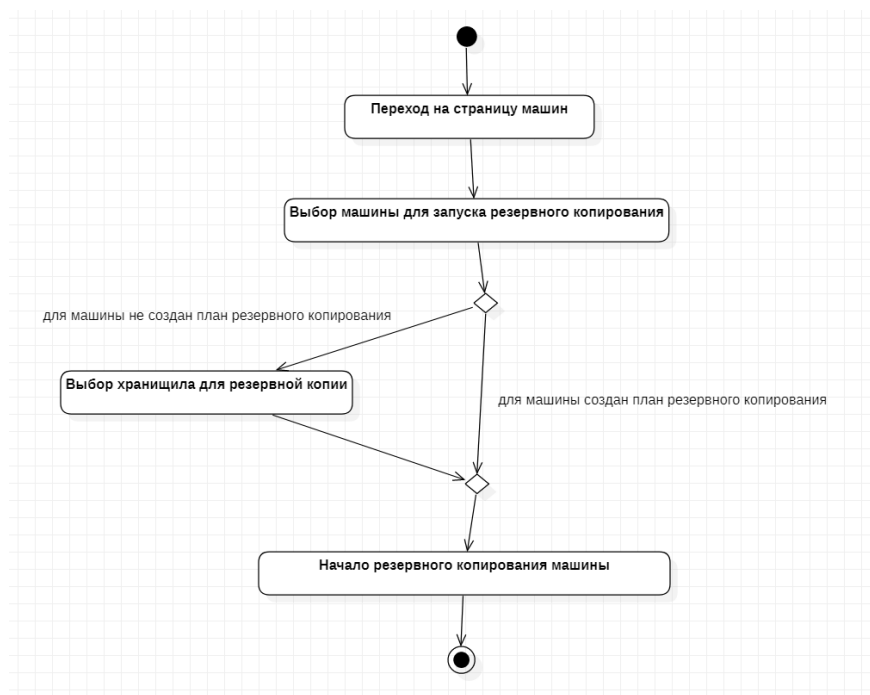


Рисунок 1.9 - Диаграмма деятельности «Запуск резервного копирования в ручном режиме»



Рисунок 1.10 - Диаграмма деятельности «Удаление резервной копии с хранилища»

1.2.5 Разработка диаграммы классов

Диаграмма классов используется для описания статичных субъектов системы и связей между ними в парадигме объектно-ориентированного программирования. В ней описываются атрибуты и методы каждого объекта, а также виды связей между объектами (наследования, зависимости, агрегации, композиции или ассоциации). В соответствии с поставленной задачей была разработана диаграмма классов АС, которая представлена в приложении В.

Описание классов приведено в таблице 1.2.

Таблица 1.2 – Описание классов

Класс	Описание класса
LoginForm	Класс для отображения формы авторизации
StoragesForm	Класс для отображения формы с хранилищами
NavbarComponent	Класс для отображения формы с информацией о пользователе
MachinesForm	Класс для отображения формы со списком машин
PlansForm	Класс для отображения формы со списком планов резервного копирования
StoragesController	Класс для отображения формы со списком хранилищ
BackProcessingController	Класс для создания и учета резервных копий и работы с планами
MachinesController	Класс для учета и редактирования машин
LoginController	Класс для авторизации
Plan	Класс представляющий сущность «Плана резервного копирования»
Machine	Класс представляющий сущность «Машина»
Backup	Класс представляющий сущность «Резервная копия»
Storage	Класс представляющий сущность «Хранилище»
User	Класс представляющий сущность «Пользователь»
Authorization	Класс представляющий сущность «Авторизация»

1.2.6 Разработка диаграммы развертывания

На диаграмме развертывания показаны основные узлы, на которых развернуты программные компоненты. Она отображает общую конфигурацию и топологию шины, показывает расположение компонентов на каждом узле в системе. Указывает на наличие физических соединений и информационных путей между аппаратными устройствами. Согласно техническому заданию была разработана диаграмма развертывания, которая состоит из двух компонентов: клиент и сервер, клиент представляет собой клиентское приложение которое открывается в браузере пользователя, сервер представляет собой программу, которая обрабатывает запросы пользователя, приходящие по каналу связи (Сети Интернет), диаграмма развертывания представлена на рисунке 1.11.

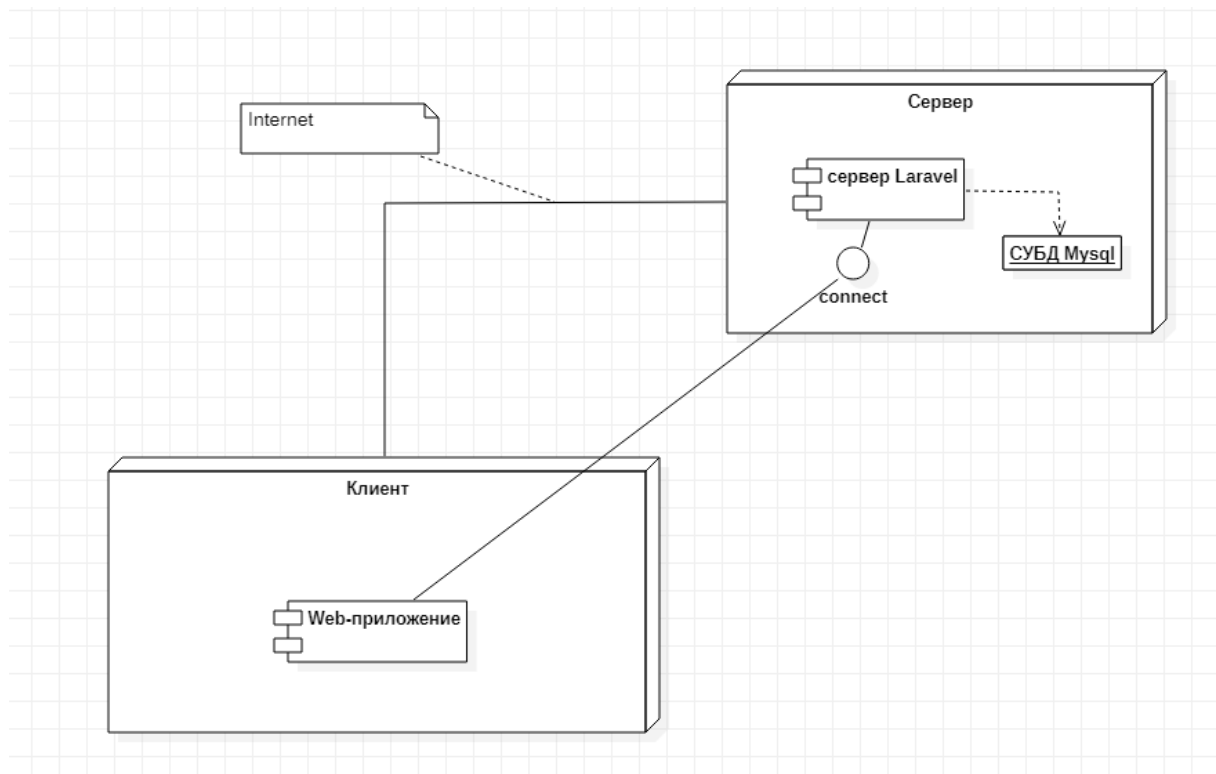


Рисунок 1.11 – Диаграмма развертывания

1.3 Проектирование базы данных

Процесс проектирования базы данных представляет собой процесс преобразования словесно описанной задачи к формализованной структуре объектов в терминах определенной модели. Выделяют следующие этапы проектирования:

- системный анализ, выделение и описание основных субъектов предметной области,
- проектирование концептуальной модели, представляющей собой формализованное описание этих сущностей в рамках семантической модели,
- проектирование логической модели, то есть описание базы данных в рамках принятой логической модели данных,
- создание физической модели, представляющей собой физическую базу данных, используемую в приложении.

Эффективность использования готовой базы данных и рациональное расходование аппаратных ресурсов во многом зависит от правильности проектирования моделей на всех этапах.

Для проектирования была выбрана система с открытым кодом Open ModelSphere 3.2, позволяющая осуществлять полный цикл проектирования реляционных баз данных для множества целевых платформ, в том числе СУБД PostgreSQL. Данная система включает в себя модули для проверки целостности архитектуры и согласованности конкретной СУБД, что позволяет экономить время на проектировании при различных ограничениях.

1.3.1 Разработка концептуальной модели данных

Концептуальная модель представляет собой информационную модель с высокой степенью абстракции, которая иллюстрирует основные сущности будущей базы данных и отношения между ними без учета особенностей конкретной СУБД.

На основе анализа предметной области была разработана концептуальная модель данных. Концептуальная схема представлена на рисунке 1.12 и содержит 7 сущностей. Описание сущностей и их атрибутов представлено в таблицах 1.3 - 1.9.

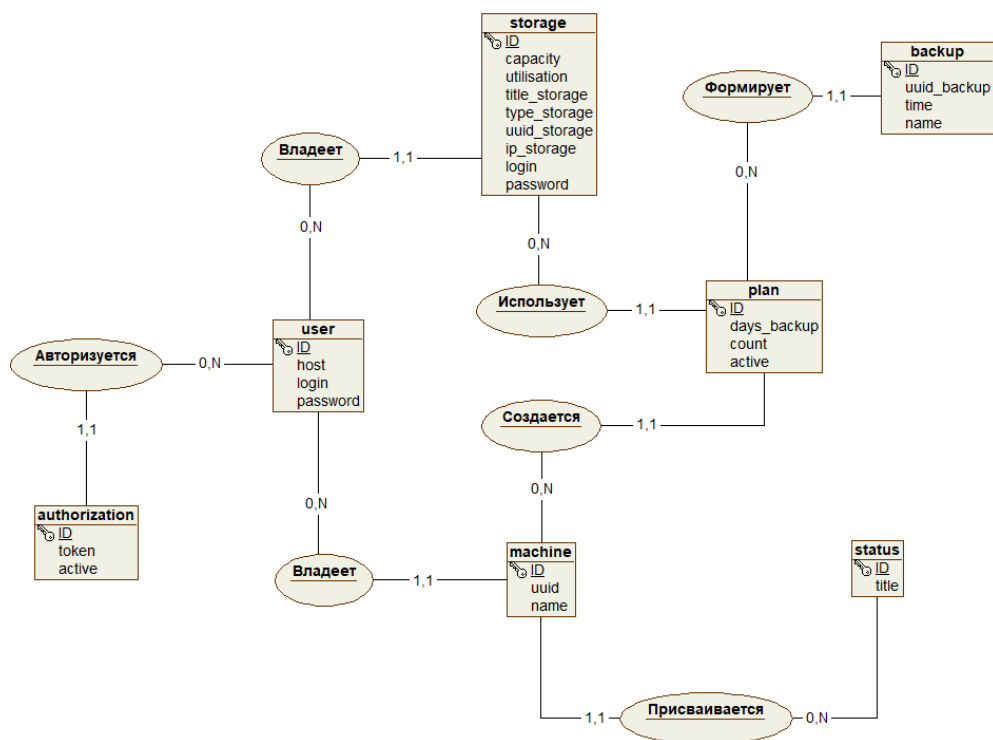


Рисунок 1.12 – Концептуальная модель данных

Таблица 1.3 – Описание атрибутов сущности «authorization»

Имя атрибута	Назначение	Первичный ключ
ID	Код авторизации	РК
Token	Уникальный ключ авторизации	-
Active	Маркер валидности ключа	-

Таблица 1.4 – Описание атрибутов сущности «user»

Имя атрибута	Назначение	Первичный ключ
ID	Код пользователя	РК
Host	Ip-адрес хоста	-
Login	Имя пользователя	-
Password	Пароль пользователя	-

Таблица 1.5 – Описание атрибутов сущности «machine»

Имя атрибута	Назначение	Первичный ключ
ID	Код машины	РК
Uuid	Uuid машины	-
Name	Имя машины	-

Таблица 1.6 – Описание атрибутов сущности «storage»

Имя атрибута	Назначение	Первичный ключ
ID	Код хранилища	РК
Capacity	Объем хранилища	-
Utilisation	Занятое место на хранилище	-
Title_storage	Название хранилища	-
Type_storage	Тип хранилища	-
Uuid_storage	Uuid хранилища	-
Ip_storage	Ip-адрес хранилища	-
Login	Логин	-
Password	Пароль	-

Таблица 1.7 – Описание атрибутов сущности «plan»

Имя атрибута	Назначение	Первичный ключ
ID	Код плана	РК
Days_backup	Период копирования	-
Count	Количество копий	-
Active	План активен	-

Таблица 1.8 – Описание атрибутов сущности «backup»

Имя атрибута	Назначение	Первичный ключ
ID	Код бекапа	РК
Uuid_backup	Uuid бекапа	-
Time	Время создания бекапа	-
Name	Имя бекапа	-

Таблица 1.9 – Описание атрибутов сущности «status»

Имя атрибута	Назначение	Первичный ключ
ID	Код статуса	РК
Title	Название статуса	-

Сущность «user» связана с сущностью «authorization» связью с показателем кардинальности 0:N и 1:1, так как у одного пользователя может быть как несколько ключей авторизации, так и ни одного, но в то же время ключ авторизации обязательно должен быть привязан к конкретному пользователю. Степень участия связи между сущностями «authorization» и «user» полная, так как сущность «authorization» не может существовать без сущности «user».

Сущность «user» связана с сущностью «storage» связью с показателем кардинальности 0:N и 1:1, так как один пользователь может владеть как несколькими хранилищами, так и ни одним, но в то же время хранилище обязательно должно быть привязано к конкретному пользователю. Степень участия связи между сущностями «storage» и «user» полная, так как сущность «storage» не может существовать без сущности «user».

Сущность «user» связана с сущностью «machine» связью с показателем кардинальности 0:N и 1:1, так как один пользователь может владеть как несколькими машинами, так и ни одной, но в то же время каждая машина обязательно должна быть привязана к конкретному пользователю. Степень

участия связи между сущностями «machine» и «user» полная, так как сущность «machine» не может существовать без сущности «user».

Сущность «status» связана с сущностью «machine» связью с показателем кардинальности 1:N и 1:1, так как машине можно присвоить один из статусов, в то же время каждая машина обязательно должна иметь статус. Степень участия связи между сущностями «status» и «machine» полная, так как сущность «machine» не может существовать без сущности «status».

Сущность «machine» связана с сущностью «plan» связью с показателем кардинальности 0:N и 1:1, так как для одной машины можно создать как несколько планов, так и не одного, но в то же время каждый план обязательно должен быть привязан к конкретной машине. Степень участия связи между сущностями «plan» и «machine» полная, так как сущность «plan» не может существовать без сущности «machine».

Сущность «storage» связана с сущностью «plan» связью с показателем кардинальности 0:N и 1:1, так как для одного хранилища может использоваться сразу несколькими планами, а может ни одним, но в то же время каждый план обязательно должен использовать хранилище. Степень участия связи между сущностями «plan» и «storage» полная, так как сущность «plan» не может существовать без сущности «storage».

Сущность «plan» связана с сущностью «backup» связью с показателем кардинальности 0:N и 1:1, так как для в рамках одного плана может быть сформировано несколько бекапов, а может ни одного, но в то же время каждый бекап обязательно должен быть привязан к конкретному плану. Степень участия связи между сущностями «backup» и «plan» полная, так как сущность «backup» не может существовать без сущности «plan».

1.3.2 Разработка логической модели данных

Логическое проектирование – это создание схемы базы данных на основе определенной модели данных, например, реляционная или иерархическая. Для реляционной модели данных логическая модель представляет собой набор схем

отношений, в которых обычно указываются первичные ключи, а также «ссылки» между отношениями, являющиеся внешними ключами. Преобразование концептуальной модели в логическую обычно происходит по формальным правилам.

На основе построенной концептуальной модели данных разработана логическая (реляционная) модель. Логическая модель данных приведена на рисунке 1.13 и содержит 7 сущностей.

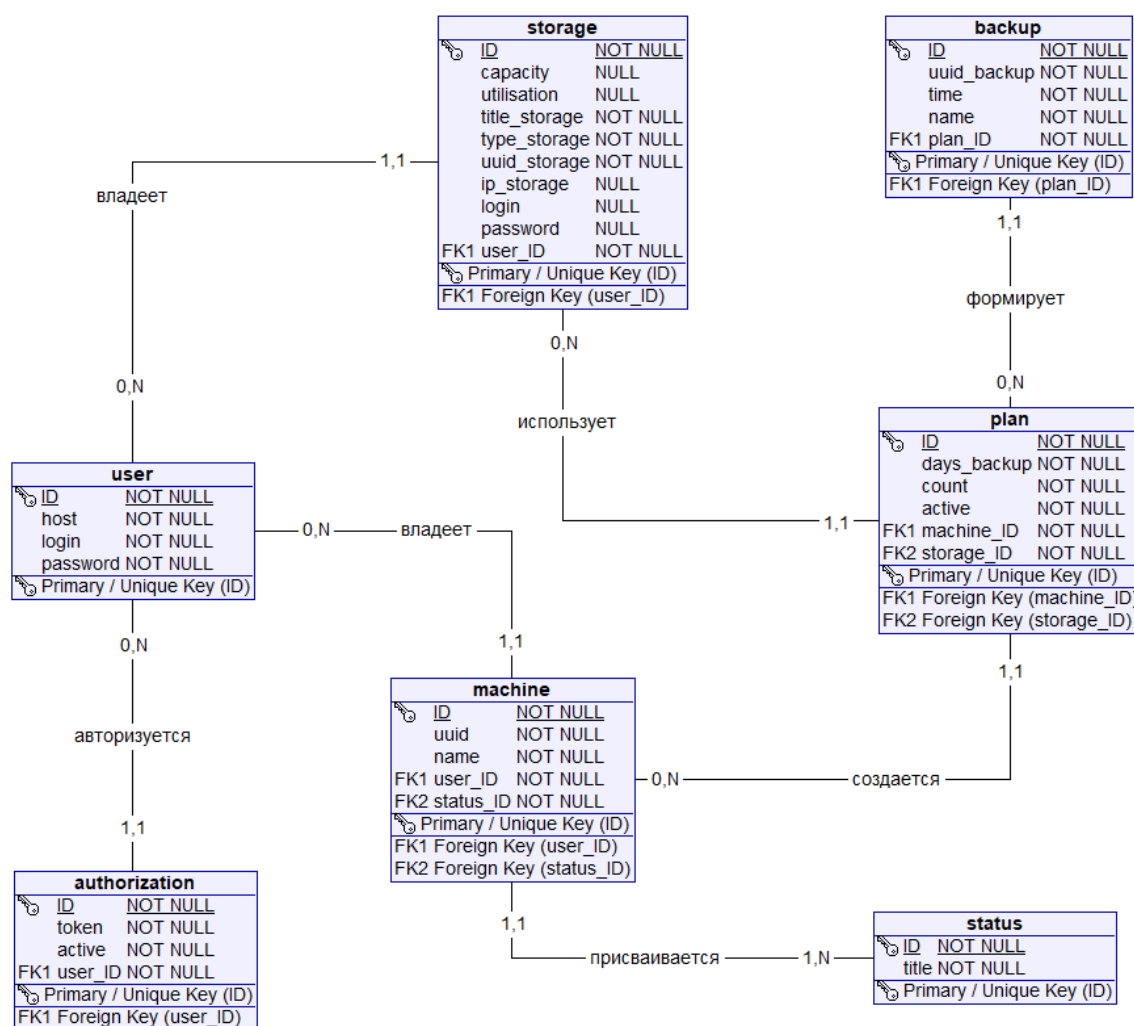


Рисунок 1.13 – Логическая модель данных

Описание атрибутов сущностей логической модели приведено в таблицах 1.10-1.16.

Таблица 1.10 - Описание атрибутов сущности «authorization»

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
ID	ID	Код авторизации	РК	-
Token	Token	ключ авторизации	-	-
Active	Active	Маркер валидности ключа	-	-
user_ID	user_ID	Код пользователя	-	FK

Таблица 1.11 - Описание атрибутов сущности «user»

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
ID	ID	Код пользователя	РК	-
Host	Host	Ip-адрес хоста	-	-
Login	Login	Имя пользователя	-	-
Password	Password	Пароль пользователя	-	-

Таблица 1.12 - Описание атрибутов сущности «machine»

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
ID	ID	Код машины	РК	-
Uuid	Uuid	Uuid машины	-	-
Name	Name	Имя машины	-	-
user_ID	user_ID	Код пользователя	-	FK
status_ID	status_ID	Код статуса	-	FK

Таблица 1.13 - Описание атрибутов сущности «storage»

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
ID	ID	Код хранилища	РК	-
Capacity	Capacity	Объем хранилища	-	-
Utilisation	Utilisation	Занятое место на хранилище	-	-
Title_storage	Title_storage	Название хранилища	-	-
Type_storage	Type_storage	Тип хранилища	-	-
Uuid_storage	Uuid_storage	Uuid хранилища	-	-
Ip_storage	Ip_storage	Ip-адрес хранилища	-	-
Login	Login	Логин	-	-
Password	Password	Пароль	-	-
user_ID	user_ID	Код пользователя	-	FK

Таблица 1.14 - Описание атрибутов сущности «plan»

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
ID	ID	Код плана	РК	-
Days_backup	Days_backup	Период бекапа	-	-
Count	Count	Количество копий	-	-
Active	Active	План активен	-	-
machine_ID	machine_ID	Код машины	-	FK
storage_ID	storage_ID	Код хранилища	-	FK

Таблица 1.15 - Описание атрибутов сущности «backup»

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
ID	ID	Код копии	РК	-
Uuid_backup	Uuid_backup	Uuid бекапа	-	-
Time	Time	Время создания копии	-	-
Name	Name	Имя копии	-	-
plan_ID	plan_ID	Код плана	-	FK

Таблица 1.16 - Описание атрибутов сущности «status»

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
ID	ID	Код статуса	РК	-
Title	Title	Название статуса	-	-

1.3.3 Нормализация базы данных

Нормализацией называется процесс организации данных в базе данных путем создания таблиц и установления связей в соответствии с правилами, которые защищают данные и делают базу данных гибкой за счет устранения избыточности и конфликтующих зависимостей. Избыточность данных ведет к не рациональному использованию аппаратных мощностей и места на жестких дисках, а также замедлению работы системы в целом.

Отношение находится в 1НФ, если все строки в таблице различны и все атрибуты неразделимы (атомарны), то есть на пересечении строки и столбца имеется единственное значение. База данных, представленная на рисунке 1.13, находится в 1НФ. Для подтверждения того, что база данных находится в 1НФ, необходимо рассмотреть каждую сущность.

Сущность «authorization» имеет ключевой атрибут «ID». Атрибуты «token», «active», «user_ID» данной сущности являются простыми и неделимыми,

Сущность «user» имеет ключевой атрибут «ID». Атрибуты «host», «login», «password» данной сущности являются простыми и неделимыми.

