МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

02.03.03 МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ И
АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

БАКАЛАВРСКАЯ РАБОТА

на тему: Система единой аутентификации ТГУ

Студент Д.А. Наговицын _____

Руководитель Н.И. Лиманова _____

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский _____

«_____» _____ 2016 г.

Тольятти 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная
математика и информатика»
_____ А.В. Очеповский

«____»_____ 2016 г.

ЗАДАНИЕ
на выполнение бакалаврской работы

Студент Наговицын Дмитрий Андреевич

1. Тема Разработка системы единой аутентификации ТГУ

2. Срок сдачи студентом законченной выпускной квалификационной работы 19 июня 2016 года

3. Исходные данные к выпускной квалификационной работе:
сведения и данные о компании, существующие разработки подобных ресурсов, электронные и печатные материалы, данные и материалы преддипломной практики, действующие стандарты.

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов):

Введение

1. Анализ исследуемой области

1.1. Анализ существующих аналогов

1.2. Внутренняя структура организации

1.3. Технология единого входа

1.4. Основные механизмы реализации систем единого входа

2. Способ реализации системы единого входа

2.1. Обзор продуктов, реализующих SSO – технологию

2.2. Выбор продукта для внедрения

3. Методология создания сервера единой аутентификации

3.1. Настройка веб – сервиса

3.2. Настройка Jasig CAS

3.3. Рассмотрение результатов внедрения

Заключение

Библиографический список

Приложения

5. Ориентировочный перечень графического и иллюстративного материала: рисунки, графики и диаграммы, поясняющие результат работы системы, презентация.

6. Дата выдачи задания « 11 » января 2016 г.

Руководитель выпускной квалификационной работы _____ Н.И. Лиманова

Задание принял к исполнению _____ Д.А.Наговицын

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

УТВЕРЖДАЮ
Зав. кафедрой «Прикладная
математика и информатика»
_____ А.В. Очеповский

«____»_____ 2016 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения бакалаврской работы

Студента Наговицына Дмитрия Андреевича
по теме Разработка системы единой аутентификации ТГУ

Наименование раздела работы	Плановый срок выполнения раздела	Фактический срок выполнения раздела	Отметка о выполнении	Подпись руководителя
Поиск и исследование литературы по теме выпускной квалификационной работы	19.02.2016	19.02.2016	выполнено	
Написание первой главы БР	22.02.2016	22.02.2016	выполнено	
Написание второй главы БР	11.03.2016	11.03.2016	выполнено	
Реализация разработки сервиса	10.04.2015	10.04.2015	выполнено	
Тестирование и отладка сервиса	17.04.2015	17.04.2015	выполнено	
Написание третьей главы	19.04.2016	19.04.2016	выполнено	
Представление выпускной квалификационной работы на кафедру	16.05.2016	16.05.2016	выполнено	
Подготовка доклада и графического материала	20.05.2016	20.05.2016	выполнено	

Предварительная защита	25.05.2016	25.05.2016	выполнено	
Сдача пояснительной записки ВКР	19.06.2016	19.06.2016	выполнено	

Руководитель выпускной квалификационной работы _____ Н.И. Лиманова

Задание принял к исполнению _____ Д.А. Наговицын

Аннотация

Тема данной выпускной квалификационной работы: Система единой аутентификации ТГУ

Актуальность темы данной работы заключается в том, что повысится эффективность и скорость работы персонала с сервисами ТГУ.

Целью данной бакалаврской работы является организация системы единой аутентификации ТГУ.

Для достижения поставленной цели в ходе работы были поставлены и решены следующие задачи:

– провести анализ предметной области;

– проанализировать существующие решения;

– выбрать соответствующий характеристикам заказчика продукт;

– на основе выбранного продукта реализовать технологию единого входа;

– внедрить полученное решение в информационное пространство ТГУ.

Работа состоит из введения, трех глав, заключения, списка использованной литературы и приложений.

В работе использованы современные системы и технологии проектирования аналитических систем: технология структурного моделирования, концепция SSO, MySQL, Oracle Database и др.

Первая глава посвящена анализу предметной области, обзору и обоснованию выбора средств для разработки.

