МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ДЛЯ СОЗДАНИЯ И УЧЕТА РЕЗЕРВНЫХ КОПИЙ В КЛАСТЕРЕ ВИРТУАЛЬНЫХ МАШИН

Студент 14.06.22 (подпись, дата) Рассказов Никита Сергеевич (ФИО полностью)

Руководитель 14.06.2022 (подпись, дата) Попов К.В. (фамилия, инициалы)

Нормоконтролёр 14.06.2022 (подпись, дата) Бычков А.С. (фамилия, инициалы)

Рецензент нач. сектора АО НПП Рубин (должность, место работы) Васильев 15.06.2022 (подпись, дата) Васильков А.В. (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. № 13)

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

Работа защищена с отметкой отлично (протокол заседания ГЭК от 17 июня 2022 № 2)

Секретарь ГЭК _____ Е.И. Гурин

Пенза 2022

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» апреля 2022 г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Рассказов Никита Сергеевич гр. 17ВО2 факультета вычислительной техники.

2.Тема работы «Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин»

Тема утверждена приказом ПГУ №0357/о от "26" апреля 2022 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.)
Автоматизированная система предназначена для создания и учета резервных копий виртуальных машин в кластере в автоматическом и ручном режимах. Система должна иметь клиент-серверную архитектуру, хранить данные в базе данных и предназначена для использования в кластере с системой виртуализации *Citrix XenServer*.

5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметной области, выбрать инструменты для разработки клиентской и серверной части автоматизированной системы, выполнить инфологическое проектирование системы, разработать базу данных, разработать клиентскую и серверную части системы. Система должна позволять настраивать параметры резервного копирования виртуальных машин, иметь авторизацию, в автоматическом и ручном режимах осуществлять резервное копирования виртуальных машин. Информация о виртуальных машинах, хранилищах и резервных копиях должна храниться в базе данных. Также выполнить задание по БЖД и осуществить экономическое обоснование разработки системы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	8%	26.04.2022 – 29.04.2022	
2.	Выбор инструментов разработки	6%	30.04.2022 – 02.05.2022	
3.	Инфологическое проектирование	12%	03.05.2022 – 08.05.2022	
4.	Проектирование базы данных	8%	09.05.2022 – 12.05.2022	
5.	Разработка автоматизированной системы	40%	13.05.2022 – 01.06.2022	
6.	Выполнение задания по БЖД	10%	02.06.2022 – 06.06.2022	
7.	Выполнение экономического обоснования	8%	07.06.2022 – 10.06.2022	
8.	Оформление пояснительной записки	8%	11.06.2022 – 14.06.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель работы _____ 26.04.2022
(подпись, дата)

Задание к исполнению принял студент _____ 26.04.2022
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 06.06.2022
(подпись, дата)

Реферат

Пояснительная записка содержит 188 листов, 35 рисунков, 26 таблиц, 10 источников, 4 приложения.

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

Цель дипломного проектирования - 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.

Графическая часть дипломного проекта содержит следующие чертежи:

- диаграммы *UML* – 2 листа формата А1;
- схемы программ – 3 листа формата А1.

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 *MySQL*.

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

Содержание

Перечень обозначений и сокращений 5
Введение 6
1 Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин 8
1.1 Анализ предметной области 8
1.1.1 Анализ существующих решений в России 9
1.1.2 Постановка задачи 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 20
1.2.6 Разработка диаграммы развертывания 21
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 34
1.4 Разработка приложения. 39
1.4.1 Анализ функций приложения 39
1.4.2 Разработка логической структуры приложения 39
1.4.3 Описание модулей клиентской части приложения 41
1.4.4 Описание модулей серверной части приложения 46
1.5 Описание применения 49
1.6 Тестирование приложения 57
2 Безопасность жизнедеятельности 59
2.1 Введение 59

2.2 Неблагоприятные факторы воздействия 59
2.3 Противодействие вредным факторам 60
2.3.1 Санитарно-гигиеническая характеристика рабочего места. 60
2.3.2 Организация труда и отдыха 62
2.3.3 Требования по пожарной безопасности 63
2.4 Расчет освещенности рабочего места 65
3 Экономическое основание разработки 68
3.1 Определение трудоемкости разработки АС. 68
3.2 Расчет затрат на разработку автоматизированной системы. 69
3.3 Расчет минимальной цены разработки АС 73
3.4 Расчет затрат на внедрение АС 73
3.5 Расчет экономической эффективности 75
Заключение 77
Список использованных источников 78
Приложение А – Листинг программы 79
Приложение Б – Диаграммы *UML* 164
Приложение В – Схемы программ 167
Приложение Г – Презентация 171

Перечень обозначений и сокращений

FK - (англ. *Foreign Key*) - внешний ключ

PK - (англ. *Primary Key*) - первичный ключ

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

Бекап – резервная копия виртуальной машины

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

КТС – комплекс технических средств

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

СУБД - система управления базами данных

Снепшот – снимок виртуальной машины

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, так же, 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. 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.

Основные задачи, которые должны быть выполнены в ходе данного дипломного проектирования:

- анализ предметной области;
- выбор инструментов разработки;
- проведение инфологического проектирования системы;
- разработка базы данных;
- разработка клиентской и серверной части автоматизированной системы;
- выполнение задания по БЖД;
- осуществление экономического расчета разработки системы.

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 снепшот - снимок виртуальной машины, который позволяет вернуть машину в рабочее состояние в случае сбоя. Процесс копирования с использование снимка виртуальной машины представлен на рисунке 1.1, состоит из трех этапов: создание снепшота, копирование виртуальной машины и удаление (слияние) снимка с машиной.

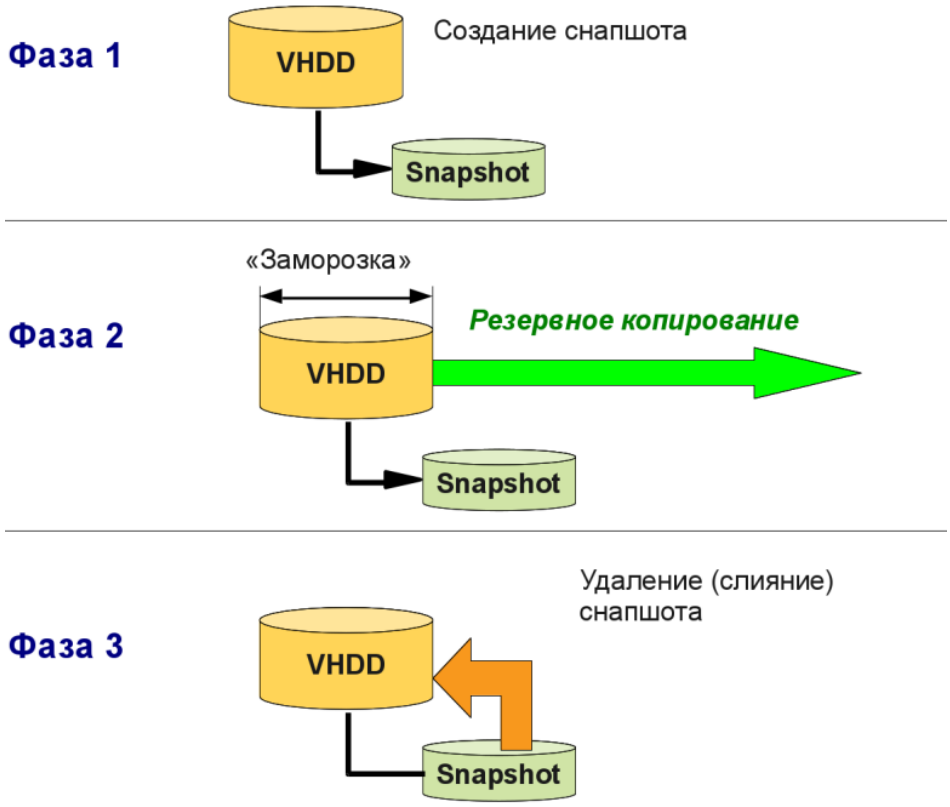


Рисунок 1.1 – Создание копии с использованием снепшота

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 Sever* 11.7;

- *Veeam Backup & Replication v*9.5.

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.

Также на рынке существуют другие системы резервного копирования, поддерживающие *Citrix*: *Veritas Backup Exec* 21, *Commvault* 11 и *NetBackup* 9. Данное ПО позволяют осуществлять резервное копирование, восстановление и хранение копий физических и виртуальных систем, в том числе по *CIFS* и *NFS*, однако копирование возможно только на уровне гостевой машины.

По результатам опроса *iKS-Consulting*, по состоянию на 2018 год 12.7 % российских компаний используют платформу виртуализации рабочих столов *Citrix* [3], также среди зарубежных компаний, использующих или рассматривающих в ближайшее время внедрение виртуализации рабочих столов 20% используют и 11% планируют внедрение *Citrix* [4].

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 копия;

- 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*. Развитая 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 десктоп 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 *MySQL* 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 компании *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 1.1, 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.

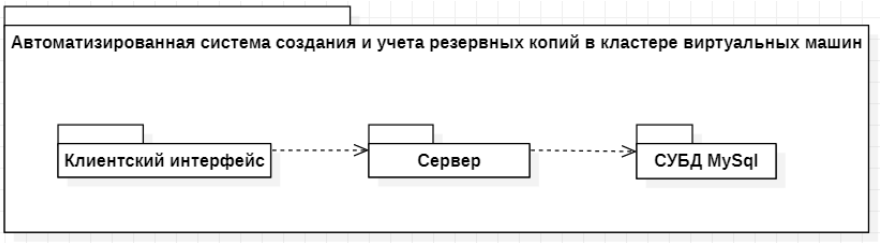


Рисунок 1.2 – Диаграмма пакетов автоматизированной системы

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 *MySql*» предназначен для хранения всех необходимых данных для АС.

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.

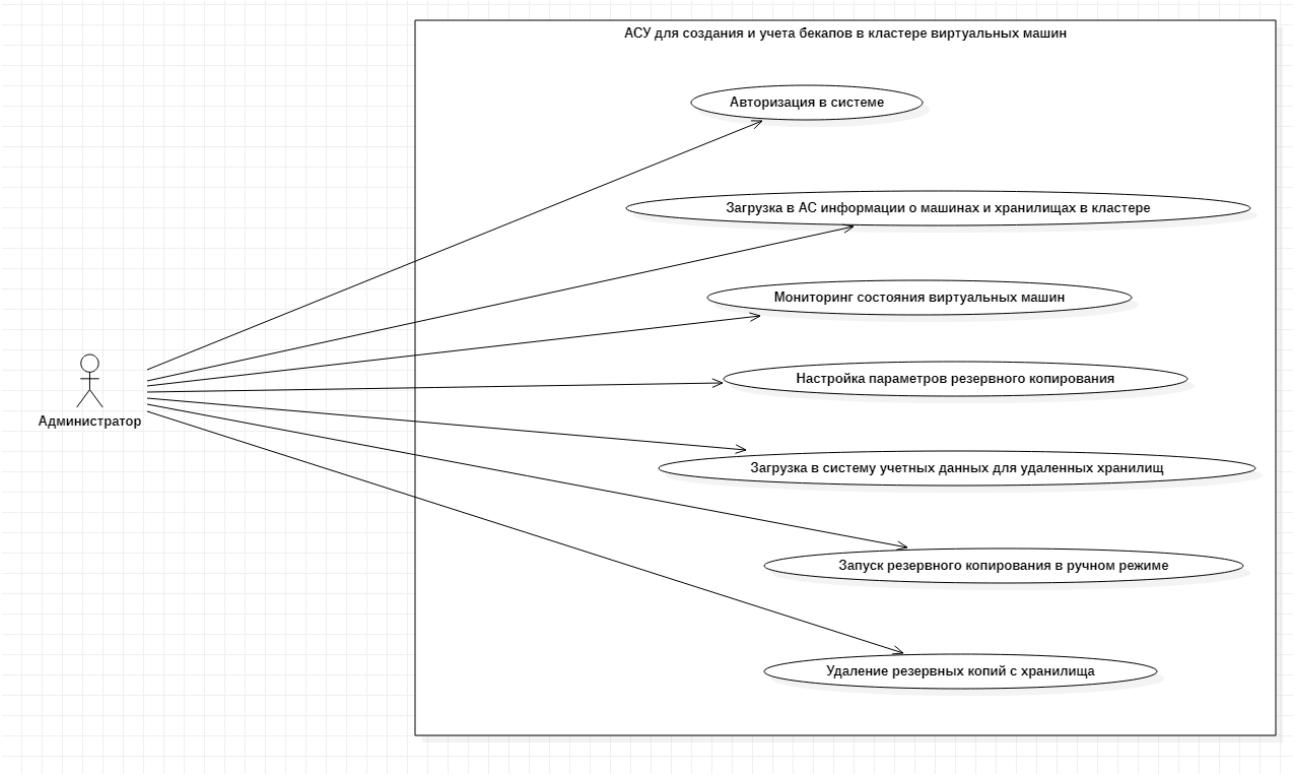


Рисунок 1.3 – Диаграмма вариантов использования

1.2.3 Разработка диаграммы последовательности АС

Диаграммы последовательности показывают коммуникацию между объектами системы. Они иллюстрируют, как различные части системы взаимо-

действуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного варианта использования.

Диаграмма последовательности для разрабатываемой автоматизированной системы представлена в приложении Б.

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 1.4-1.9.

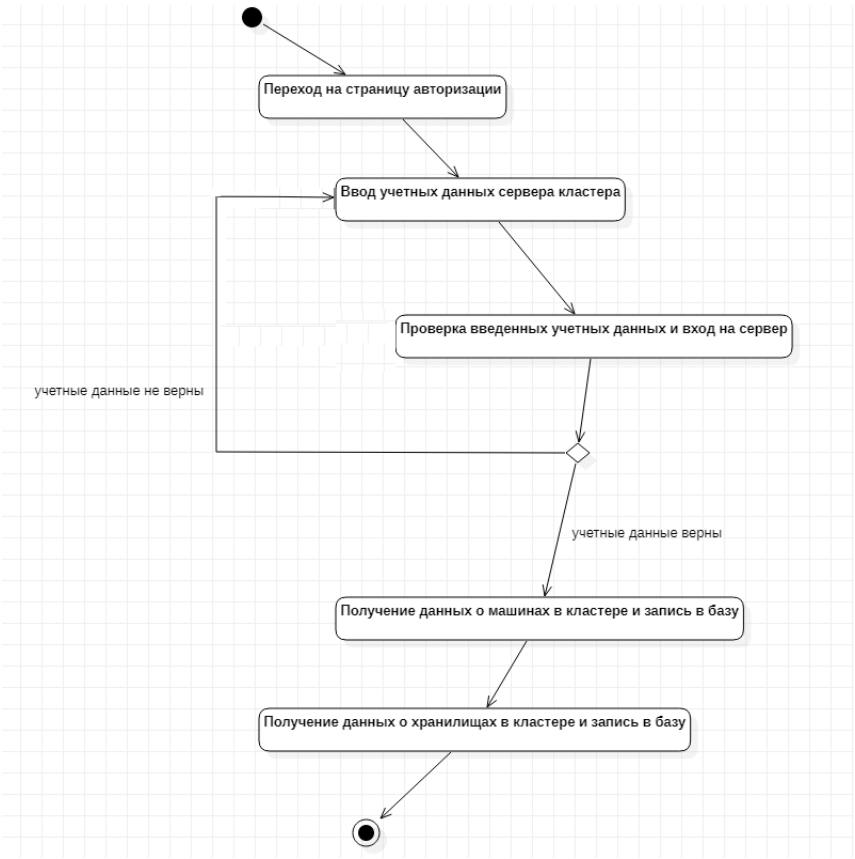


Рисунок 1.4 - Диаграмма деятельности «Загрузка в АС информации о машинах и хранилищах в кластере»

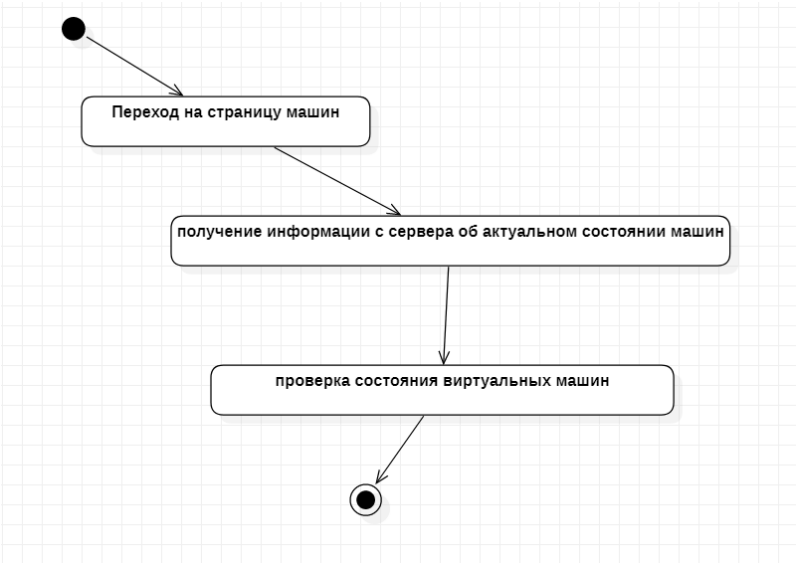


Рисунок 1.5 - Диаграмма деятельности «Мониторинг состояния виртуальных машин»

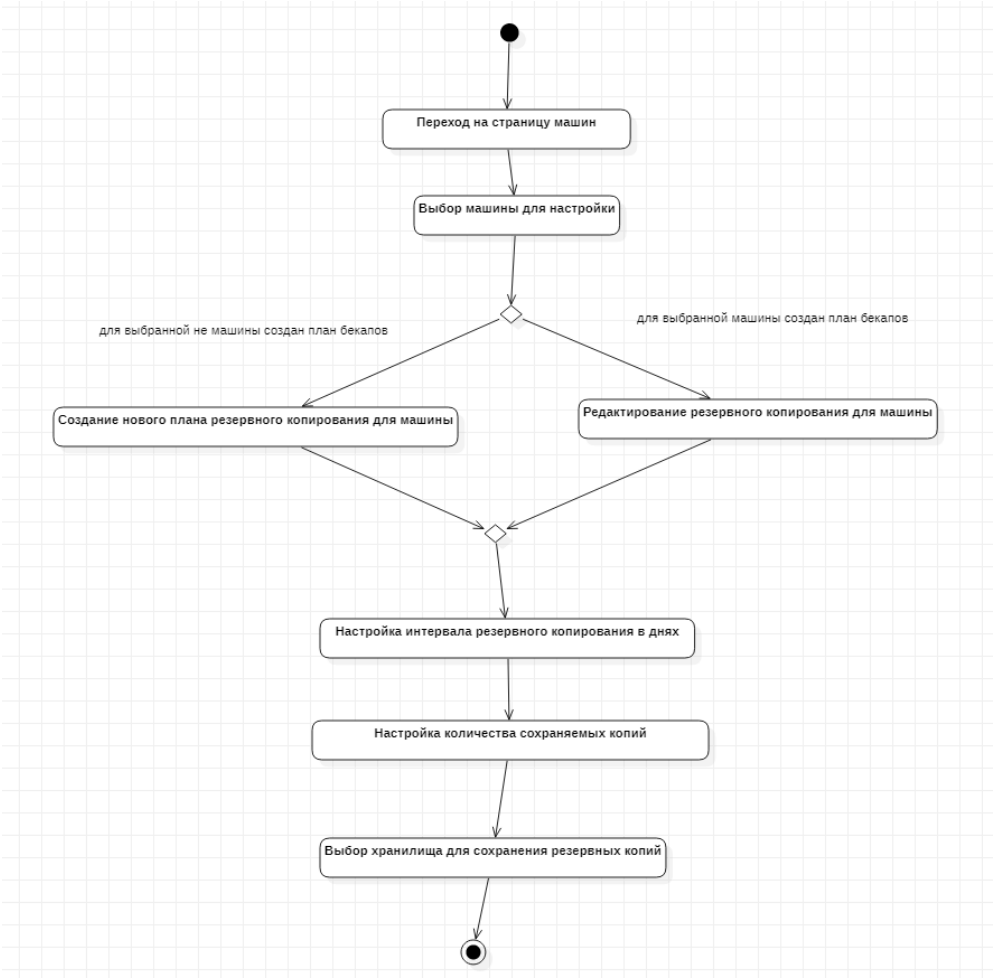


Рисунок 1.6 - Диаграмма деятельности «Настройка параметров резервного копирования»