Сущность «machine» имеет ключевой атрибут «ID». Атрибуты «uuid», «name», «user_ID», «status_ID» являются простыми и неделимыми.

Сущность «storage» имеет ключевой атрибут «ID», Атрибуты «capacity», «utilisation», «title_storage», «type_storage», «uuid_storage», «ip_storage», «login», «password», «user_ID» данной сущности являются простыми и неделимыми.

Сущность «backup» имеет ключевой атрибут «ID». Атрибуты «uuid_backup», «time», «name», «plan_ID» данной сущности являются простыми и неделимыми.

Сущность «plan» имеет ключевой атрибут «ID». Атрибуты «days_backup», «count», «active», «machine_ID», «storage_ID» данной сущности являются простыми и неделимыми.

Сущность «status» имеет ключевой атрибут «ID». Атрибут «title» данной сущности являются простым и неделимым.

Отношение находится во 2НФ, если оно находится в 1НФ и каждый неключевой атрибут в таблице функционально-полно зависит от первичного ключа.

База данных, представленная на рисунке 1.13, находится во 2НФ.

Сущность «authorization» находится во 2НФ, так как имеет единственный ключевой атрибут «ID». Неключевые атрибуты «token», «active», «user_ID» данной сущности функционально-полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Сущность «user» находится во 2НФ, так как имеет единственный ключевой атрибут «ID». Неключевые атрибуты «host», «login», «password» данной сущности функционально-полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Сущность «storage» находится во 2НФ, так как имеет единственный ключевой атрибут «ID». Неключевые атрибуты «capacity», «utilisation», «title_storage», «type_storage», «uuid_storage», «ip_storage», «login», «password», «user_ID» данной сущности функционально-полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Сущность «machine» находится во 2НФ, так как имеет единственный ключевой атрибут «ID». Неключевые атрибуты «uuid», «name», «user_ID», «status_ID» данной сущности функционально-полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Сущность «backup» находится во 2НФ, так как имеет единственный ключевой атрибут «ID». Неключевые атрибуты «uuid_backup», «time», «name», «plan_ID» данной сущности функционально-полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Сущность «plan» находится во 2НФ, так как имеет единственный ключевой атрибут «ID». Неключевые атрибуты «days_backup», «count», «active», «machine_ID», «storage_ID» данной сущности функционально-полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Сущность «status» находится во 2НФ, так как имеет единственный ключевой атрибут «ID». Неключевой атрибут «title» данной сущности функционально-полно зависит от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2НФ.

Отношение находится в 3НФ, если оно находится во 2НФ и нет транзитивных зависимостей между неключевыми атрибутами.

База данных, представленная на рисунке 1.13, находится в 3НФ, так как её сущности не имеют транзитивных зависимостей.

В сущности «authorization» неключевые атрибуты «token», «active», «user_ID» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

В сущности «user» неключевые атрибуты «host», «login», «password» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

В сущности «storage» неключевые атрибуты «capacity», «utilisation», «title_storage», «type_storage», «uuid_storage», «ip_storage», «login», «password», «user_ID» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

В сущности «machine» неключевые атрибуты «uuid», «name», «user_ID», «status_ID» данного отношения не зависят друг от друга.

Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

В сущности «backup» неключевые атрибуты «uuid_backup», «time», «name», «plan_ID» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

В сущности «plan» неключевые атрибуты «days_backup», «count», «active», «machine_ID», «storage_ID» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

В сущности «status» есть один неключевой атрибут «title». Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3НФ.

1.3.4 Разработка физической модели данных

Физической моделью данных называют логическую модель, выраженную для конкретной СУБД, она содержит все данные, нужные СУБД для создания базы данных: названия полей, таблиц, типы данных, внешние и первичные ключи.

На основе разработанной логической модели данных была разработана физическая модель данных. Физическая модель базы данных разработана для СУБД PostgreSQL и приведена в приложении Г.

Физическая модель данных содержит 7 таблиц. Описание полей таблиц физической модели приведено в таблицах 1.17-1.23.

Таблица 1.17 - Описание полей таблицы «authorization»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Внешний ключ	Тип данных
ID	ID	Недопустимо	PK	-	INTEGER
Token	Token	Недопустимо	-	-	VARCHAR (255)
Active	Active	Недопустимо	-	-	BINARY
user_ID	user_ID	Недопустимо	-	FK	INTEGER

Таблица 1.18 - Описание полей таблицы «user»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Внешний ключ	Тип данных
ID	ID	Недопустимо	PK	-	INTEGER
Host	Host	Недопустимо	-	-	VARCHAR (255)
Login	Login	Недопустимо	-	-	VARCHAR (255)
Password	Password	Недопустимо	-	-	VARCHAR (255)

Таблица 1.19 - Описание полей таблицы «machine»

Имя атрибута	Физическое имя	Нулевое значение	Первичный ключ	Внешний ключ	Тип данных
ID	ID	Недопустимо	РК	-	INTEGER
Uuid	Uuid	Недопустимо	-	-	VARCHAR (255)
Name	Name	Недопустимо	-	-	VARCHAR (255)
user_ID	user_ID	Недопустимо	-	FK	INTEGER
status_ID	status_ID	Недопустимо	-	FK	INTEGER

Таблица 1.20 - Описание полей таблицы «storage»

Имя атрибута	Физическое имя	Нулевое значение	Первичный ключ	Внешний ключ	Тип данных
ID	ID	Недопустимо	РК	-	INTEGER
Capacity	Capacity	Недопустимо	-	-	BIGINT
Utilisation	Utilisation	Недопустимо	-	-	BIGINT
Title_storage	Title_storage	Недопустимо	-	-	VARCHAR (255)
Type_storage	Type_storage	Недопустимо	-	-	BINARY
Uuid_storage	Uuid_storage	Недопустимо	-	-	VARCHAR (255)
Ip_storage	Ip_storage	Допустимо	-	-	VARCHAR (255)
Login	Login	Допустимо	-	-	VARCHAR (255)
Password	Password	Допустимо	-	-	VARCHAR (255)
user_ID	user_ID	Недопустимо	-	FK	INTEGER

Таблица 1.21 - Описание полей таблицы «plan»

Имя атрибута	Физическое имя	Нулевое значение	Первичный ключ	Внешний ключ	Тип данных
ID	ID	Недопустимо	PK	-	INTEGER
Days_backup	Days_backup	Недопустимо	-	-	INTEGER
Count	Count	Недопустимо	-	-	INTEGER
Active	Active	Недопустимо	-	-	BINARY
machine_ID	machine_ID	Недопустимо	-	FK	INTEGER
storage_ID	storage_ID	Недопустимо	-	FK	INTEGER

Таблица 1.22 - Описание полей таблицы «backup»

Имя атрибута	Физическое имя	Нулевое значение	Первичный ключ	Внешний ключ	Тип данных
ID	ID	Недопустимо	PK	-	INTEGER
Uuid_backup	Uuid_backup	Недопустимо	-	-	VARCHAR (255)
Time	Time	Недопустимо	-	-	TIME
Name	Name	Недопустимо	-	-	VARCHAR (255)
plan_ID	plan_ID	Недопустимо	-	FK	INTEGER

Таблица 1.23 - Описание полей таблицы «status»

Имя атрибута	Физическое имя	Нулевое значение	Первичный ключ	Внешний ключ	Тип данных
ID	ID	Недопустимо	PK	-	INTEGER
Title	Title	Недопустимо	-	-	VARCHAR (255)

На основе физической модели данных был сгенерирован «backup.sql», содержащий скрипт базы данных на языке SQL. Текст скрипта приведен в приложении А.

С помощью инструментального средства RHPMyAdmin данный скрипт был выполнен и создана база данных находящаяся на сервере СУБД.

1.4 Разработка приложения

1.4.1 Анализ функций приложения

Перед началом разработки автоматизированной системы был произведен анализ функций будущего приложения, который основывался на анализе предметной области и требованиям к функциональным характеристикам автоматизированной системы.

Результат анализа функций представлен на рисунке 1.14.

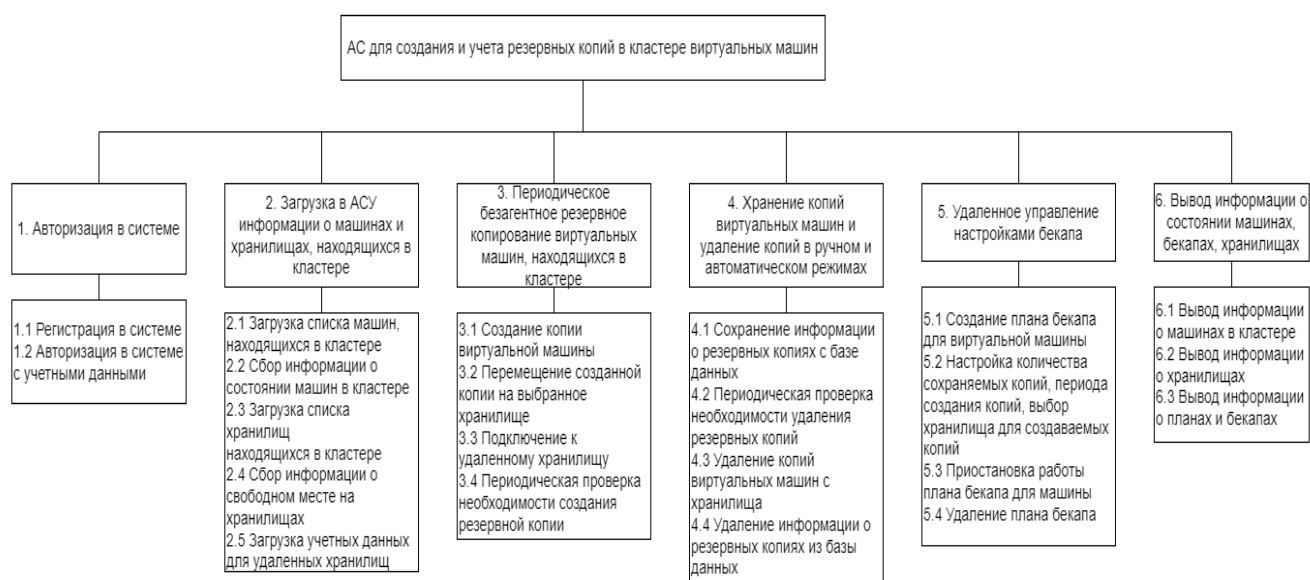


Рисунок 1.14 - Иерархия функций приложения

1.4.2 Разработка логической структуры приложения

После анализа необходимого функционала была разработана логическая структура приложения, которая состоит из двух основных частей: клиентская (Client) и серверная (Server), которые в свою очередь состоят из отдельных подмодулей, структурная схема приложения представлена на рисунке 1.15.

Описание модулей приведено в таблице 1.24.

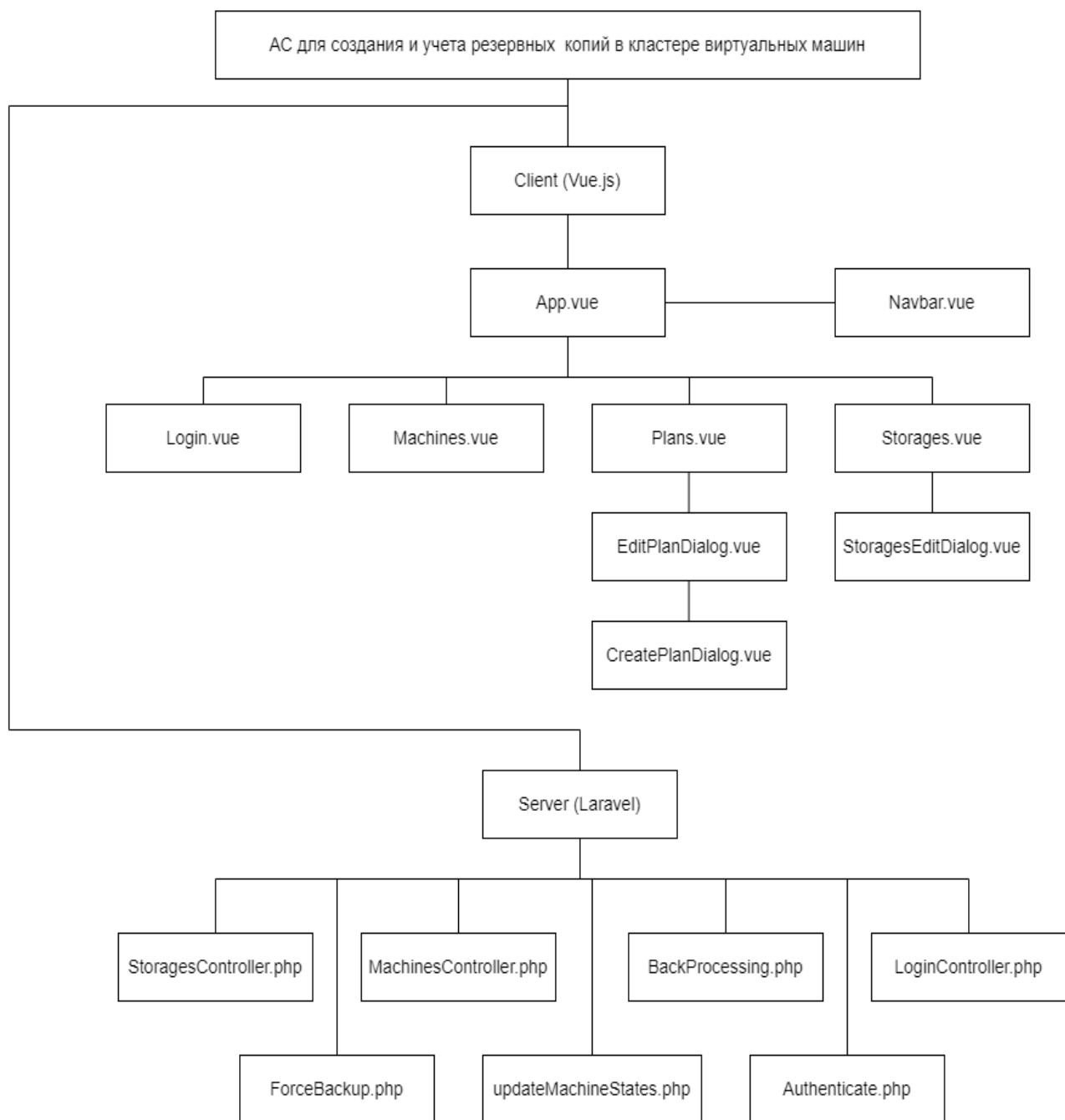


Рисунок 1.15 – Логическая структура приложения

Таблица 1.24 – Описание модулей приложения

Название модуля	Название подмодуля	Назначение
Client	App.vue	Вызывает остальные подмодули
	Navbar.vue	Навигация по приложению
	Login.vue	Страница авторизации
	Machines.vue	Страница со списком машин
	Plans.vue	Страница со списком планов
	EditPlanDialog.vue	Диалоговое окно редактирования параметров резервного копирования
	CreatePlanDialog.vue	Диалоговое окно для создания плана резервного копирования
	Storages.vue	Страница со списком хранилищ
	StoragesEditDialog.vue	Диалоговое окно для задания учетных данных удаленного хранилища
Server	StoragesController.php	Модуль для работы с хранилищами
	ForceBackup.php	Модуль для создания резервных копий машин
	MachinesController.php	Модуль для работы с машинами
	BackProcessing.php	Модуль для работы с резервными копиями и планами
	Authenticate.php	Модуль для проверки аутентификации
	LoginController.php	Модуль для регистрации и аутентификации

1.4.3 Описание модулей клиентской части приложения

Клиентская часть приложения была реализована на языке JavaScript с использованием программной платформы Vue.js по принципу одностраничного веб-приложения, которое благодаря динамическому обновлению при помощи JavaScript и асинхронных запросов позволяет работать с различными модулями

приложения без необходимости перезагружать страницу или загружать другие страницы. Для хранения уникального ключа авторизации используется стандартное хранилище браузера – `localStorage`.

Рассмотрим каждый компонент и его набор обработчиков событий клиентской части приложения:

Компонент `App.vue` является родительским для всех остальных компонентов и вызывает их по мере надобности, с помощью встроенного в `Vue.js` инструмента маршрутизации, и не содержит никаких методов.

Компонент `NavBar.vue` предназначен для навигации по приложению пользователем и содержит ссылки для перехода к основным модулям приложения, а также позволяет выйти из системы, и содержит следующие методы:

- метод `showMenu` предназначен отображение и скрывания основного меню,
- метод `logout` предназначен для выхода из системы,
- метод `checkLogin` предназначен для проверки авторизации пользователя путем проверки записи уникального ключа авторизации в хранилище `localStorage`,
- метод `goHome` – предназначен для перехода на основную страницу со списком машин.

Компонент содержит следующие поля:

- поле `logged` предназначено для хранения информации об авторизации пользователя,
- поле `drawer` предназначено для хранения состояния показа меню,
- поле `links` предназначено для хранения ссылок для навигации.

Компонент `Login.vue` предназначен для авторизации в системе и включает в себя основные поля для авторизации пользователя в системе и кластере виртуальных машин и содержит следующие методы:

- метод `letLogin` предназначен для авторизации путем отправки запроса на сервер, при успешной попытке в хранилище записывается уникальный ключ

авторизации и пользователя переносит на основную страницу со списком машин.

Компонент содержит следующие поля:

- `wrong_login` - хранит состояние об успешности авторизации,
- `host` – имя хоста,
- `login` – имя пользователя,
- `password` – пароль,
- `logging` – текущее состояние авторизации.

Компонент `Machines.vue` предназначен для отображения информации о виртуальных машинах в кластере, а также создания для них планов резервного копирования и содержит следующие методы:

- метод `fetchMachinesList` предназначен для отправки запроса на сервер и получения информации о машинах,
- метод `confirmForceBackup` предназначен для подтверждения создания резервной копии в ручном режиме, отправляет запрос на сервер для начала резервного копирования,
- метод `createPlan` вызывает компонент `CreatePlanDialog.vue` в котором осуществляется создание плана резервного копирования для машины,
- метод `editPlan` переносит пользователя на страницу со списком планов с последующим вызовом модального окна для редактирования параметров плана резервного копирования,
- метод `stopPlan` предназначен для приостановки действия плана резервного копирования для машины,
- метод `ucFirst` форматирует строку,
- метод `formatDate` форматирует дату,
- метод `getNextDate` формирует следующую дату резервного копирования,
- метод `sortBy` сортирует список машин по заданному параметру,
- метод `closeModal` предназначен для закрытия модальных окон.

В компоненте реализованы следующие поля:

- selectedMachine – выбранная машина,
- showButtonsMachine – состояние отображения дополнительных кнопок у машины,

- statuses – список статусов для машин,
- machines – список машин,
- sort – параметр по которому осуществляется сортировка.

Компонент Plans.vue предназначен для отображения списка планов резервного копирования, их редактирования, приостановки, возобновления удаления, а также списков сделанных резервных копий в рамках этих планов. В компоненте реализованы следующие методы:

- метод fetchPlans предназначен для получения списка планов резервного копирования с сервера путем отправки запроса,
- метод deletePlan отправляет запрос на сервер для удаления плана,
- метод editModalPlan открывает диалоговое окно, которое реализовано в рамках компонента editPlanDialog.vue для редактирования настроек плана резервного копирования,
- метод editPlanActive осуществляет возобновление и приостановку работы плана резервного копирования,
- метод deleteBackup предназначен для удаления резервной копии путем отправки запроса на сервер,
- метод formatDate предназначен для форматирования даты,
- метод closeModal предназначен для закрытия модальных окон.

Компонент содержит следующие поля:

- selectedPlanEdit – выбранный план для редактирования,
- plans – список планов,
- editPlan – параметр, переданный из компонента Machines.vue.

Компонент editPlanDialog.vue предназначен для редактирования настроек плана резервного копирования и содержит следующие методы:

- метод fetchStorages предназначен для получения с

сервера списка хранилищ,

- метод `close` предназначен для закрытия диалогового окна,
- метод `save` предназначен для сохранения настроек плана,
- метод `editPlanActive` предназначен для включения и отключения плана.

Компонент содержит следующие поля:

- `storages` – список хранилищ,
- `plan` – редактируемый план,
- `inputRules` – правила для ввода в текстовые поля.

Компонент `CreatePlanDialog.vue` предназначен создание нового плана резервного копирования с заданными настройками и содержит следующие методы:

- метод `fetchStorages` предназначен для получения с сервера списка хранилища,
- метод `close` предназначен для закрытия модального окна,
- метод `create` предназначен для создания нового плана.

Компонент содержит следующие поля:

- `machine` – машина, для которой создается план,
- `plan` – объект плана,
- `storages` – список хранилищ,
- `inputRules` – правила для заполнения текстовых полей.

Компонент `Storages.vue` предназначен для вывода и редактирования информации о хранилищах в кластере и содержит следующие методы:

- метод `fetchStoragesList` предназначен для получения списка хранилищ,
- метод `updateStorages` предназначен для обновления информации о хранилищах,
- метод `editStorage` предназначен для вызова компонента `StorageEditDialog.vue`, который предназначен для редактирования информации о хранилище и задать учетные данные, если это удаленное хранилище,
- метод `closeModal` предназначен для закрытия модального окна.

И содержит следующие поля:

- storages – список хранилищ,
- editingStorage – редактируемое хранилище.

Компонент `StoragesEditDialog.vue` предназначен для редактирования информации о хранилище и задания учетных данных если это хранилище удаленное и содержит следующие методы:

- метод `checkConnection` предназначен для проверки учетных данных к хранилищу путем отправки запроса на сервер и авторизации на указанном хранилище с данными учетными данными,
- метод `save` предназначен для сохранения информации о хранилище,
- метод `close` предназначен для закрытия модального окна.

Компонент содержит следующие поля:

- editingStorage – редактируемое хранилище,
- checking – состояние проверки хранилища,
- inputRules – правила для ввода в текстовые поля.

Таким образом были разработаны модули клиентской части приложения.

Примеры запросов от клиента предоставлены на следующих рисунках:

При запуске клиента открывается страница авторизации (рис. 1.16).