Вторая глава посвящена архитектуре информационной среды системы единой аутентификации ТГУ.

Третья глава посвящена реализации системы единой аутентификации ТГУ.

В заключении сформулированы основные выводы, которые были сделаны в процессе написания бакалаврской работы, описаны результаты практической реализации проекта.

В приложениях представлены фрагменты программного кода и другие дополнительные материалы.

Структура и объем работы. Работа состоит из введения, трёх разделов, заключения, списка литературы из 20 источников и 4 приложений. Общий объем работы – 44 страницы, 39 рисунков.

Оглавление

Введение

Жизнь и практическая деятельность в информационном обществе неразрывно связаны с грамотной организацией информационных процессов, использованием современных информационных технологий.

Как показывает практика, люди тратят много времени для перехода между разными информационными сервисами. Также при частых сменах ресурсов рассеивается внимание, не говоря уже о том, что чем больше логинов и паролей – тем проще в них запутаться.

В современном быстро развивающемся мире появляется множество технологий, обеспечивающих удобство использования и скорость доступа к программам и сервисам. Зачастую внедрение и установка являются дорогим и трудным процессом, но результат окупает себя с лихвой.

Для успешной разработки и внедрения технологии единого входа необходимо решить следующие задачи:

1. Провести анализ предметной области.
2. Проанализировать существующие решения.
3. Выбрать соответствующий характеристикам заказчика продукт.
4. На основе выбранного продукта реализовать технологию единого входа.
5. Внедрить полученное решение в информационное пространство ТГУ.

Объектом исследования является процесс аутентификации пользователя.

Предметом исследования является организация системы единого входа.

Целью данной выпускной квалификационной работы является разработка и внедрение системы единой аутентификации для Тольяттинского Государственного Университета.

В ходе выполнения выпускной квалификационной работы реализована система единого входа, которая позволит переходить по информационным сервисам ТГУ без

повторной аутентификации.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка литературы и приложений.

Во введении описывается актуальность проводимого исследования, формируется цель, и ставятся задачи, которые необходимо решить для достижения цели.

В первой главе описывается анализ деятельности предприятия для определения ключевых компонентов процесса, анализ методов и способов реализации систем единой аутентификации.

Во второй главе проводится разработка и реализация проектных решений.

В третьей главе приводится руководство пользователя.

В заключении приводятся основные выводы по работе, достигнутые в ходе выполнения выпускной квалификационной работы.

Глава 1 Анализ исследуемой области

1.1 Анализ существующих аналогов

Системы единой аутентификации внедряются практически повсеместно. Лидером в этой сфере, несомненно, является Google. Практически все сервисы, принадлежащие этой организации, способны использовать единую аутентификацию. Система SSO доступна для Google Apps for Work, Education и Government. Она позволяет войти сразу во все сервисы Google Apps (включая консоль администратора), не указывая каждый раз учетные данные. Если система включена, пользователь, выполняющий вход в консоль администратора или другой сервис Google, перенаправляется на страницу единого входа.

Они предоставляют API для систем единого входа на базе SAML (язык разметки декларации безопасности), позволяющий интегрировать Google Apps в существующую систему SSO на основе LDAP (Lightweight Directory Access Protocol) и других протоколов. Сетевой протокол LDAP используется для изменения сервисов каталогов, работающих в сетях TCP/IP, и отправки к ним запросов.

Система SSO также доступна на устройствах Chrome.

Стоит отметить, что в государственном аппарате Российской федерации так же используется единая аутентификация. Единая система идентификации и аутентификации (ЕСИА) является информационным сервисом Российской федерации. Была разработана ОАО Ростелеком в 2010 году. Постоянно модернизируется и расширяется. Работает на базе SAML 2.0 и поддерживает OpenID. Алгоритм работы включает в себя следующие действия:

- запрос пользователя к ресурсу;
- отправка запроса в ЕСИА на аутентификацию;
- ЕСИА проверяет пользователя на наличие активной сессии;
- при нахождении таковой, ЕСИА отправляет данные пользователя;

− на основе полученных данных, система авторизует пользователя на ресурсе.