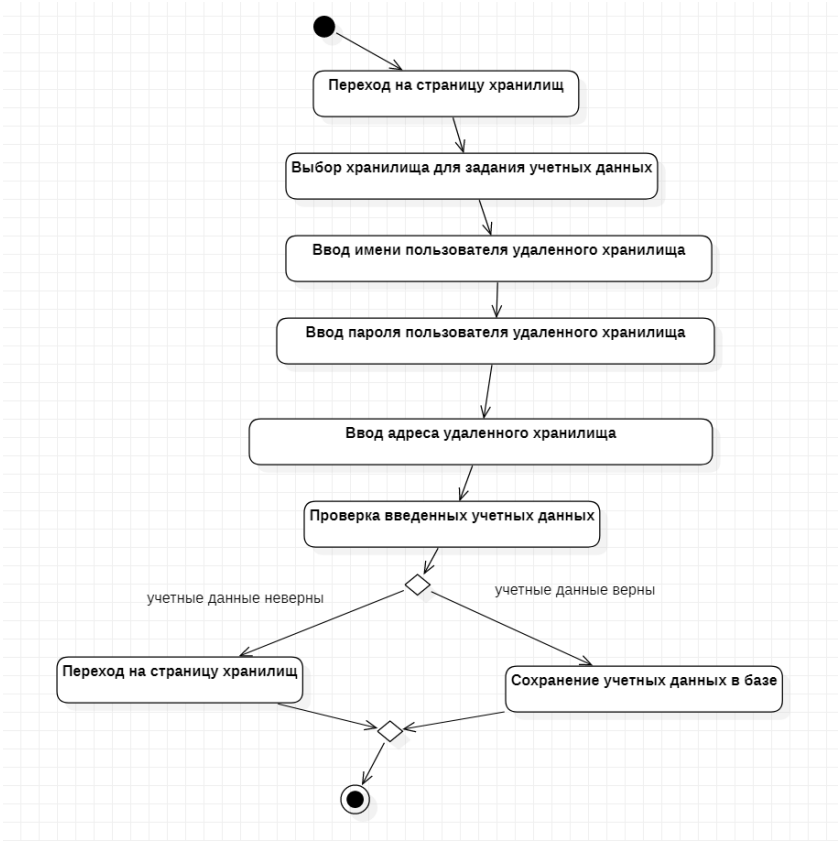


Рисунок 1.7 - Диаграмма деятельности «Загрузка в систему учетных данных для удаленных хранилищ»

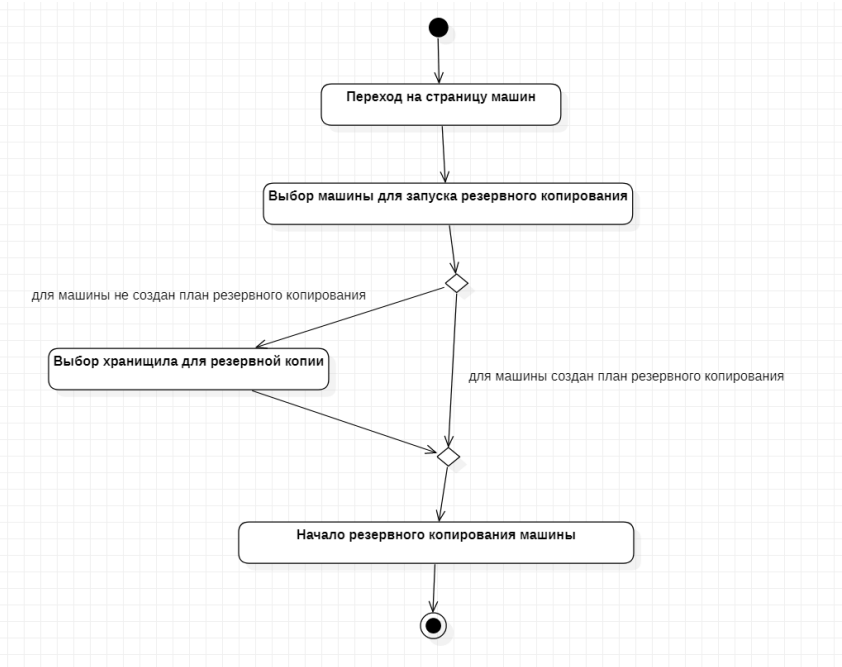


Рисунок 1.8 - Диаграмма деятельности «Запуск резервного копирования в ручном режиме»

Рисунок 1.9 - Диаграмма деятельности «Удаление резервной копии с хранилища»

1.2.5 Разработка диаграммы классов

Диаграмма классов используется для описания статичных объектов 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 Б.

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

Класс	Описание класса
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 1.10.

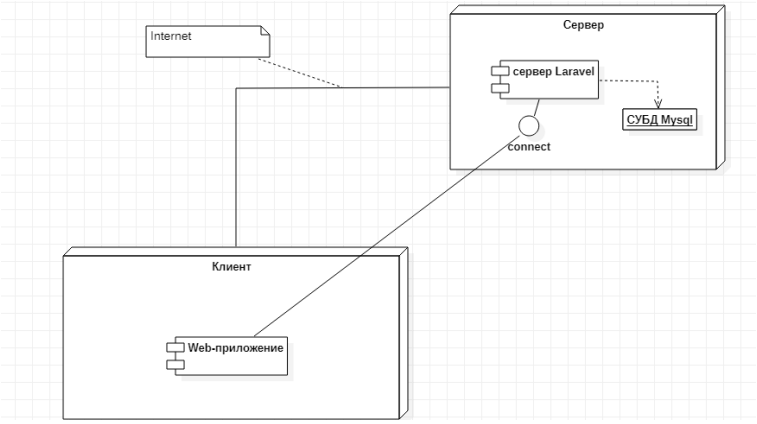


Рисунок 1.10 – Диаграмма развертывания

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 *MySQL*. 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 1.11 и содержит 7 сущностей. Описание сущностей и их атрибутов представлено в таблицах 1.3 – 1.9.

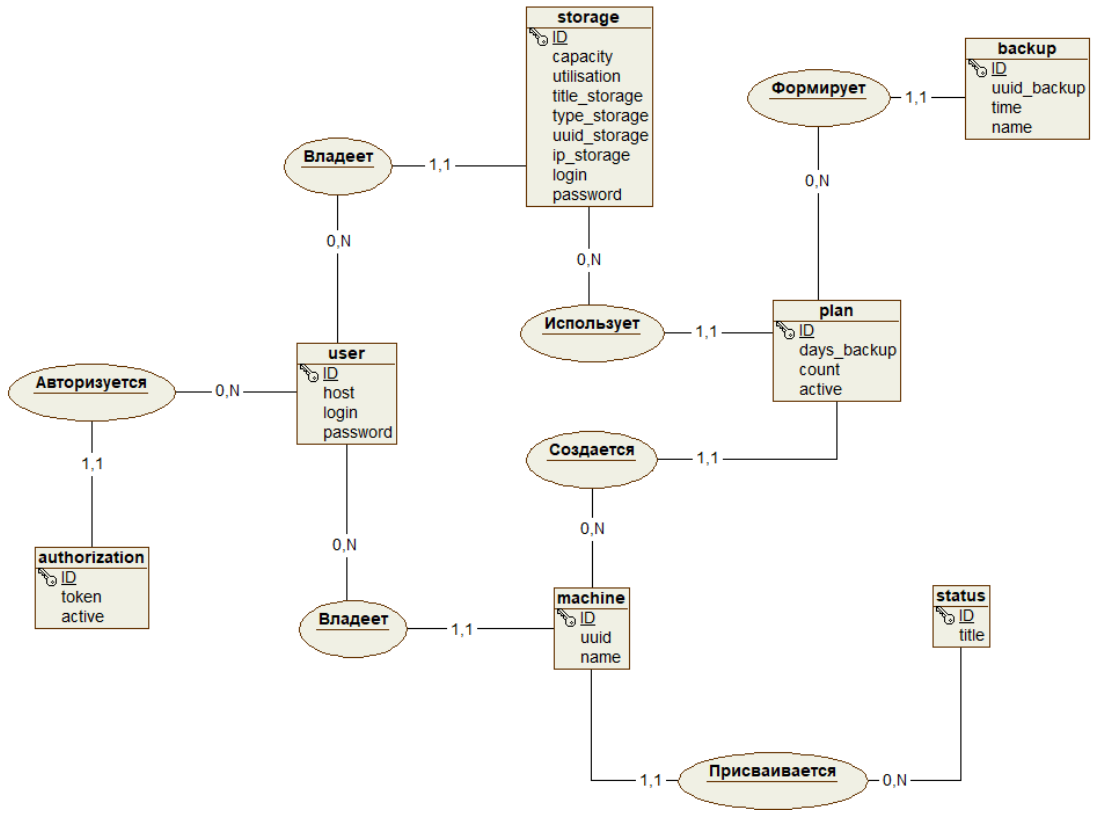


Рисунок 1.11 – Концептуальная модель данных

Таблица 1.3 – Описание атрибутов сущности «*authorization*»

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

Таблица 1.4 – Описание атрибутов сущности «*user*»

Имя атрибута	Назначение	Первичный ключ
ID	Код пользователя	*PK*
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	*PK*
Uuid	*Uuid* машины	-
Name	Имя машины	-

Таблица 1.6 – Описание атрибутов сущности «*storage*»

Имя атрибута	Назначение	Первичный ключ
ID	Код хранилища	*PK*
Capacity	Объем хранилища	-

Имя атрибута	Назначение	Первичный ключ
Utilisation	Количество занятых байт 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	*PK*
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	*PK*
Uuid_backup	*Uuid* бекапа	-
Time	Время создания копии	-
Name	Название копии	-

Таблица 1.9 – Описание атрибутов сущности «*status*»

Имя атрибута	Назначение	Первичный ключ
ID	Код статуса	*PK*
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 одно хранидище 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 1.12 и содержит 7 сущностей.

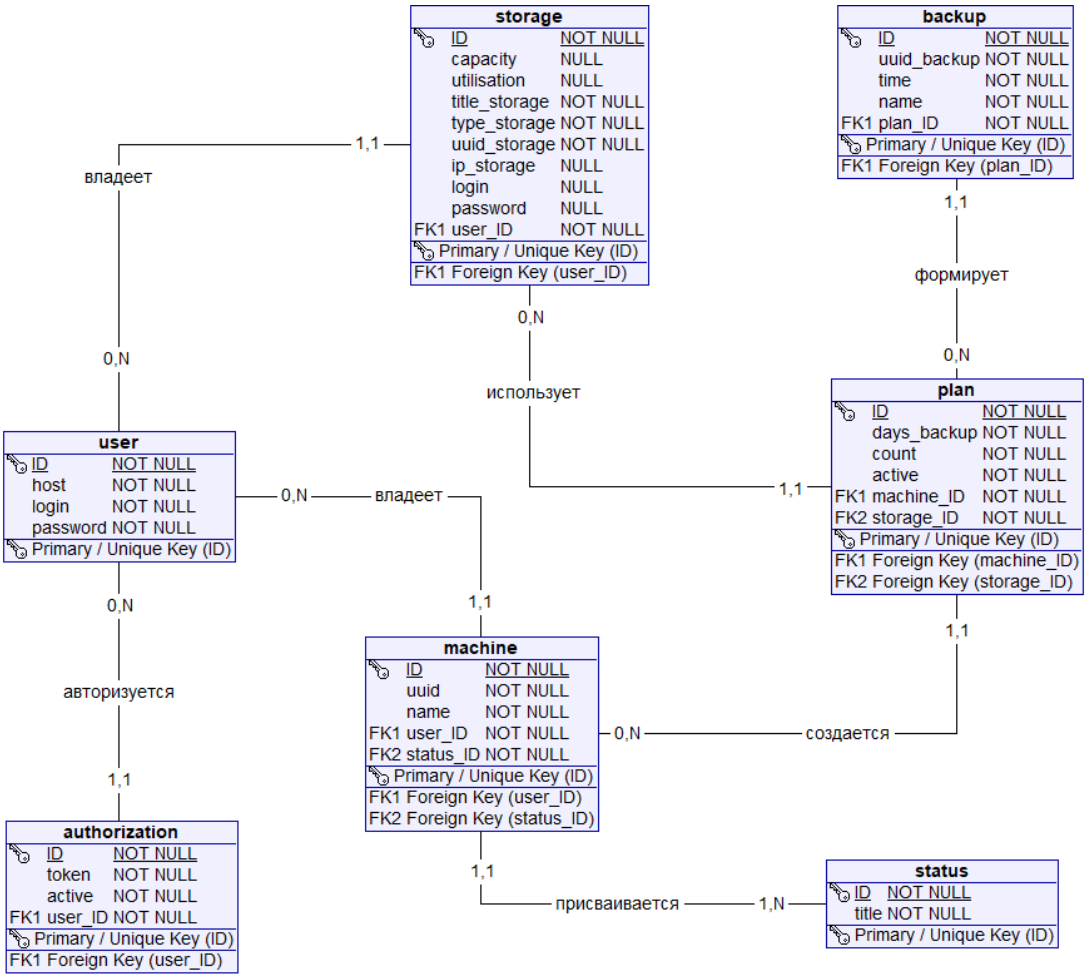


Рисунок 1.12 – Логическая модель данных

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	*PK*	-
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	*PK*	-
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	*PK*	-
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	*PK*	-
Capacity	*Capacity*	Объем хранилища	-	-
Utilisation	*Utilisation*	Занятое место на хранилище	-	-

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
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	*PK*	-
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	*PK*	-

Имя атрибута	Физическое имя	Назначение	Первичный ключ	Внешний ключ
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	*PK*	-
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 1.12, 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 1.12, 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, 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 1.12, 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, 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 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 со-

здания 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 *MySql* и приведена на рисунке 1.13.

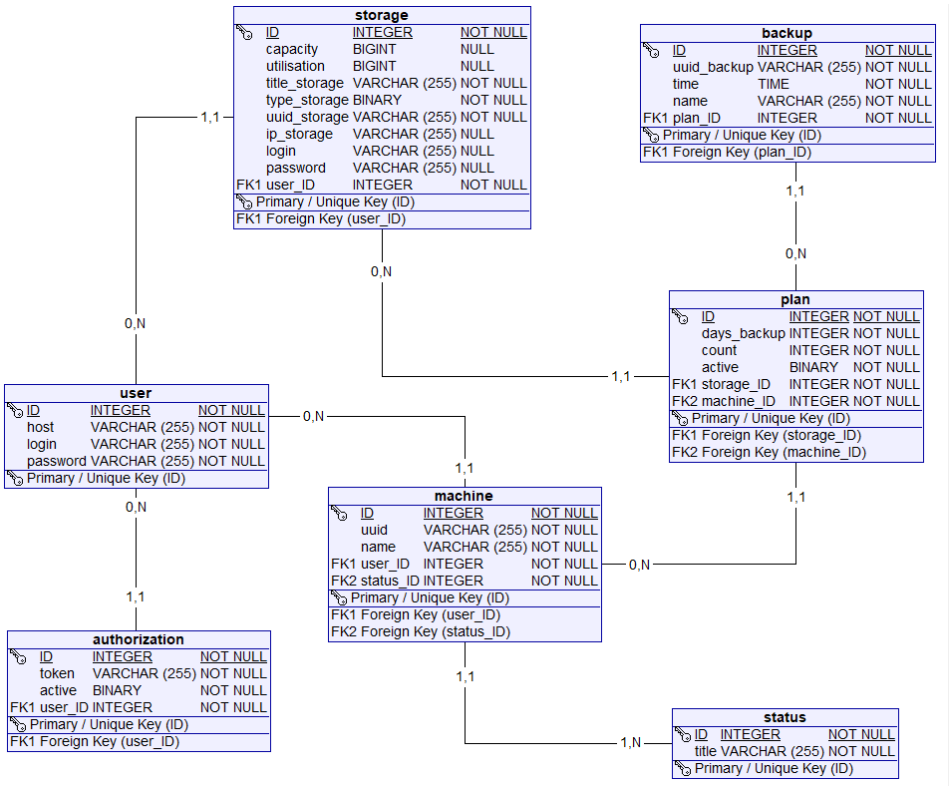


Рисунок 1.13 – Физическая модель данных

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)

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Внешний ключ	Тип данных
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	*PK*	-	*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	*PK*	-	*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 *PHPMyAdmin* 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.

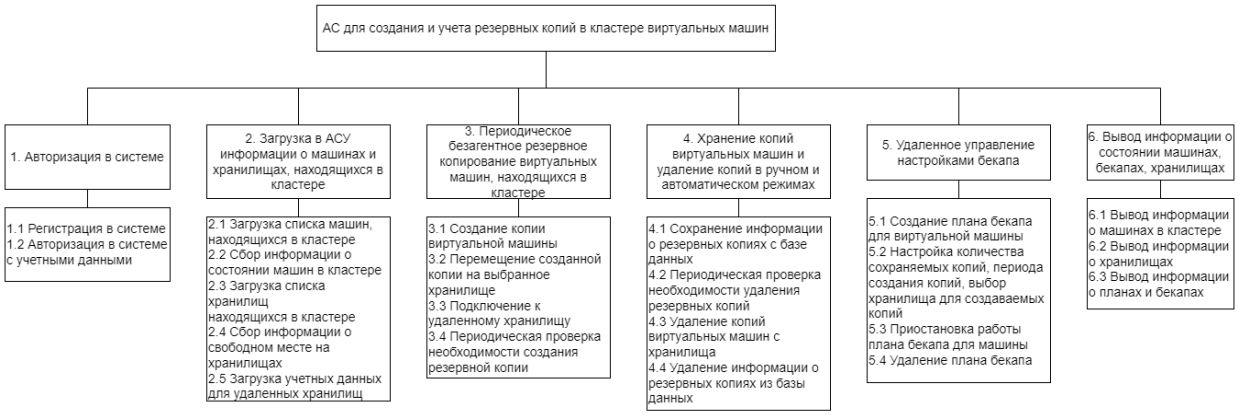


Рисунок 1.14 - Иерархия функций приложения

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, которые в свою очередь состоят из отдельных подмодулей, структурная схема приложения представлена на рисунке 1.15, описание модулей приведено в таблице 1.24.

Таблица 1.24 – Описание модулей приложения

Название модуля	Название подмодуля	Назначение
Client	*App.vue*	Вызывает остальные подмодули
	Navbar.vue	Навигация по приложению

Название модуля	Название подмодуля	Назначение
	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

Рисунок 1.15 – Логическая структура приложения

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 прило-

жения, а также позволяет выйти из системы, и содержит следующие методы:

- метод *showMenu* предназначен отображение и скрытия 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 *loginned* 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;

- *loggining* – 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* обновляет информацию 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.

Компонент 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.

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 данный, использованная система управления базой данных – *MySql*. 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 виртуальны 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 В;

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 необходимо-

сти начинает его, путем вызова функции *handle* модуля *ForceBackup.php*. Данная функция вызывается периодически с помощью утилиты *cron* операционной системы *Linux*, и вызывает функции основные функции *parseCurrentMachinesStates* и *ProcessingClusterPlans*, а также функцию *handle* модуля *ForceBackup.php* блок-схемы которых представлены в приложении В;

- 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 В;

- функция *parseCurrentMachinesStates* предназначена для получения актуального состояния виртуальных машин в кластере, блок-схема функции *parseCurrentMachinesStates* приведена в приложении В;

- 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:

- функция *checkAcccount* предназначена для проверки учетных данных путем авторизации на удаленном хосте с введенными данными, путем подключения по протоколу *Secure Shell*;

- функция *login* предназначена для аутентификации в системе, при успешной попытке, пользователю отправляется ключ аутентификации;

- функция *logout* предназначена для выхода из системы.