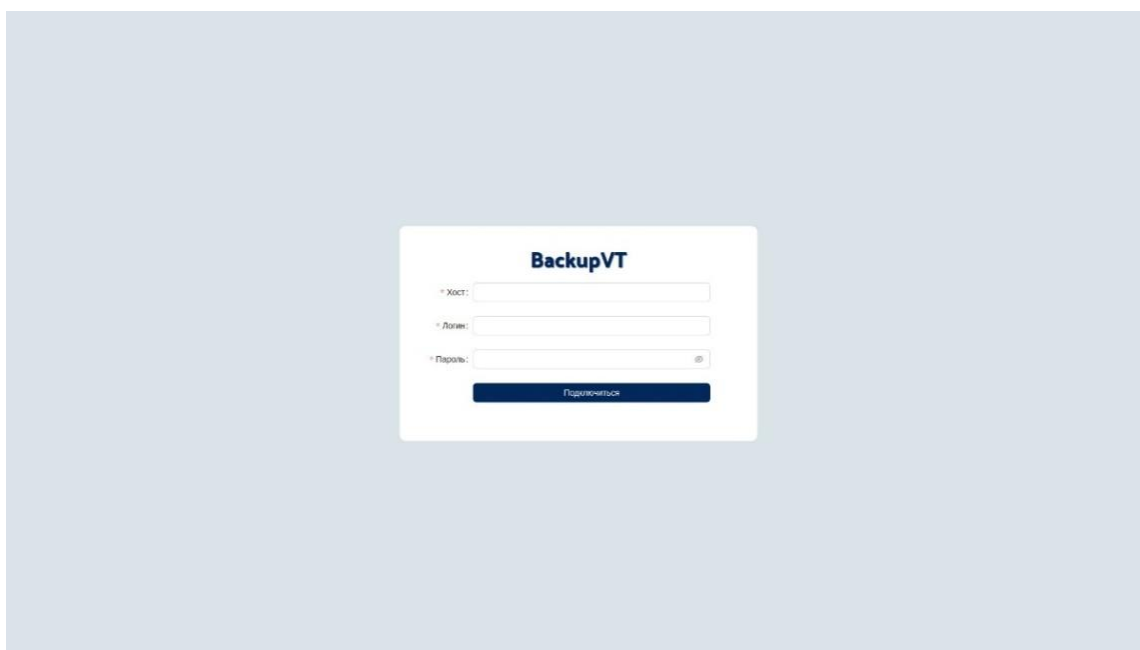


Рисунок 1.16 – Авторизация

После подключения пользователь перенаправляется на страницу с доступными устройствами (рис. 1.17).

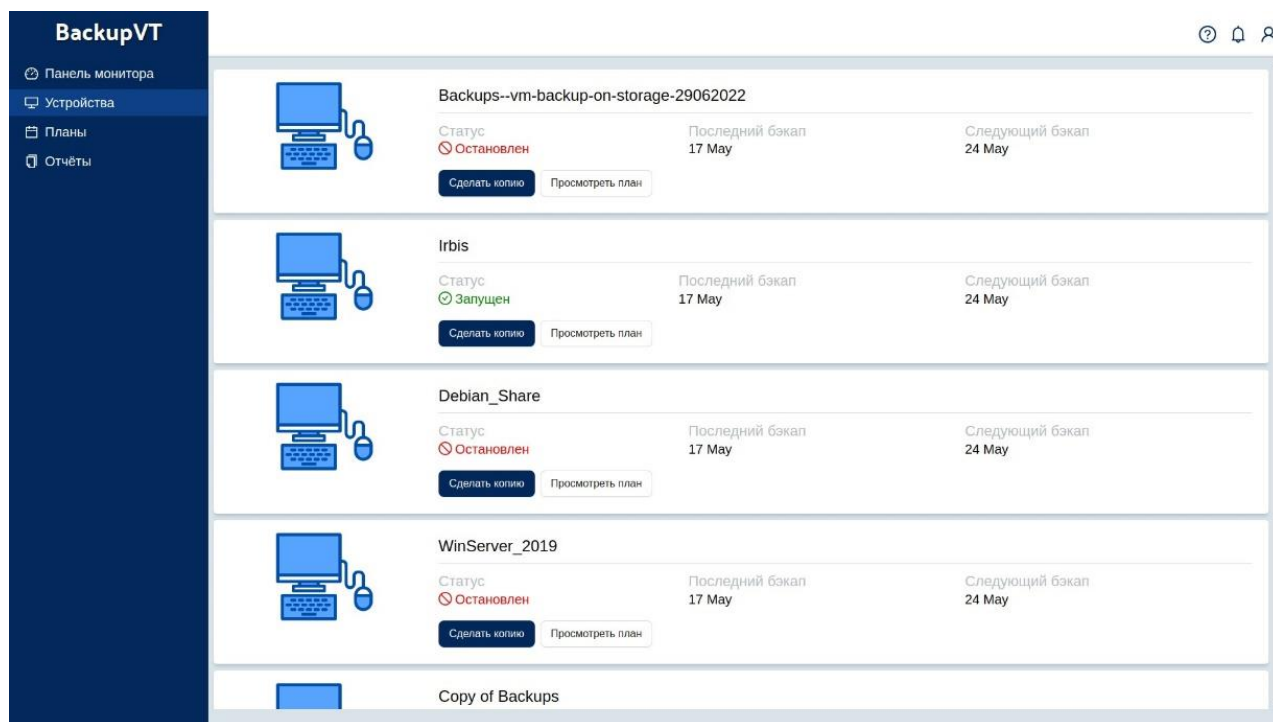


Рисунок 1.17 – Подключение

В заголовке страницы можно посмотреть уведомления и выйти из подключения.

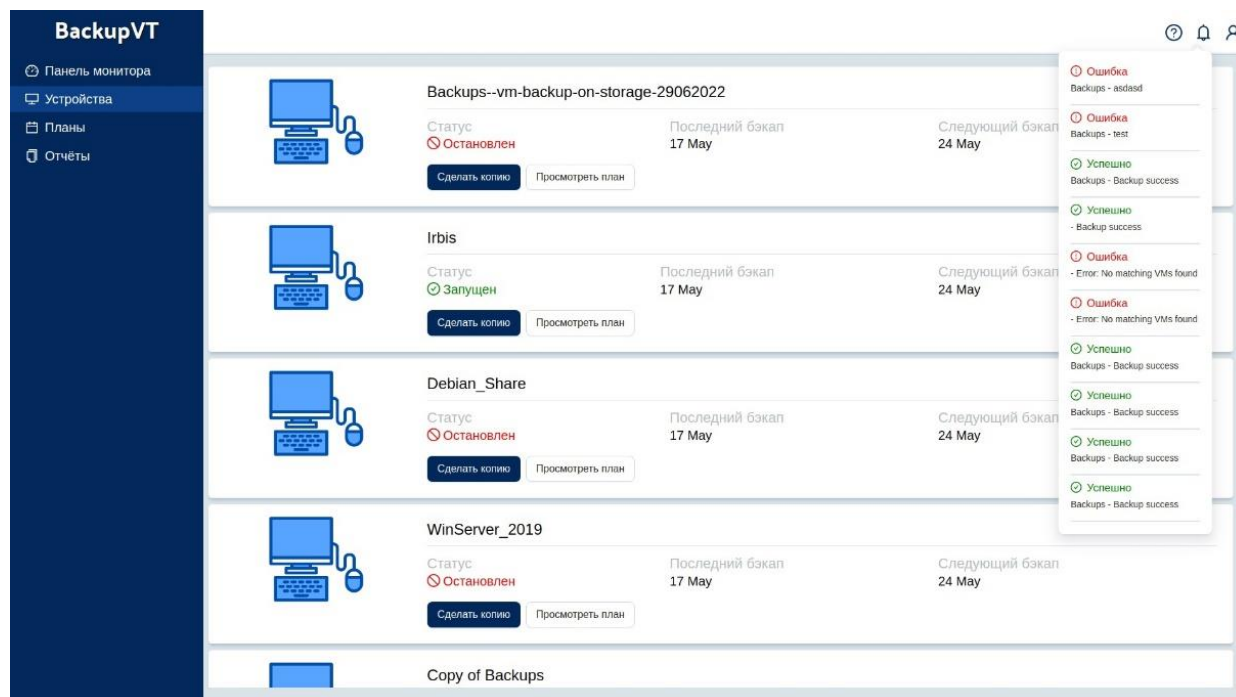


Рисунок 1.18 – Просмотр уведомлений

При нажатии кнопки посмотреть план, открывается страница с планами, где отмечено выбранное устройство. Также, можно перейти на эту страницу с помощью меню.

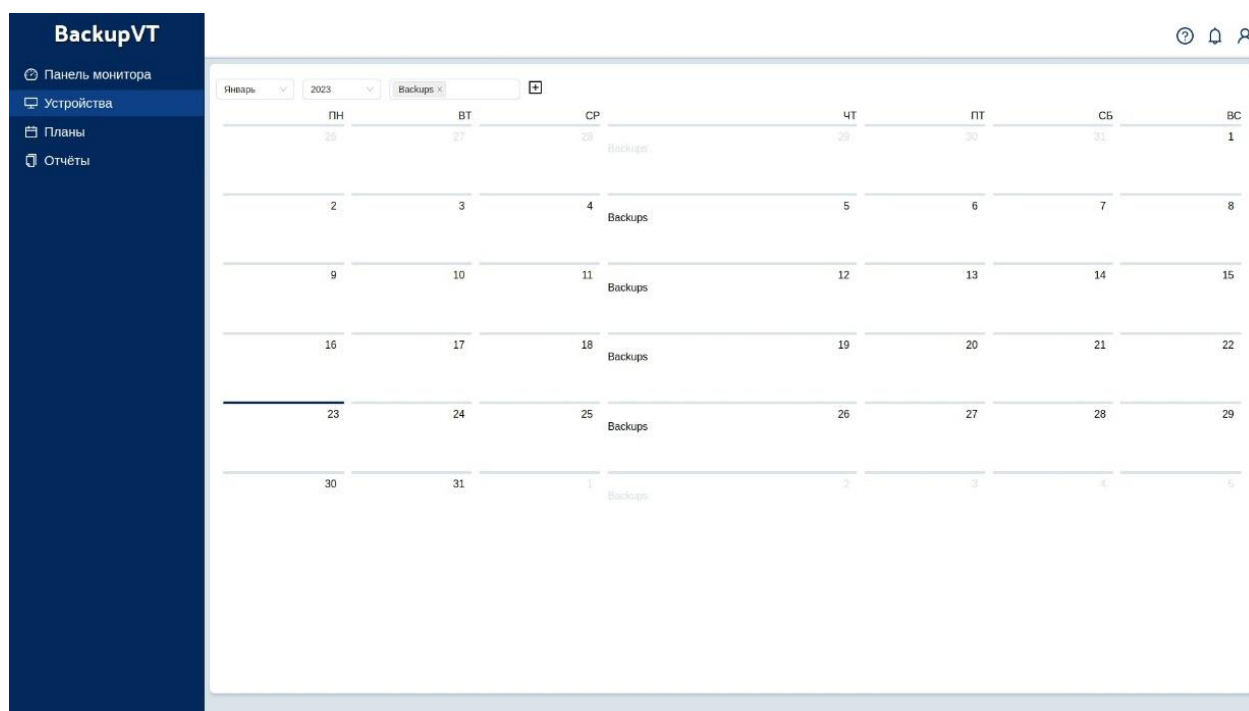


Рисунок 1.19 – Страница планов

При нажатии на любой день можно увидеть какие задачи на него запланированы.

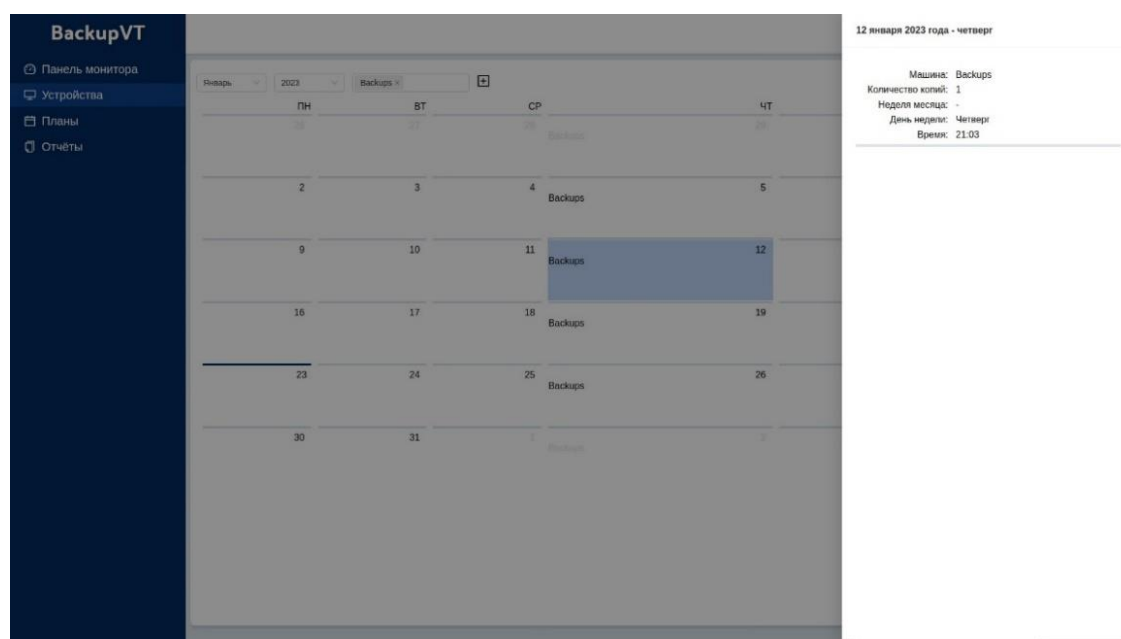


Рисунок 1.20 – Просмотр планов на день

Чтобы добавить или изменить план нужно нажать на кнопку рядом с фильтром устройств. Если план существует, то рядом с его названием отображается иконка удаления.

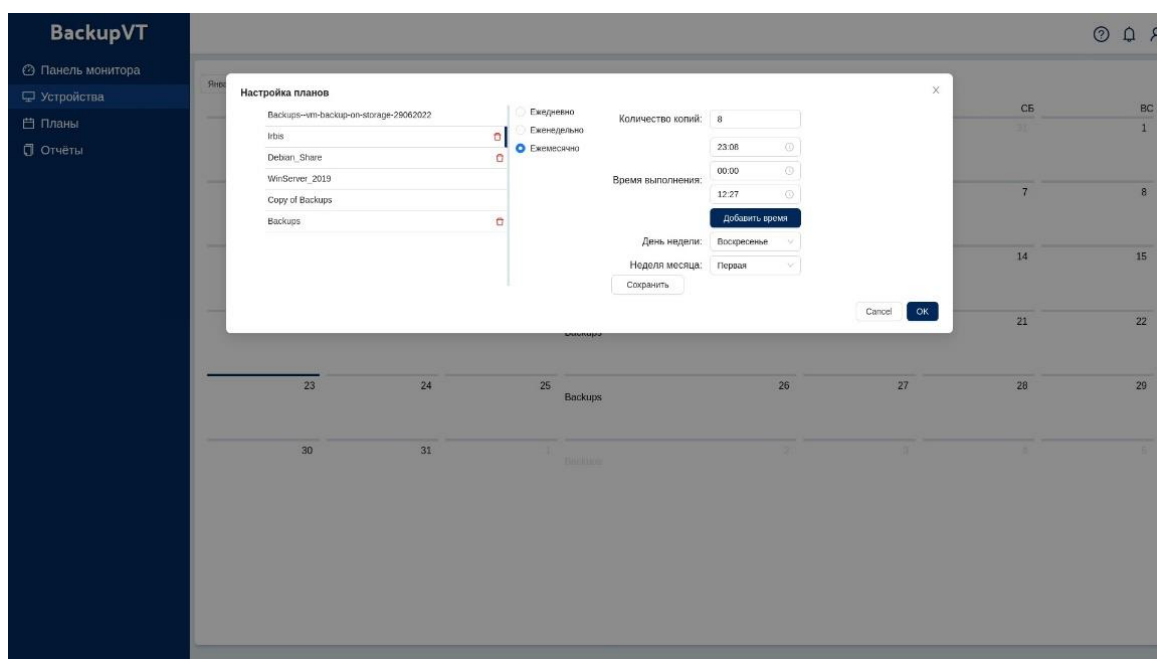


Рисунок 1.21 – Удаление данных

1.4.4 Описание модулей серверной части приложения

Серверная часть приложения была реализована на языке PHP с использованием фреймворка Laravel версии 5.8. Сервер взаимодействует с базой данных, использованная система управления базой данных – PostgreSQL. Код сервера разбит на модули для работы с конкретными сущностями базы данных. Сервер располагается на сервере с операционной системой Linux и использует внутренний планировщик задач операционной системы – Cron для реализации периодического запуска создания резервных копий виртуальных машин, а также инструмент ssh, позволяющий с помощью протокола Secure Shell, подключаться к удаленному хосту, который непосредственно располагает всеми виртуальными машинами и хранилищами для работы с ними, данные для авторизации совпадают с фактическим пользователем на хосте.

Далее представлено описание основных модулей и их функций.

Модуль StoragesController.php предназначен для работы с хранилищами и реализует следующие функции:

- функция parseStorages предназначена для сбора актуальной информации о хранилищах на хосте, для чего происходит подключение к хосту и выполнение команды, выводящей список всех хранилищ,
- функция setCredentialsStorage выполняет сохранение учетных данных для хранилища, для чего подключается к удаленному хосту и производит подключение к удаленному хранилищу, если подключение успешно, сохраняет учетные данные,
- функция editStorage сохраняет данные о хранилище в базе,
- функция listStorages предназначена для отправки списка хранилищ,
- функция updateStorages предназначена для обновления информации о хранилищах,
- функция parseFreeSpaceStorages получает актуальную информацию о количестве занятого и свободного места на хранилищах кластера, для чего происходит подключение к хосту и выполнение команды с выводом детализированной информации по каждому хранилищу.

Модуль ForceBackup.php предназначен для выполнения резервной копии машины и располагает одной функцией - функция handle предназначена для выполнения резервного копирования машины, блок-схема функции handle приведена в приложении Д,

Модуль MachinesController.php предназначен для работы с машинами и состоит из функций:

- функция machinesList отправляет клиенту список машин,
- функция getLastDateBackup получает дату последней резервной копии,

Модуль BackProcessing предназначен для работы с резервными копиями и планами резервных копий и включает в себя следующие функции:

- функция createPlan создает новый план резервного копирования,
- функция editPlan предназначена для редактирования

плана резервного копирования,

- функция `listPlans` предназначена для отправки клиенту списка планов резервного копирования,

- функция `authorization` предназначена для авторизации на удаленном хосте для выполнения различных команд,

- функция `forceBackup` предназначена для старта резервного копирования в ручном режиме и вызывает функцию `handle` модуля `ForceBackup.php`,

- функция `MainBackup` предназначена для проверки необходимости начала резервного копирования для каждой машины и в случае необходимости начинает его, путем вызова функции `handle` модуля `ForceBackup.php`. Данная функция вызывается периодически с помощью утилиты `cron` операционной системы `Linux`, и вызывает функции `parseCurrentMachinesStates` и `ProcessingClusterPlans`, а также функцию `handle` модуля `ForceBackup.php` блок-схемы которых представлены в приложениях Е и Ж соответственно,

- функция `removeOldBackups` предназначена для удаления с хранилищ старых резервных копий,

- функция `ProcessingClusterPlans` предназначена для обработки плана каждой машины и определения необходимости начала резервного копирования, блок-схема функции `ProcessingClusterPlans` приведена в приложении Е,

- функция `parseCurrentMachinesStates` предназначена для получения актуального состояния виртуальных машин в кластере, блок-схема функции `parseCurrentMachinesStates` приведена в приложении Ж,

- функция-предикат `checkPlanBackupTimeCome` проверяет необходимость начала резервного копирования в рамках конкретного плана,

- функция `VMsInformation` предназначена для выполнения команды для получения информации о виртуальных машинах на хосте,

- функция `backupExecuteCommands` предназначена для выполнения необходимых команд на хосте для создания резервной копии машины,

- функция `getBackupUUIDByName` предназначена для получения уникального идентификатора резервной копии,
- функция `mountStorage` предназначена для подключения удаленного хранилища,
- функция `unmountStorage` предназначена для отключения удаленного хранилища,
- функция `removeMachine` предназначена для выполнения команды на хосте, осуществляющей удаление резервной копии с хранилища,
- функция `setStatusMachine` предназначена для установки статуса машине,
- функция `deleteBackup` удаляет резервную копию с хранилища,
- функция `deletePlan` удаляет план резервного копирования,
- функция `editActivePlan` предназначена для активизации/деактивизации плана.

Модуль `Authenticate.php` является промежуточным обработчиком запросов и предназначен для проверки аутентификации пользователя, отправляющего запросы на сервер и состоит из одной функции `handle`, которая проверяет в объекте запроса уникальный ключ авторизации на подлинность, при успешной проверке, запрос отправляется далее на контроллеры сервера.

Модуль `LoginController.php` предназначен для аутентификации пользователя в системе и состоит из трех функций:

- 1) функция `checkAccount` предназначена для проверки учетных данных путем авторизации на удаленном хосте с введенными данными, путем подключения по протоколу `Secure Shell`,
- 2) функция `login` предназначена для аутентификации в системе, при успешной попытке, пользователю отправляется ключ аутентификации,
- 3) функция `logout` предназначена для выхода из системы и помечает ключ аутентификации как недействительный.

Примеры реализации интерфейса:

Запуск сервера происходит командой cargo run. (рис. 1.22).

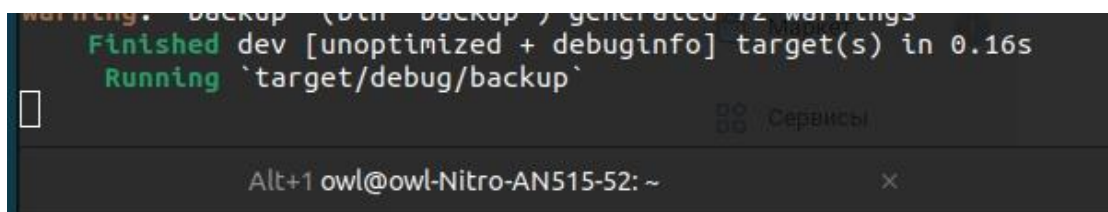
A terminal window with a dark background. The text shows a warning about backup files, followed by 'Finished dev [unoptimized + debuginfo] target(s) in 0.16s' and 'Running `target/debug/backup`'. The terminal title bar at the bottom reads 'Alt+1 owl@owl-Nitro-AN515-52: ~'.

Рисунок 1.22 – Запуск сервера

Запрос на соединение с управляющей машиной (рис. 1.23).