Единую аутентификацию использует и Яндекс. Сервисы Яндекса авторизуют приложения по токенам. Каждый токен — это цифро-буквенная последовательность, в которой зашифрована следующая информация:

− идентификатор учетной записи, к которой разрешен доступ;

− идентификатор приложения, которому разрешен доступ;

− набор прав (действий, доступных приложению).

1. Приложение направляет пользователя на OAuth-сервер. На открывшейся странице он может разрешить приложению доступ к определенным данным своей учетной записи.

2. Пользователь разрешает доступ к своим данным, и OAuth-сервер перенаправляет его на указанный разработчиком адрес.

Выданный токен (или код для его получения) включается в URL перенаправления. Если пользователь отказал в доступе, или произошла ошибка, в URL включается описание ошибки.

3. Приложение включает полученный токен в запрос к сервису Яндекса, который поддерживает OAuth.

Тольяттинский государственный университет является градообразующим учреждением, ежегодно собирающим в своих стенах тысячи студентов. Информационное пространство ТГУ включает множество различных сайтов и сервисов. За обеспечение их стабильной работы отвечает Центр новых информационных технологий Тольяттинского государственного университета (ЦНИТ). Также в его юрисдикции находятся разработка и внедрение новых программных продуктов и сервисов, управление ИТ – услугами, разработка и модернизация программного обеспечения.

1.2 Внутренняя структура организации

Информационная структура Тольяттинского государственного университета базируется на системе «Галактика». Под ее контролем

находятся следующие модули:

- заработная плата;
- бухгалтерская отчетность;
- приемная кампания;
- кадровый учет;
- управление бюджетом;
- управление договорами;
- управление контингентом студентов;
- управление учебным процессом;
- складской учет;
- платежный календарь;
- финансово – расчетные операции;
- расчет заработной платы и стипендий;
- платное обучение;
- расписание учебных занятий.

Вне системы «Галактика» задействованны:

- документооборот;
- деканаты;
- кафедры;
- битрикс24;
- readmine;
- iTop.

Исходя из количества ресурсов, мы можем представить сколько раз сотрудник будет вводить логин – пароль. Использование технологии единого вход может решить эту проблему.

1.3 Технология единого входа

Технология единого входа (Single Sign – On) позволяет пользователю переходить из одного раздела портала в другой без повторной

аутентификации.

SSO – системы представлены несколькими основными типами. К ним относятся:

- традиционный SSO;
- клиентский SSO;
- серверный SSO;
- комбинированное решение;
- web-SSO.

Клиентский SSO основан на SSO – приложении. Его алгоритм работы представлен на рисунке 1.2.

К плюсам клиентского SSO можно отнести:

- запоминание пароля;
- быстрое внедрение, которое не зависит от сетевой инфраструктуры
- доступ к параметрам аутентификации имеет только пользователь

Из минусов же стоит отметить такие особенности:

- проблемы с мобильными пользователями
- сложности с администрированием

Серверный SSO, как следует из его названия, основан на сервере аутентификации. Алгоритм его работы представлен на рисунке 1.3.

Рисунок 1.2 – Алгоритм работы клиентского SSO

Рисунок 1.3 – Алгоритм работы серверного SSO

К плюсам серверных SSO относят:

- возможность удаленного доступа к сервисам, минуя один определенный компьютер;
- удобное централизованное управление.

В числе минусов следует отметить:

- зависимость от сетевой инфраструктуры;
- единую точку отказа.

Поскольку, внедрение SSO – решений, зачастую довольно трудоемкий и дорогой процесс, следует убедиться, что они соответствуют определенным требованиям, среди которых следует отметить:

- гибкость и масштабируемость;
- поддержку открытых архитектур и стандартов;
- быстрое развертывание и эффективное управление;
- поддержку мобильных устройств;
- простоту использования конечными пользователями;
- безопасность;
- оправданную стоимость.

К основным преимуществам технологии единого входа относятся:

− уменьшение парольного хаоса между различными комбинациями имени пользователя и пароля;

− уменьшение времени на повторный ввод пароля для одной и той же учетной записи;

− поддержка традиционных механизмов аутентификации, таких как имя пользователя и пароль;

− снижение расходов на IT – службу за счёт уменьшения количества запросов по восстановлению забытых паролей;

− обеспечение безопасности на каждом уровне входа/выхода/доступа к системе без причинения неудобств пользователям.