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 1.16).

При неверно введенных данных соответствующие поля формы будут подчеркнуты красным цветом, что означает то, что нужно проверить вводимые данные, рисунок 1.17.

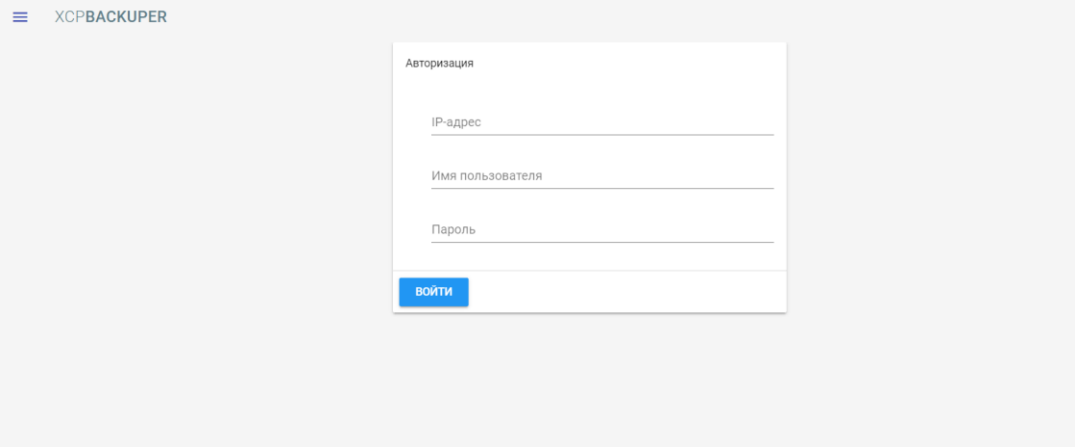


Рисунок 1.16 - Форма авторизации в приложении

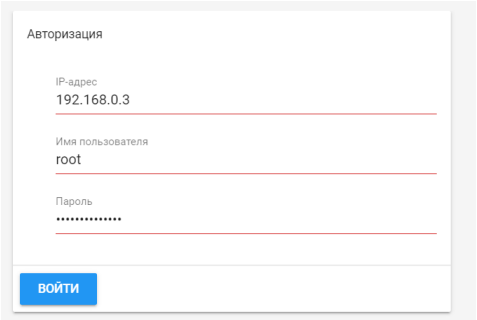


Рисунок 1.17 – Форма авторизации при вводе неверных учетных данных

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 1.18.

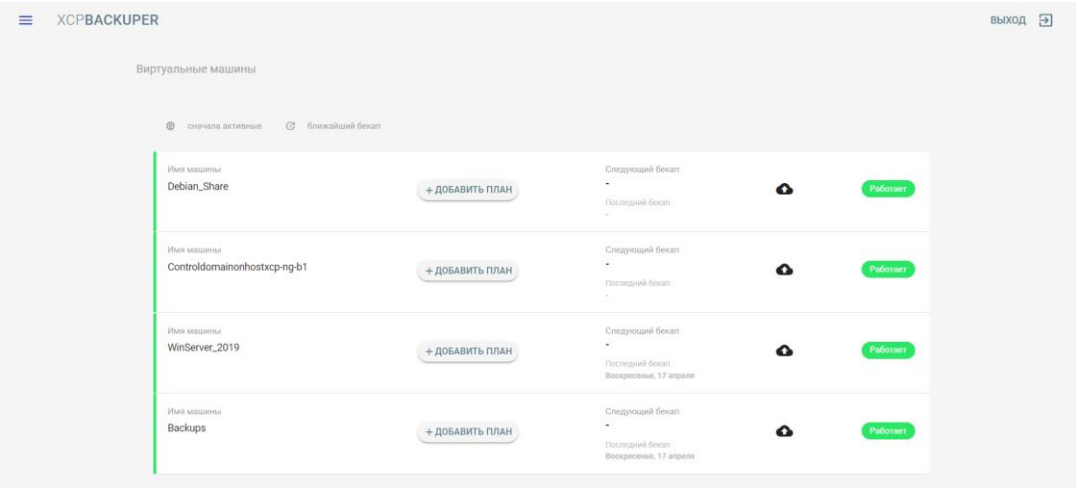


Рисунок 1.18 – Форма приложения со списком машин

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 1.19).

Рисунок 1.19 – 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 1.20), у указанной машины будут отображаться настройки текущего плана резервного копирования, а также информация о том, когда будет создана следующая резервная копия и дата последней.

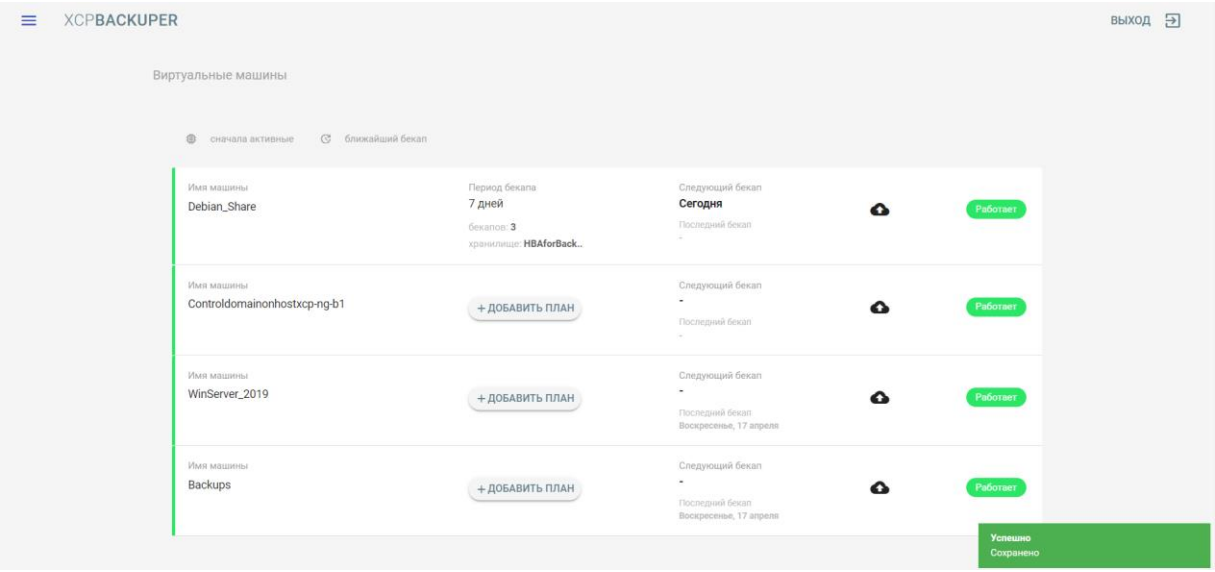


Рисунок 1.20 – Форма со списком машин после создания плана

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 1.21), и нажать на кнопку «Изменить план», после чего откроется модальное окно редактирования настроек резервного копирования (рисунок 1.22), или на странице со списком планов резервного копирования (рисунок 1.23) напротив выбранного плана нажать на кнопку «Редактировать» с иконкой в виде карандаша, после чего откроется такое-же модальное окно (рисунок 1.22). После редактирования настроек для их сохранения необходимо нажать кнопку «Сохранить».

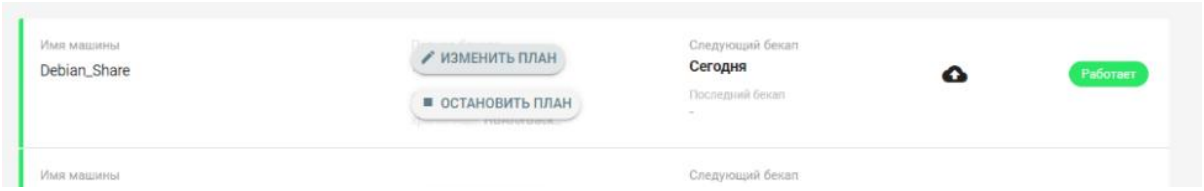


Рисунок 1.21 – Дополнительные кнопки управления настройками резервного копирования

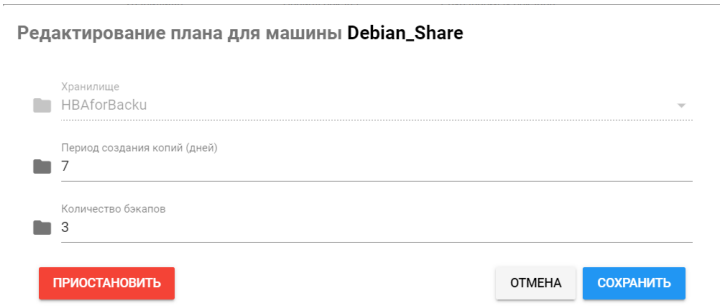


Рисунок 1.22 – Модальное окно редактирования настроек резервного копирования

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 бекапов», после чего откроется выпадающий список созданных резервных копий (рисунок 1.23).

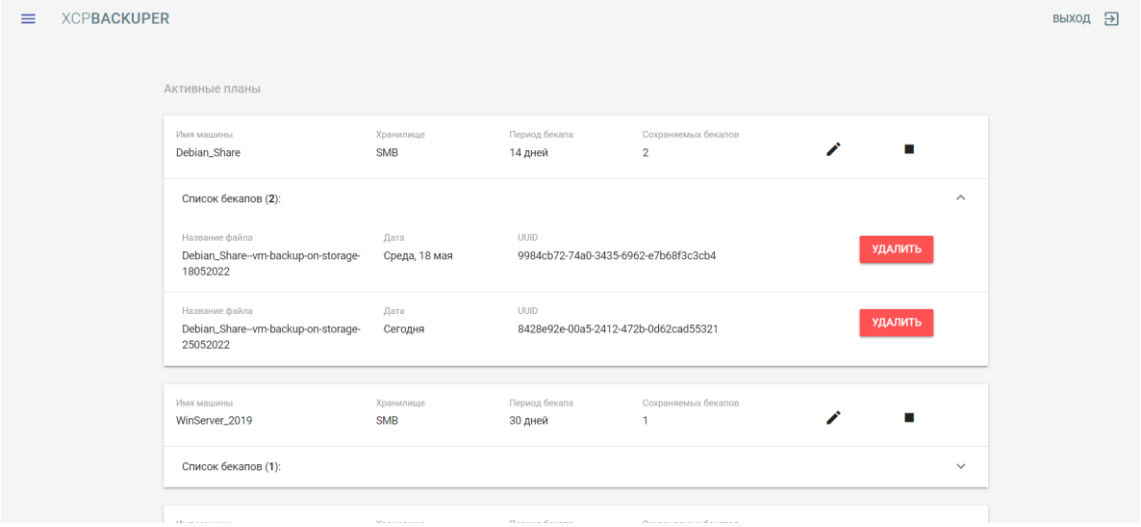


Рисунок 1.23 – Страница списка планов резервного копирования с раскрытым списком резервных копий

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 1.24) в котором необходимо нажать на кнопку «Да».

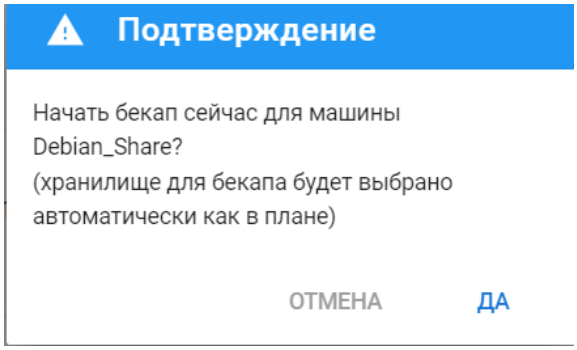


Рисунок 1.24 – Модальное окно подтверждения начала резервного копирования

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 1.21) и нажать на кнопку «Остановить план», после чего в модальном окне (рисунок 1.25) подтвердить приостановку, нажав на кнопку «Да», или на стра-

нице со списком планов резервного копирования (рисунок 1.23) напротив выбранного плана нажать на кнопку «Остановить план» с иконкой в виде квадрата, после чего откроется такое-же модальное окно (рисунок 1.25).

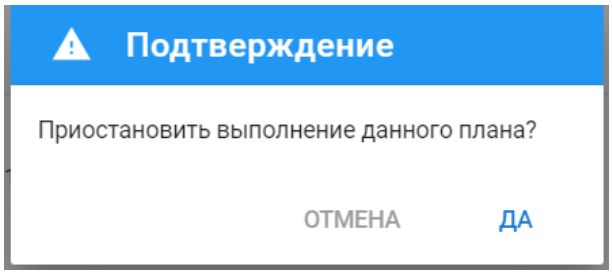


Рисунок 1.25 – Модальное окно подтверждения остановки работы плана

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 1.23), после чего в появившемся модальном окне подтвердить удаление нажав кнопку «Да» (рисунок 1.26).

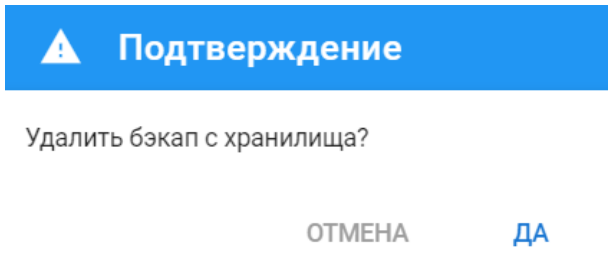


Рисунок 1.26 – Окно подтверждения удаления резервной копии

Для удаления неактивного плана резервного копирования необходимо на странице со списком планов напротив выбранного плана (рисунок 1.27), нажать на кнопку «Удалить план» с иконкой в виде корзины, после чего в модальном окне (рисунок 1.28) подтвердить удаления нажав на кнопку «Да». Для возобновления работы плана резервного копирования необходимо на странице со списком планов выбрать неактивный план (рисунок 1.27) и нажать на кнопку «Запустить план» с иконкой в виде стрелки, после чего в появившемся модальном окне (рисунок 1.29) подтвердить нажатием кнопки «Да».

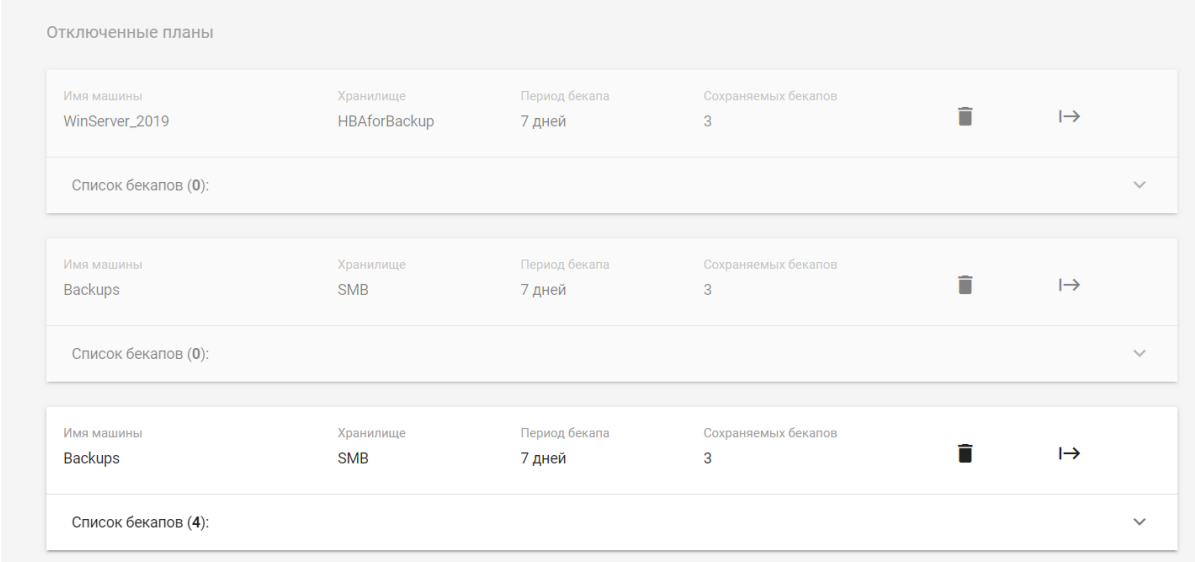


Рисунок 1.27 – Список неактивных планов резервного копирования

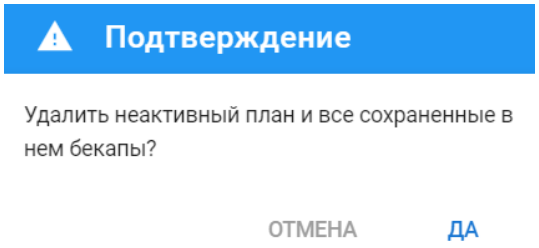


Рисунок 1.28 – Модальное окно подтверждения удаления плана

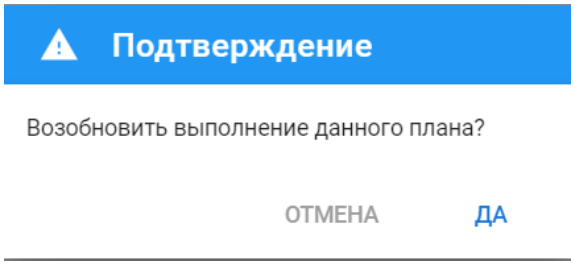


Рисунок 1.29 – Окно подтверждения возобновления работы плана

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 1.30), после чего в появившемся модальном окне (рисунок 1.31) 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 1.32), после чего можно закрыть модальное окно нажав кнопку «Закрыть».

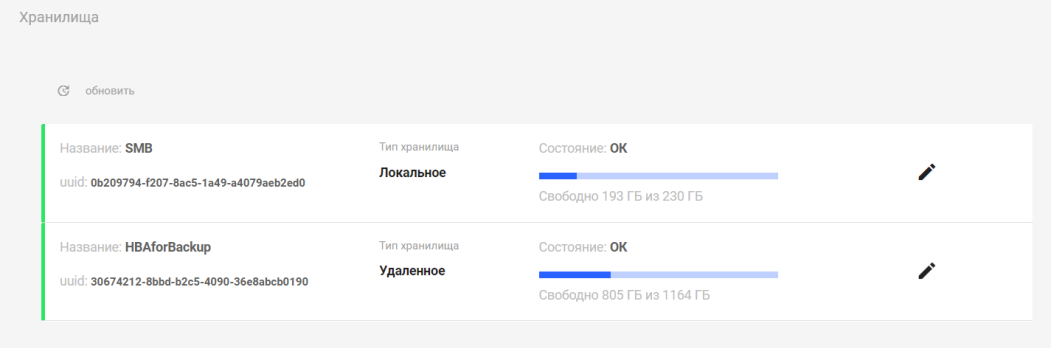


Рисунок 1.30 – Форма со списком хранилищ

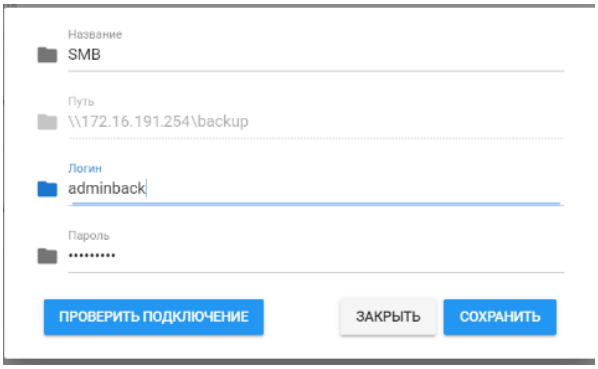


Рисунок 1.31 – Окно редактирования информации о хранилище

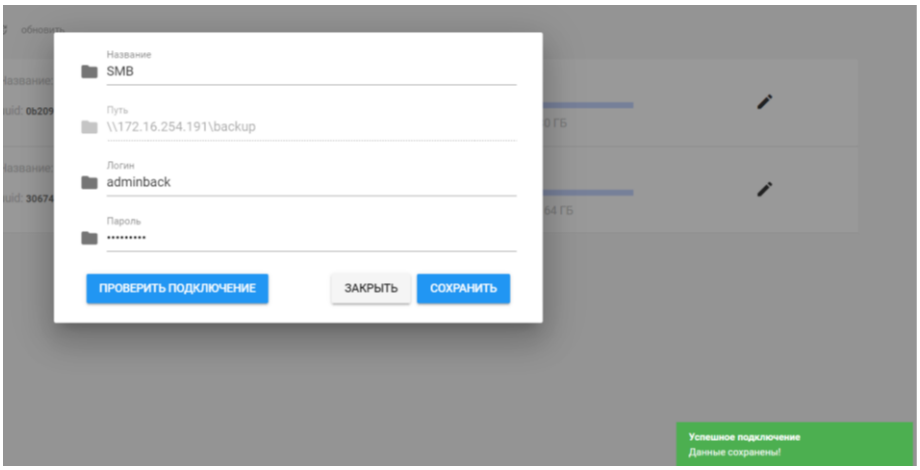


Рисунок 1.32 – Сообщение успешности подключения хранилища

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 1.21), и нажать на кнопку «Изменить план», после чего откроется модальное окно редактирования настроек резервного копирования (рисунок 1.22), после чего заменить количество резервных копий (установить 1), изменить период резервного копирования (установить 14), далее нажать на кнопку сохранить, после чего должно появится сообщение об успешном сохранении данных (рисунок 1.33), при проверке на странице со списком машин, указанные данные должны быть отображены (рисунок 1.34).