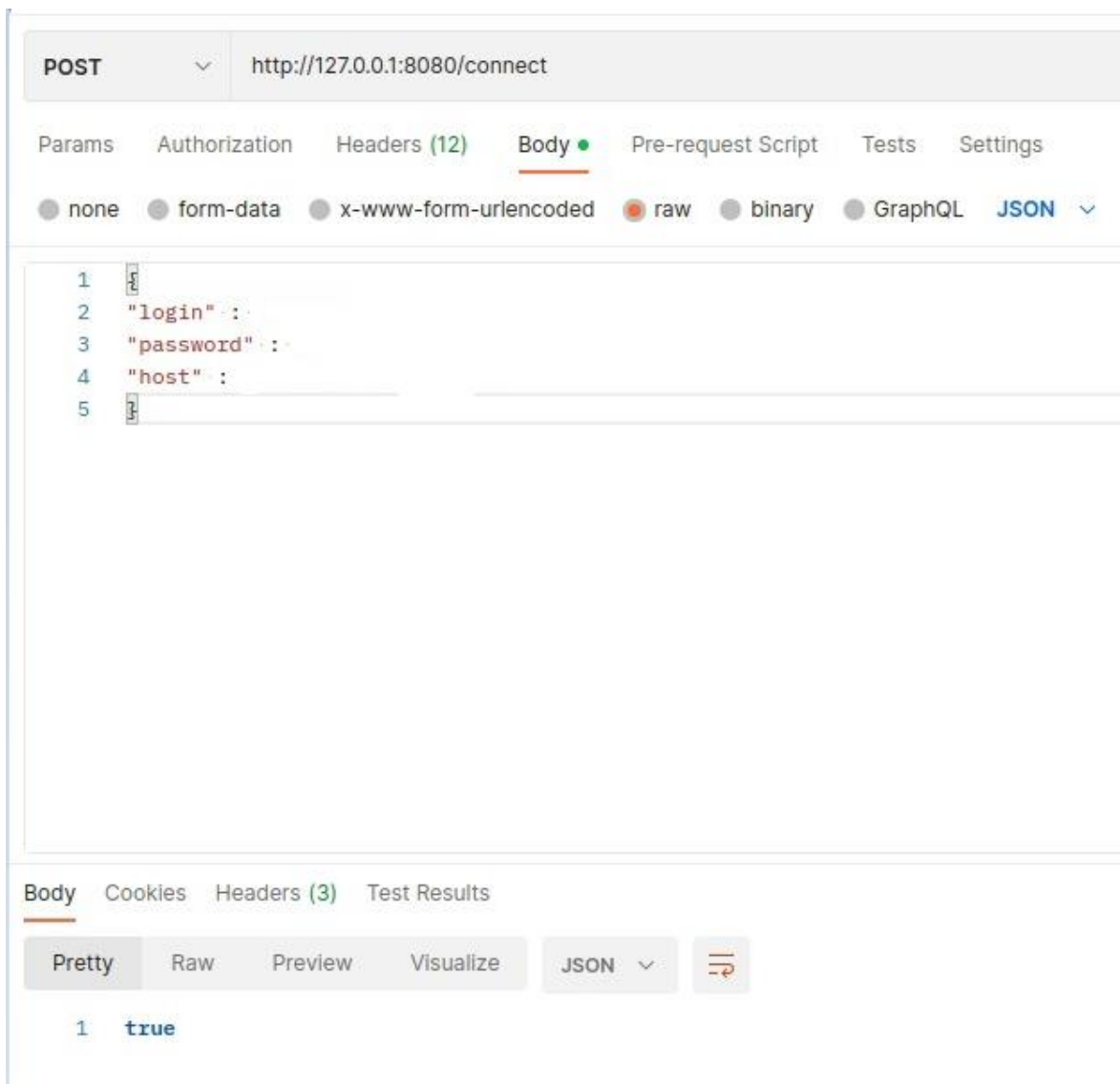


Рисунок 1.23 – Соединение с управляющей машиной

Запрос на получение времени из расписания из базы данных (рис. 1.24).

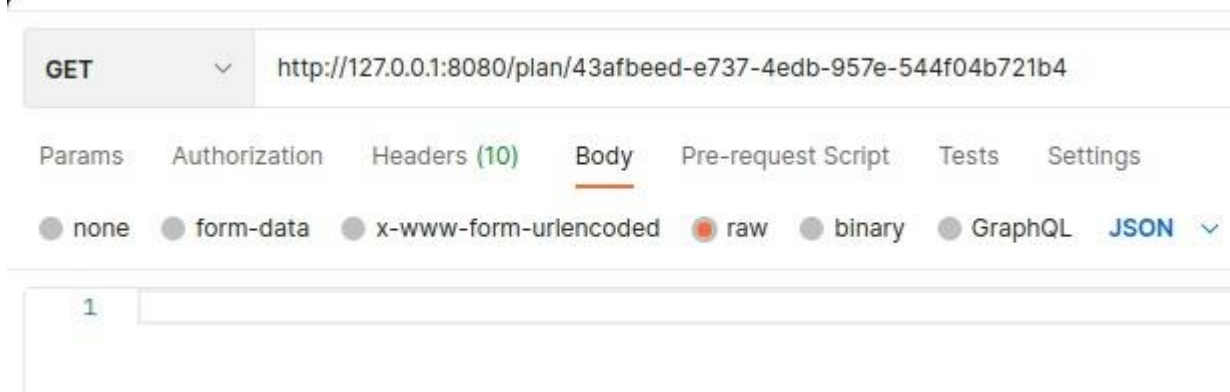


Рисунок 1.24 – Запрос времени из базы данных

Ответ на запрос о получении времени из расписания (рис. 1.25).

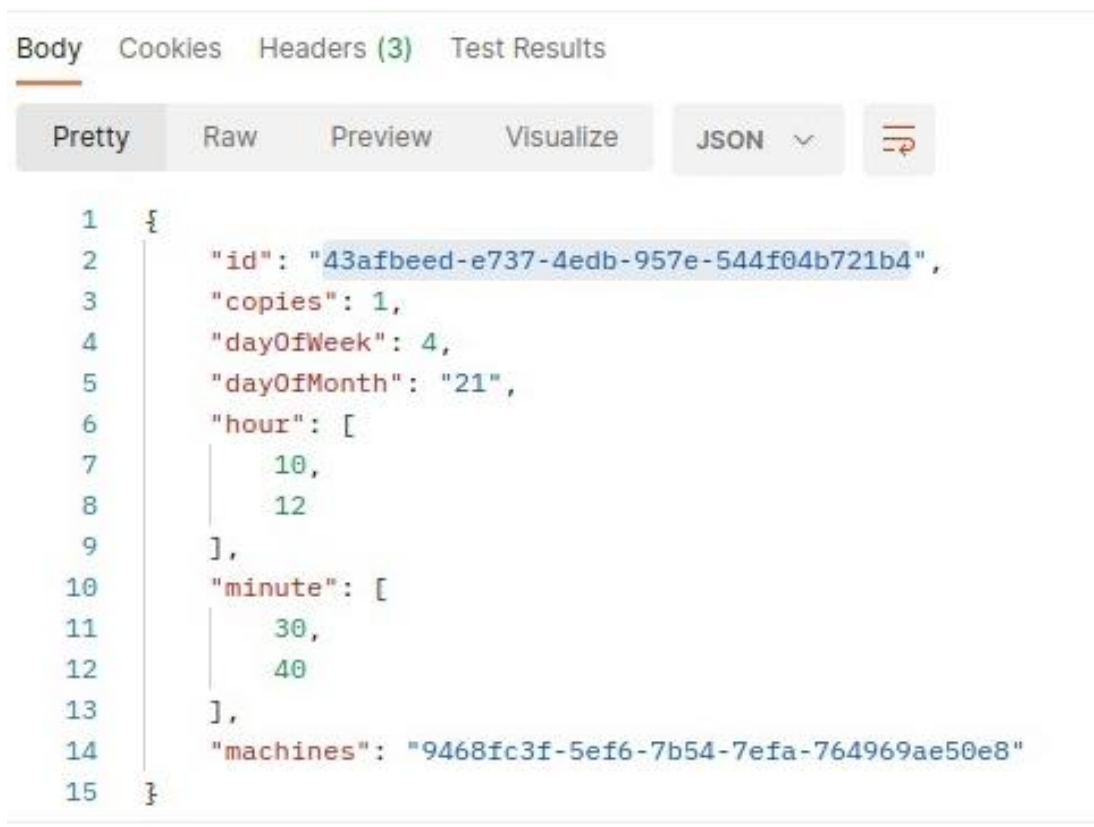


Рисунок 1.25 – Ответ на запрос о получении времени из базы данных.

Запись времени в cron и базу данных (рис. 1.26, 1.27).

```
30,40 10,12 21 * 4 [6 = 4] && echo '$copies : 1\nuuid : 9468fc3f-5ef6-7b54-7efa-764969ae50e8' > /root/test/xen  
-uuids.txt && /root/test/backupscrip.sh >> /root/test/crontab.log > /dev/null 2>&1
```

Рисунок 1.26 – Запись времени в cron

	uuid	updu	updt	copies	hour	minute	day_of_week	day_of_month	machines
1	43afbeed-e737-4edb-957e-54	NULL	2023-01-21 19:49:22.000	1	{10,12}	{30,40}	4	21	9468fc3f-5ef6-7b

Рисунок 1.27 – Запись времени в базу данных

Запрос на удаление времени из расписания и положительный ответ (рис. 1.28).

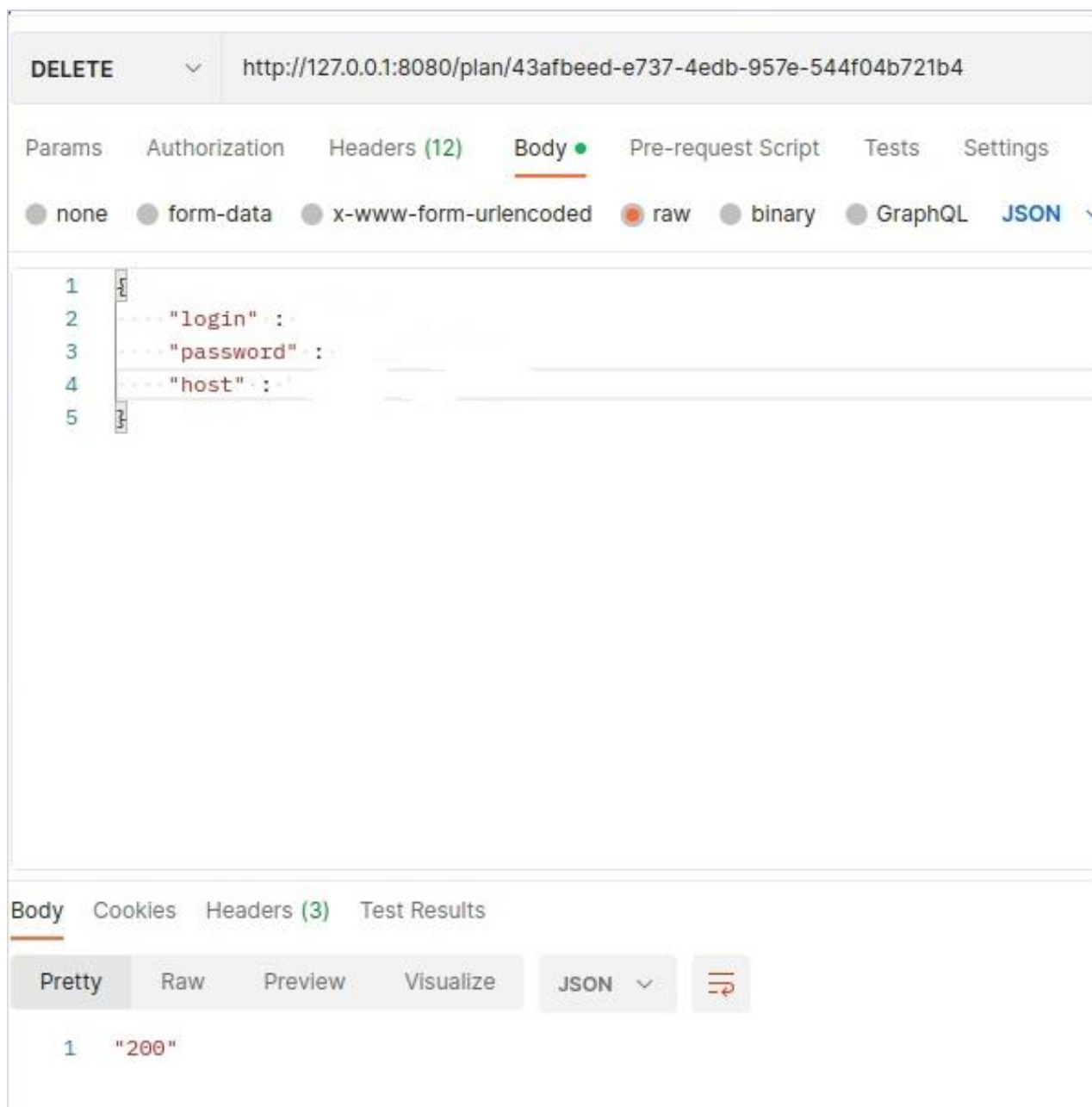


Рисунок 1.28 – Запрос удаления времени из расписания

Результаты удаления времени из расписания (рис. 1.29, 1.30).

```
[23:32 xcp-ng-b1 test]# crontab -l
[23:33 xcp-ng-b1 test]#
```

Рисунок 1.29 – Результат удаления времени из cron

uuid	updu	updt	copies	hour	minute	day_of_week	day_of_month	machines

Рисунок 1.30 – Результат удаления времени из базы данных

Запрос на создание уведомления (рис. 1.31).

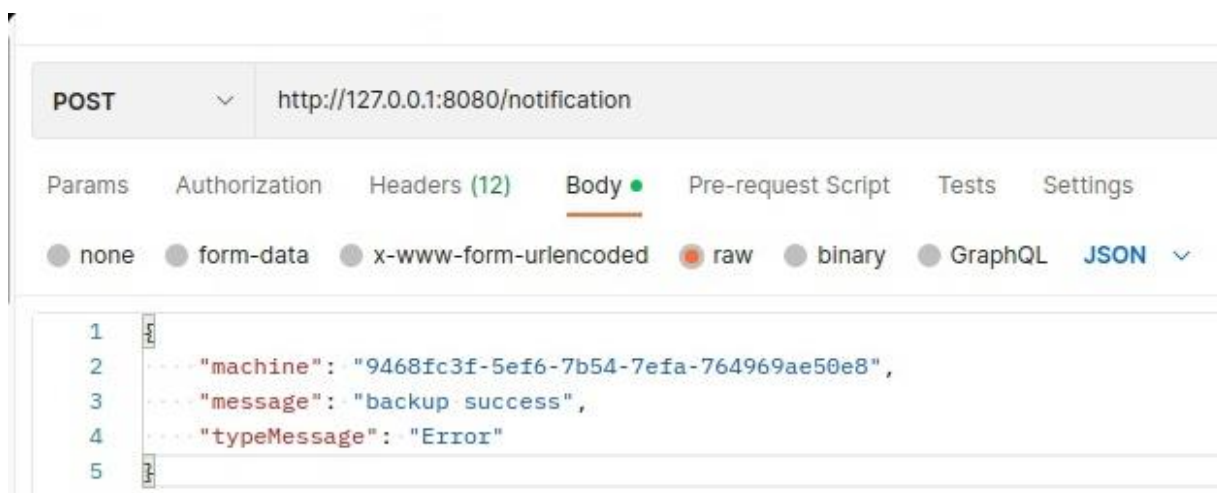


Рисунок 1.31 – Запрос на добавление уведомления

Ответ на запрос о создании уведомления (рис. 1.32).

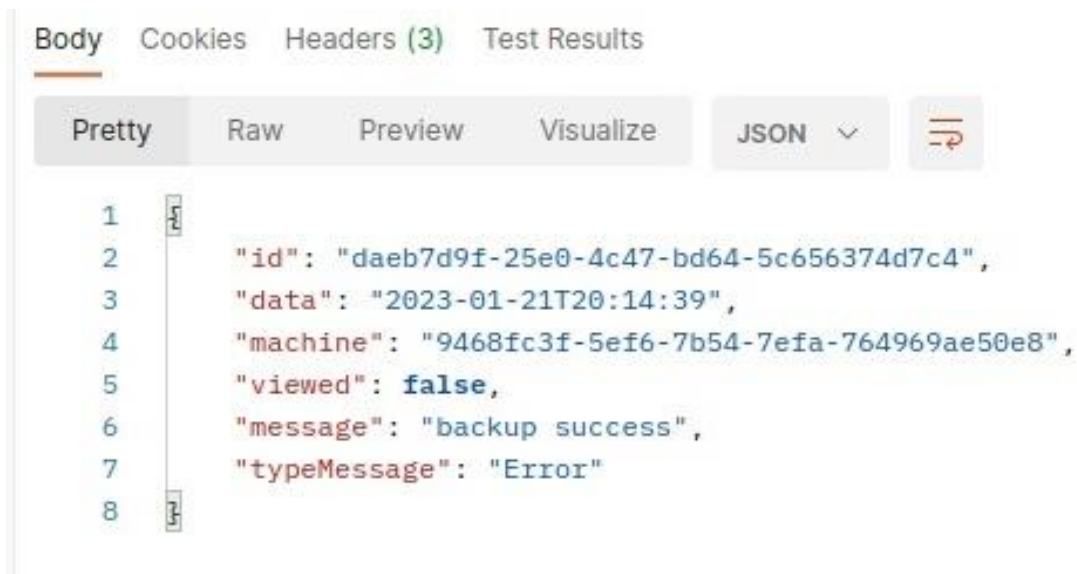
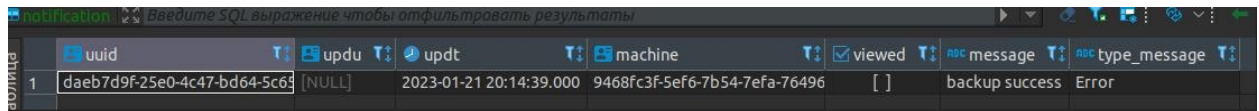


Рисунок 1.32 – Ответ на запрос о создании уведомления

Результат создания уведомления в базе данных (рис. 1.33).



	uuid	updu	updt	machine	viewed	message	type_message
1	daeb7d9f-25e0-4c47-bd64-5c63	[NULL]	2023-01-21 20:14:39.000	9468fc3f-5ef6-7b54-7efa-76496	[]	backup success	Error

Рисунок 1.33 – Уведомление в базе данных

Запрос на изменение уведомления в базе данных (рис. 1.34).

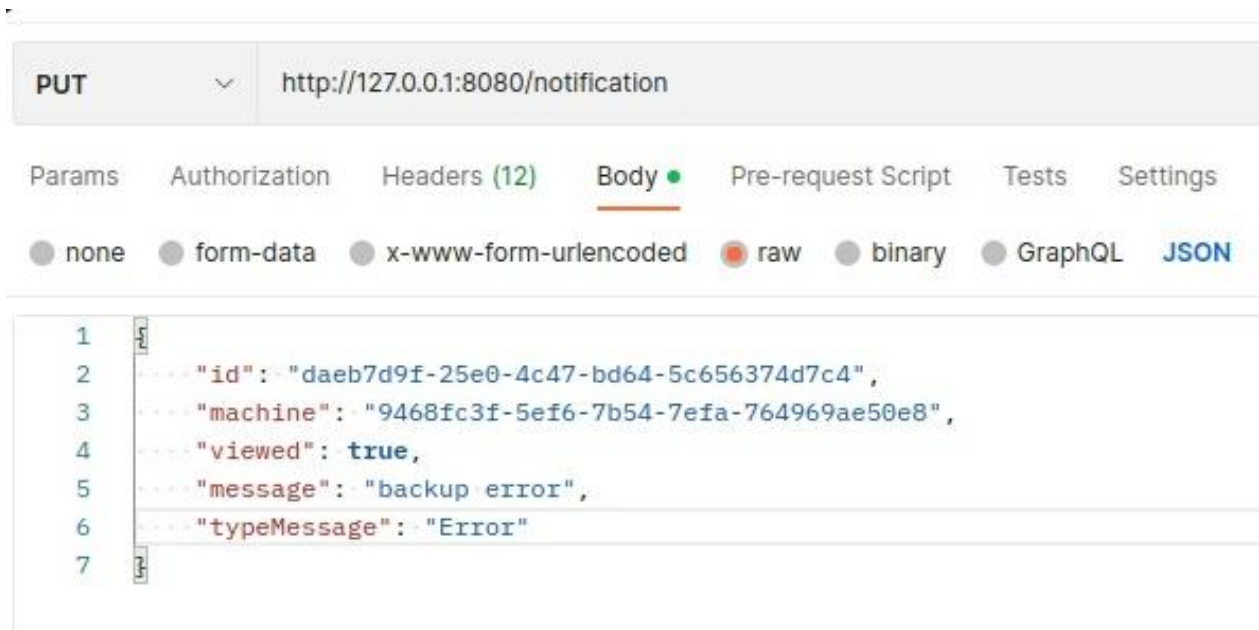
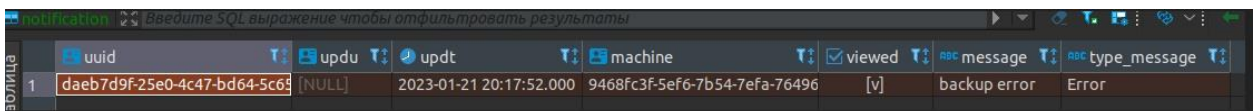


Рисунок 1.34 – Изменение уведомления

Результат изменения уведомления в базе данных (рис. 1.35).



	uuid	updu	updt	machine	viewed	message	type_message
1	daeb7d9f-25e0-4c47-bd64-5c63	[NULL]	2023-01-21 20:17:52.000	9468fc3f-5ef6-7b54-7efa-76496	[v]	backup error	Error

Рисунок 1.35 – Измененное уведомление в базе данных

1.5 Описание применения

Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин имеет пользовательский интерфейс в виде веб-приложения. При первом открытии приложения пользователя направит на страницу авторизации, на которой необходимо ввести учетные данные от реального пользователя на удаленном хосте, а также адрес этого хоста (рисунок 1.36).

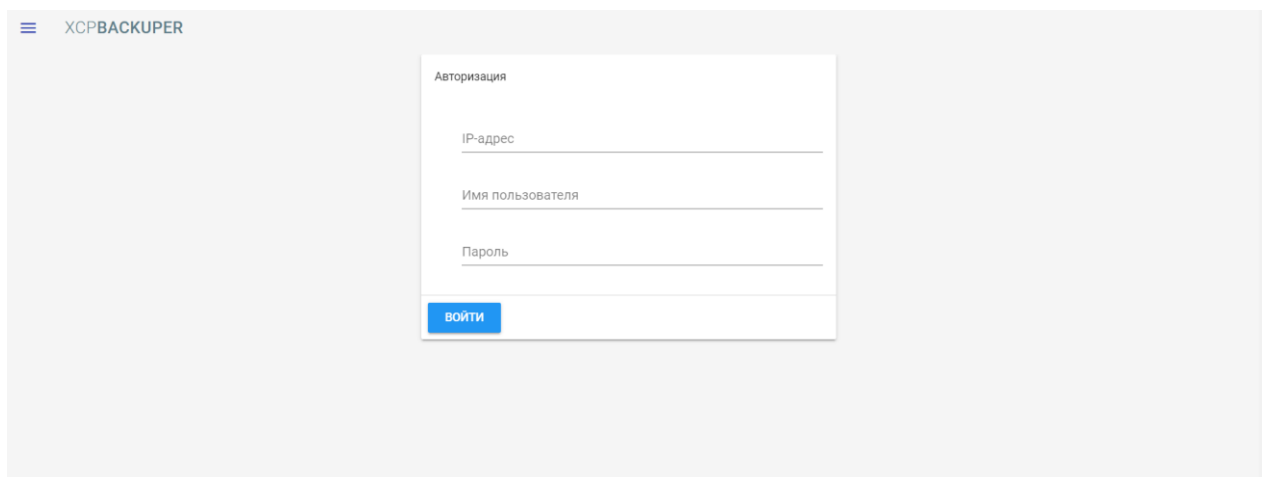


Рисунок 1.36 - Форма авторизации в приложении

При неверно введенных данных соответствующие поля формы будут подчеркнуты красным цветом, что означает то, что нужно проверить вводимые данные, рисунок 1.37.