В технологии единого входа применяются централизованные серверы аутентификации, используемые другими приложениями и системами, которые обеспечивают ввод пользователем своих учётных данных только один раз.

Экономия времени и улучшение эргономики рабочего места благотворно сказывается на продуктивности работы сотрудников, именно поэтому внедрение технологий единого входа является актуальным.

1.4 Основные механизмы реализации систем единого входа

Технология единого входа может быть реализована множеством путей.

Первым из них является сетевой протокол аутентификации Kerberos.

Он основан на механизме взаимной аутентификации клиента и сервера перед установлением связи между ними. Протокол Kerberos 4 содержит два логических компонента:

– сервер аутентификации;

– сервер выдачи билетов. Обычно эти компоненты поставляются как единая программа, которая запускается на центре распределения ключей

(ЦРК — содержит базу данных логинов/паролей для пользователей и сервисов использующих Kerberos).

Схема работы Kerberos 5 в настоящее время происходит следующим образом:

Вход пользователя в систему:

1) Пользователь вводит имя и пароль на клиентской машине.

2) Клиентская машина выполняет над паролем одностороннюю функцию (обычно хэш), и результат становится секретным ключом клиента/пользователя [1-7].

Аутентификация клиента:

1) Клиент отсылает запрос (AS_REQ) на CA для получения аутентификационных верительных данных и последующего их предоставления TGS серверу (впоследствии он будет их использовать для получения билетов без дополнительных запросов на применение секретного ключа пользователя.) Данный запрос содержит идентификатор клиента, его метку времени и идентификатор сервера.

2) Если политика ЦРК требует предварительной аутентификации, то пользователь получает сообщение KRB_ERROR, в ответ на которое посылает повторный запрос, но уже с данными для установления подлинности.

3) CA проверяет, есть ли такой клиент в базе. Если есть, то назад CA отправляет сообщение (AS_REP), включающее.

- сессионный ключ Клиент/TGS, идентификатор TGS и время жизни билета, зашифрованные секретным ключом клиента;
- TGT (который включает идентификатор и сетевой адрес клиента, метку времени ЦРК, период действия билета и сессионный ключ Клиент/TGS), зашифрованный секретным ключом TGS.

Если же нет, то клиент получает новое сообщение, говорящее о произошедшей ошибке.

Получив сообщение, клиент расшифровывает свою часть для получения Сессионного Ключа Клиент/TGS. Этот сессионный ключ

используется для дальнейшего обмена с сервером TGS. (Важно: Клиент не может расшифровать TGT, так как оно зашифровано секретным ключом TGS) В этот момент у пользователя достаточно данных, чтобы авторизоваться на TGS.

Второй путь – авторизация клиента на TGS. Происходит это следующим образом:

1) Для запроса сервиса клиент формирует запрос на TGS (TGS_REQ).

2) После получения TGS_REQ, TGS извлекает из него TGT и расшифровывает его используя секретный ключ TGS. Это дает ему Сессионный Ключ Клиент/TGS. Им он расшифровывает аутентификатор. Затем он генерирует сессионный ключ клиент/сервис и посылает ответ (TGS_REP) включающий:

- билет сервиса (который содержит ID клиента, сетевой адрес клиента, метку времени ЦРК, время действия билета и Сессионный Ключ клиент/сервис) зашифрованный секретным ключом сервиса;
- сессионный ключ клиент/сервис, идентификатор сервиса и время жизни билета, зашифрованные на Сессионном Ключе Client/TGS.

3) Запрос сервиса клиентом.

После получения TGS_REP, у клиента достаточно информации для авторизации на сервисе. Клиент соединяется с ним и посылает сообщение содержащее:

- зашифрованный билет сервиса полученный ранее;
- новый аутентификатор, зашифрованный на сессионном ключе клиент/сервис, и включающий ID клиента и метку времени.