Рисунок 1.33 – Сообщение об успешности сохранения данных

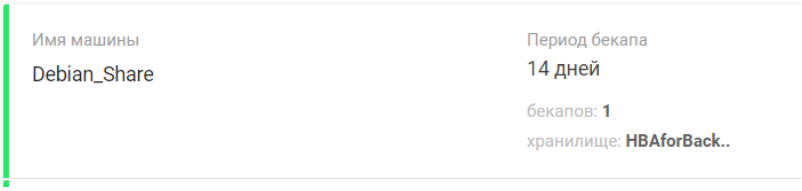


Рисунок 1.34 – Отображение сохраненных данных на странице со списком машин

2 Безопасность жизнедеятельности

2.1 Введение

В современном мире, как никогда раньше в истории информация играет большую роль в жизни человека. Для работы с информацией используются различные инструменты, среди которых одно из главных мест занимает компьютер, который в последние годы стал важнейшим помощником в любой сфере деятельности. Возможность передавать данные посредством сети Интернет, позволила беспрепятственно обмениваться опытом специалистам по всему миру. Однако на фоне стремительного научного прогресса рядовой пользователь все меньше отводит внимания вопросу безопасного использования информационных технологий. Недавние исследования ученых показали, что не столько компьютерная техника является прямым фактором негативного воздействия на организм человека, сколько ее неправильная конфигурация, несоблюдение элементарных правил компьютерной гигиены. Компьютер — новая реальность в нашей жизни, и в силу его развития люди не приспособлены к постоянной работе с ним. Поэтому с профилактической точки зрения лучше учитывать проблемы, связанные с влиянием компьютера на здоровье человека, чем при лечении неотложных состояний [5].

2.2 Неблагоприятные факторы воздействия

Существует три группы основных воздействующих неблагоприятным образом факторов компьютера на организм человека [6]:

- вредное воздействие, вызванное различными изучениями;
- воздействие, связанное с эргономическими факторами;
- абиотические факторы воздействия.

К источникам вредных излучений относится: накопление статического электрического заряда на экране дисплея, электромагнитное излучение монитора и процессора компьютера в пределах от 5 герц до 300 мегагерц, излуче-

ние в ультрафиолетовом диапазоне с длинной волны 200-400 нанометров., а также инфракрасное излучение с длинной волны от 1050 нанометров до 1 метра и рентгеновское излучение более 1.2 килоэлектронвольт.

К эргономическим факторам относится: сниженная контрастность изображения в условиях интенсивного внешнего освещения, зеркальные отражения от передней поверхности экрана, мерцание изображения на экране.

Абиотические факторы включают в себя: нервно-эмоциональное напряжение, снижение работоспособности при монотонных и однообразных действиях, пониженная подвижность организма, нахождение в однообразной вынужденной рабочей позе, необходимость выполнения точной зрительной работы на ярком экране в условиях перепада освещенности и другие.

Характерной чертой рабочего процесса с использованием ПК является высокая информационная нагрузка, связанная с используемыми возможностями операционной системы и специфическими прикладными задачами, которые выполняет пользователь.

Комплекс перечисленных факторов может оказывать влияние как на физиологическое, так и на психоэмоциональное состояние пользователей как персональных, так и промышленных компьютеров.

2.3 Противодействие вредным факторам

2.3.1 Санитарно-гигиеническая характеристика рабочего места

Обеспечение комфортных и безопасных условий работы является одной из первоочередных задач при организации рабочего места. Работодатель обязан при создании организации рабочих мест иметь ввиду не только должностные обязанности и характер выполняемых работ, а еще индивидуальных физиологические и психические особенности своих сотрудников.

Основные требования диктуются Трудовым кодексом Российской Федерации, а также различными нормативными документами, в том числе, санитарно-эпидемиологическими правилами и нормативами (СанПиН). При работе

с компьютером необходимо руководствоваться СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы».

В помещении, в котором работники находятся непрерывно более 2 часов, должно быть естественное освещение [6]. Исключение составляют помещения, которые могут быть размещены в цокольном и цокольном этажах, если это необходимо для соблюдения технологического процесса. Окна в помещениях, где работают с компьютерами, должны выходить на север и северо-восток. Оконные проемы должны быть оборудованы регулируемыми жалюзи, наружными крышками и т.п.

При работе за компьютером с плоским экраном рабочее место должно иметь площадь не менее 4,5 квадратных метров, при использовании кинескопического монитора - не менее 6 квадратных метров. После каждого часа работы помещение необходимо проветривать [7].

Экран монитора должен находиться на расстоянии 600 - 700 мм от глаз работника, но не менее 500 мм с учетом размера буквенно-цифровых знаков и символов. Для облегчения чтения документов можно использовать мобильные подставки (подставки), которые размещают на одном уровне и на той же высоте, что и экран.

Рабочий стул должен регулироваться по высоте и повороту, по высоте и углу наклона сиденья и спинки, а также по расстоянию между спинкой и передним краем сиденья. При этом регулировка каждого параметра должна быть самостоятельной, простой в исполнении и иметь надежную фиксацию.

При выборе разновидности рабочего стула необходимо иметь ввиду рост пользователя, вид и продолжительность работы с компьютером. Устройство рабочего кресла должна обеспечивать соблюдение оптимальной рабочей позы при работе на компьютере, позволять менять позу с целью снижения статического напряжения мышц шейно-плечевого и спинного отдела, предотвращать развитие усталости. Поверхности сиденья, спинки и других элементов кресла

должны быть полумягкими, с нескользящим, плохо электризующимся и воздухопроницаемым покрытием, позволяющим легко очищаться от загрязнений.

При работе, выполняемой в сидячем положении без физических нагрузок, идеальными параметрами микроклимата на рабочем месте являются [6]:

- температура воздуха в холодное время года - от 22 до 24°С, в теплое время года - от 23 до 25°С;

- относительная влажность воздуха на рабочем месте – от 40 до 60 процентов, скорость движения воздуха - 0,1 м/с. Для повышения влажности воздуха в помещении могут использоваться увлажнители воздуха.

Освещенность на столе в радиусе расположения рабочего документа должна быть в пределах 300 - 500 люкс. Освещение не должно быть источником бликов на поверхности экрана монитора. Освещенность поверхности дисплея не должна превышать 300 люкс. Коэффициент пульсации должен быть не более 5%.

При выполнении работе на мониторах и компьютерах инженерно-техническими работниками, уровень шума не должен превышать 60 акустических децибел, в помещениях операторов ЭВМ (без мониторов) - 65 акустических децибел, на рабочих местах в помещениях с шумными компьютерными блоками - 75 акустических децибел [6].

2.3.2 Организация труда и отдыха

Действующее законодательство предусматривает определенный режим работы на компьютере, включающий перерывы, частота и длительность которых обуславливается видом, категорией выполняемой работы, а также продолжительностью смены. При восьмичасовой рабочей неделе перерывы длительностью 15 минут проводятся через два часа после начала работы и два часа после обеда для сотрудников, деятельность которых связана с чтением информации с экрана.

Во избежание преждевременного утомления у сотрудников, работающих на компьютерах, зрительного дискомфорта и других нежелательных

субъективных ощущений, несмотря на выполнение санитарно-гигиенических, эргономических требований, режимов труда и отдыха, необходимо применять индивидуальный подход к ограничению времени работы с ПК, изменение продолжительности перерывов на отдых или переключения с деятельности на другую, не связанную с использованием компьютера.

В случае, если выполняемый вид работы обязывает постоянно работать с компьютером и связан с психологическим напряжением и концентрацией внимания, при отсутствии возможности регулярного переключения на другие роды деятельности, не связанные работой на компьютере, рекомендуется каждые 45-60 минут делать перерыв на 10 - 15 минут.

Во время организованных перерывов рекомендуется выполнять комплексы упражнений ввиду эмоциональной разгрузки, снижения усталости глаз и исключения влияния гиподинамии и гипокинезии. Кроме того, тем сотрудникам, кто работает в условиях повышенной психоэмоциональной нагрузки, во время регулярных перерывов и после работы положена психологическая разгрузка в специально оборудованных местах (комната психологической разгрузки).

Излучение компьютера является вредным фактором условий труда, если сотрудник работает за ним более 50% рабочего времени согласно СанПиНу 2.2.2/2.4.134003, поэтому офисные работники относящиеся к этой категории должны подвергаться периодическому обязательному медицинскому обследованию независимо от того, признаны ли их условия труда приемлемыми или нет. Также офисные работники должны проходить обязательный медосмотр при устройстве на работу (Письмо Минтруда РФ от 21.03.2014 № 152/ООГ-242).

2.3.3 Требования по пожарной безопасности

В офисном помещении как правило одновременно находится изрядное количество техники и персонала. Возникновение пожара в таких условиях может привести к тяжелым последствиям, в плоть до летального исхода. Для

избежания подобных последствий необходимо соблюдать правила пожарной безопасности в офисе.

В законодательстве прописаны следующие основные требования:

- должны быть специальные места оборудованные информационными стендами с номерами служб спасения;
- каждый этаж должен быть снабжен планом эвакуации;
- должны быть в наличии первичные средства пожаротушения;
- должна работать противопожарная сигнализация и автономные противопожарные системы;
- все сотрудники должны знать правила пожарной безопасности.

В офисе должен быть ответственный человек, который следит за соблюдением правил пожарной безопасности и проводит регулярные инструктажи и обучающие занятия с персоналом. Все сотрудники должны соблюдать указанные правила по обеспечению пожарной безопасности в помещениях.

В офисных помещениях должна быть установлена система пожарной сигнализации и система предупреждения сотрудников о пожаре. Требования к системе сигнализации:

- надежность - устройство должно работать в любых условиях;
- большая область работы - датчики должны фиксировать все пространство офиса;
- бесперебойность работы - система должна работать даже в случае сбоя питания.

В офисных помещениях к сети подключено большое количество устройств. При подключении устройств необходимо учитывать ограничение по максимальному току. Если не принять меры предосторожности, перегруженная проводка может стать причиной пожара. Категорически запрещается перегружать электрическую сеть, подключая к розетке несколько устройств с помощью удлинителей.

Необходимо регулярно проверять целостность розеток, выключателей и аксессуаров. Источник питания должен быть оборудован автоматическим выключателем.

2.4 Расчет освещенности рабочего места

Расчет освещения рабочего места заключается в выборе системы освещения, определению необходимого количества источников света, их типа и расположения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение осуществляется с использованием двух видов электрических источников света: ламп накаливания и люминесцентных ламп. В данном случае были выбраны люминесцентные лампы, которые обладают рядом достоинств в отличии от ламп накаливания [8]:

- спектральный состав люминесцентных ламп близок к натуральному дневному свету;
- коэффициент полезного действия данных ламп в среднем в полтора-два раза выше чем у ламп накаливания;
- светоотдача у люминесцентных ламп выше чем у ламп накаливания в три-четыре раза;
- обладают большим сроком службы.

Для расчета используются следующие исходные данные: комната имеет площадь 20 квадратных метров, длину 5 метров, ширину 4 метра и высоту 3 метра. Определим освещенность F [Лм] по методу светового потока [9], воспользовавшись формулой:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n}, \qquad (2.1)$$

где E - нормированная минимальная освещенность, [Лк] (согласно нормативам освещенности офисных и жилых объектов по СНиП, для офисного помещения с компьютерами минимальная освещенность равна 300 Лк);

S - площадь освещаемого помещения, [м^2], (в нашем случае S = 20 м^2);

Z - отношение средней освещенности к минимальной (обычно принимается равным в пределах от 1,1 до 1,2, пусть Z = 1,1);

K - коэффициент, характеризующий снижение светового потока лампы в результате загрязнения источника света в процессе эксплуатации (возьмем K = 1,5);

n - коэффициент использования светового потока, (отражается в отношении светового потока к совокупному потоку всех ламп и обуславливается характеристиками светильника, размерами помещения и коэффициентами отражения от стен ($P_С$) и потолка ($P_П$)).

В помещениях, где находится компьютер, величины коэффициента отражения должны быть в следующих пределах: для потолка: 60…70%, для стен: 40…50%, для пола: около 30%. Для других поверхностей и рабочей мебели: 30…40%.

Рассчитаем освещенность исходя из следующих значений коэффициентов: $P_С$ = 40%, $P_П$ = 65%. Для этого сначала вычислим индекс помещения I, определяемый геометрическую характеристику помещения по формуле:

$$I = \frac{S}{h \cdot (A + B)} \quad , \tag{2.2}$$

где S - площадь помещения, [м^2] (в нашем случае S = 20 м^2);

h - расчетная высота подвеса, [м] (в нашем случае h = 2.90 м);

A - ширина помещения, [м] (в нашем случае A = 4 м);

B - длина помещения, [м] (в нашем случае B = 5 м).

Подставив значения в формулу (2.2) получим:

$$I = \frac{20}{2.90 \cdot (4+5)} = 0.77$$

Зная индекс помещения I, по таблице расчета использования светового потока, приведенной на рисунке 2.1 находим коэффициент n = 0,25.

Светильник	«Глубокоизлучатель»			«Универсаль» без затенителя			«Люцетта»			ВЗГ 200 с отражателем			ОД			ПВЛ		
$\rho_п$, %	30	50	70	30	50	70	30	50	70	30	50	70	30	50	70	30	50	70
$\rho_с$, %	10	30	50	10	30	50	10	30	50	10	30	50	10	30	50	10	30	50
i	*Коэффициент использования* $\eta \cdot 100$																	
0,5	19	21	25	21	24	28	14	16	22	12	14	17	23	26	31	14	16	19
0,6	24	27	31	27	30	34	19	21	27	16	18	21	30	33	37	18	20	22
0,7	29	31	34	32	35	38	23	24	30	19	21	24	35	38	42	21	23	25
0,8	32	34	37	35	38	41	25	26	33	21	24	26	39	41	45	23	25	27
0,9	34	36	39	38	40	44	27	29	35	23	25	28	42	44	48	25	27	29
1,0	36	38	40	40	42	45	29	31	37	25	27	29	44	46	49	26	28	30
1,1	37	39	41	42	44	46	30	32	38	26	27	30	46	48	51	27	29	31
1,25	39	41	43	44	46	48	31	34	41	27	29	31	48	50	53	29	30	32
1,5	41	43	46	46	48	51	34	37	44	29	30	39	50	52	56	30	31	34
1,75	43	44	48	48	50	53	36	39	46	30	32	34	52	55	58	31	33	35
2,0	44	46	49	50	52	55	38	41	48	32	33	35	55	57	60	33	34	36
2,25	46	48	51	52	54	56	40	43	50	33	35	37	57	59	62	34	35	37
3,0	49	51	53	55	57	60	44	47	54	35	37	39	60	62	66	36	37	40
3,5	50	52	54	56	58	61	45	49	57	36	38	40	61	64	67	37	38	40
4,0	51	52	55	57	59	62	46	50	59	37	39	41	63	65	68	38	39	41
5,0	52	54	57	58	60	63	48	52	61	38	40	42	64	66	70	38	40	42

Рисунок 2.1 – таблица расчета использования светового потока

Рассчитаем световой поток I, подставив значения в формулу (2.1):

$$F = \frac{300 \cdot 1.5 \cdot 20 \cdot 1.1}{0.25} = 39600 \text{ Лм}$$

Для установки выберем люминесцентные лампы типа ЛБ-80 со световым потоком 5250 люкс.

Чтобы получить количество ламп для установки, разделим полученный необходимый световой поток на значение светового потока одной лампы, и получим необходимое количество ламп $N = 8$. В качестве осветительных приборов будут использоваться светильники типа ОД, каждый светильник экипируется двумя лампами.

3 Экономическое основание разработки

Автоматизированная система как правило разрабатывается если данная разработка экономически целесообразна. Определим целесообразность и экономический эффект от разработки автоматизированной системы для создания и учета резервных копий в кластере виртуальных машин, для этого необходимо рассчитать следующие параметры:

- трудоемкость разработки автоматизированной системы;
- стоимость разработки автоматизированной системы;
- минимальная стоимость разработки автоматизированной системы;
- затраты на внедрение автоматизированной системы.

3.1 Определение трудоемкости разработки АС

Трудоемкость разработки автоматизированной системы складывается из времени, затраченного на каждый этап проектирования, выраженный в человеко-часах, в процессе разработки должны быть выполнены следующие задачи:

- описание задачи и разработка плана работ;
- выбор инструментов разработки;
- разработка схем структуры проекта;
- программирование клиентского модуля приложения;
- программирование серверного модуля приложения;
- отладка программы;
- подготовка документации.

Для определения трудоемкости разработки АС необходимо рассчитать трудоемкость каждого этапа разработки системы, ожидаемое значение трудоемкости каждого этапа t_i [чел./дн] может быть рассчитано эмпирически по следующей формуле и выражено в человеко-днях [10]:

$$t_i = \frac{a_i + 4m_i + b_i}{6}, \quad (3.1)$$

где a_i – минимально возможное время выполнения работ, [чел./дн.];

b_i – максимально возможное время выполнения работ, [чел./дн.];

m_i – наиболее вероятное время выполнения работ, [чел./дн.].

Длительность выполнения каждой работы D_i [дн.], рассчитывается по следующей формуле и выражается в количестве дней, затрачиваемых на работу:

$$D_i = \frac{t_i}{n}, \quad (3.2)$$

где n_i – количество исполнителей, [чел].

Количество исполнителей в данном случае равно 1, таким образом результат оценки трудоемкости каждого вида работ приведена в таблице 3.1.