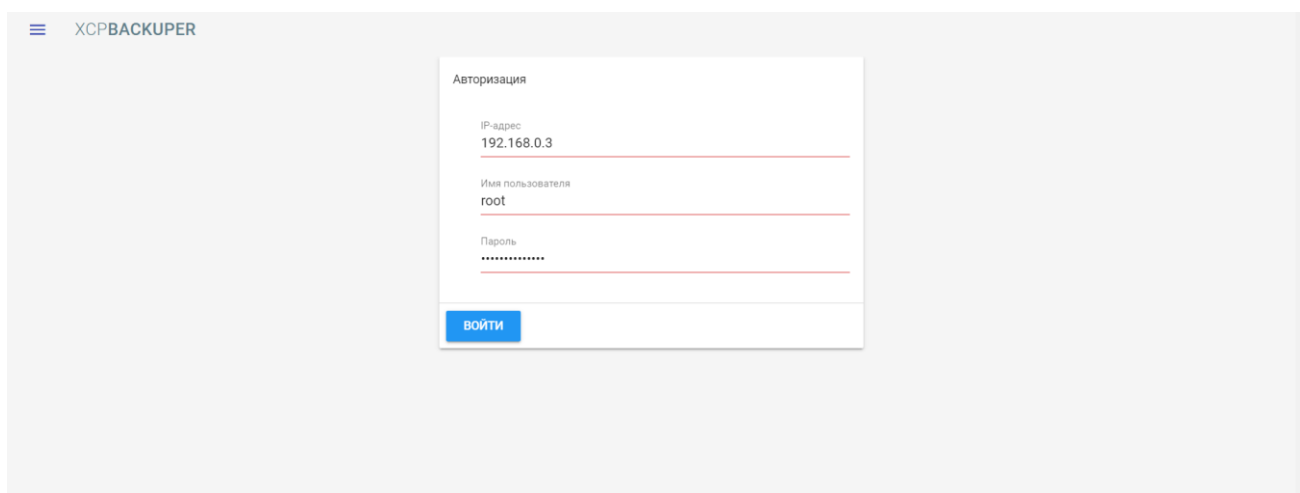


Рисунок 1.37 – Форма авторизации при вводе неверных учетных данных

При правильно введенных данных учетных данных, система загрузит виртуальные машины и хранилища с хоста, и далее пользователь будет переведен на страницу со списком машин рисунок 1.38.

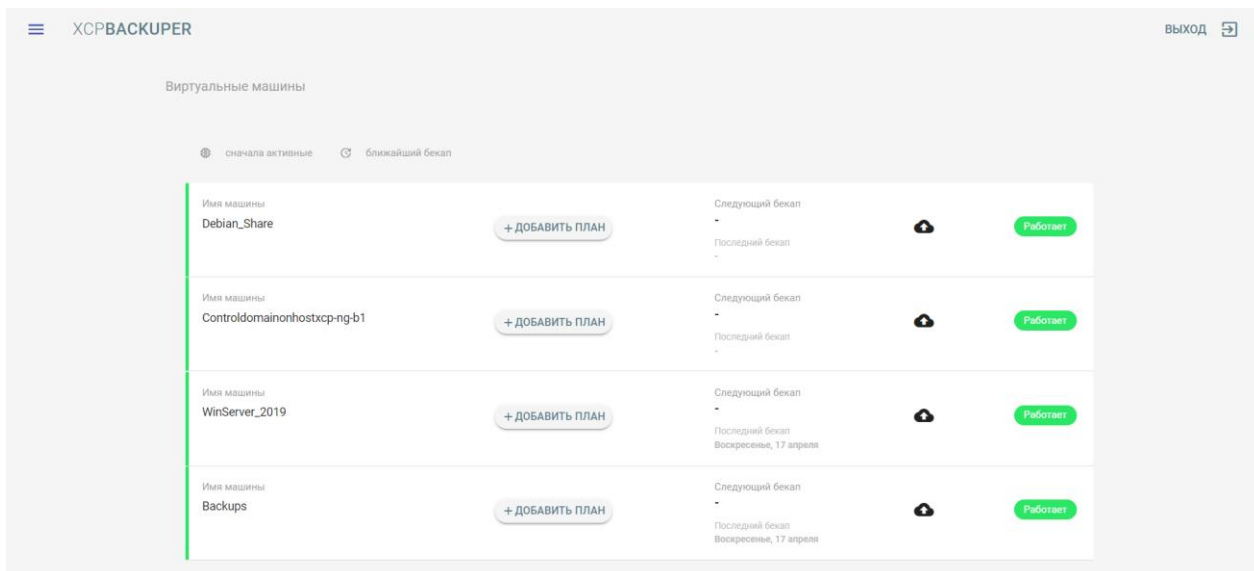


Рисунок 1.38 – Форма приложения со списком машин

Для машины появится возможность создать план резервного копирования, для этого необходимо напротив нужной машины в списке нажать на кнопку «Добавить план», после чего, появится диалоговое окно, в котором необходимо из выпадающего списка выбрать хранилище для сохраняемых резервных копий, в соответствующих текстовых полях ввести период создания копий в днях, и количество сохраняемых копий (рисунок 1.39).

Создание плана для машины "Debian_Share"

Хранилище

Период создания копий (дней)

Количество бэкапов

Рисунок 1.39 – Модальное окно создания плана резервного копирования

После заполнения всех полей модального окна, необходимо нажать кнопку «Добавить», после чего форма со списком машин примет следующий вид (рисунок 1.40), у указанной машины будут отображаться настройки текущего плана резервного копирования, а также информация о том, когда будет создана следующая резервная копия и дата последней.

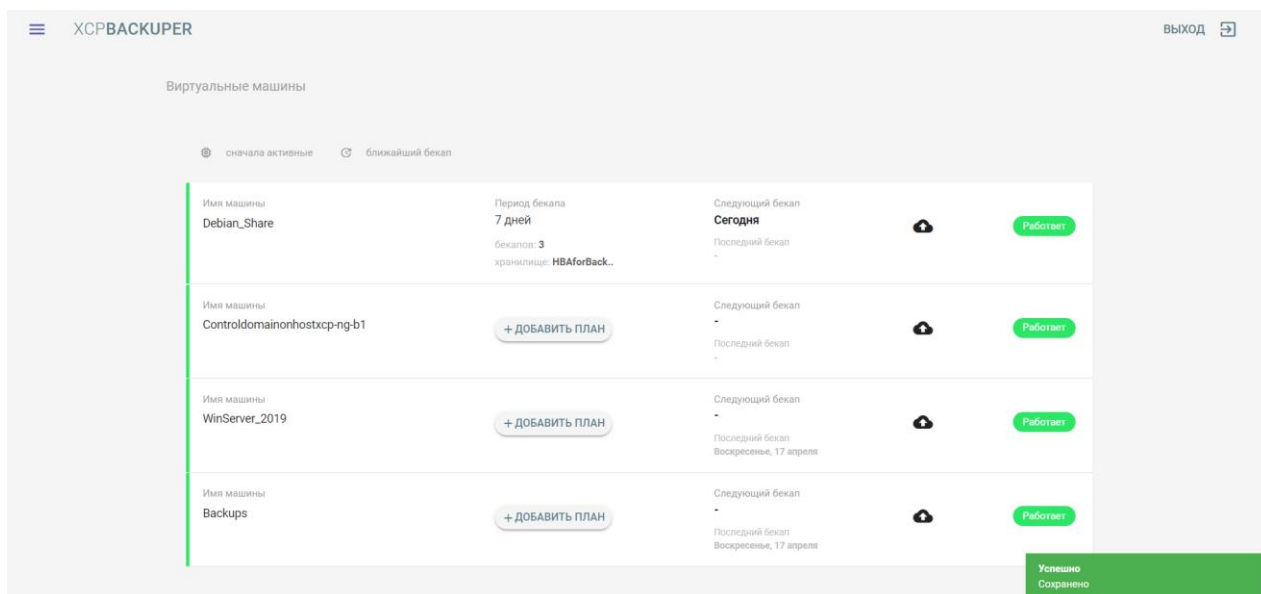


Рисунок 1.40 – Форма со списком машин после создания плана

Для редактирования настроек резервного копирования машины, необходимо навести курсором мыши на строку с указанной машиной на форме списка машин, после чего покажутся дополнительные кнопки настроек (рисунок 1.41), и нажать на кнопку «Изменить план», после чего откроется модальное окно редактирования настроек резервного копирования (рисунок 1.42), или на странице со списком планов резервного копирования (рисунок 1.43) напротив выбранного плана нажать на кнопку «Редактировать» с иконкой в виде карандаша, после чего откроется такое-же модальное окно (рисунок 1.42). После редактирования настроек для их сохранения необходимо нажать кнопку «Сохранить».

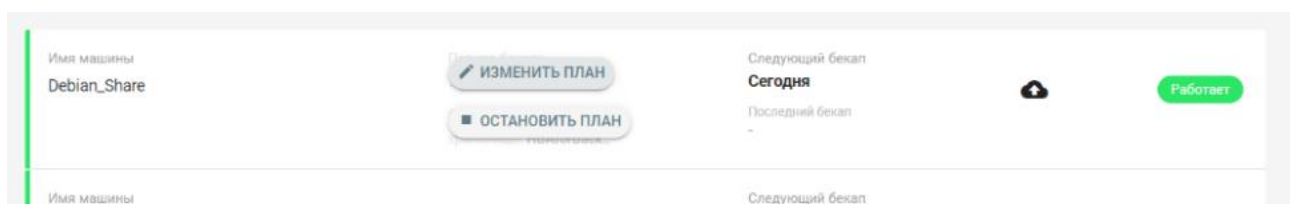


Рисунок 1.41 – Дополнительные кнопки управления настройками резервного копирования

Редактирование плана для машины Debian_Share

Хранилище
HBAforBacku

Период создания копий (дней)
7

Количество бэкапов
3

ПРИОСТАНОВИТЬ **ОТМЕНА** **СОХРАНИТЬ**

Рисунок 1.42 – Модальное окно редактирования настроек резервного копирования

Для того, чтобы просмотреть сделанные резервные копии, необходимо перейти на страницу списка планов, для чего в меню нажать на кнопку «Планы бекапа» в основном меню, после чего на странице планов выбрать план для нужной машины и нажать на кнопку «Список бэкапов», после чего откроется выпадающий список созданных резервных копий (рисунок 1.43).

ХСРВАКУПЕР ВЫХОД

Активные планы

Имя машины	Хранилище	Период бекапа	Сохраняемых бэкапов		
Debian_Share	SMB	14 дней	2		
Список бэкапов (2):					
Название файла Debian_Share-vm-backup-on-storage-18052022	Дата Среда, 18 мая	UUID 9984cb72-74a0-3435-6962-e7b68f3c3cb4	УДАЛИТЬ		
Название файла Debian_Share-vm-backup-on-storage-25052022	Дата Сегодня	UUID 8428e92e-00a5-2412-472b-0d62cad55321	УДАЛИТЬ		
Имя машины WinServer_2019	Хранилище SMB	Период бекапа 30 дней	Сохраняемых бэкапов 1		
Список бэкапов (1):					

Рисунок 1.43 – Страница списка планов резервного копирования с раскрытым списком резервных копий

Для старта резервного копирования в ручном режиме, необходимо напротив выбранной машины нажать на кнопку «Запустить бекап сейчас» с иконкой в виде облака, после чего появится модальное окно подтверждения действия (рисунок 1.44) в котором необходимо нажать на кнопку «Да».

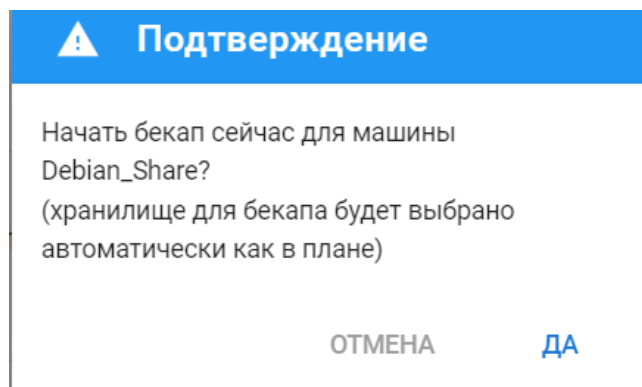


Рисунок 1.44 – Модальное окно подтверждения начала резервного копирования

Для приостановки работы плана резервного копирования необходимо на странице навести курсором мыши на строку с указанной машиной на форме списка машин, после чего покажутся дополнительные кнопки настроек (рисунок 1.41) и нажать на кнопку «Остановить план», после чего в модальном окне (рисунок 1.45) подтвердить приостановку, нажав на кнопку «Да», или на странице со списком планов резервного копирования (рисунок 1.43) напротив выбранного плана нажать на кнопку «Остановить план» с иконкой в виде квадрата, после чего откроется такое-же модальное окно (рисунок 1.45).

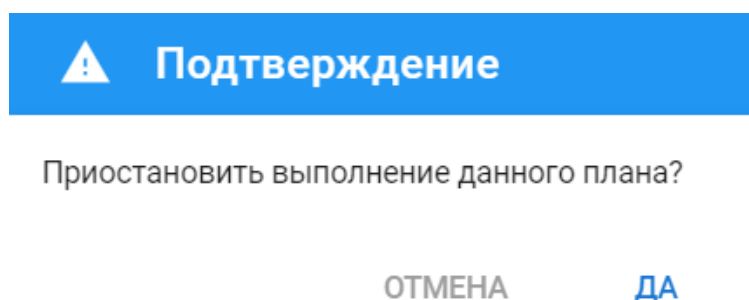
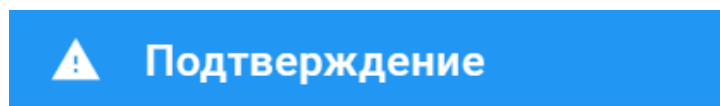


Рисунок 1.45 – Модальное окно подтверждения остановки работы плана

Для того, чтобы удалить резервную копию с хранилища, необходимо перейти на страницу списка планов резервного копирования и напротив выбранной резервной копии нажать на кнопку «удалить» (рисунок 1.43), после чего в появившемся модальном окне подтвердить удаление нажав кнопку «Да» (рисунок 1.46).



Удалить бэкап с хранилища?

ОТМЕНА

ДА

Рисунок 1.46 – Окно подтверждения удаления резервной копии

Для удаления неактивного плана резервного копирования необходимо на странице со списком планов напротив выбранного плана (рисунок 1.47), нажать на кнопку «Удалить план» с иконкой в виде корзины, после чего в модальном окне (рисунок 1.48) подтвердить удаления нажав на кнопку «Да». Для возобновления работы плана резервного копирования необходимо на странице со списком планов выбрать неактивный план (рисунок 1.47) и нажать на кнопку «Запустить план» с иконкой в виде стрелки, после чего в появившемся модальном окне (рисунок 1.49) подтвердить возобновление нажатием кнопки «Да».

Отключенные планы					
Имя машины	Хранилище	Период бекапа	Сохраняемых бекапов		
WinServer_2019	NBAforBackup	7 дней	3		I→
Список бекапов (0):					
Имя машины	Хранилище	Период бекапа	Сохраняемых бекапов		
Backups	SMB	7 дней	3		I→
Список бекапов (0):					
Имя машины	Хранилище	Период бекапа	Сохраняемых бекапов		
Backups	SMB	7 дней	3		I→
Список бекапов (4):					

Рисунок 1.47 – Список неактивных планов резервного копирования

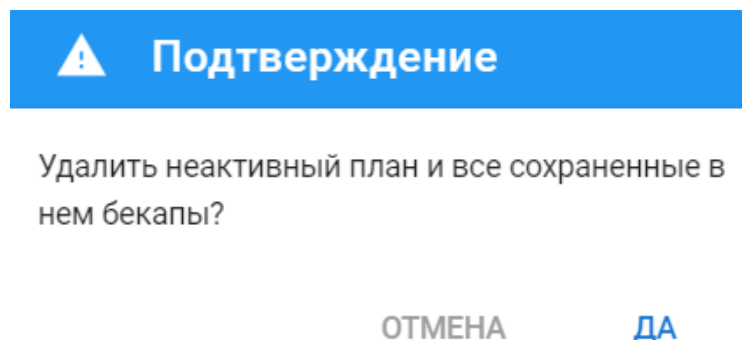


Рисунок 1.48 – Модальное окно подтверждения удаления плана

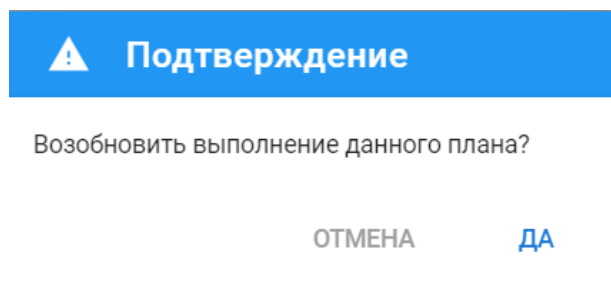


Рисунок 1.49 – Окно подтверждения возобновления работы плана

Для того, чтобы задать учетные данные для хранилища, необходимо перейти на страницу со списком хранилищ, нажав на кнопку «Хранилища» главного меню, после чего в списке выбрать удаленное хранилище и нажать на кнопку «Редактировать данные» с иконкой в виде карандаша (рисунок 1.50), после чего в появившемся модальном окне (рисунок 1.51) заполнить учетные данные, и нажать на кнопку «Проверить подключение», после чего в случае правильно введенных данных на экране появится информационное сообщение об успешности подключения (рисунок 1.52), после чего можно закрыть модальное окно нажав кнопку «Закрыть».

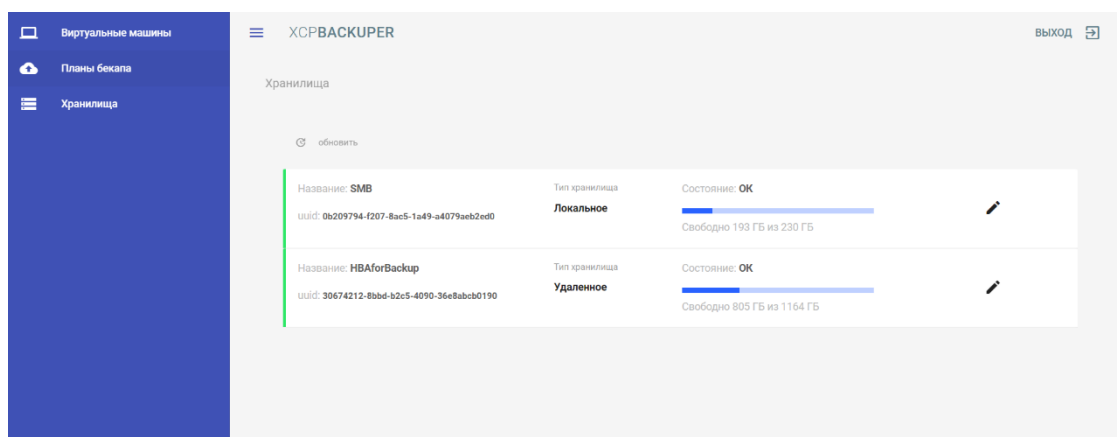


Рисунок 1.50 – Форма со списком хранилищ

The screenshot shows a web interface for configuring an SMB storage. It contains four input fields: 'Название' (Name) with the value 'SMB', 'Путь' (Path) with the value '\\172.16.191.254\backup', 'Логин' (Login) with the value 'adminback', and 'Пароль' (Password) with masked characters. At the bottom, there are three buttons: 'ПРОВЕРИТЬ ПОДКЛЮЧЕНИЕ' (Check connection), 'ЗАКРЫТЬ' (Close), and 'СОХРАНИТЬ' (Save).

Рисунок 1.51 – Окно редактирования информации о хранилище

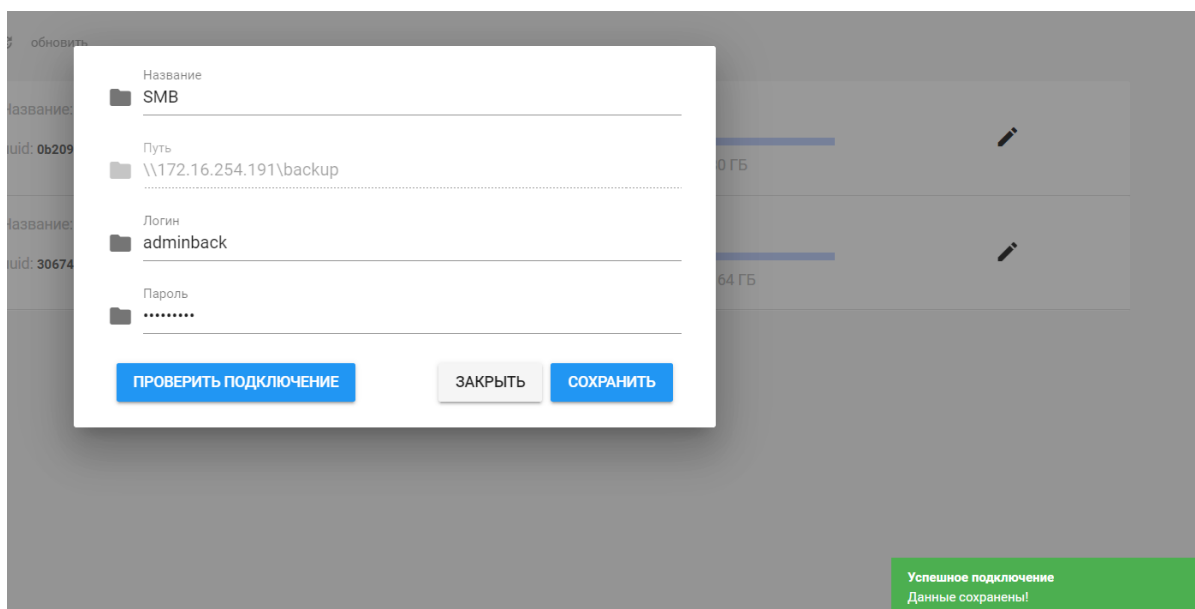


Рисунок 1.52 – Сообщение успешности подключения хранилища

1.6 Тестирование приложения

Тестирование является важнейшим этапом в процессе разработки программного обеспечения. В ходе тестирования выявляются и устраняются недоработки и ошибки в программном коде. Без полноценного тестирования нельзя быть уверенным в дальнейшей стабильности работы системы, что создает потенциальные риски для бизнеса в будущем.