Затем сервис расшифровывает билет используя свой секретный ключ и получает сессионный ключ клиент/сервис. Используя новый ключ, он расшифровывает аутентификатор и посылает клиенту следующее сообщение для подтверждения готовности обслужить клиента и показать, что сервер действительно является тем, за кого себя выдает метку времени, указанную клиентом + 1, зашифрованную на сессионном ключе клиент/сервис.

После этого клиент расшифровывает подтверждение, используя сессионный ключ клиент/сервис и проверяет, действительно ли метка времени корректно обновлена.

В итоге сервер предоставляет клиенту требуемый сервис.

Третьим способом является использование SAML. Этот язык разметки, основанный на языке XML. Открытый стандарт обмена данными аутентификации и авторизации между участниками, в частности, между поставщиком учётных записей и поставщиком сервиса . SAML — продукт OASIS, разработанный Техническим Комитетом Безопасности Сервисов. SAML создан в 2001 году; последнее значимое обновление SAML было опубликовано в 2005 году, но расширения протокола постоянно выпускались через дополнительные, опциональные стандарты.

Одной из важных проблем, которую пытается решить SAML, является обеспечение сквозной аутентификации (технология единого входа, англ. Single Sign On) при работе через Web – браузер. Использование SAML в качестве технологии единого входа на уровне сети распространено, но расширение за пределы частной сети было проблематично и привело к созданию несовместимых запатентованных технологий

Четвертый путь – OpenID. Являясь открытым стандартом децентрализованной системы аутентификации, предоставляющей пользователю возможность создать единую учётную запись для аутентификации на множестве не связанных друг с другом интернет – ресурсов, используя услуги третьих лиц.

Механизм работы:

1. Конечный пользователь желает аутентифицироваться с помощью Предъявляемого ID на Интернет – сервисе, через свой браузер.

2. Из Предъявляемого Идентификатора Интернет – сервис определяет URL Провайдера OpenID, используемого конечным пользователем. Предъявляемый ID может содержать только OP URL, который позволяет конечному пользователю, каким – то образом взаимодействуя с OP, передать

Единый ID, или любую другую информацию о себе, Интернет – сервису. Переданная информация ещё не проверена Интернет – сервисом.

3. Интернет – сервис и OP вместе создают общий секретный ключ для кода аутентификации сообщения по протоколу Диффи – Хеллмана. С помощью кода аутентификации сообщения Интернет – сервис аутентифицирует сообщение от OP без дополнительных запросов подлинности к провайдеру.

4. Интернет – сервис перенаправляет браузер пользователя на OP с запросом аутентификации.

5. Провайдер проверяет, авторизирован ли пользователь на сервере и хочет ли он аутентифицироваться на Интернет – сервисе. В общем случае, конечный пользователь предъявляет OP ID. Спецификация OpenID не описывает, каким образом конечный пользователь аутентифицируется у OP.

6. OP перенаправляет браузер пользователя назад в Интернет – сервис с утверждением, что пользователь аутентифицирован или не аутентифицирован, и с результатом аутентификации — одноразовой меткой. Каждая аутентификация на сервере генерирует новую метку, которая не повторяется со сгенерированными ранее одноразовыми метками для того же пользователя. Одноразовая метка позволяет предотвратить атаки повторного воспроизведения.

7. Интернет – сервис проверяет информацию, полученную от провайдера, включая возвращённый URL, информацию о пользователе, результат аутентификации на сервере — одноразовую метку, код аутентификации сообщения, с помощью секретного общего ключа, если он создавался на шаге 3, или, посылая прямой запрос OP.

8. В случае успешной проверки Интернет – сервис аутентифицирует пользователя.

Глава 2 Способ реализации системы единого входа

2.1 Обзор продуктов, реализующих SSO – технологию

В наше время есть множество готовых продуктов, которые реализуют технологию единого входа. Их можно разделить на «бесплатные» и «платные». Рассмотрим оба варианта и начнем с бесплатных.

OpenAm является «open source» серверной платформой. Поддерживает множество механизмов реализации SSO.Обеспечивающих соблюдение политики безопасности и защиту ресурсов. Реализован с использованием SAML.

Shibbolet является singl – sign on системой для компьютерных сетей и интернета. Основан на технологии SAML.