Таблица 3.1 – Трудовые затраты

Вид работ	Оценка трудоемкости			Расчетные величины	
	a_i	m_i	b_i	t_i	D_i
Описание задачи и разработка плана работ	2	3	4	3	3
Выбор инструментов разработки	2	2	2	2	2
Разработка схем структуры проекта	4	5	6	5	5
Программирование клиентского модуля приложения	20	24	28	24	24
Программирование Серверного модуля приложения	20	24	28	24	24
Отладка программы	10	12	14	12	12
Подготовка документации	4	5	6	5	5
Всего					75

3.2 Расчет затрат на разработку автоматизированной системы

Расчет затрат на разработку $К_П$ [руб.] можно рассчитать по формуле:

$$К_п = Ф_{з/п}[(1 + \beta_Д)(1 + \beta_с) + \beta_н + \beta_{пр} + t_{ЭВМ} С_{м\text{-}ч}], \quad (3.3)$$

где $Ф_{з/п}$ – фонд окладов разработчиков и других исполнителей работ, [руб.];

$\beta_Д$ – коэффициент надбавки, (примем, что $\beta_Д$ = 0,08);

$\beta_с$ – коэффициент отчислений на социальные нужды основной и дополнительной заработной платы, (примем $\beta_с$ = 0,3);

$\beta_н$ – коэффициент накладных расходов компании, разрабатывающей проект (можно принять $\beta_н$ = 0,65);

$\beta_{пр}$ – коэффициент других возможных трат, (принимаем $\beta_{пр}$ = 0,1);

$t_{ЭВМ}$ – машинное время, затрачиваемое на отладку программы, [ч.];

$С_{м–ч}$ – стоимость одного машино-часа работы ЭВМ, [руб.].

Фонд зарплат разработчиков $Ф_{з/п}$ [руб.] можно рассчитать по следующей формуле:

$$Ф_{З/П}=\frac{t_{\Sigma}}{t_{ср}}\cdot С_{МРОТ}\cdot \beta_{Т.С.}, \quad (3.4)$$

где t_{Σ} - сумма трудовых затрат, [чел./дн.] (результат расчета трудовых затрат на проект представлен в таблице 3.1 и составляет t_{Σ} = 75 чел./дн.);

$t_{ср}$ - количество дней в месяце, [дн.] (в среднем это 21 день);

$С_{МРОТ}$ – размер минимального размера труда, [руб./мес.] (согласно Федеральному закон от 19.06.2000 *N* 82-ФЗ (ред. от 06.12.2021) "О минимальном размере оплаты труда", минимальный размер оплаты труда по состоянию на май 2022 года составляет 13 890 рублей в месяц);

$\beta_{Т.С}$ - коэффициент, согласно тарифной сетке для программиста, согласно постановлению Минтруда РФ от 10.11.1992 *N* 30 (ред. от 31.10.1995) Об утверждении тарифно-квалификационных характеристик по общеотраслевым должностям служащих для программиста может устанавливаться тарифные разряд от 6 до 13. Для программиста с высшим образованием и опытом работы до 3 лет устанавливается 7 тарифный разряд с коэффициентом 2.20.

Таким образом рассчитаем фонд заработной платы, подставив значения в формулу (3.4):

$$Ф_{з/п}=\frac{75}{21}\cdot 13890\cdot 2.2=109135 \text{ руб.}$$

Себестоимость машино-часа работы ПК $С_{м-ч}$ [руб.] определяется по формуле:

$$С_{м-ч}=\frac{З_П+А+З_Э+З_Р+З_Н+З_М}{Ф_Д}, \quad (3.5)$$

где $З_п$ – затраты на оплату труда технического персонала с учетом всех отчислений, [руб.];

А – годовая сумма амортизации, [руб.];

$З_э$ – стоимость электроэнергии, [руб.];

$З_р$ – годовые затраты на ремонт и обслуживание оборудования, [руб.];

$З_м$ – стоимость материалов в год, [руб.];

$З_н$ – накладные расходы, [руб.];

$Ф_д$ – действительный годовой фонд времени работы комплекса средств автоматизации, [ч.].

В связи с отсутствием необходимости обслуживания компьютера, нет и необходимости рассчитывать затраты на оплату труда обслуживающего персонала, поэтому ($З_п = 0$).

Затраты на амортизацию по использованию комплекса средств автоматизации за год А [руб.] можно рассчитать по следующей формуле:

$$А=\frac{С_{КСА}\cdot Н_а}{100}, \quad (3.6)$$

где $С_{КСА}$ - стоимость оборудования, используемого при разработке, [руб.] (стоимость компьютера, необходимого для разработки системы составляет 40000 руб.);

$Н_а$ – норма годовой амортизации, ($Н_а = 25\%$).

Таким образом, годовые траты на амортизацию составят:

$$А=40000\cdot 0.25=10000 \text{ руб.}$$

Годовые затраты на электроэнергию $З_э$ [руб.] можно рассчитать по следующей формуле:

$$З_Э=W_А\cdot C_Э\cdot T_В, \quad (3.7)$$

где W_A - установленная мощность, [кВт] (установленное потребление аппаратного комплекса автоматизации W_A = 0.3 кВт/ч);

$С_Э$ - стоимость силовой электроэнергии, [руб./кВт], (для расчета возьмем цену 1 кВт/ч для промышленных предприятий, которая составляет $С_Э$ = 3.72 руб./кВт);

$T_В$ — время, в течение года, когда КСА потребляет электроэнергию, [ч] (для расчета возьмем количество рабочих дней в году, умноженное на количество часов в рабочем дне - 8).

Рассчитаем затраты на электроэнергию $З_э$ [руб.] согласно формуле (3.7):

$$З_э = 0.3 \cdot 3.72 \cdot 12 \cdot 21 \cdot 8 = 2249.8 \text{ руб.}$$

Затраты на текущий ремонт $З_р$ и на материалы $З_м$ за год учитываются на основании данных бухгалтерского учета, в сумме могут составлять от 4,5 до 10 % от стоимости компьютера, рассчитаем при коэффициенте равном 8%:

$$З_р+З_м = 40000 \cdot 0{,}08 = 3200 \text{ руб.}$$

К накладным расходам $З_Н$ относятся трудозатраты административно управленческого персонала, амортизация площадей и затраты на содержание, отопление, освещение и другие затраты. Для расчета примем что они равны 20% стоимости компьютера и рассчитаем:

$$З_Н = 40000 \cdot 0.20 = 8000 \text{ руб.}$$

Годовой фонд времени $Ф_Д$ [ч.] рассчитывается исходя из времени работы, времени обслуживания и ремонтов:

$$Ф_д = S \cdot h \cdot D - T_{ПР}, \quad (3.8)$$

где S - длительность смены, [ч], (S = 8 ч.);

h - число смен в сутках, (h = 1);

D - число рабочих дней в году,[дн.], (D = 252 дн.);

$T_{ПР}$ - время на ремонт и техническое обслуживание, [ч.], ($T_{ПР}$ = 96 ч.).

Рассчитаем годовой фонд времени $Ф_Д$ [ч.]:

$$Ф_д = 8 \cdot 1 \cdot 252 - 96 = 1920 \text{ ч.}$$

Рассчитаем стоимость машино-часа $С_{м-ч}$ согласно формуле (3.5):

$$C_{м\text{-}ч}=\frac{0+10000+2249.8+3200+8000}{1920}=12.21 \text{ руб.}$$

Таким образом рассчитаем затраты на разработку автоматизированной системы $К_П$ согласно формуле (3.3):

$$К_П = 103315 \cdot (1.08 \cdot 1.3 + 0.65 + 0.1) + 71 \cdot 12.21 = 235943.7 \text{ руб.}$$

3.3 Расчет минимальной цены разработки АС

Рассчитаем минимальную предполагаемую цену разработки автоматизированной системы, она включает затраты на разработку системы $К_П$ и минимальной необходимой прибыли, которая позволила бы профинансировать компанию, занимающуюся разработкой после всех обязательных платежей и налогов.

Если принять уровень рентабельности для разработчика равным 20 %, то выручка составит:

$$235943.7 \cdot 0.2 = 47188.74 \text{ руб.}$$

Поскольку автоматизированная система создается для заказчика, продажа будет осуществляться по договорной цене. Таким образом то минимальная стоимость автоматизированной системы вместе с НДС 20% составит:

$$(235943.7 + 47188.74) \cdot 1.2 = 339758 \text{ руб.}$$

3.4 Расчет затрат на внедрение АС

Единовременные затраты на внедрение системы К [руб.] определяются как сумма затрат на покупку системы и размера капитальных затрат $К_К$ [руб.].

Сумма капитальных затрат вычисляется по формуле:

$$К_к = К_{ктс} + К_м + К_{инв} + К_{зд} + К_{ос} + К_{тр} + К_{соп} – К_{выс}, \quad (3.9)$$

где $К_{ктс}$ – стоимость комплекса технических средств, [руб.] (ввиду того, что разрабатываемая система не ограничена по количеству машин в кластере, а

следовательно по мощности, возьмем стоимость минимальной конфигурации, $К_{ктс}$ = 20000 руб.);

$К_м$, $К_{инв}$, $К_{зд}$ – стоимость монтажа, используемого инвентаря и трат на перепланировку здания для установки комплекса технических средств, [руб.]. Так как монтирование не подразумевает собой реконструкцию здания и отсутствует необходимость использования дополнительного инвентаря, $К_{инв}$ = 0 руб. и $К_{зд}$ = 0 руб., а $К_м$ обозначим как 10% от цены КТС, $К_М$ = 2000 руб.;

$К_{ОС}$ - оборотные средства , [руб.] ($К_{ОС}$ = 0 руб.);

$К_{ТР}$ – затраты на транспорт, [руб.] (положим $К_{ТР}$ равно 10% от стоимости КТС, $К_{ТР}$ = 2000 руб.);

$К_{СОП}$ - ориентировочная стоимость системы стандартного обеспечения комплекса технических средств, [руб.] (равна 10% от стоимости КТС, $К_{СОП}$ = 2000 руб.);

$К_{ВЫС}$ – высвобожденные средства, [руб.] ($К_{ВЫС}$ = 0 руб.).

Рассчитаем капитальные затраты $К_К$ по формуле 3.9:

$$К_К = 20000 + 2000 + 2000 + 2000 = 26000 \text{ руб.}$$

Таким образом единовременные затраты К на введение системы будут равны:

$$К = 339758 + 26000 = 365758 \text{ руб.}$$

Чтобы рассчитать текущие траты на эксплуатацию системы $З_{тек}$ [руб.], можно воспользоваться следующей формулой, которая учитывает все решаемые в течение года задачи при условии функционирования АС:

$$З_{тек} = \sum_{i=1}^{n} З_i + З_{сист}, \quad (3.10)$$

где $З_i$ – затраты на решение *i*-й задачи, [руб./год];

n – количество задач, решаемых за год;

$З_{сист}$ – общесистемные затраты, [руб./год], ($З_{сист}$ = 6000 руб./год.).

Затраты на решение задачи $З_i$ [руб./год.] рассчитываются по формуле:

$$З_i = З^i_{п} + З^i_{м.в.}, \quad (3.11)$$

где $З^{i}_{п}$ - затраты на зарплаты работникам, решающим данную задачу, [руб./год];

$З^{i}_{м.в.}$ - стоимость работы КСА для данной задачи [руб./год].

Затраты на работу КСА для конкретной задачи $З^{i}_{м.в}$ можно рассчитать по следующей формуле:

$$З^{i}_{м.в.} = С_{м\text{-}ч} \cdot t_i^{КСА}, \quad (3.12)$$

где $С_{м\text{-}ч}$ – стоимость часа работы КСА, [руб./ч.], ($С_{м\text{-}ч}$ = 8 руб./ч.);

$t_i^{КСА}$ – время, потраченное на выполнение задачи при условии использования КСА, [ч.] (задача по мониторингу виртуальных машин и по запуску резервного копирования в ручном режиме занимает 2 часа работы специалиста в месяц, поэтому $t_i^{КСА}$ = 28 ч.).

Рассчитаем затраты на работу КСА $З^{i}_{м.в}$ согласно формуле (3.12):

$$З^{1}_{м.в} = 8 * 24 = 192 \text{ руб./год.}$$

При стоимости часа работы специалиста 360 руб., затраты на решение задачи будут составлять

$$З_1 = 360 * 24 + 192 = 8832 \text{ руб./год.}$$

Таким образом текущие затраты на эксплуатацию системы $З_{тек}$ равны:

$$З_{тек} = 8832 + 6000 = 14832 \text{ руб./год.}$$

3.5 Расчет экономической эффективности

Для расчета экономической эффективности от внедрения автоматизированной системы для создания и учета резервных копий в кластере виртуальных машин необходимо проанализировать существующие на рынке системы с аналогичным функционалом. Такими системами являются «*Veeam Backup* & *Replication*» и «*Acronis Backup* & *Recovery*». Данные системы предоставляют модель подписки на программное обеспечение.

Стоимость лицензии «*Veeam Backup* & *Replication*» на 12 месяцев будет составлять 145414.80 руб. для пакета из 10 экземпляров без учета затрат на

внедрение. Каждый экземпляр предназначен для защиты 1 виртуальной машины или одного физического или облачного сервера.

Стоимость лицензии «*Acronis Backup* & *Recovery*» на 12 месяцев составляет от 26900 руб. до 36465 руб. за лицензию для одной виртуальной или физической машины при покупке различных пакетов лицензий без учета затрат на внедрение. Каждый экземпляр предназначен для установки на одну машину.

При условии, что в кластере находится 10 машин и, так как разработанная система не зависит от количества машин в кластере, окупаемость внедрения данной системы составит 3 года, из чего можно сделать вывод, что внедрение автоматизированной системы является целесообразной.

Заключение

В ходе выполнения дипломного проекта 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 освещенности рабочего места в помещении.

В экономическом 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.

Таким образом, задание на дипломное проектирование выполнено в полном объеме.

Список использованных источников

1. Питкевич П. 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*://*express*42.*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. СанПиН 2.2.4.335916 «Санитарно-эпидемиологические требования к физическим факторам на рабочих местах», утвержденные Постановлением Главного государственного санитарного врача РФ от 21.06.2016 № 81.

7. СанПиН 2.2.2/2.4.134003 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» (утверждены Постановлением Главного государственного санитарного врача РФ от 03.06.2003 № 118).

8. Справочная книга для проектирования электрического освещения. / Под ред. Г.Б. Кнорринга. – Л.: Энергия, 1976.

9. Самгин Э.Б. Освещение рабочих мест. – М.: МИРЭА, 1989. – 186с.

10. Куренкова В.П. Технико-экономическое обоснование создания автоматизированных систем и программных продуктов (методические указания), 2006.

Приложение А

(обязательное)

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

Листинг программы

Код 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 COL-
LATE=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      COL-
LATE=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      COL-
LATE=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,   AU-
TO_INCREMENT=21;


--
```

```
-- AUTO_INCREMENT for table `backup`

--

ALTER TABLE `backup`

  MODIFY   `ID`   int(11)   NOT   NULL   AUTO_INCREMENT,   AU-
TO_INCREMENT=178;

-

-- AUTO_INCREMENT for table `failed_jobs`

--

ALTER TABLE `failed_jobs`

  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT, AU-
TO_INCREMENT=7;


--

-- AUTO_INCREMENT for table `jobs`

--

ALTER TABLE `jobs`

  MODIFY `id` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT, AU-
TO_INCREMENT=1696;


--

-- AUTO_INCREMENT for table `machine`

--

ALTER TABLE `machine`

  MODIFY   `ID`   int(11)   NOT   NULL   AUTO_INCREMENT,   AU-
TO_INCREMENT=51;


--

-- AUTO_INCREMENT for table `migrations`

--

ALTER TABLE `migrations`

  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT, AU-
TO_INCREMENT=5;
```

```
--

-- AUTO_INCREMENT for table `plan`

--

ALTER TABLE `plan`

  MODIFY  `ID_PLAN`  int(11)  NOT  NULL  AUTO_INCREMENT,  AU-
TO_INCREMENT=37;


--

-- AUTO_INCREMENT for table `status`

--

ALTER TABLE `status`

  MODIFY  `ID`  int(11)  NOT  NULL  AUTO_INCREMENT,  AU-
TO_INCREMENT=8;


--

-- AUTO_INCREMENT for table `storage`

--

ALTER TABLE `storage`

  MODIFY  `ID_STORAGE`  int(11)  NOT  NULL  AUTO_INCREMENT,  AU-
TO_INCREMENT=6;


--

-- AUTO_INCREMENT for table `user`

--

ALTER TABLE `user`

  MODIFY  `ID`  int(11)  NOT  NULL  AUTO_INCREMENT,  AU-
TO_INCREMENT=6;


--

-- Constraints for dumped tables

-- Constraints for table `authorization`
```

```
--

ALTER TABLE `authorization`

  ADD CONSTRAINT `USER_ID` FOREIGN KEY (`USER`) REFERENCES `us-
er` (`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`) REFER-
ENCES `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   (`US-
ER_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 */;
```

Листинг модулей клиентской части приложения:

Модуль *Machines.vue*

```
<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 маши-
ны</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 ${sta-
tuses[machine.status-1].style}`">
                    <v-flex md4>
                        <div class="caption grey--text">Имя маши-
ны</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">
                                        {{ ma-
chine.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">
                                        {{
                                            ma-
chine.storage.TITLE_STORAGE.length > 10 ?
                                                ma-
chine.storage.TITLE_STORAGE.slice(0, 10) + '..' :
                                                ma-
chine.storage.TITLE_STORAGE
                                        }}
                                    </span>
                                </p>
                            </div>
                        </div>
                        <div class=""
                             style="position: absolute; top:0;
transition: all .3s linear;"
                             v-if="machine.plan && showBut-
tonsMachine === 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 !im-
portant;" class="caption text--lighten-1 grey--text  my-0">
                                Последний бекап</p>
                            <p style="font-size: 11px !im-
portant;"
                               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: ma-
chine["ID_MODE_BACKUP"],
                        days_backup: machine.plan ? ma-
chine.plan['DAYS_BACKUP'] : null,
                        title_mode_backup: ma-
chine["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.$confirm('Для данной маши-
ны нет активного плана, создать его сейчас?');
                if (res) {
                    this.createPlan(machine);
                }
            } else {
                // подтверждение
                const res = await this.$confirm(`Начать бекап
сейчас для машины ${machine.title}? <br>(хранилище для бекапа бу-
дет 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.$notify({
                                        group: 'foo',
                                        type: 'success',
                                        title: 'Успешно',
                                        text: 'В ближайшее время за-
пустится бекап'
                                    });
                                    this.$emit('closeModal');
                                } else if (r.data.errorMessage ===
'today already') {
                                    this.$notify({
                                        group: 'foo',
                                        type: 'error',
                                        title: 'Ошибка',
                                        text: 'Сегодня бэкап уже вы-
полнялся'
                                    });
                                } else {
                                    this.$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 выпол-
нение 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>
```

Модуль *Storages.vue*

```
<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 ${stor-
age.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 ? 'Уда-
ленное' : '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>
```

Модуль *Plans.vue*

```
<template>
    <v-container class="backups">
        <v-container class="my-0">
            <h2 class="subheading grey--text">
                Активные планы
            </h2>
            <div
                style="display: flex; flex-direction: column;
width: 100%;"
                v-for="plan in plans.activePlans"
                :key="plan.ID_PLAN"
                class="my-4 elevation-2"
            >
                <v-card flat>
                    <v-layout row wrap :class="`pt-3 pl-3 pr-3
pb-1 project`">
                        <!--${statuses[machine.status-1].style}`-
->
                        <v-flex md3>
                            <div class="caption grey--text">Имя
машины</div>
                            <div class="mt-1">
                                {{ plan.machine.NAME }}
                                <b></b>
                            </div>
                        </v-flex>
                        <v-flex md2>
                            <div class="caption grey--
text">Хранилище</div>
```

```
                                <div class="mt-1">
                                    {{ plan.storage.TITLE_STORAGE }}
                                    <b></b>
                                </div>
                            </v-flex>
                            <v-flex md2>
                                <div class="caption grey--
text">Период бекапа</div>
                                <div class="mt-1">
                                    {{ plan.DAYS_BACKUP }} дней
                                    <b></b>
                                </div>
                            </v-flex>
                            <v-flex md2>
                                <div class="caption grey--
text">Сохраняемых бекапов</div>
                                <div class="mt-1">
                                    {{ plan.COUNT_BACKUPS }}
                                    <b></b>
                                </div>
                            </v-flex>
                            <v-flex md1>
                                <div class="right" ti-
tle="Редактировать">
                                    <v-btn
                                        @click="editPlanModal(plan)"
                                        class="mx-0 mt-0"
                                        left
                                        elevation="2"
                                        fab
                                        flat
                                        rounded
```