В ходе разработки автоматизированной системы она подвергалась модульному тестированию для каждой реализованной функции. Для

тестирования были разработаны тестовые примеры, которые иллюстрируют правильную работу функционала. Для демонстрации приведем один такой пример для функции «Редактирование настроек резервного копирования».

Для тестирования данной функции, необходимо перейти на главную страницу со списком машин, после чего необходимо навести курсором мыши на строку с 1 машиной, после чего покажутся дополнительные кнопки настроек (рисунок 1.41), и нажать на кнопку «Изменить план», после чего откроется модальное окно редактирования настроек резервного копирования (рисунок 1.42), после чего заменить количество резервных копий (установить 1), изменить период резервного копирования (установить 14), далее нажать на кнопку сохранить, после чего должно появиться сообщение об успешном сохранении данных (рисунок 1.53), при проверке на странице со списком машин, указанные данные должны быть отображены (рисунок 1.54).



Рисунок 1.53 – Сообщение об успешности сохранения данных

Имя машины	Период бекапа
Debian_Share	14 дней
	бекапов: 1
	хранилище: HBAforBack..

Рисунок 1.54 – Отображение сохраненных данных на странице со списком машин

ЗАКЛЮЧЕНИЕ

В ходе выполнения магистерской диссертации была изучена предметная область. В результате чего было сформулировано техническое задание на разработку автоматизированной системы для создания и учета резервных копий в кластере виртуальных машин.

Был произведен выбор инструментов для разработки клиентской и серверной части автоматизированной системы, выполнено инфологическое проектирование в результате которого были диаграмма: пакетов, вариантов использования, последовательности, деятельности, классов и развертывания автоматизированной системы.

Было осуществлено проектирование базы данных, в результате которого были разработаны концептуальная, логическая и физическая модели данных, а также проведена нормализация базы данных.

Было разработано клиент-серверное приложение, которое позволяет осуществлять создание и учет резервных копий виртуальных машин, настраивать параметры резервного копирования. В приложении реализована авторизация пользователей, информация о виртуальных машинах, хранилищах и резервных копиях хранится в базе данных.

В разделе безопасности жизнедеятельности были выявлены основные вредные факторы воздействия компьютера на человека, описаны требования к офисным помещениям для минимизации воздействия этих факторов, а также произведен расчет необходимой освещенности рабочего места в помещении.

В разделе был произведен расчет трудоемкости и затрат на разработку автоматизированной системы для создания и учета резервных копий в кластере виртуальных машин, а также произведен примерный подсчет минимальной цены разработки и затрат на внедрение автоматизированной системы.

Магистерская диссертация представляет собой законченную работу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Питкевич Павел Игоревич Методы Резервирования Данных Для Критически Важных Ит- Систем Предприятия // Universum: технические науки. 2021. №10-1 (91).
2. Бережной А. Резервное Копирование Виртуальных Машин - Системный Администратор. 2013. № 10 (131).
3. Николай Н. Российский рынок серверной виртуализации: тенденции и игроки // ИксМедиа, [Электронный ресурс]: Режим доступа: <https://www.iksmedia.ru/articles/5542238-Rossijskij-rynok-servernoj-virtuali.html>.
4. Экспресс 42: Состояние технологий виртуализации в 2020 году, [Электронный ресурс]: Режим доступа: <https://express42.com/news/state-of-virtualization-2020/>.
5. Чомаева М.Н. Компьютер как фактор вредного воздействия на здоровье человека // Международный журнал гуманитарных и естественных наук. 2020. №7-2. Режим доступа: <https://cyberleninka.ru/article/n/kompyuter-kak-faktor-vrednogo-vozdeystviya-na-zdorovie-cheloveka>.
6. [Электронный ресурс]: https://www.vinchin.com/en/res/pdf/User_Guide_Vinchin_Backup_&_Recovery_v7.0.pdf.
7. [Электронный ресурс]: <https://cyberprotect.ru/documentation/backup>.
8. [Электронный ресурс]: <https://www.iperiusbackup.net/iperius-backup-7-8-best-backup-software-vmware-esxi-vsphere-8/>.
9. [Электронный ресурс]: <https://www.acronis.com/en-us/support/documentation/>.

ПРИЛОЖЕНИЕ А

Листинг кода базы данных

```

-- phpMyAdmin SQL Dump
-- version 5.1.1
-- https://www.phpmyadmin.net/

--

-- Table structure for table `authorization`
--

CREATE TABLE `authorization` (
  `ID` int(11) NOT NULL,
  `TOKEN` varchar(255) NOT NULL,
  `ACTIVE` tinyint(1) NOT NULL DEFAULT 1,
  `USER` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4,

--

-- Dumping data for table `authorization`

--

-- Table structure for table `backup`
--

CREATE TABLE `backup` (
  `ID` int(11) NOT NULL,
  `UUID_BACKUP` varchar(255) NOT NULL,
  `TIME` datetime DEFAULT NULL,
  `NAME` text NOT NULL,

```

```

`RESTORING` tinyint(1) NOT NULL DEFAULT 0,

`PLAN` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8,

--

-- Dumping data for table `backup`

-

--

-- Table structure for table `failed_jobs`

--

CREATE TABLE `failed_jobs` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `connection` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `queue` text COLLATE utf8mb4_unicode_ci NOT NULL,
  `payload` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
  `exception` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
  `failed_at` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_c

--

-- Table structure for table `jobs`

--

CREATE TABLE `jobs` (
  `id` bigint(20) UNSIGNED NOT NULL,
  `queue` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,

```

```

`payload` longtext COLLATE utf8mb4_unicode_ci NOT NULL,
`attempts` tinyint(3) UNSIGNED NOT NULL,
`reserved_at` int(10) UNSIGNED DEFAULT NULL,
`available_at` int(10) UNSIGNED NOT NULL,
`created_at` int(10) UNSIGNED NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_c

```

```
--
```

```
-- Table structure for table `machine`
```

```
--
```

```

CREATE TABLE `machine` (
  `ID` int(11) NOT NULL,
  `UUID` varchar(255) NOT NULL,
  `NAME` varchar(120) NOT NULL,
  `STATUS` int(11) NOT NULL DEFAULT 1,
  `USER` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8,

```

```
--
```

```
-- Dumping data for table `machine`
```

```
--
```

```
-- Table structure for table `migrations`
```

```
--
```

```

CREATE TABLE `migrations` (
  `id` int(10) UNSIGNED NOT NULL,

```

```

    `migration` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
    `batch` int(11) NOT NULL
)          ENGINE=InnoDB          DEFAULT          CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci,

--

-- Dumping data for table `migrations`

--

-- Table structure for table `plan`
--

CREATE TABLE `plan` (
  `ID_PLAN` int(11) NOT NULL,
  `PLAN_MACHINE_ID` int(11) NOT NULL,
  `PLAN_STORAGE_ID` int(11) NOT NULL,
  `DAYS_BACKUP` int(11) NOT NULL,
  `COUNT_BACKUPS` int(11) NOT NULL DEFAULT 3,
  `ACTIVE` tinyint(1) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4,

--

-- Dumping data for table `plan`

--

-- Table structure for table `status`

```

--

```
CREATE TABLE `status` (  
  `ID` int(11) NOT NULL,  
  `TITLE` varchar(255) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8,
```

--

-- Dumping data for table `status`

-

--

-- Table structure for table `storage`

--

```
CREATE TABLE `storage` (  
  `ID_STORAGE` int(11) NOT NULL,  
  `CAPACITY` bigint(20) DEFAULT NULL,  
  `UTILISATION` bigint(20) DEFAULT NULL,  
  `USER_ID_STORAGE` int(11) NOT NULL,  
  `TITLE_STORAGE` varchar(255) DEFAULT NULL,  
  `TYPE_STORAGE` tinyint(4) NOT NULL DEFAULT 0,  
  `UUID_STORAGE` varchar(255) NOT NULL,  
  `IP_STORAGE` varchar(255) DEFAULT NULL,  
  `LOGIN` varchar(255) DEFAULT NULL,  
  `PASSWORD` varchar(255) DEFAULT NULL,  
  `ACTIVE` tinyint(1) NOT NULL DEFAULT 0  
) ENGINE=InnoDB DEFAULT CHARSET=utf8,
```

```

--
-- Dumping data for table `storage`

--
-- Table structure for table `user`
--

CREATE TABLE `user` (
  `ID` int(11) NOT NULL,
  `HOST` varchar(255) NOT NULL,
  `LOGIN` varchar(255) NOT NULL,
  `PASSWORD` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8,

--
-- Dumping data for table `user`
--
--
-- Indexes for dumped tables
--

--
-- Indexes for table `authorization`
--

ALTER TABLE `authorization`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `USER_ID` (`USER`),

```

```

--
-- Indexes for table `backup`
--
ALTER TABLE `backup`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `PLAN` (`PLAN`),

--
-- Indexes for table `failed_jobs`
--
ALTER TABLE `failed_jobs`
  ADD PRIMARY KEY (`id`),

--
-- Indexes for table `jobs`
--
ALTER TABLE `jobs`
  ADD PRIMARY KEY (`id`),
  ADD KEY `jobs_queue_index` (`queue`),

--
-- Indexes for table `machine`
--
ALTER TABLE `machine`
  ADD PRIMARY KEY (`ID`),
  ADD KEY `Machine_fk1` (`STATUS`),
  ADD KEY `Machine_fk3` (`USER`),

```

```

--
-- Indexes for table `migrations`
--
ALTER TABLE `migrations`
    ADD PRIMARY KEY (`id`),

--
-- Indexes for table `plan`
--
ALTER TABLE `plan`
    ADD PRIMARY KEY (`ID_PLAN`),
    ADD KEY `MACHINE_ID` (`PLAN_MACHINE_ID`),
    ADD KEY `STORAGE_ID` (`PLAN_STORAGE_ID`),

--
-- Indexes for table `status`
--
ALTER TABLE `status`
    ADD PRIMARY KEY (`ID`),

--
-- Indexes for table `storage`
--
ALTER TABLE `storage`
    ADD PRIMARY KEY (`ID_STORAGE`),
    ADD KEY `USER_ID_STORAGE` (`USER_ID_STORAGE`),

```

```

--
-- Indexes for table `user`
--
ALTER TABLE `user`
    ADD PRIMARY KEY (`ID`),

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `authorization`
--
ALTER TABLE `authorization`
    MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=21,

--
-- AUTO_INCREMENT for table `backup`
--
ALTER TABLE `backup`
    MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT,
    AUTO_INCREMENT=178,
-
-- AUTO_INCREMENT for table `failed_jobs`
--
ALTER TABLE `failed_jobs`
    MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
    AUTO_INCREMENT=7,

```

```

--
-- AUTO_INCREMENT for table `jobs`
--
ALTER TABLE `jobs`
    MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
    AUTO_INCREMENT=1696,

--
-- AUTO_INCREMENT for table `machine`
--
ALTER TABLE `machine`
    MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=51,

--
-- AUTO_INCREMENT for table `migrations`
--
ALTER TABLE `migrations`
    MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
    AUTO_INCREMENT=5,

--
-- AUTO_INCREMENT for table `plan`
--
ALTER TABLE `plan`
    MODIFY `ID_PLAN` int(11) NOT NULL AUTO_INCREMENT,
    AUTO_INCREMENT=37,

--
-- AUTO_INCREMENT for table `status`

```

```

--
ALTER TABLE `status`

    MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8,

--
-- AUTO_INCREMENT for table `storage`
--
ALTER TABLE `storage`

    MODIFY `ID_STORAGE` int(11) NOT NULL AUTO_INCREMENT,
    AUTO_INCREMENT=6,

--
-- AUTO_INCREMENT for table `user`
--
ALTER TABLE `user`

    MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6,

--
-- Constraints for dumped tables
-- Constraints for table `authorization`
--
ALTER TABLE `authorization`

    ADD CONSTRAINT `USER_ID` FOREIGN KEY (`USER`) REFERENCES `user`
    (`ID`),

--
-- Constraints for table `backup`
--
ALTER TABLE `backup`

```

```

    ADD CONSTRAINT `PLAN_ID` FOREIGN KEY (`PLAN`) REFERENCES `plan`
    (`ID_PLAN`),

--

-- Constraints for table `machine`

--

ALTER TABLE `machine`

    ADD CONSTRAINT `Machine_fk1` FOREIGN KEY (`STATUS`) REFERENCES
    `status` (`ID`),

    ADD CONSTRAINT `Machine_fk3` FOREIGN KEY (`USER`) REFERENCES
    `user` (`ID`),

--

-- Constraints for table `plan`

--

ALTER TABLE `plan`

    ADD CONSTRAINT `MACHINE_ID` FOREIGN KEY (`PLAN_MACHINE_ID`)
    REFERENCES `machine` (`ID`),

    ADD CONSTRAINT `STORAGE_ID` FOREIGN KEY (`PLAN_STORAGE_ID`)
    REFERENCES `storage` (`ID_STORAGE`),

--

-- Constraints for table `storage`

--

ALTER TABLE `storage`

    ADD CONSTRAINT `USER_ID_STORAGE` FOREIGN KEY
    (`USER_ID_STORAGE`) REFERENCES `user` (`ID`),

COMMIT,

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

```

```
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */,
```

ПРИЛОЖЕНИЕ Б

Листинг программы

```

<template>

  <v-container class="machines">

    <h1 class="subheading grey--text">

      Виртуальные машины

    </h1>

    <v-container class="my-4">

      <v-layout row justify-start class="mb-3">

        <v-tooltip top>

          <v-btn

            :style="sort === 'active' ? 'background-
color: #8080803b,' : ''"

            small flat color="grey"
@click="sortBy('active')" slot="activator">

              <v-icon small left>memory</v-icon>

              <span class="caption text-
lowercase">сначала активные</span>

            </v-btn>

            <span>Сортировка: сначала активные
машины</span>

          </v-tooltip>

          <v-tooltip top>

            <v-btn

              :style="sort === 'backup' ? 'background-
color: #8080803b,' : ''"

              small flat color="grey"
@click="sortBy('backup')" slot="activator">

                <v-icon small left>update</v-icon>

                <span class="caption text-
lowercase">Ближайший бекап</span>

```

```

        </v-btn>

        <span>Сортировка: по дате бекапа</span>

    </v-tooltip>

</v-layout>

    <v-card flat v-for="(machine,i) in machines"
:key="machine.title"

        @mouseenter="showButtonsMachine=i"

        @mouseleave="showButtonsMachine=null"

    >

        <v-layout row wrap :class="`pa-3 project
${statuses[machine.status-1].style}`">

            <v-flex md4>

                <div class="caption grey--text">Имя
машины</div>

                <div class="mt-1">

                    {{ machine.title }}

                    <b></b>

                </div>

            </v-flex>

            <v-flex md3 style="position:relative">

                <div v-if="machine.plan"

                    :style="showButtonsMachine === i ?

{opacity: 0.3}:'">

                    <div>

                        <p class="caption grey--text
darken-3 my-0">Период бекапа</p>

                        <p class="grey--text text--darken-
3 my-0">{{ machine.plan.DAYS_BACKUP }} дней</p>

```

```

        </div>

        <div class="mt-2">

            <p class="my-0">

                <span class="caption text--
lighten-1 grey--text my-0">бекапов:</span>

                <span class="caption grey--
text text--darken-2 font-weight-bold my-0">

                    {{
machine.plan.COUNT_BACKUPS }}

                </span>

            </p>

            <p class="text--secondary grey--
text my-0">

                <span class="caption text--
lighten-1 grey--text my-0">хранилище:</span>

                <span class="caption grey--
text text--darken-2 font-weight-bold my-0">

                    {{

machine.storage.TITLE_STORAGE.length > 10 ?

machine.storage.TITLE_STORAGE.slice(0, 10) + '..' :

machine.storage.TITLE_STORAGE

                    }}

                </span>

            </p>

        </div>

    </div>

    <div class=""

        style="position: absolute, top:0,
transition: all .3s linear,"

```

```

        v-if="machine.plan &&
showButtonsMachine === i"
    >
        <v-btn
            small
            class="blue-grey--text ml-0 mt-2"
            round
            style=""
            @click="editPlan(machine)"
        >
            <v-icon small class="align-self-
center mb-1 mr-1">edit</v-icon>
                ИЗМЕНИТЬ ПЛАН
        </v-btn>
        <v-btn
            small
            class="blue-grey--text ml-0 mt-2"
            round
            style=""
            @click="stopPlan(machine)"
        >
            <v-icon small class="align-self-
center mb-1 mr-1">stop</v-icon>
                ОСТАНОВИТЬ ПЛАН
        </v-btn>
    </div>
    <div class="mt-2" v-if="!machine.plan">
        <v-btn
            small

```

```

        class="blue-grey--text ml-0 mt-3"
        round
        style=""
        @click="createPlan(machine)"
    >
        <v-icon small class="align-self-
center">add</v-icon>

        Добавить план

    </v-btn>

</div>

</v-flex>

<v-flex md2>

    <div>

        <p class="caption grey--text darken-3
my-0">Следующий бекап</p>

        <p class="font-weight-bold text--
darken-1 my-0">{{ formatDate(machine.nextDate) }}</p>

    </div>

    <div class="mt-2">

        <p style="font-size: 11px !important,"
class="caption text--lighten-1 grey--text my-0">

            Последний бекап</p>

        <p style="font-size: 11px !important,"
class="caption grey--text text--
lighten-1 font-weight-bold my-0">

            {{ formatDate(machine.lastBackup)
}}</p>

    </div>

</v-flex>

```

```

        <v-flex md1>

            <div class="right" title="Запустить бэкап
сейчас">

                <v-btn

@click="confirmForceBackup(machine)"

                    class="mx-0 mt-2"
                    left
                    elevation="2"
                    fab
                    flat
                    rounded
                >

                    <v-icon>

                        backup
                    </v-icon>

                </v-btn>

            </div>

        </v-flex>

        <v-flex md2>

            <div class="right">

                <v-chip class="mt-4" small

: class="\${statuses[machine.status-1].style} white--text my-2
caption`">

                    {{ statuses[machine.status -
1].title }}

                </v-chip>

            </div>

        </v-flex>

```

```

        </v-layout>

        <v-divider></v-divider>

    </v-card>

    <CreatePlanDialog
        v-if="selectedMachine"
        :machine="selectedMachine"
        @closeModal="closeModal"
    >

    </CreatePlanDialog>

</v-container>

</v-container>
</template>

<script>
import axios from "axios",
import CreatePlanDialog from "../components/CreatePlanDialog",

export default {
    components: {CreatePlanDialog},
    name: 'MachinesComponent',
    data() {
        return {
            selectedMachine: null,
            showButtonsMachine: null,
            statuses: [
                {
                    style: 'overdue',

```

```

        title: 'Выключена'
      },
      {
        style: 'complete',
        title: 'Работает'
      },
      {
        style: 'ongoing',
        title: 'Бэкап'
      }
    ],
    machines: [],
    sort: ''
  }
},
mounted() {
  this.$emit('loginCheck'),
  this.fetchMachinesList(),
  setInterval(() => {
    this.fetchMachinesList(),
  }, 5000)
},
methods: {
  fetchMachinesList() {
    axios.get(process.env.VUE_APP_HOST + '/machines_list',
{
    headers: {
      token:
JSON.parse(localStorage.getItem('token'))

```

```

    }

    }).then(r => {

        const vms = r.data,

        this.machines = [],

        for (let vm in vms) {

            const machine = vms[vm],

            this.machines.push({

                lastBackup: machine['lastBackup'],

                id: machine['ID'],

                plan: machine['plan'],

                uuid: machine["UUID"],

                title: machine["NAME"],

                storage: machine["storage"],

                status: machine["STATUS"],

                sort_priority_status: machine["STATUS"]
=== 2 ? 1 : machine["STATUS"] === 3 ? 2 : 3,

                mode: machine["MODEBACKUP"],

                user: machine["USER"],

                count_backups: machine["COUNT_BACKUPS"],

                id_mode_backup: machine["ID_MODE_BACKUP"],

                days_backup: machine.plan ?
machine.plan['DAYS_BACKUP'] : null,

                title_mode_backup:
machine["TITLE_MODE_BACKUP"],

                nextDate: machine.plan ?
this.getNextDate(machine.plan["DAYS_BACKUP"], machine.lastBackup)
: null

            })

        }

        this.sorted()
    }

```

```

        // console.log(this.machines),
    })
},
async confirmForceBackup(machine) {
    if (!machine.plan) {
        // предлагаю создать план

        const res = await this.$confirm('Для данной машины
нет активного плана, создать его сейчас?'),

        if (res) {
            this.createPlan(machine),
        }
    } else {
        // подтверждение

        const res = await this.$confirm(`Начать бекап
сейчас для машины ${machine.title}? <br>(хранилище для бекапа
будет выбрано автоматически как в плане)`),

        if (res) {
            axios.post(process.env.VUE_APP_HOST +
'/force_backup', {
                id_plan: machine.plan.ID_PLAN
            }, {
                headers: {
                    token:
JSON.parse(localStorage.getItem('token'))
                }
            })

            .then((r) => {
                if (r.data.status === 'ok') {
                    this.$notify({
                        group: 'foo',

```

```

        type: 'success',
        title: 'Успешно',
        text: 'В ближайшее время
запустится бекап'

    )),
    this.$emit('closeModal'),
} else if (r.data.errorMessage ===
'today already') {

    this.$notify({
        group: 'foo',
        type: 'error',
        title: 'Ошибка',
        text: 'Сегодня бэкап уже
выполнялся'

    )),
} else {

    this.$notify({
        group: 'foo',
        type: 'error',
        title: 'Ошибка',
        text: 'Неизвестная ошибка,
повторите еще раз'

    )),
}
})

.catch(e => {
    console.log(e),
    this.$notify({
        group: 'foo',

```

```

        type: 'error',
        title: 'Ошибка',
        text: JSON.stringify(e)
    })),
    })
}

},
createPlan(machine) {
    this.selectedMachine =
JSON.parse(JSON.stringify(machine)),
},
editPlan(machine) {
    const plan = machine.plan.ID_PLAN,
    this.$router.push({name: 'plans', params: {plan}})
},
async stopPlan(machine) {
    const res = await this.$confirm('Приостановить
выполнение данного плана?'),
    if (res) {
        axios.post(process.env.VUE_APP_HOST +
'/edit_active_plan', {
            plan_id: machine.plan.ID_PLAN,
            new_status: false
        }, {
            headers: {
                token:
JSON.parse(localStorage.getItem('token'))
            }
        }
    }
}

```

```

    })

    .then(r => {

        this.fetchMachinesList(),

    }),

}

},

ucFirst(str) {

    if (!str) return str,

    return str[0].toUpperCase() + str.slice(1),

},

formatDate(d) {

    if (!d) return '-',

    try {

        d = new Date(d),

        const now = new Date(),

        if (

            d.getFullYear() === now.getFullYear() &&

            d.getMonth() === now.getMonth() &&

            d.getDate() === now.getDate()

        ) {

            return 'Сегодня'

        }

    }

}

return this.ucFirst(d.toLocaleString('ru', {

    month: 'long',

    day: 'numeric',

    weekday: 'long',

    timezone: 'UTC',

```

```

        // hour: 'numeric',
        // minute: 'numeric'

    )))

    } catch (e) {

        return '-',

    }

},

getNextDate(count, lastBackup) {

    if (!lastBackup) return new Date(),

    const date = new Date(lastBackup),
    date.setDate(date.getDate() + count),
    return date,

},

sorted() {

    const by = this.sort,
    if (!by) return,
    if (by === 'active') {

        this.machines = this.machines.sort((a, b) => {

            if (a.sort_priority_status <
b.sort_priority_status) {

                return -1,

            }

            if (a.sort_priority_status >
b.sort_priority_status) {

                return 1,

            }

            return 0,

        }),

    },

```

```

    }

    if (by === 'backup') {
        this.machines = this.machines.sort((a, b) => {
            if (a.nextDate < b.nextDate) {
                return -1,
            }
            if (a.nextDate > b.nextDate) {
                return 1,
            }
            return 0,
        }),
    }

    },

    sortBy(by) {
        if (this.sort === by) {
            this.sort = null,
            this.fetchMachinesList(),
            return,
        }

        this.sort = by,

        if (by === 'active') {
            this.machines = this.machines.sort((a, b) => {
                if (a.sort_priority_status <
b.sort_priority_status) {
                    return -1,
                }

                if (a.sort_priority_status >
b.sort_priority_status) {
                    return 1,

```

```

        }

        return 0,

    }},

}

if (by === 'backup') {

    this.machines = this.machines.sort((a, b) => {

        if (a.nextDate < b.nextDate) {

            return -1,

        }

        if (a.nextDate > b.nextDate) {

            return 1,

        }

        return 0,

    }},

}

},

closeModal() {

    console.log('closeModal'),

    this.selectedMachine = null,

    this.selectedMachineForceBackup = null,

    this.fetchMachinesList(),

}

}

}

</script>

```

```
<style>

.container .v-chip .v-chip__content {
    cursor: pointer,
}

.project.complete {
    border-left: 4px solid #2ce766,
}

.project.ongoing {
    border-left: 4px solid #ffaa2c,
}

.project.overdue {
    border-left: 4px solid #f83e70,
}

.v-chip.complete {
    background: #2ce766,
}

.v-chip.ongoing {
    background: #ffaa2c,
}

.v-chip.overdue {
    background: #f83e70,
```

```

}
</style>
<template>
  <v-container class="storages">
    <h1 class="subheading grey--text">
      Хранилища
    </h1>
    <v-container class="my-4">
      <v-layout row justify-start class="mb-3">
        <v-tooltip top>
          <v-btn
            title="Обновить информацию"
            small flat color="grey"
            @click="updateStorages" slot="activator">
            <v-icon small left>update</v-icon>
            <span class="caption text-
lowercase">Обновить</span>
          </v-btn>
        </v-tooltip>
      </v-layout>
      <v-card flat v-for="(storage,i) in storages"
:key="storage.TITLE_STORAGE">
        <v-layout row wrap :class="'pa-3 project
${storage.ACTIVE ? 'complete' : 'ongoing'}'">
          <v-flex md4>
            <p class="text--secondary grey--text my-
0">
              <span class="text--accent-1 text--
lighten-1 grey--text my-0">Название: </span>

```

```

        <span class="text--accent-1 grey--text
text--darken-2 font-weight-bold my-0">

                {{ storage.TITLE_STORAGE }}

        </span>

</p>

<p class="text--secondary grey--text my-
3">

        <span class="text--accent-1 text--
lighten-1 grey--text my-0">uuid: </span>

        <span class="caption grey--text text--
darken-2 font-weight-bold my-0">

                {{ storage.UUID_STORAGE }}

        </span>

</p>

</v-flex>

<v-flex md2>

        <div>

                <p class="caption grey--text darken-3
my-0">Тип хранилища</p>

                <p class="font-weight-bold text--
darken-1 my-0 mt-2">

                        {{ storage.TYPE_STORAGE ?
'Удаленное' : 'Локальное' }}</p>

                </div>

        </v-flex>

<v-flex md3>

        <div>

                <p class=" my-0">

                        <span class="text--accent-1 text--
lighten-1 grey--text my-0">Состояние: </span>

```

```

        <span class="text--accent-1 grey--
text text--darken-2 font-weight-bold my-0">
        {{ storage.ACTIVE ? 'OK' : 'Не
заданы учетные данные' }}

        </span>
    </p>

    <v-progress-linear
        class="mt-3 mb-2"
        v-model="storage.percents"
        :active="true"
        :indeterminate="false"
        :color="storage.percents < 50 ?
'blue accent-4' : storage.percents < 75 ? 'orange accent-4' : 'red
accent-4'"
        :query="true"
    ></v-progress-linear>

    <p class="text--accent-1 text--
lighten-1 grey--text my-0">
        Свободно {{ storage.CAPACITY -
storage.UTILISATION }} ГБ из {{ storage.CAPACITY }} ГБ

    </p>
</div>
</v-flex>
<v-flex md2>
    <div class="right" title="Редактировать
данные">

        <v-btn
            @click="editStorage(storage)"
            class="mx-0 mt-2"
            left

```

```

        elevation="2"

        fab

        flat

        rounded

    >

    <v-icon>

        edit

    </v-icon>

</v-btn>

</div>

</v-flex>

</v-layout>

<v-divider></v-divider>

</v-card>

<StorageEditDialog

    v-if="editingStorage"

    :editingStorage="editingStorage"

    @closeModal="closeModal"

>

</StorageEditDialog>

</v-container>

</v-container>

</template>

<script>

import axios from "axios",

import StorageEditDialog from "../components/StorageEditDialog",

```

```

export default {
  name: "Storages",
  components: {
    StorageEditDialog
  },
  data() {
    return {
      storages: [],
      editingStorage: null
    }
  },
  methods: {
    updateStorages() {
      axios.get(process.env.VUE_APP_HOST +
        '/update_storages', {
        headers: {
          token:
JSON.parse(localStorage.getItem('token'))
        }
      }).then(r => {
        this.fetchStoragesList(),
      })
    },
    fetchStoragesList() {
      axios.get(process.env.VUE_APP_HOST + '/list_storages',
{
      headers: {
        token:
JSON.parse(localStorage.getItem('token'))

```

```

    }

    }).then(r => {

        const vms = r.data,

        this.storages = r.data,

        this.storages.map(storage => {

            storage.CAPACITY = storage.CAPACITY ?
Math.floor(storage.CAPACITY / (8 * 1024 * 1024 * 1024)) : 0,

            storage.UTILISATION = storage.UTILISATION ?
Math.floor(storage.UTILISATION / (8 * 1024 * 1024 * 1024)) : 0,

            storage.percents = storage.CAPACITY ?
Math.floor(100 * (storage.UTILISATION / storage.CAPACITY)) : 0,

            return storage,

        })

        console.log(this.storages),

    })

},

editStorage(storage) {

    this.editingStorage = storage,

},

closeModal() {

    this.editingStorage = null,

    this.fetchStoragesList(),

}

},

mounted() {

    this.fetchStoragesList(),

}

}

</script>

```

```
<style scoped>
```

```
</style>
```

```
<?php
```

```
namespace App\Jobs,
```

```
use App\Http\Controllers\BackProcessing,
```

```
use DateTime,
```

```
use Illuminate\Bus\Queueable,
```

```
use Illuminate\Queue\SerializesModels,
```

```
use Illuminate\Queue\InteractsWithQueue,
```

```
use Illuminate\Contracts\Queue\ShouldQueue,
```

```
use Illuminate\Foundation\Bus\Dispatchable,
```

```
use Illuminate\Support\Facades\App,
```

```
use Illuminate\Support\Facades\DB,
```

```
use App\Http\Controllers,
```

```
use Exception,
```

```
use Illuminate\Support\Facades\Log,
```

```
class ForceBackup implements ShouldQueue
```

```
{
```

```
    use Dispatchable, InteractsWithQueue, Queueable,  
    SerializesModels,
```

```
    public $timeout = 172800,
```

```
    protected $user,
```

```

protected $plan,

/**
 * Create a new job instance.
 *
 * @return void
 */
public function __construct($user, $plan)
{
    $this->user = $user,
    $this->plan = $plan,
}

/**
 * Execute the job.
 *
 * @return void
 * @throws Exception
 */

private function logError($text)
{
    info(json_encode($text)),
    info(json_encode($this->plan)),
    info(json_encode($this->user)),
}

/**

```

```

    * @throws Exception
    */
    public function handle()
    {

        $ssh = (new \App\Http\Controllers\BackProcessing)
            ->authentication(
                $this->user->HOST,
                $this->user->LOGIN,
                $this->user->PASSWORD),

        if (!$ssh) {
            $this->logError('error connect SSH'),
            return,
        }

        set_time_limit(0),

        $storage = DB::table('storage')
            ->where('ID_STORAGE', '=', $this->plan-
>PLAN_STORAGE_ID)
            ->get()
            ->first(),

        if (!$storage) {
            $this->logError('error fetch storage'),
            return,
        }
    }

```

```

        // если удаленное хранилище - коннектим его

        if ($storage->TYPE_STORAGE) {

            (new \App\Http\Controllers\BackProcessing)-
>unmountStorage($ssh),

            if ((new \App\Http\Controllers\BackProcessing)
                ->mountStorage(

                    $ssh,

                    $storage->IP_STORAGE,

                    $storage->LOGIN,

                    $storage->PASSWORD

                ) !== 'ok') {

                Log::info('error mount in forceBackup'),

                return,

            },

        }

        // устанавливаем статус машины

        (new \App\Http\Controllers\BackProcessing)-
>setStatusMachine($this->plan->PLAN_MACHINE_ID, 3),

        // бэкап

        $machine = DB::table('machine')

            ->where('ID', '=', $this->plan->PLAN_MACHINE_ID)

            ->get()

            ->first(),

        if (!$machine) {

            $this->logError('error fetch machine'),

            return,

        }

```

```

        $backupInfo = (new \App\Http\Controllers\BackProcessing)-
>backupExecuteCommands($ssh, $machine->UUID, $storage-
>UUID_STORAGE),

        if (!$backupInfo) {

            $this->logError('error backup'),

            return,

        }

        // попытка получить uuid, если он есть - сохранение бекапа

        $ssh = (new \App\Http\Controllers\BackProcessing)

            ->authentication(

                $this->user->HOST,

                $this->user->LOGIN,

                $this->user->PASSWORD),

        $backupUUID = (new \App\Http\Controllers\BackProcessing)-
>getBackupUUIDByName($ssh, $backupInfo),

        if (!$backupUUID) {

            $this->logError('error fetch backup uuid'),

            return,

        }

        $backupInfo['backupUUID'] = $backupUUID,

        (new \App\Http\Controllers\BackProcessing)-
>setStatusMachine($this->plan->PLAN_MACHINE_ID, 2),

        // сейв бекапа

        DB::table('backup')

            ->insert([

```

```

        'UUID_BACKUP' => $backupInfo['backupUUID'],
        'TIME' => new DateTime(),
        'NAME' => $backupInfo['backupName'],
        'PLAN' => $this->plan->ID_PLAN
    ),

    try {
        (new \App\Http\Controllers\StoragesController)-
>parseFreeSpaceStorages(),
    } catch (\Exception $e) {
        Log::info(json_encode($e)),
    }

    (new \App\Http\Controllers\BackProcessing)-
>removeMachine($ssh, $backupInfo['snapUUID']),

    if ($storage->TYPE_STORAGE) {
        (new \App\Http\Controllers\BackProcessing)-
>unmountStorage($ssh),
    }

}

}

<?php

namespace App\Http\Controllers,

use App\Jobs\ForceBackup,
use App\Jobs\CronTest,
use App\Jobs\RestoreBackup,

```

```

use App\Jobs\updateStatesMachines,
use DateTime,
use Exception,
use Illuminate\Http\Request,
use App\Http\Controllers\Controller,
use Illuminate\Support\Facades\Bus,
use Illuminate\Support\Facades\DB,
use APP\Jobs,
use Illuminate\Support\Facades\Log,
use phpseclib\Net\SSH2,
use Symfony\Component\Process\Process,

$loader = new \Composer\Autoload\ClassLoader(),
$loader->addPsr4('phpseclib\\', __DIR__ . '/phpseclib'),
$loader->register(),

class BackProcessing extends Controller
{

    /**
     * создание плана
     * @throws \Exception
     */
    public function createPlan(Request $request)
    {
        $data = $request->all(),
        $id_machine = $data['id_machine'],
        $id_storage = $data['id_storage'],

```

```

$days_backup = $data['days_backup'],
$count_backups = $data['count_backups'],

try {
    // деактивация текущего
    DB::table('plan')
        ->where('PLAN_MACHINE_ID', '=', $id_machine)
        ->update(['ACTIVE' => false]),

    DB::table('plan')
        ->insert([
            'PLAN_MACHINE_ID' => $id_machine,
            'PLAN_STORAGE_ID' => $id_storage,
            'DAYS_BACKUP' => $days_backup,
            'COUNT_BACKUPS' => $count_backups,
            'ACTIVE' => true
        ]),
    return json_encode([
        'status' => 'ok',
        'errorMessage' => ''
    ]),
} catch (Exception $e) {
    return json_encode([
        'status' => 'error',
        'errorMessage' => $e
    ]),
}

```

```

}

public function editPlan(Request $request)
{
    $data = $request->all(),
    $id_plan = $data['id_plan'],
    //    $id_storage = $data['id_storage'],
    $days_backup = $data['days_backup'],
    $count_backups = $data['count_backups'],

    try {
        DB::table('plan')
            ->where('ID_PLAN', '=', $id_plan)
            ->update([
                'DAYS_BACKUP' => $days_backup,
                'COUNT_BACKUPS' => $count_backups
            ]),
        return json_encode([
            'status' => 'ok',
            'errorMessage' => ''
        ]),
    } catch (Exception $e) {
        return json_encode([
            'status' => 'error',
            'errorMessage' => $e
        ]),
    }
}

```

```

}

/**
 * @throws Exception
 */
public function listPlans(Request $request)
{
    $data = $request->all(),
    $id_user = $data['user'],

    $machines = json_decode((new
\App\Http\Controllers\MachinesController()-
>machinesList($request)),

    $activePlans = [],
    $turnedOffPlans = [],
    foreach ($machines as $machine) {
        $plans = DB::table('plan')
            ->where('PLAN_MACHINE_ID', '=', $machine->ID)
            ->get(),

        foreach ($plans as $plan) {
            $plan->machine = $machine,
            $plan->backups_list = DB::table('backup')
                ->where('PLAN', '=', $plan->ID_PLAN)
                ->get(),

            $plan->storage = DB::table('storage')
                ->where('ID_STORAGE', '=', $plan-
>PLAN_STORAGE_ID)
                ->get()

```

```

        ->first(),

        if ($plan->ACTIVE) {
            $activePlans[] = $plan,
        } else {
            $turnedOffPlans[] = $plan,
        }
    }
}

    return json_encode(['activePlans' => $activePlans,
'turnedOffPlans' => $turnedOffPlans]),

}

/**
 * ssh - аутентификация
 * @throws \Exception
 */
public function authentication($host, $login, $password)
{
    $ssh = new SSH2($host),
    try {
        $ssh->login($login, $password),
        return $ssh,
    } catch (Exception $e) {
        LOG::error($host . ' ' . $login . ' Login failed'),
        return null,
    }
}

```

```

    }
}

/** Принудительный бекап машины
 * @throws Exception
 */
public function forceBackup(Request $request)
{
    $data = $request->all(),
    $id_user = $data['user'],
    $id_plan = $data['id_plan'],

    $plan = DB::table('plan')
        ->where('ID_PLAN', '=', $id_plan)
        ->get()
        ->first(),

    $user = DB::table('user')
        ->where('ID', '=', $id_user)
        ->get()
        ->first(),

    $lastDateBackup = (new
    \App\Http\Controllers\MachinesController)-
    >getLastDateBackup($plan->PLAN_MACHINE_ID),

    $permission = false,

    // Чек, что сегодня бекап уже был
    if (!$lastDateBackup) {

```

```

        $permission = true,
    } else if ((new DateTime())->format('d') !== // now day
        (new DateTime($lastDateBackup))->format('d')) { // day
last backup
        $permission = true,
    }

    if ($permission) {
        ForceBackup::dispatch($user, $plan),
        updateStatesMachines::dispatch(),
        return json_encode([
            'status' => 'ok'
        ]),
    } else {
        return json_encode([
            'status' => 'error',
            'errorMessage' => 'today already'
        ]),
    }

}

```

```

/** ОСНОВНАЯ функция бекапа
 * @throws Exception
 */
public function MainBackup()
{

```

```

// сначала обновление текущих статусов машин
try {
    $this->parseCurrentMachinesStates(),
} catch (Exception $e) {
    Log::info(json_encode($e)),
}

try {
    (new \App\Http\Controllers\StoragesController())->parseStorages(),
} catch (Exception $e) {
    Log::info(json_encode($e)),
}

// массивы для теста чисто
$all_machines = [],
$users = DB::table('user')->get(),
foreach ($users as $user) {
    try {
        $this->ProcessingClusterPlans($user),
    } catch (Exception $e) {
        Log::info(json_encode($e)),
    }
}

updateStatesMachines::dispatch(),
return json_encode(['status' => 'ok']),
}

/**
 * @throws Exception

```

```

*/

public function removeOldBackups($user, $plan)
{
    $backups = DB::table('backup')
        ->where('PLAN', '=', $plan->ID_PLAN)
        ->orderBy('TIME', 'desc')
        ->get()
        ->toArray(),

    $count = $plan->COUNT_BACKUPS - 1,
    $backups_to_delete = array_slice($backups, $count),
    if (count($backups_to_delete)) {
        $ssh = $this->authentication(
            $user->HOST,
            $user->LOGIN,
            $user->PASSWORD),
        foreach ($backups_to_delete as $backup) {
            $result = $this->removeMachine($ssh, $backup->
>UUID_BACKUP),
            if ($result) {
                DB::table('backup')->where(['ID' => $backup->
>ID])->delete(),
            }
        }
    }
    try {
        (new \App\Http\Controllers\StoragesController())->
>parseFreeSpaceStorages(),

```

```

    } catch (Exception $e) {

        Log::info(json_encode($e)),

    }

}

/**
 * @throws Exception
 */

private function ProcessingClusterPlans($user)
{
    $id_user = $user->ID,

    $user_machines = DB::table('machine')
        ->where('USER', '=', $id_user)
        ->get(),

    $ssh = $this->authentication($user->HOST, $user->LOGIN,
    $user->PASSWORD),

    if (!$ssh) return,

    // проверки и загрузка в очередь на бекап
    foreach ($user_machines as $machine) {

        $plan = DB::table('plan')
            ->where('PLAN_MACHINE_ID', '=', $machine->ID)
            ->where('ACTIVE', '=', 1)
            ->get()
            ->first(),

        if (!$plan) continue,

        // разрешил несколько планов на одну машину, но не
        должно быть несколько бекапов в 1 день
    }
}

```

```

        // чек тут - $diffDays != 0

        $lastDateBackup = (new
\App\Http\Controllers\MachinesController)-
>getLastDateBackup($plan->PLAN_MACHINE_ID),

        $diffDays = $lastDateBackup ?

            (new DateTime())->format('d') - (new
DateTime($lastDateBackup))->format('d') : 1,

        // Чек, что сегодня бекап уже был

        Log::info($machine->NAME),

        Log::info($diffDays),

        if (/*$this->checkPlanBackupTimeCome($plan) &&*/
$diffDays != 0) {

            $this->removeOldBackups($user, $plan),

            ForceBackup::dispatch($user, $plan),

        }

    }

//         updateStatesMachines::dispatch(),

    }

/**
 * @throws Exception
 */

//     public function restoreBackup(Request $request)
//     {
//         $data = $request->all(),
//         $id_user = $data['id_user'],
//         $id_backup = $data['id_backup'],

```

```

//      DB::table('backup')
//      ->where('ID', '=', $id_backup)
//      ->update([
//          'RESTORING' => 1
//      ]),
//      RestoreBackup::dispatch($id_user, $id_backup),
//      return json_encode([
//          'status' => 'ok',
//          'errorMessage' => ''
//      ]),
//  }

```

```

// ГОТОВО

public function parseCurrentMachinesStates()
{
    $users = DB::table('user')->get(),
    // в каждом кластере чекаем машины
    foreach ($users as $user) {
        $id_user = $user->ID,
        $host = $user->HOST,
        $login = $user->LOGIN,
        $password = $user->PASSWORD,
        $ssh = $this->authentication($host, $login,
$password),
        if (!$ssh) continue,

        $machineList = $this->VMsInformation($ssh),
    }
}

```

```

// чек каждой машины в бд, если нет, запись
foreach ($machineList as $machine) {

    // бэкапы сюда не пишутся

    if (strpos($machine['name'], 'vm-backup-on-
storage')) continue,

    $existing_machine = DB::table('machine')
        ->where('UUID', '=', $machine['uuid'])
        ->where('USER', '=', $id_user)
        ->get()
        ->first(),

    // обновление статуса работает/не работает
    if ($existing_machine) {
//        if ($existing_machine->STATUS !== 3) {
            DB::table('machine')
                ->where('UUID', '=', $machine['uuid'])
                ->where('USER', '=', $id_user)
                ->update([
                    'STATUS' => $machine['status']
                ]),
//        }
        continue,
    }

    // если новая машина - добавляем
    DB::table('machine')

```

```

        ->insert([
            'UUID' => $machine['uuid'],
            'NAME' => $machine['name'],
            'STATUS' => $machine['status'],
            'USER' => $id_user
        ]),
    }
}

/**
 * @throws Exception
 */
private function checkPlanBackupTimeCome($plan)
{
    // если нет бекапов - true, если есть и пришло время -
    количество дней, иначе - 0

    $lastBackup = DB::table('backup')
        ->where('PLAN', '=', $plan->ID_PLAN)
        ->orderBy('TIME', 'desc')
        ->first(),

    if (!$lastBackup) return true,

    if (date_diff(new DateTime(), new DateTime($lastBackup->TIME))->days + 1 >= $plan->DAYS_BACKUP)
        return true,
    return false,
}