```
                                    >
                                        <v-icon>
                                            edit
                                        </v-icon>
                                    </v-btn>
                                </div>
                            </v-flex>
                            <v-flex md1>
                                <div style="margin: 0 50%; transform:
translateX(-50%)"
                                     :title="plan.ACTIVE ? 'Остано-
вить план' : 'Запустить план'">
                                    <v-btn
                                        @click="editPlanActive(plan,
false)"
                                        class="mx-0 mt-0"
                                        left
                                        elevation="2"
                                        fab
                                        flat
                                        rounded
                                    >
                                        <v-icon>
                                            {{ plan.ACTIVE ? 'stop' :
'play' }}
                                        </v-icon>
                                    </v-btn>
                                </div>
                            </v-flex>
                        </v-layout>
                        <v-divider></v-divider>
                    </v-card>
```

```
                <v-expansion-panel
                    class="elevation-0"
                >
                    <v-expansion-panel-content>
                        <div slot="header"
                             class="py-1 elevation-0"
                        >
                            <div class="text--darken-4 grey--
text">
                                Список бекапов (<b>{{
plan.backups_list.length }}</b>):
                            </div>
                        </div>
                        <v-card
                            v-if="plan.backups_list.length"
                            flat v-for="(backup,i) in
plan.backups_list"
                            :key="backup.ID"
                        >
                            <v-layout row wrap class="pl-4 pt-3
pb-3">
                                <v-flex md3>
                                    <div class="caption grey--
text">Название файла</div>
                                    <div class="mt-1">{{ back-
up.NAME }}</div>
                                </v-flex>
                                <v-flex md2>
                                    <div class="caption grey--
text">Дата</div>
                                    <div class="mt-1">{{ format-
Date(backup.TIME) }}</div>
                                </v-flex>
                                <v-flex md5>
```

```
                                    <div class="caption grey--
text">UUID</div>
                                    <div class="mt-1">{{ back-
up.UUID_BACKUP }}</div>
                                </v-flex>


<!--                                <v-flex md2-->
<!--
@click="restoreBackup(backup)"-->
<!--                                >-->
<!--                                    <v-btn class="primary"
style="width: 80%;">-->
<!--                                        <div id="circularG"
v-if="backup.RESTORING">-->
<!--                                            <div
id="circularG_1" class="circularG"></div>-->
<!--                                            <div
id="circularG_2" class="circularG"></div>-->
<!--                                            <div
id="circularG_3" class="circularG"></div>-->
<!--                                            <div
id="circularG_4" class="circularG"></div>-->
<!--                                            <div
id="circularG_5" class="circularG"></div>-->
<!--                                            <div
id="circularG_6" class="circularG"></div>-->
<!--                                            <div
id="circularG_7" class="circularG"></div>-->
<!--                                            <div
id="circularG_8" class="circularG"></div>-->
<!--                                        </div>-->
<!--                                        <div v-
else>Восстановить</div>-->
<!--                                    </v-btn>-->
<!--                                </v-flex>-->
```

```
                                    <v-flex
                                        @click="deleteBackup(backup)"
                                        md2>
                                        <v-btn
class="error">Удалить</v-btn>
                                    </v-flex>

                                </v-layout>
                                <v-divider></v-divider>
                            </v-card>
                        </v-expansion-panel-content>
                    </v-expansion-panel>
                </div>

                <br>
                <h2 class="subheading grey--text">
                    Отключенные планы
                </h2>
                <div
                    style="display: flex; flex-direction: column;
width: 100%;"
                    v-for="plan in plans.turnedOffPlans"
                    :key="plan.ID_PLAN"
                    class="my-4 elevation-2 disabled-plan"
                >
                    <v-card flat>
                        <v-layout row wrap :class="`pt-3 pl-3 pr-3
pb-1 project`">
                            <!--${statuses[machine.status-1].style}`-
->
                            <v-flex md3>
```

```
                                <div class="caption grey--text">Имя
машины</div>
                                <div class="mt-1">
                                    {{ plan.machine.NAME }}
                                    <b></b>
                                </div>
                            </v-flex>
                            <v-flex md2>
                                <div class="caption grey--
text">Хранилище</div>
                                <div class="mt-1">
                                    {{ plan.storage.TITLE_STORAGE }}
                                    <b></b>
                                </div>
                            </v-flex>
                            <v-flex md2>
                                <div class="caption grey--
text">Период бекапа</div>
                                <div class="mt-1">
                                    {{ plan.DAYS_BACKUP }} дней
                                    <b></b>
                                </div>
                            </v-flex>
                            <v-flex md2>
                                <div class="caption grey--
text">Сохраняемых бекапов</div>
                                <div class="mt-1">
                                    {{ plan.COUNT_BACKUPS }}
                                    <b></b>
                                </div>
                            </v-flex>
                            <v-flex md1>
```

```
                                <div class="right" title="Удалить
план">
                                    <v-btn
                                        @click="deletePlan(plan)"
                                        class="mx-0 mt-0"
                                        left
                                        elevation="2"
                                        fab
                                        flat
                                        rounded
                                    >
                                        <v-icon>
                                            delete
                                        </v-icon>
                                    </v-btn>
                                </div>
                            </v-flex>
                            <v-flex md1>
                                <div style="margin: 0 50%; transform:
translateX(-50%)"
                                     :title="plan.ACTIVE ? 'Остано-
вить план' : 'Запустить план'">
                                    <v-btn
                                        @click="editPlanActive(plan,
true)"
                                        class="mx-0 mt-0"
                                        left
                                        elevation="2"
                                        fab
                                        flat
                                        rounded
                                    >
```

```
                                    <v-icon>start</v-icon>
                                </v-btn>
                            </div>
                        </v-flex>
                    </v-layout>
                    <v-divider></v-divider>
                </v-card>
                <v-expansion-panel
                    class="elevation-0"
                >
                    <v-expansion-panel-content>
                        <div slot="header"
                             class="py-1 elevation-0"
                        >
                            <div class="text--darken-4 grey--
text">
                                Список бекапов (<b>{{
plan.backups_list.length }}</b>):
                            </div>
                        </div>
                        <v-card
                            v-if="plan.backups_list.length"
                            flat v-for="(backup,i) in
plan.backups_list"
                            :key="backup.ID"
                        >
                            <v-layout row wrap class="pl-4 pt-3
pb-3">
                                <v-flex md3>
                                    <div class="caption grey--
text">Название файла</div>
                                    <div class="mt-1">{{ back-
up.NAME }}</div>
```

```
                                    </v-flex>
                                    <v-flex md2>
                                        <div class="caption grey--
text">Дата</div>
                                        <div class="mt-1">{{ format-
Date(backup.TIME) }}</div>
                                    </v-flex>
                                    <v-flex md5>
                                        <div class="caption grey--
text">UUID</div>
                                        <div class="mt-1">{{ back-
up.UUID_BACKUP }}</div>
                                    </v-flex>

                                    <v-flex
                                        @click="deleteBackup(backup)"
                                        md2>
                                        <v-btn
class="error">Удалить</v-btn>
                                    </v-flex>

                                </v-layout>
                                <v-divider></v-divider>
                            </v-card>
                        </v-expansion-panel-content>
                    </v-expansion-panel>
                </div>
            </v-container>
            <EditPlanDialog
                v-if="selectedPlanEdit"
                :plan="selectedPlanEdit"
                @closeModal="closeModal"
```

```
        >

        </EditPlanDialog>
    </v-container>
</template>

<script>
import axios from "axios";
import EditPlanDialog from "../components/EditPlanDialog";

export default {
    name: "Plans",
    components: {EditPlanDialog},
    data() {
        return {
            selectedPlanEdit: null,
            plans: [],
            editPlan: null
        }
    },
    mounted() {
        this.editPlan = this.$route.params.plan;
        this.fetchPlans();
        setInterval(() => {
            this.fetchPlans();
        }, 5000)
    },
    methods: {
        fetchPlans() {
            axios.get(process.env.VUE_APP_HOST + '/list_plans', {
```

```
                headers: {
                    token:
JSON.parse(localStorage.getItem('token'))
                }
            })
                .then(r => {
                    this.plans = r.data;
                    if (this.editPlan) {
                        const neededPlan =
this.plans.activePlans.find(el => el.ID_PLAN === this.editPlan);
                        this.editPlan = null;
                        this.editPlanModal(neededPlan);
                    }
                });
        },
        async deletePlan(plan) {
            if (!plan.ACTIVE) {
                const res = await this.$confirm('Удалить неактив-
ный план и все сохраненные в нем бекапы?');
                if (res) {
                    axios.post(process.env.VUE_APP_HOST +
'/delete_plan', {
                        id_plan: plan.ID_PLAN,
                    }, {
                        headers: {
                            token:
JSON.parse(localStorage.getItem('token'))
                        }
                    })
                        .then(r => {
                            if (r.data.status === 'ok') {
                                this.$notify({
```

```
                                group: 'foo',
                                type: 'success',
                                title: 'Успешно',
                                text: 'Удалено'
                            });
                        } else {
                            this.$notify({
                                group: 'foo',
                                type: 'error',
                                title: 'Ошибка',
                                text: r.data.errorMessage
                            });
                        }
                        this.fetchPlans();
                    });
                }
            } else {
                const res = await this.$confirm('План активен,
необходимо его сначала приостановить, приостановить?');
                if (res) {
                    axios.post(process.env.VUE_APP_HOST +
'/edit_active_plan', {
                        plan_id: plan.ID_PLAN,
                        new_status: false
                    }, {
                        headers: {
                            token:
JSON.parse(localStorage.getItem('token'))
                        }
                    })
                        .then(r => {
                            this.fetchPlans();
```

```
                    });
                }
            }

        },


        editPlanModal(plan) {
            this.selectedPlanEdit =
JSON.parse(JSON.stringify(plan));
        },


        async editPlanActive(plan, status) {
            const res = await this.$confirm(`${status ? 'Возобно-
вить' : 'Приостановить'} выполнение данного плана?`);
            if (res) {
                axios.post(process.env.VUE_APP_HOST +
'/edit_active_plan', {
                    plan_id: plan.ID_PLAN,
                    new_status: status
                }, {
                    headers: {
                        token:
JSON.parse(localStorage.getItem('token'))
                    }
                })
                    .then(r => {
                        this.fetchPlans();
                    });
            }
        },
        async deleteBackup(backup) {
```

```
            // ID
            // UUID_BACKUP
            const res = await this.$confirm('Удалить бэкап с
хранилища?');
            if (res) {
                axios.post(process.env.VUE_APP_HOST +
'/delete_backup', {
                    uuid_for_delete: backup.UUID_BACKUP,
                    id_backup: backup.ID
                }, {
                    headers: {
                        token:
JSON.parse(localStorage.getItem('token'))
                    }
                })
                    .then(r => {
                        if (r.data.status === 'ok') {
                            this.$notify({
                                group: 'foo',
                                type: 'success',
                                title: 'Успешно',
                                text: 'Удалено'
                            });
                        } else {
                            this.$notify({
                                group: 'foo',
                                type: 'error',
                                title: 'Ошибка',
                                text: 'Ошибка удаления'
                            });
                        }
                        this.fetchPlans();
```

```
                });
            }
        },
        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 '-';
            }
        },
        ucFirst(str) {
            if (!str) return str;
```

```
            return str[0].toUpperCase() + str.slice(1);
        },
        closeModal() {
            this.selectedPlanEdit = null;
            this.fetchPlans();
        }
    },
    computed: {},
}
</script>

<style scoped>

@keyframes bounce_circularG {
    0% {
        transform: scale(1);
    }

    100% {
        transform: scale(.3);
    }
}

@-o-keyframes bounce_circularG {
    0% {
        -o-transform: scale(1);
    }

    100% {
        -o-transform: scale(.3);
```

```
    }
}

@-ms-keyframes bounce_circularG {
    0% {
        -ms-transform: scale(1);
    }

    100% {
        -ms-transform: scale(.3);
    }
}

@-webkit-keyframes bounce_circularG {
    0% {
        -webkit-transform: scale(1);
    }

    100% {
        -webkit-transform: scale(.3);
    }
}

@-moz-keyframes bounce_circularG {
    0% {
        -moz-transform: scale(1);
    }

    100% {
        -moz-transform: scale(.3);
```

```
    }
}

.disabled-plan {
    opacity: .55;
}

.disabled-plan:hover {
    opacity: 1;
}

</style>
```

Листинг модулей серверной части приложения:

Модуль *BackProcessing.php*

```
<?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 - аутинтентификация
     * @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, $pass-
word);
            if (!$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;
    }

    // готово
    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, $stor-
ageUUID)
    {
        // снепшот
        $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';
    }

    // ок
    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';
    }

    // ок
    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);
        }
    }
}
```

Модуль *ForceBackup.php*

```
<?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, Serializes-
Models;
    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);

        }


    }

}
```

Модуль *LoginController.php*

```
<?php


namespace App\Http\Controllers;


use phpseclib\Net\SSH2;
use Exception;
use Illuminate\Support\Facades\Log;



use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;


class LoginController extends Controller
{
    public function index()
    {
    }


    /**
     * @throws Exception
     */
    private function checkAccount($logRes, $host, $login, $pass-
word)
    {
        if (!$logRes) {
            return json_encode(['status' => 'error', 'er-
rorMessage' => 'wrong credentials']);
        }


        if (!DB::table('user')
```

```
            ->where('HOST', '=', trim($host))
            ->where('LOGIN', '=', trim($login))
            ->where('PASSWORD', '=', trim($password))
            ->get()
            ->first()) {
            DB::table('user')
                ->insert([
                    'HOST' => $host,
                    'LOGIN' => $login,
                    'PASSWORD' => $password,
                ]);

            try {
                (new \App\Http\Controllers\StoragesController())->parseStorages();
            } catch (Exception $e) {
                Log::info(json_encode($e));
            }
            try {
                (new \App\Http\Controllers\BackProcessing)->parseCurrentMachinesStates();
            } catch (Exception $e) {
                Log::info(json_encode($e));
            }
            try {
                (new \App\Http\Controllers\StoragesController)->parseFreeSpaceStorages();
            } catch (Exception $e) {
                Log::info(json_encode($e));
            }
```

```
        }
        $user = DB::table('user')
            ->where('HOST', '=', trim($host))
            ->where('LOGIN', '=', trim($login))
            ->where('PASSWORD', '=', trim($password))
            ->get()
            ->first();

        if (!$user) return json_encode(['status' => 'error', 'er-
rorMessage' => 'try again']);
        $key = hash('sha256', $login . time());
        Log::info('$key:');
        Log::info($key);
        DB::table('authorization')
            ->insert([
                'TOKEN' => $key,
                'ACTIVE' => true,
                'USER' => $user->ID
            ]);
        return json_encode($key);
    }

    /**
     * @throws Exception
     */
    public function login(Request $request)
    {
        $data = $request->all();
        $host = $data['host'];
        $login = $data['login'];
        $password = $data['password'];
```

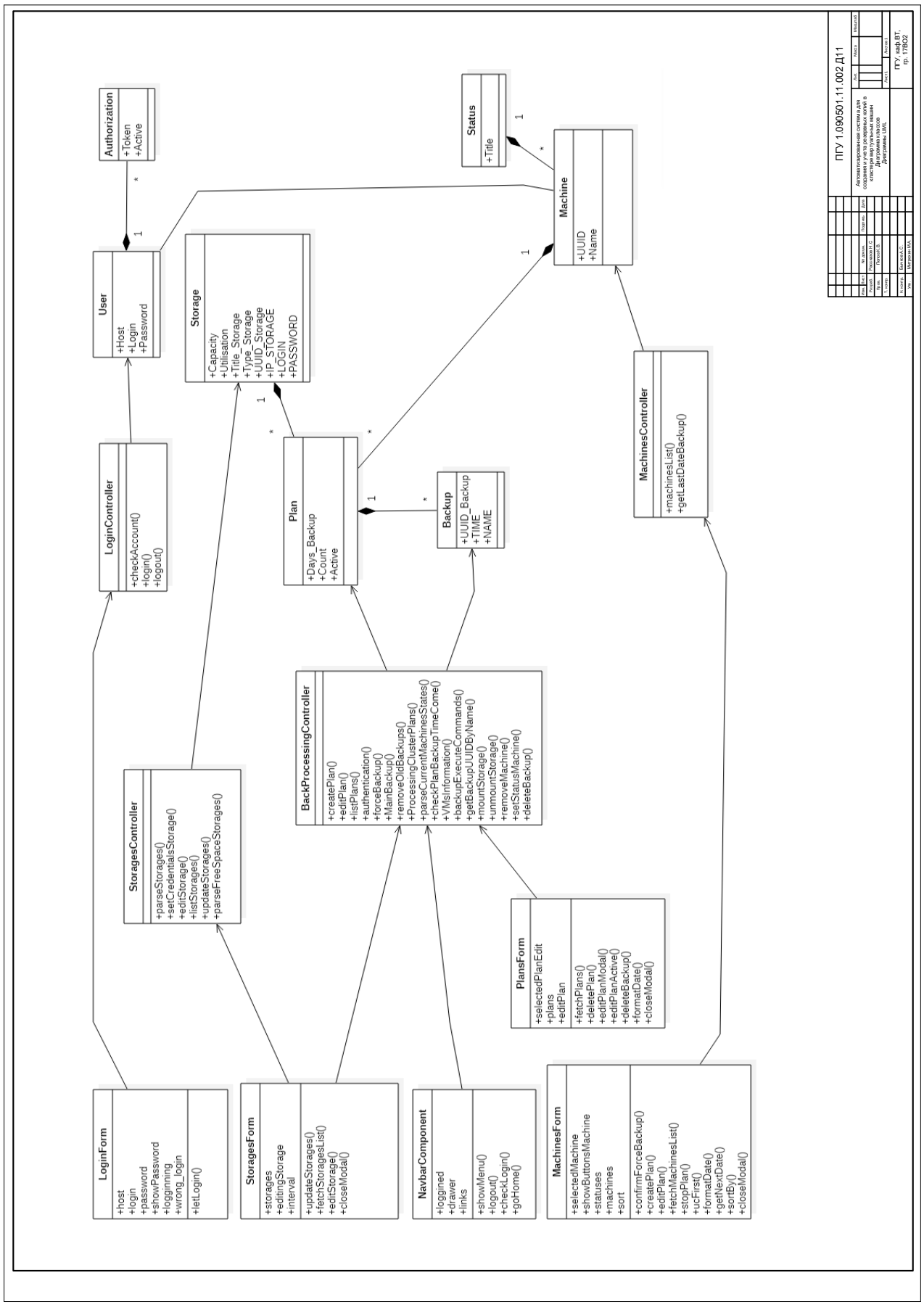
```
        $ssh = (new \App\Http\Controllers\BackProcessing)
            ->authentication(
                $host,
                $login,
                $password);

        return $this->checkAccount($ssh, $host, $login, $pass-
word);

    }

    public function logout(Request $request)
    {
        $token = $request->header('token');
        Log::info('logout token');
        Log::info($token);
        Log::info(DB::table('authorization')
            ->where('TOKEN', '=', $token)
            ->update(['ACTIVE' => 0]));
        return json_encode(['status' => 'ok']);
    }
}
```

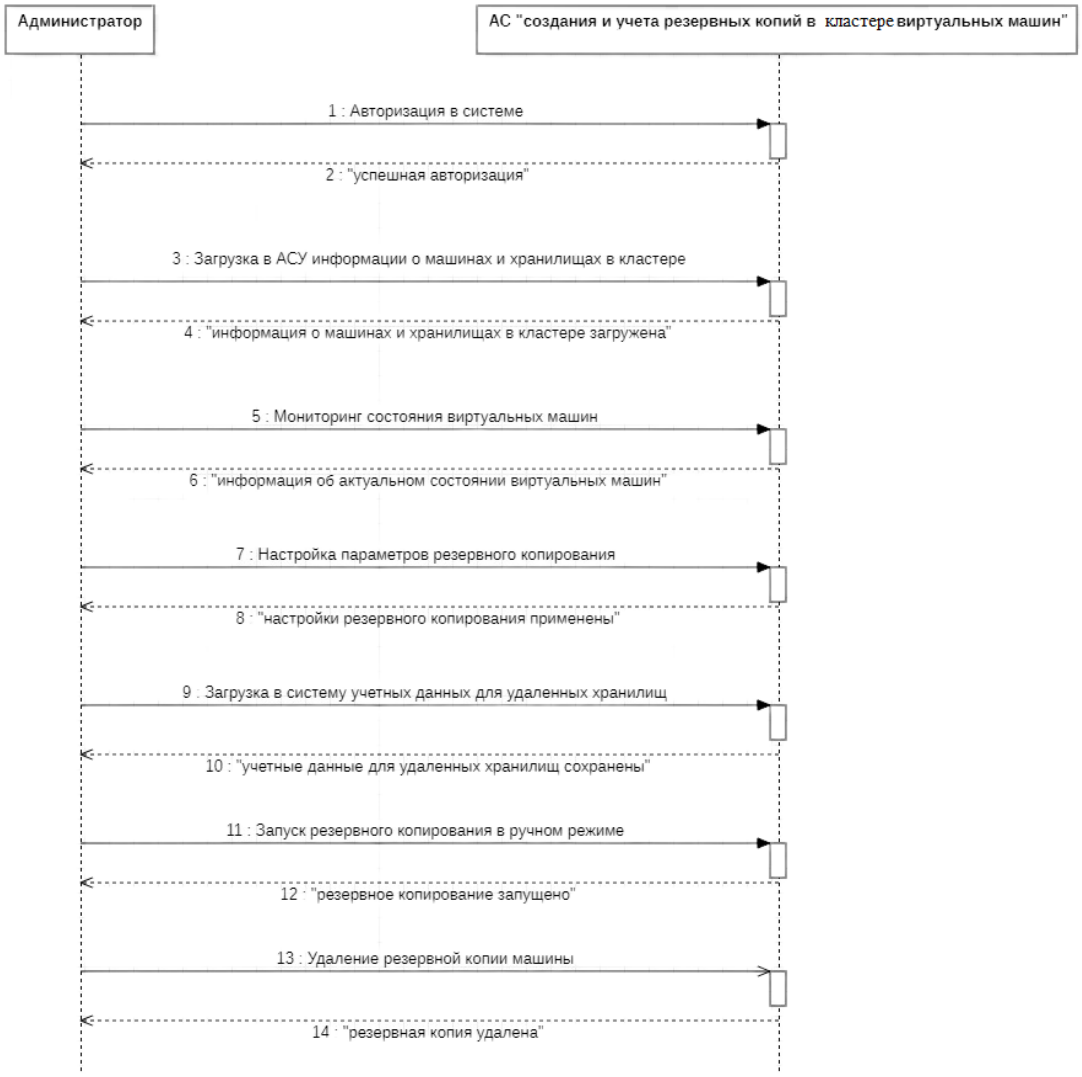
Приложение Б

(обязательное)

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

Диаграммы *UML*


Authorization
+Token
+Active
User
+Host
+Login
+Password
Status
+Title
Machine
+UUID
+Name
Storage
+Capacity
+Utilisation
+Title_Storage
+Type_Storage
+UUID_Storage
+IP_STORAGE
+LOGIN
+PASSWORD
LoginController
+checkAccount()
+login()
+logout()
Plan
+Days_Backup
+Count
+Active
Backup
+UUID_Backup
+TIME
+NAME
MachinesController
+machinesList()
+getLastDateBackup()
StoragesController
+parseStorages()
+setCredentialsStorage()
+editStorage()
+listStorages()
+updateStorages()
+parseFreeSpaceStorages()
BackProcessingController
+createPlan()
+editPlan()
+listPlans()
+authentication()
+forceBackup()
+MainBackup()
+removeOldBackups()
+ProcessingClusterPlans()
+parseCurrentMachinesStates()
+checkPlanBackupTimeCome()
+VMsInformation()
+backupExecuteCommands()
+getBackupUUIDByName()
+mountStorage()
+unmountStorage()
+removeMachine()
+setStatusMachine()
+deleteBackup()
PlansForm
+selectedPlanEdit
+plans
+editPlan
+fetchPlans()
+deletePlan()
+editPlanModal()
+editPlanActive()
+deleteBackup()
+formatDate()
+closeModal()
LoginForm
+host
+login
+password
+showPassword
+logginning
+wrong_login
+letLogin()
StoragesForm
+storages
+editingStorage
+interval
+updateStorages()
+fetchStoragesList()
+editStorage()
+closeModal()
NavbarComponent
+loggined
+drawer
+links
+showMenu()
+logout()
+checkLogin()
+goHome()
MachinesForm
+selectedMachine
+showButtonsMachine
+statuses
+machines
+sort
+confirmForceBackup()
+createPlan()
+editPlan()
+fetchMachinesList()
+stopPlan()
+ucFirst()
+formatDate()
+getNextDate()
+sortBy()
+closeModal()
ПГУ 1.090501.11.002 Д11
Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин
Диаграмма UML
ПГУ, ИФ ВТ, гр. 17802



Изм.	Лист	№ докум.	Подпись	Дата	Разраб.	Рассказов Н.С.			
					Пров.	Попов К.В.			
					Т. контр.				
					Н. контр.	Бычков А.С.			
					Утв.	Митрохин М.А.			

ПГУ 1.090501.11.003 Д14

Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин
Диаграммы последовательности
Диаграммы UML

Лит.	Масса	Масштаб
Лист 1	Листов 1	

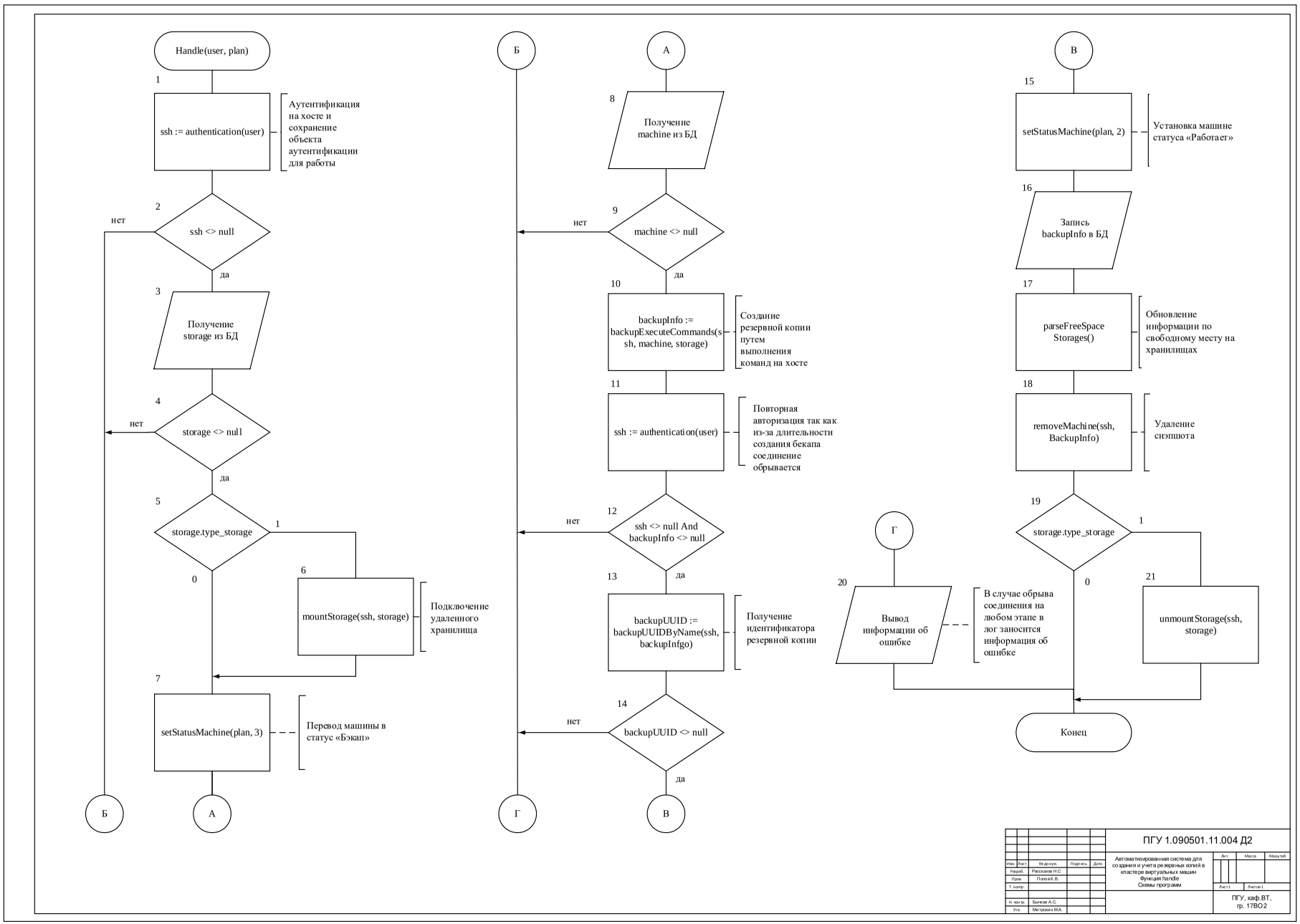
ПГУ, каф.ВТ, гр. 17ВО2

Приложение В

(обязательное)

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

Схемы программ

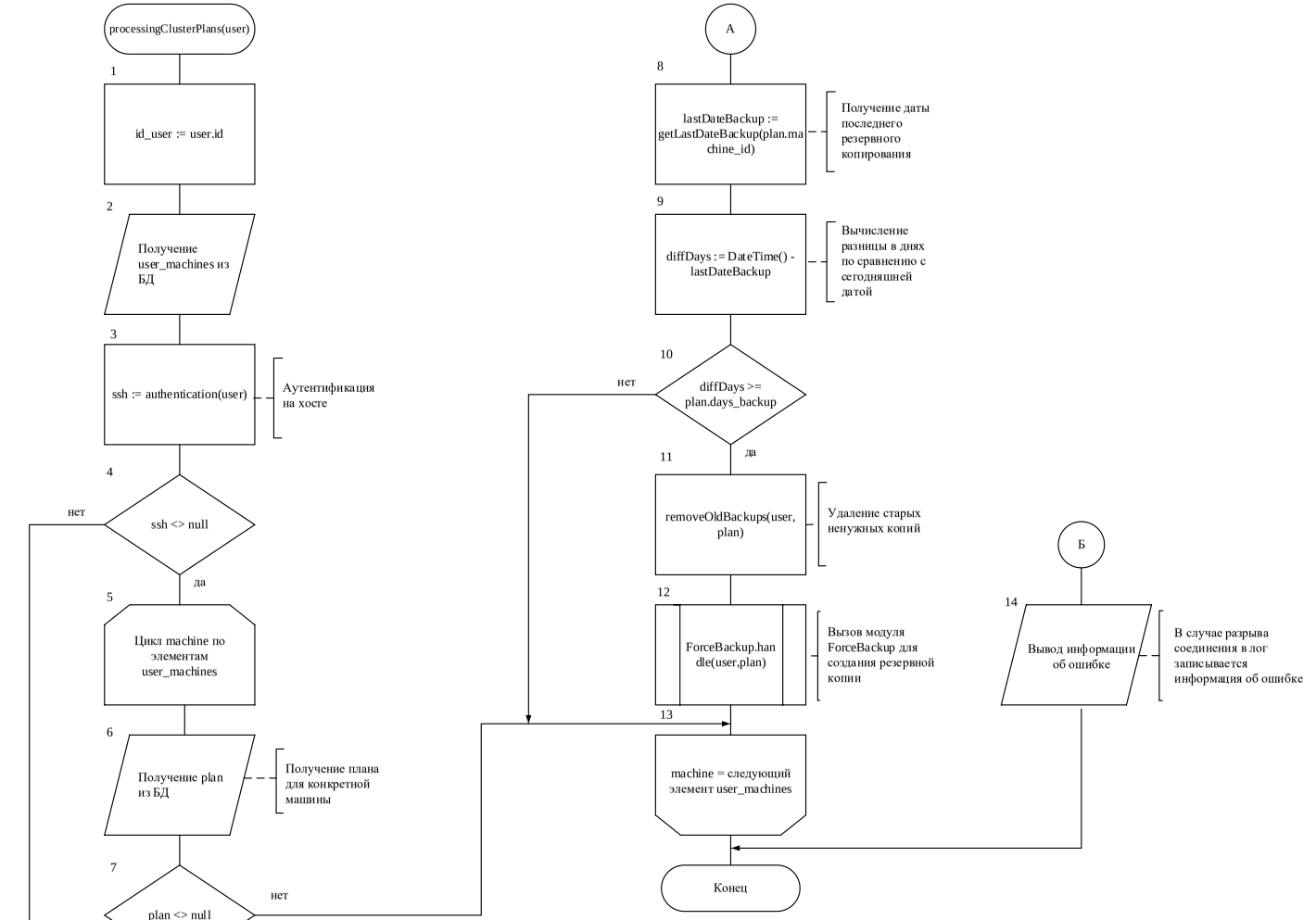


Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 1.090501.11.004 Д2
Разраб.		Рассказов Н.С			Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин. Функция handle. Схемы программ
Пров.		Попов К. В.			Лит. / Масса / Масш.
Т. контр.					Лист 1 / Листов 1
Н. контр.		Быков А.С.			ПГУ, каф. ВТ, гр. 17ВО2
Утв.		Митрохин М.А.			

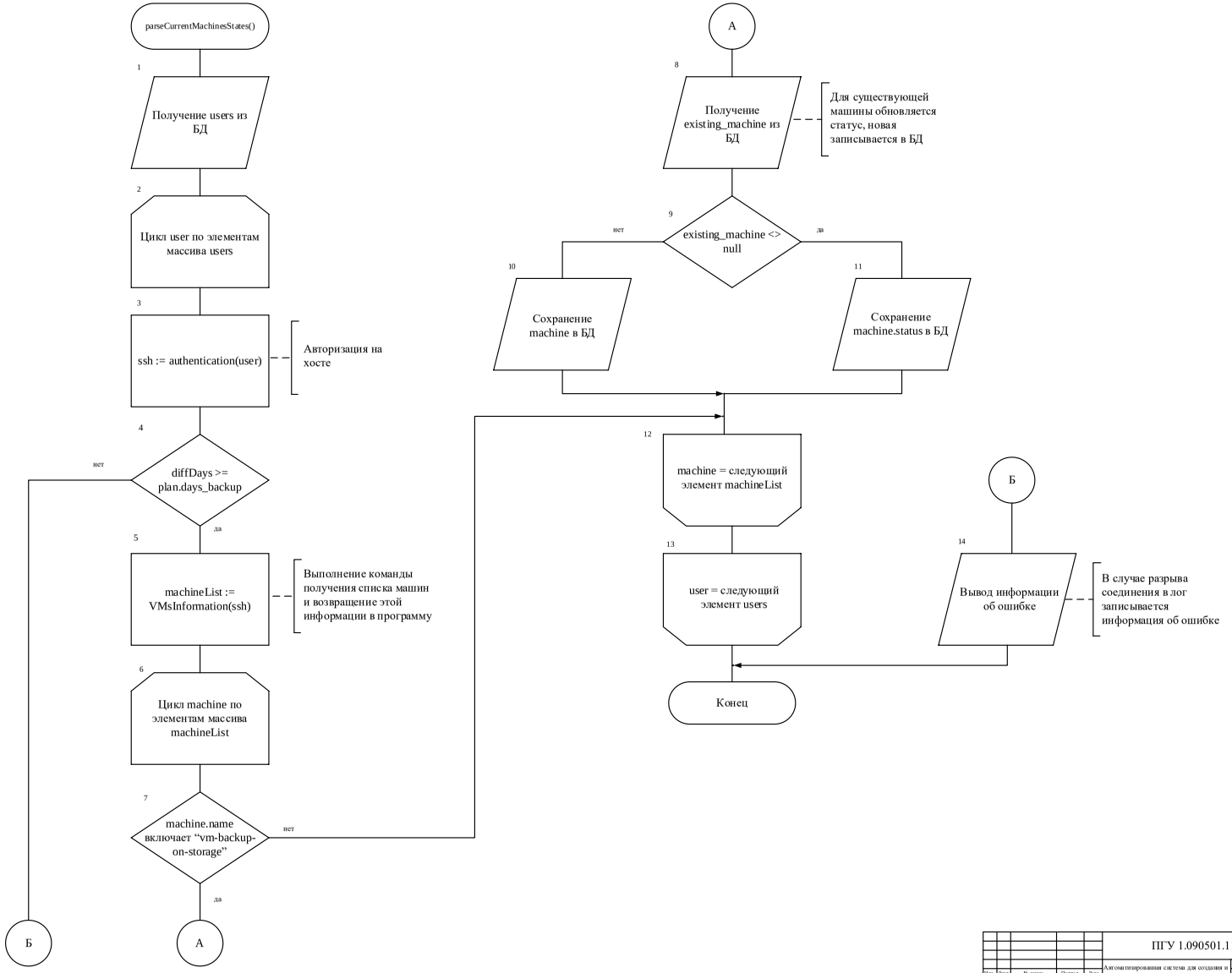
Изм. | Лист | № докум. | Подпись | Дата

ПГУ 1.090501.11.001 ПЗ

Лист 168



Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 1.090501.11.005 Д2
Разраб.		Рассомаха Н.С.			Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин. Функция processingClusterPlans Схемы программ
Пров.		Попов К.В.			Лит. / Масса / Масштаб
Т. контр.					Лист 1 / Листов 1
Н. контр.		Бычков А.С.			ПГУ, каф.ВТ, гр. 17ВО2
Утв.		Митрохин М.А.			



					ПГУ 1.090501.11.006 Д2
Изм.	Лист	№ докум.	Подпись	Дата	Автоматизированная система для создания и учета резервных копий в кластере виртуальных машин Функция parseCurrentMachinesStates Схемы программ