```

```

}

// FOTOB0
public function VMsInformation($ssh): array
{
    $cmd = 'xe vm-list',
    $status = $ssh->exec($cmd),
    parse_str($status, $arr),
    $status = explode("uuid", $status),
    for ($i = 1, $i < count($status), $i++) {
        $status[$i] = "uuid " . $status[$i],
        $status[$i] = explode("name", $status[$i]),
        $status[$i][1] = "name" . $status[$i][1],
        $temp = explode("power-state", $status[$i][1]),
        $status[$i][1] = $temp[0],
        $status[$i][2] = "power-state" . $temp[1],
        $obj[$i] = [],
        $obj[$i]['uuid'] = str_replace(array('uuid', ':', '( RO)', "\n", " "), "", $status[$i][0]),
        $obj[$i]['name'] = str_replace(array('name-label', ':', '( RW)', "\n", " "), "", $status[$i][1]),
        if (str_replace(array('power-state', ':', '( RO)', "\n", " "), "", $status[$i][2]) == "running") {
            $obj[$i]['status'] = 2,
        } else {
            $obj[$i]['status'] = 1,
        }
    }
    return $obj,

```

```

}

/**
 * @throws Exception
 */

function backupExecuteCommands($ssh, $machineUUID,
$storageUUID)
{
    // снимок
    $cmdSnap = str_replace("\r", "", '
        DATE=`date +%d%m%Y`,
        VMNAME=`xe vm-list uuid="' . $machineUUID . '" | grep
name-label | cut -d":" -f2 | sed "s/^ *///g"` ,
        SNAPUUID=`xe vm-snapshot uuid="' . $machineUUID . '"
new-name-label="$VMNAME--vm-backup-$DATE"` ,
        xe template-param-set is-a-template=false ha-always-
run=false uuid=${SNAPUUID},
        echo $SNAPUUID'),
    $snapUUID = trim($ssh->exec($cmdSnap)),
    Log::info('snap uuid:'),
    Log::info($snapUUID),
    if (!$snapUUID) {
        return null,
    }

    // todo добавить проверку на ошибки в выводах команд
    // получение имени
    $cmdNameBackup = str_replace("\r", "", '
        DATE=`date +%d%m%Y`,
        VMNAME=`xe vm-list uuid="' . $machineUUID . '" | grep
name-label | cut -d":" -f2 | sed "s/^ *///g"` ,

```

```

        echo "$VMNAME--vm-backup-on-storage-$DATE"),
$backupName = trim($ssh->exec($cmdNameBackup)),
Log::info('backup name:'),
Log::info($backupName),
if (!$backupName) {
    return null,
}
// копирование
$cmdBackup = str_replace("\r", "", '
        BACKUPUUID=`xe vm-copy sr-uuid="' . $storageUUID . '"
new-name-label="' . $backupName . '" uuid="' . $snapUUID . '"`,
        echo $BACKUPUUID,')',
Log::info('command:'),
Log::info($cmdBackup),
$backupUUID = trim($ssh->exec($cmdBackup)),
Log::info('backup uuid:'),
Log::info($backupUUID),

return [
    'backupName' => $backupName,
    'snapUUID' => $snapUUID
],
}

function getBackupUUIDByName($ssh, $backupInfo)
{
    $cmdSnap = str_replace("\r", "",

```

```

        'xe vm-list name-label="' . $backupInfo['backupName']
. '"'),

    $out = trim($ssh->exec($cmdSnap)),

    if (strpos($out, $backupInfo['backupName'])) {

        return trim(explode(':', explode('name-label',
$out)[0])[1]),

    }

    return null,

}

// ГОТОВО

function mountStorage($ssh, $path, $login, $pass): string
{

    Log::info('mount ' . $path . '..'),

    $cmd = "mkdir -p /mnt/backup,mount -t cifs '$path' -o
username='$login',password='$pass' /mnt/backup",

    $out = trim($ssh->exec($cmd)),

    Log::info(strlen($out) ? $out : 'ok'),

    return strlen($out) ? $out : 'ok',

}

// OK

function unmountStorage($ssh): string
{

    Log::info('unmounting'),

    $cmd = 'umount "/mnt/backup"',

    $out = trim($ssh->exec($cmd)),

    Log::info(strlen($out) ? $out : 'ok'),

    return strlen($out) ? $out : 'ok',

}

```

```

}

// ok

function removeMachine($ssh, $uuid): ?string
{
    Log::info('removing machine ' . trim($uuid) . '..'),
    $cmd = 'xe vm-uninstall uuid="' . $uuid . '" force=true',
    Log::info($cmd),
    $cmd = str_replace("\r", "", $cmd),
    $res = trim($ssh->exec($cmd)),
    Log::info($res),
    if (strpos($res, 'All objects destroyed') || strpos($res,
'No matching VMs found')) {
        return true,
    }
    return null,
}

public function setStatusMachine($machineID, $status)
{
    DB::table('machine')
        ->where('ID', '=', $machineID)
        ->update(['STATUS' => $status]),
}

public function deletePlan(Request $request)
{
    try {

```

```

$data = $request->all(),
$id_user = $data['user'],
$id_plan = $data['id_plan'],

$plan_backups = DB::table('backup')
    ->where('PLAN', '=', $id_plan)
    ->get(),

$user = DB::table('user')
    ->where('ID', '=', $id_user)
    ->get()
    ->first(),

$ssh = $this->authentication(
    $user->HOST,
    $user->LOGIN,
    $user->PASSWORD),

foreach ($plan_backups as $backup) {
    try {
        $result = $this->removeMachine($ssh, $backup-
>UUID_BACKUP),

        if ($result) {
            DB::table('backup')->where(['ID' =>
$backup->ID])->delete(),
        }
    } catch (Exception $e) {
        Log::info(json_encode($e)),
    }
}

```

```

        DB::table('plan')->where(['ID_PLAN' => $id_plan])->delete(),

        return json_encode(['status' => 'ok']),

    } catch (Exception $e) {

        return json_encode(['status' => 'error',
        'errorMessage' => json_encode($e)]),

    }

}

public function deleteBackup(Request $request)
{
    $data = $request->all(),

    $id_user = $data['user'],
    $uuid_for_delete = $data['uuid_for_delete'],
    $id_backup = $data['id_backup'],

    $user = DB::table('user')
        ->where('ID', '=', $id_user)
        ->get()
        ->first(),

    $ssh = $this->authentication(
        $user->HOST,
        $user->LOGIN,
        $user->PASSWORD),

    $result = $this->removeMachine($ssh, $uuid_for_delete),
    if ($result) {

```

```

        try {
            (new \App\Http\Controllers\StoragesController())->parseFreeSpaceStorages(),
        } catch (Exception $e) {
            Log::info(json_encode($e)),
        }
        DB::table('backup')->where(['ID' => $id_backup])->delete(),
        return json_encode(['status' => 'ok']),
    }
    return json_encode(['status' => 'error']),
}

public function editActivePlan(Request $request)
{
    $data = $request->all(),
    try {
        $plan_id = $data['plan_id'],
        $new_status = $data['new_status'],

        DB::table('plan')
            ->where('ID_PLAN', '=', $plan_id)
            ->update(['ACTIVE' => $new_status]),

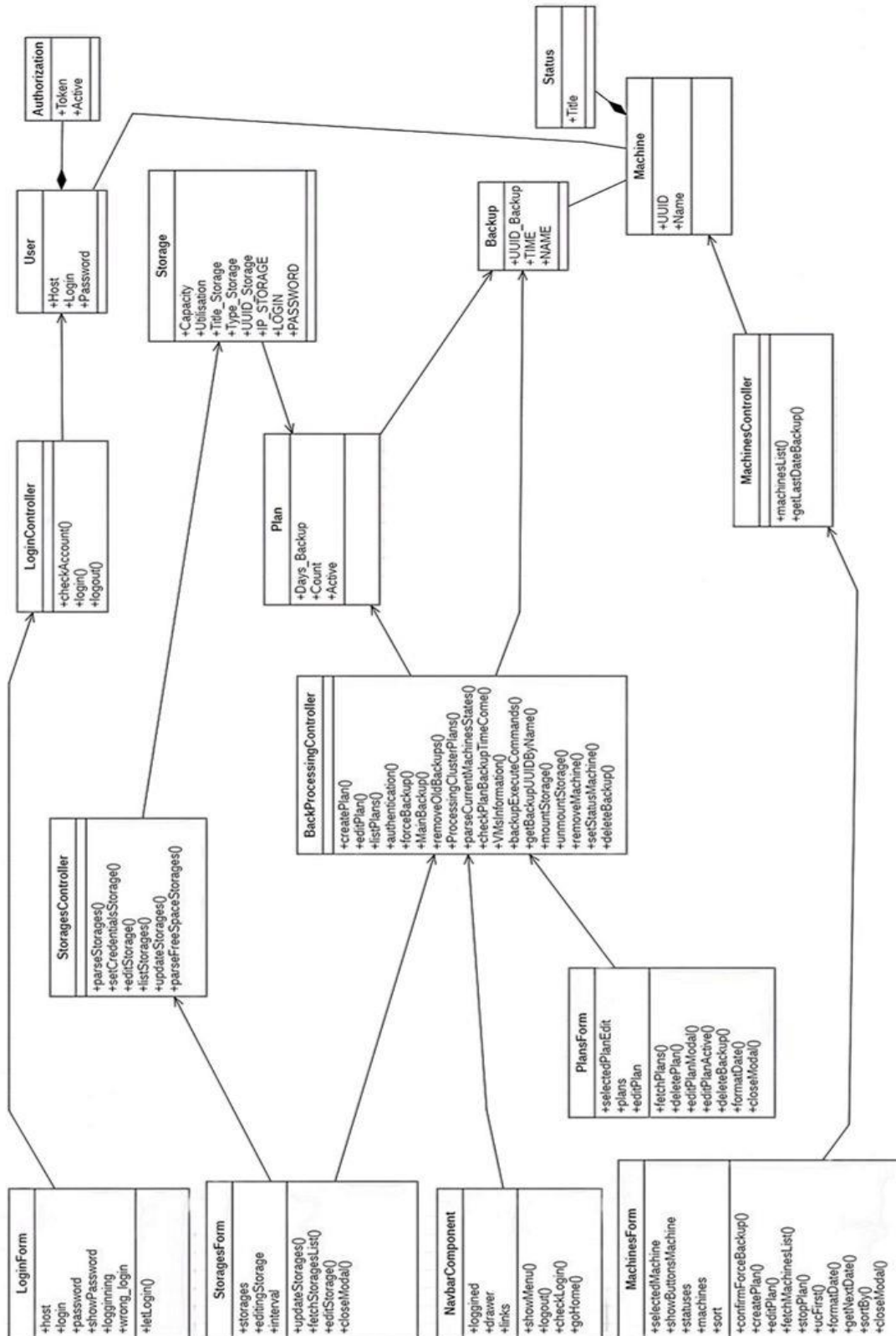
        return json_encode([
            'status' => 'ok'
        ]),
    } catch (Exception $e) {
        return json_encode($e),
    }
}

```

}
}
}

ПРИЛОЖЕНИЕ В

Диаграмма классов



ПРИЛОЖЕНИЕ Г

Схема данных

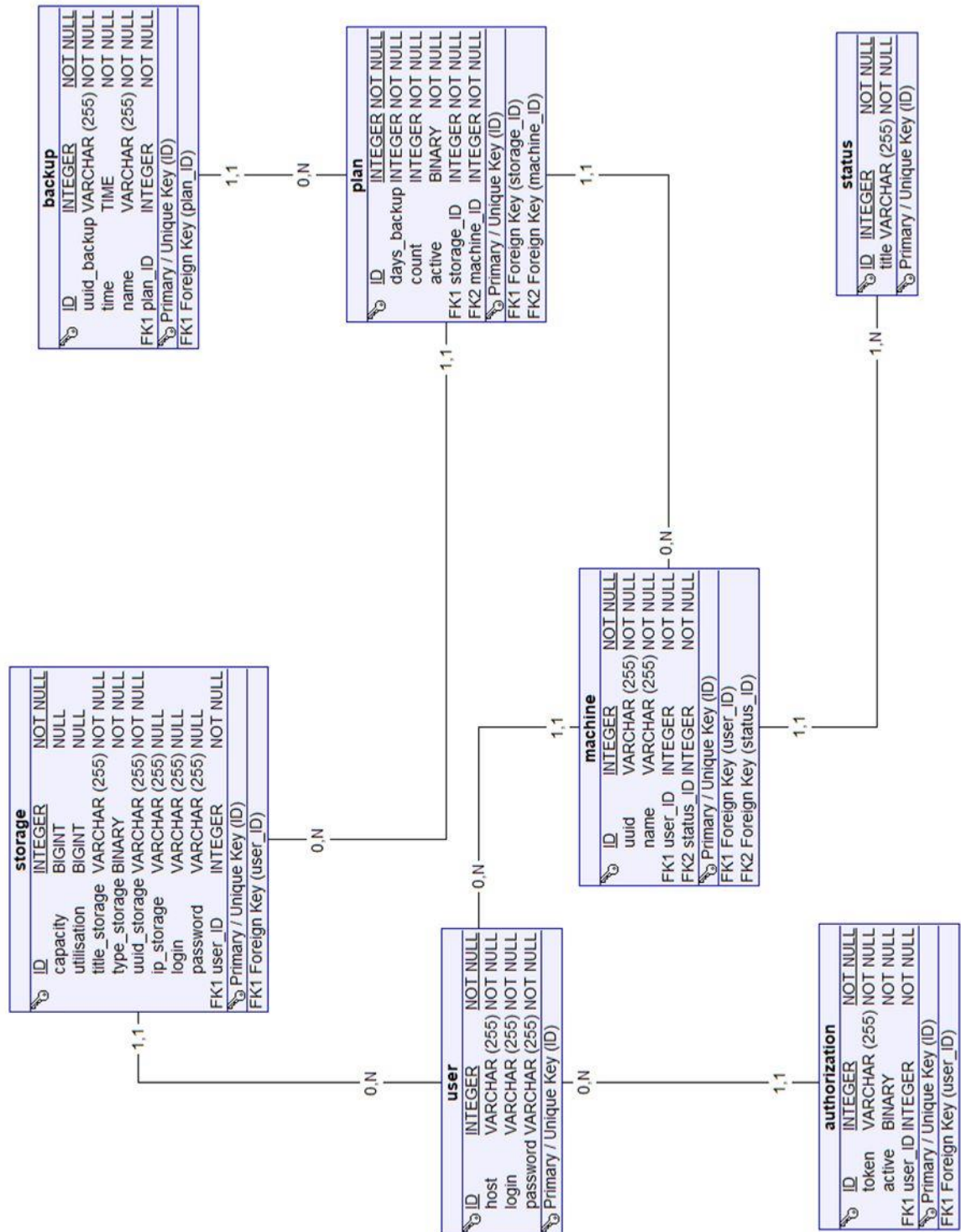
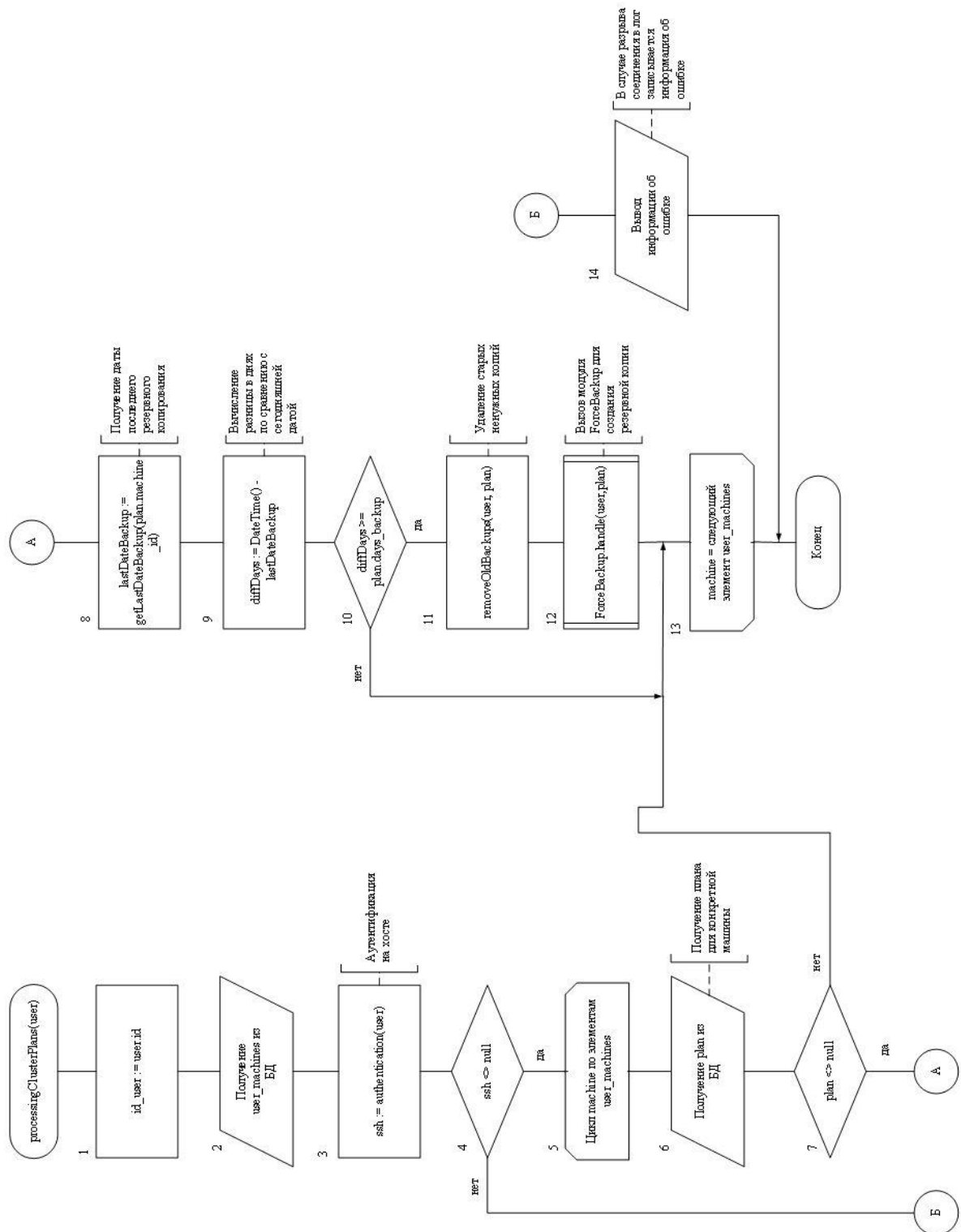


Схема функции handle



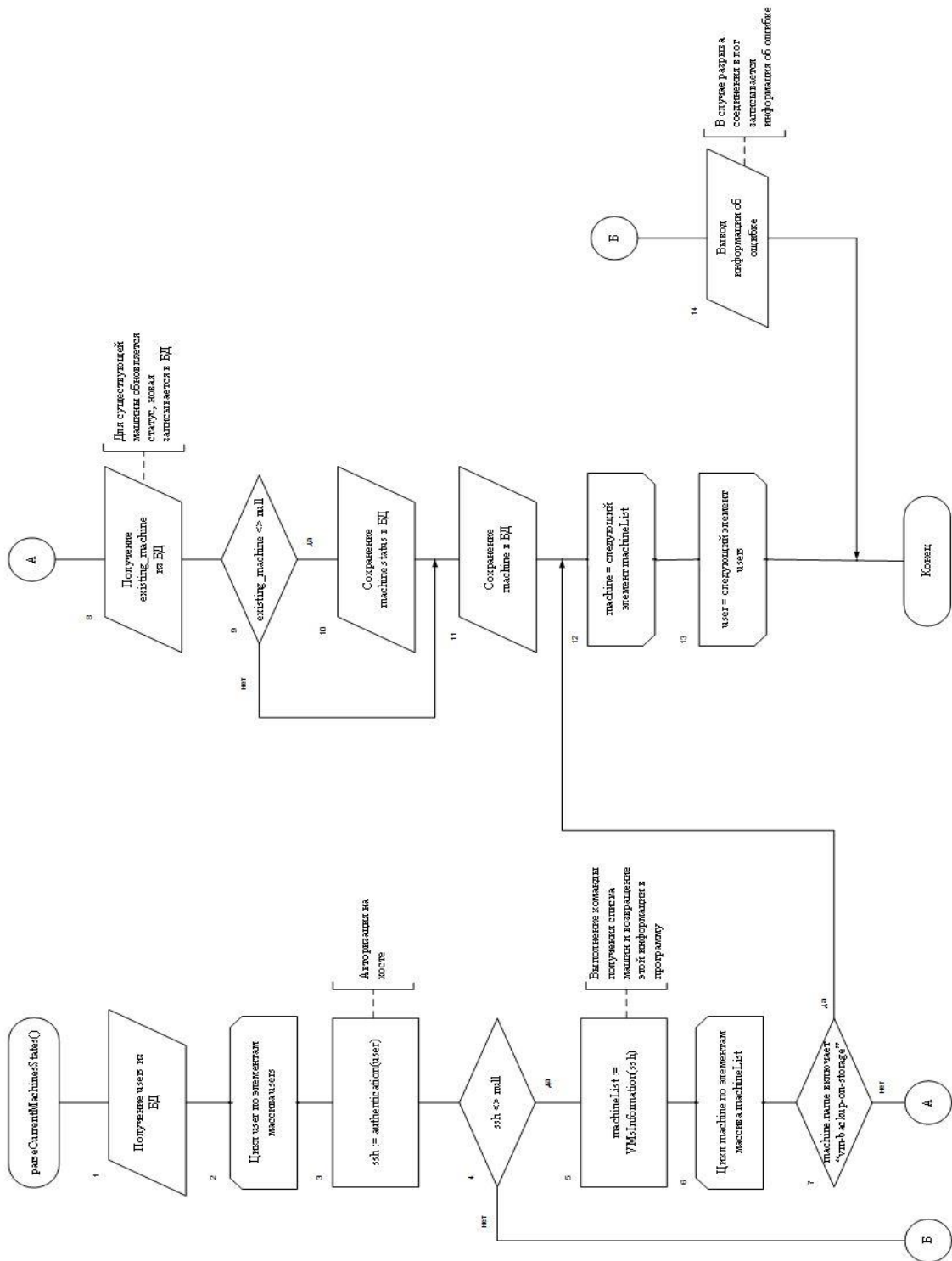
ПРИЛОЖЕНИЕ Е

Схема функции processingCluserPlans



ПРИЛОЖЕНИЕ Ж

Схема функции parseCurrentMachinesStates



ПРИЛОЖЕНИЕ 3

Презентация