Разраб.		Рязанов П.С			Лит	Масса	Масштаб
Пров.		Попов К.В.					
Т. контр.					Лист 1	Листов 1	
Н. контр.		Белов А.С.			ПГУ, каф.ВТ, гр. 17ВО2		
Утв.		Митрохин М.А.					

Изм. | Лист | № докум. | Подпись | Дата

ПГУ 1.090501.11.001 ПЗ

Лист 170

Приложение Г

(обязательное)

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

Презентация

Пензенский государственный университет
Кафедра «Вычислительная техника»

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

Разработал: студент гр. 17ВО2
Рассказов Н.С.
Научный руководитель: к.т.н., доцент каф.
ВТ Попов К.В.

Пенза - 2022

Актуальность проекта

Проблема: потеря критически важных данных в случае:

- аппаратных сбоев устройств;
- действия вредоносного ПО
- ошибок пользователей

Решение: организация системы резервного копирования виртуальных машин

2

Назначение и функции системы

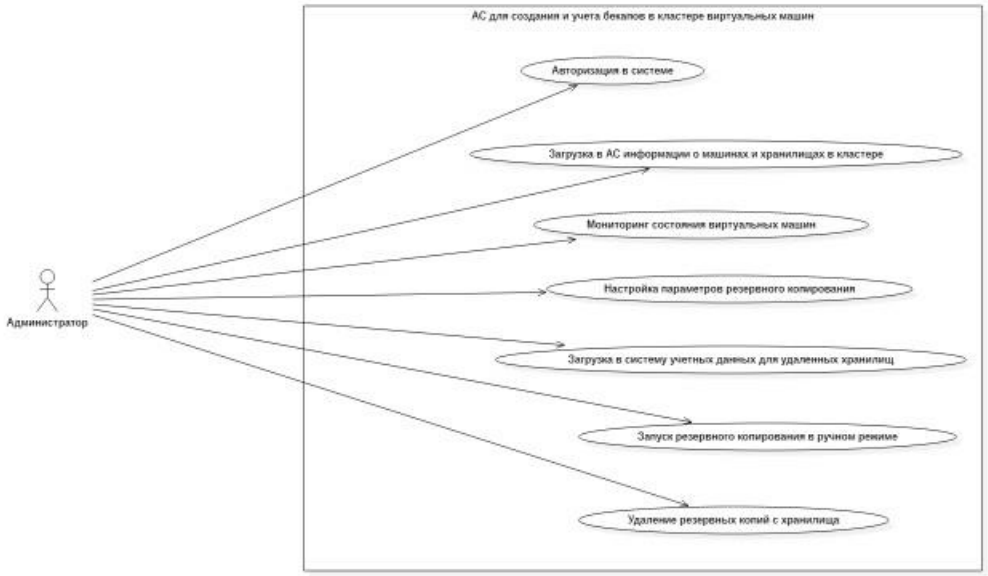
Система предназначена для периодического создания резервных копий виртуальных машин кластера в автоматическом и ручном режимах, а также хранения информации о резервных копиях, машинах и хранилищах кластера.

Функции системы:

- авторизация в системе;
- загрузка в систему информации о виртуальных машинах и хранилищах кластера;
- периодическое безагентное резервное копирование виртуальных машин кластера в автоматическом и ручном режимах;
- хранение резервных копий на хранилищах с возможностью их удаления в ручном и автоматическом режимах;
- управление настройками резервного копирования с помощью веб-формы;
- вывод информации о виртуальных машинах, хранилищах и резервных копиях;
- загрузка учетных данных для удаленных хранилищ.

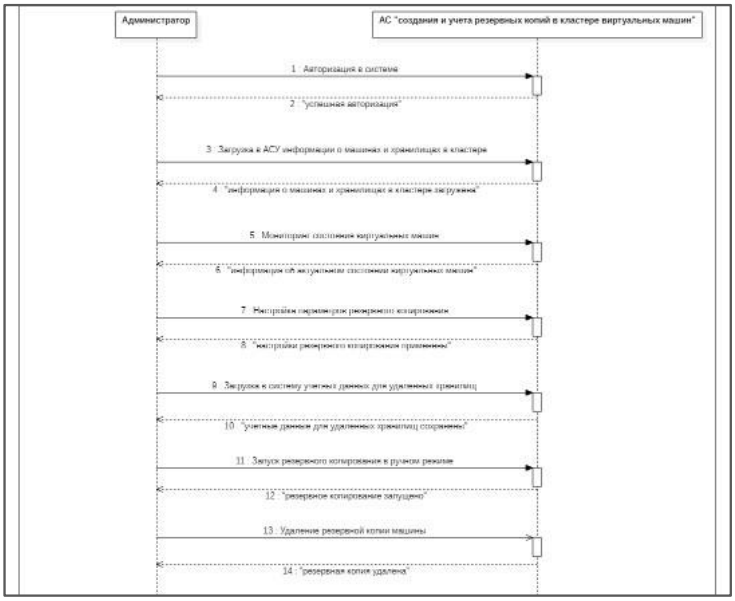
3

Диаграмма вариантов использования



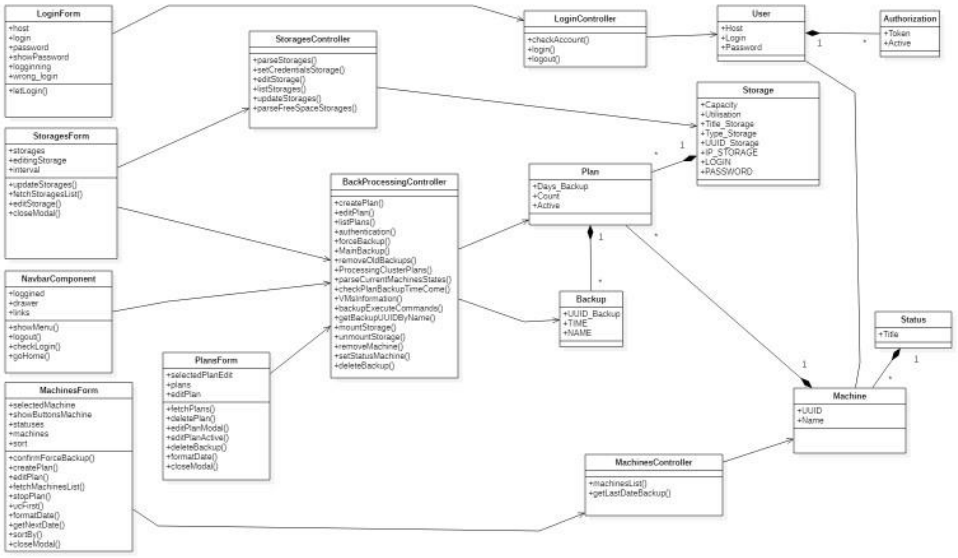
4

Диаграмма последовательности



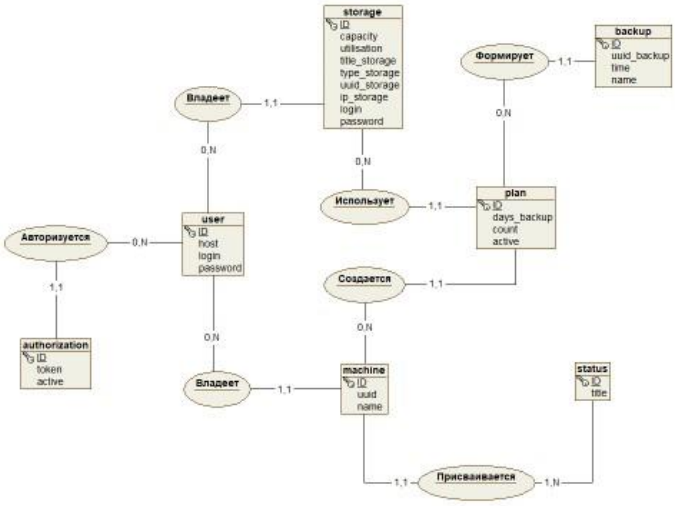
5

Диаграмма классов

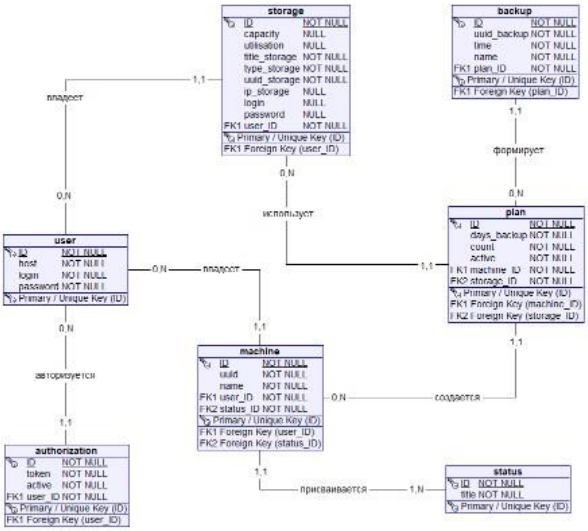


6

Концептуальная модель данных

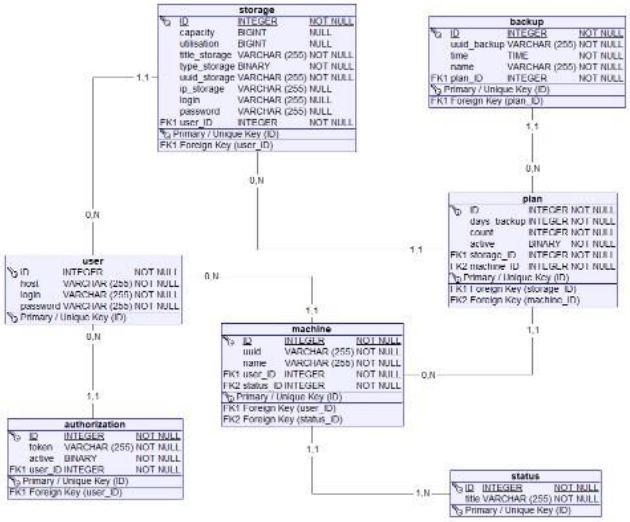


Логическая модель данных



8

Физическая модель данных



9

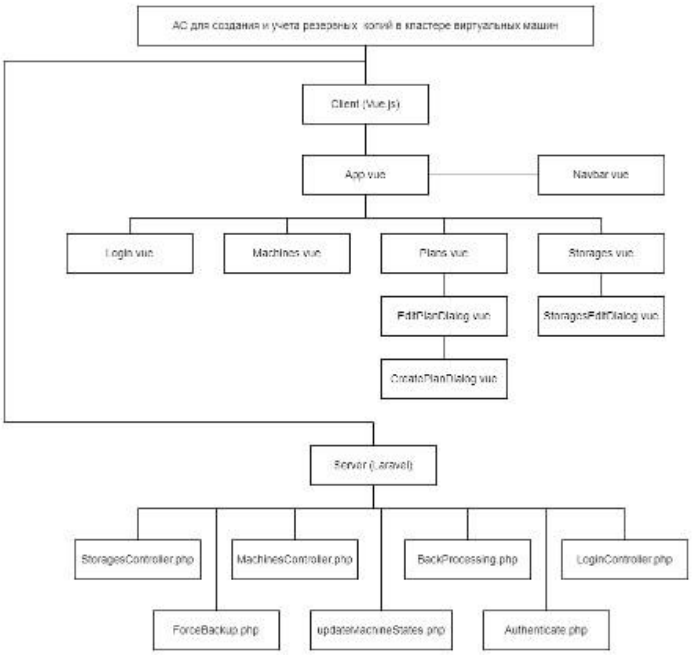
Использованные средства разработки

клиентская часть

серверная часть

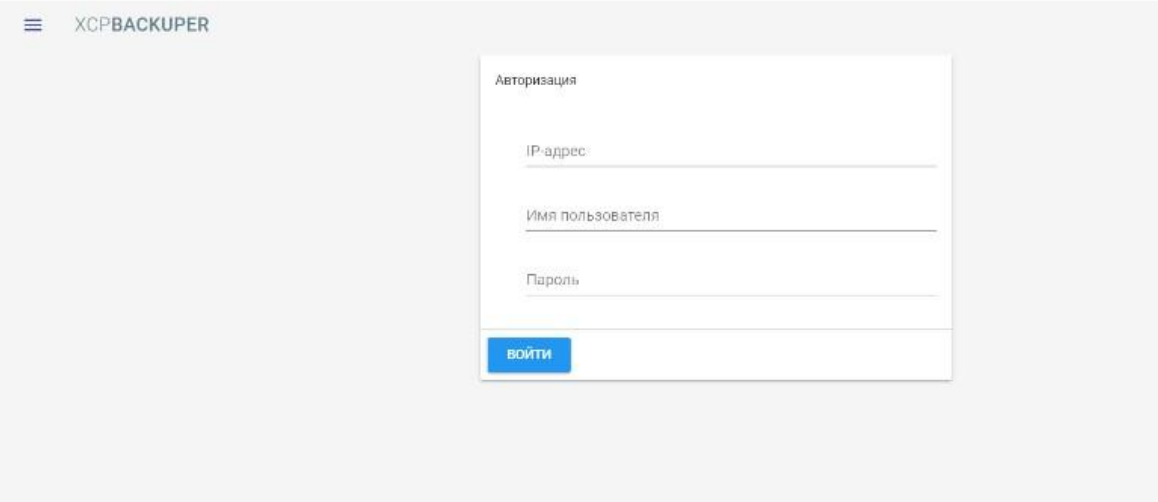
WebStorm

WebStorm


АС для создания и учета резервных копий в кластере виртуальных машин
Client (Vue.js)
App.vue
Navbar.vue
Login.vue
Machines.vue
Plans.vue
Storages.vue
EditPlanDialog.vue
StoragesEditDialog.vue
CreatePlanDialog.vue
Server (Laravel)
StoragesController.php
MachinesController.php
BackProcessing.php
LoginController.php
ForceBackup.php
updateVachineStates.php
Authenticate.php

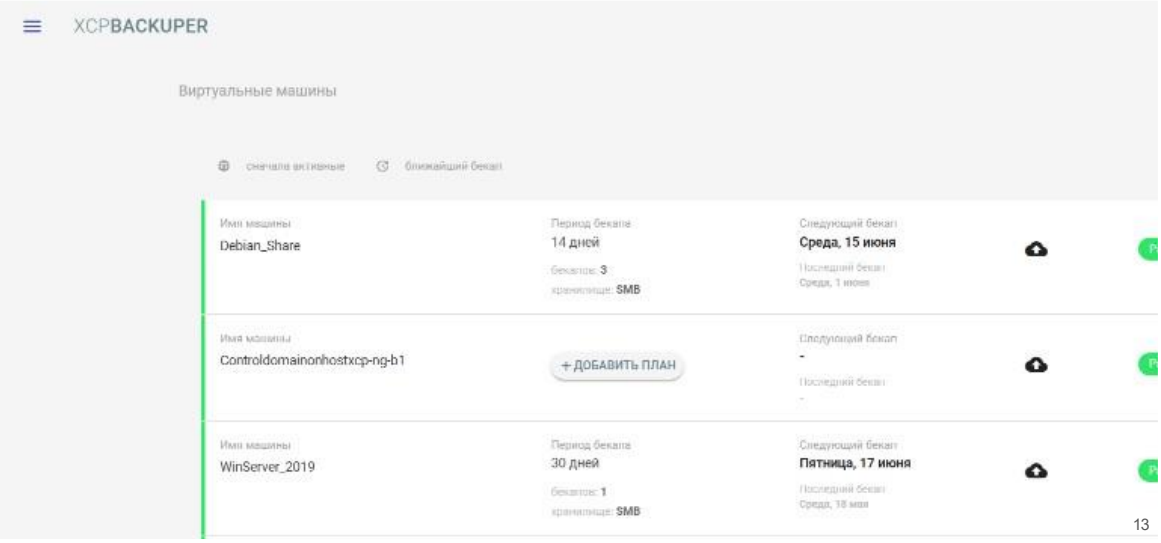
11

Форма авторизации

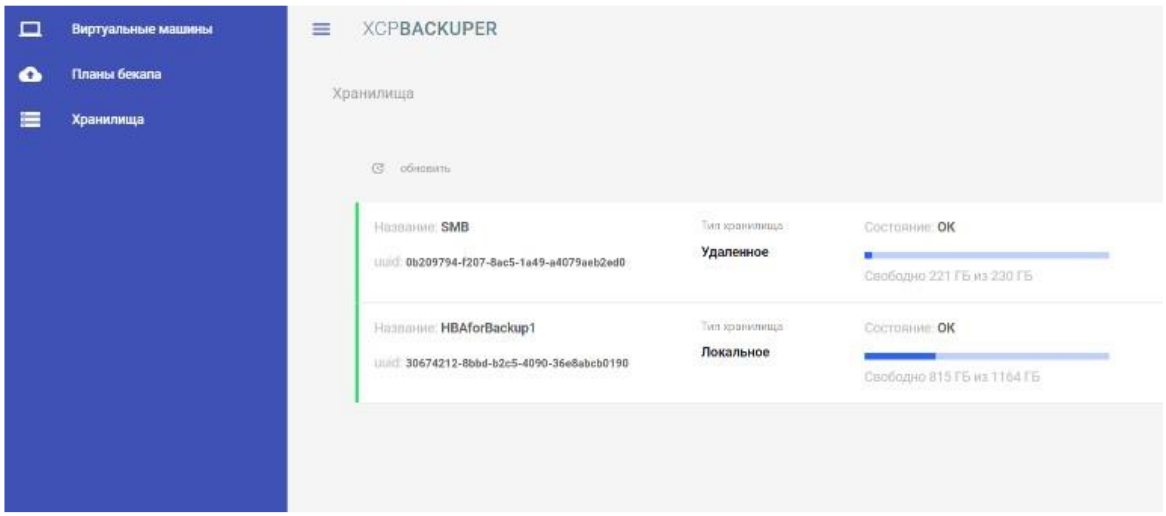


12

Форма со списком виртуальных машин

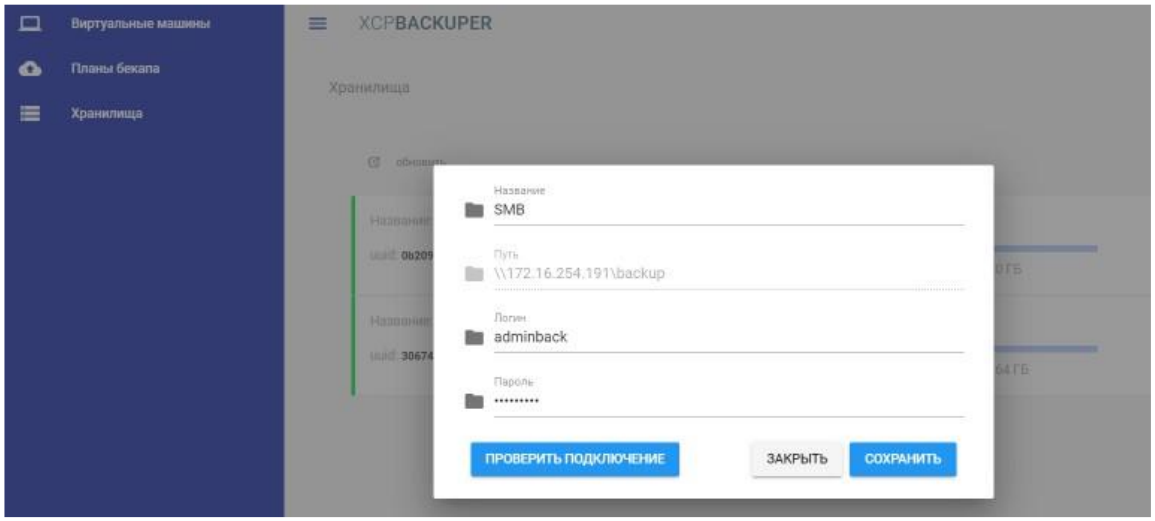


Форма со списком хранилищ



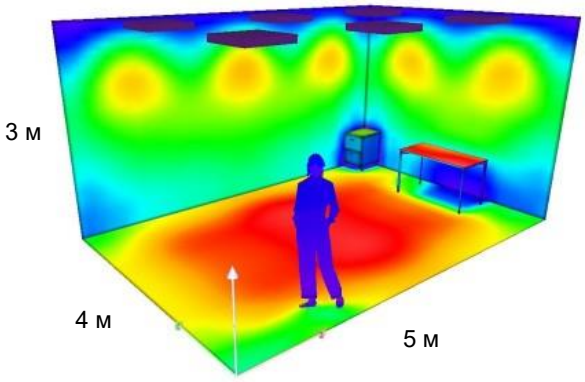
14

Форма введения учетных данных хранилища



15

Расчет освещенности помещения



S = 20 м^2
h = 3 м
F = 39600 Лм
N = 8

Лампа типа ЛБ -80

16

Расчет затрат труда

Вид работ	Оценка трудоемкости дн.			Расчетные величины дн.	
	a_i	m_i	b_i	t_i	D_i
Описание задачи и разработка плана работ	2	3	4	3	3
Выбор инструментов разработки	2	2	2	2	2
Разработка схем структуры проекта	4	5	6	5	5
Программирование клиентского модуля приложения	20	24	28	24	24
Программирование Серверного модуля приложения	20	24	28	24	24
Отладка программы	10	12	14	12	12
Подготовка документации	4	5	6	5	5
Всего					75

Стоимость разработки: 339758 руб.

Стоимость внедрения: 365758 руб.

Затраты на эксплуатацию: 14832 руб./год.

17