Distributed Access Control System сокращенно DACS, является «легковесной» SSO и role – based access control системой для web – серверов и серверных программ. Поддерживает lightweight directory access protocol и Microsoft active directory. Обладает возможностью двух – факторной аутентификации.

4) Central Authentication Service сокращенно CAS, является SSO протоколом для web. Позволяет пользователю получать доступ к множеству различных приложений, введя пароль лишь единожды.

CAS состоит из трех частей:

- клиентского web – браузера;
- web – приложения, требующего аутентификацию;
- CAS сервера.

Когда пользователь пытается пройти аутентификацию в приложении, оно переправляет данные на CAS – сервер, который сверяет логин и пароль с содержащимися в базе данных [8-14].

Среди платных стоит отметить таких представителей, как Indeed Enterprise Single Sign-On, Novell Nsure SecureLogin и eTrust Single Sign-On.

Indeed Enterprise Single Sign-On относится к комбинированным

решениям, поскольку состоит из приложения и сервера. Алгоритм работы системы заключается в запросе перечня систем, требующих аутентификации и, соответственно, данные пользователя, требующиеся для этого. После чего приложение-агент перехватывает окно аутентификации и каждый раз заполняет его предоставленными данными. Каждая аутентификация фиксируется в журнале для отслеживания возможных ошибок.

Novell Nsure SecureLogin является клиентским SSO. Обеспечивает единый вход для всех приложений, которые могут использоваться на предприятии. Способен работать в терминале и поддерживает систему двухфакторной проверки подлинности.

Способен работать с Microsoft Active Directory, Windows NT доменами, Novell eDirectory или любой службы каталогов LDAP v3.

SSO работает с 32-разрядной ОС Windows рабочем столе пользователя, сохраняя запись учетных данных для аутентификации пользователей и инструкции о том, как использовать их. Он хранит их (надежно зашифрованными с помощью Triple-DES) в каталоге, и он обнаруживает запросы на вход, извлекает соответствующие учетные данные и автоматически подает их в приложение.

eTrust Single Sign-On для корпоративного рынка и состоит из трех компонентов – сервера политик, клиентской рабочей станции, и диспетчера политик.

Сервер политик, который работает на IBM AIX, HP-UX, Sun Solaris или Windows, использует структуру базы данных и каталогов на основе протокола LDAP. Эта база данных содержит сведения о которых приложения каждый пользователь разрешен доступ.

Сервер политик поддерживает несколько методов аутентификации, и хранит учетные данные пользователей для входа в систему каждого приложения в его зашифрованной базе данных. Они прозрачно поставляются в соответствии с требованиями для каждого приложения через браузер или клиентское программное обеспечение пользователя. Сервер политик

управляет коллекцией паролей пользователя и представляет подходящие для приложений по мере необходимости.

2.2 Выбор продукта для внедрения

Исходя из собранных данных, можно сделать вывод что платные решения на голову превосходят бесплатные. Они, в целом, удобнее, надежнее, эффективнее. Так же при их использовании не придется волноваться об установке и настройке – их, зачастую, производит фирма, предоставляющая решение. Тем не менее, у них есть серьезный недостаток – стоимость. Все рассмотренные выше продукты стоят от 80 долларов за одного пользователя. Учитывая это, можно рассчитать стоимость внедрения какого-либо из продуктов в информационное пространство ТГУ. В данный момент системой «Галактика» одновременно могут пользоваться около сотни человек. В сумме это даст нам 8 тысяч долларов. В пересчете на современный курс это составит 560 тысяч рублей. Тратить подобные средства на систему которая может оказаться неэффективной в стенах университета не логично. Поэтому было принято решение воспользоваться одним из бесплатных аналогов, что бы оценить эффективность использования подобной системы в информационном пространстве ТГУ.

Из всех вышеперечисленных бесплатных продуктов, Central Authentication Service отличается гибкостью настройки, количеством поддерживающихся источников аутентификации, в числе которых LDAB и JAAS. Так же в России, CAS используется чаще его аналогов, что подразумевает под собой большую базу вспомогательных и информационных материалов по его настройке и внедрению, поэтому закономерным итогом стал выбор в его пользу. На рисунке 2.1 показан алгоритм работы CAS.

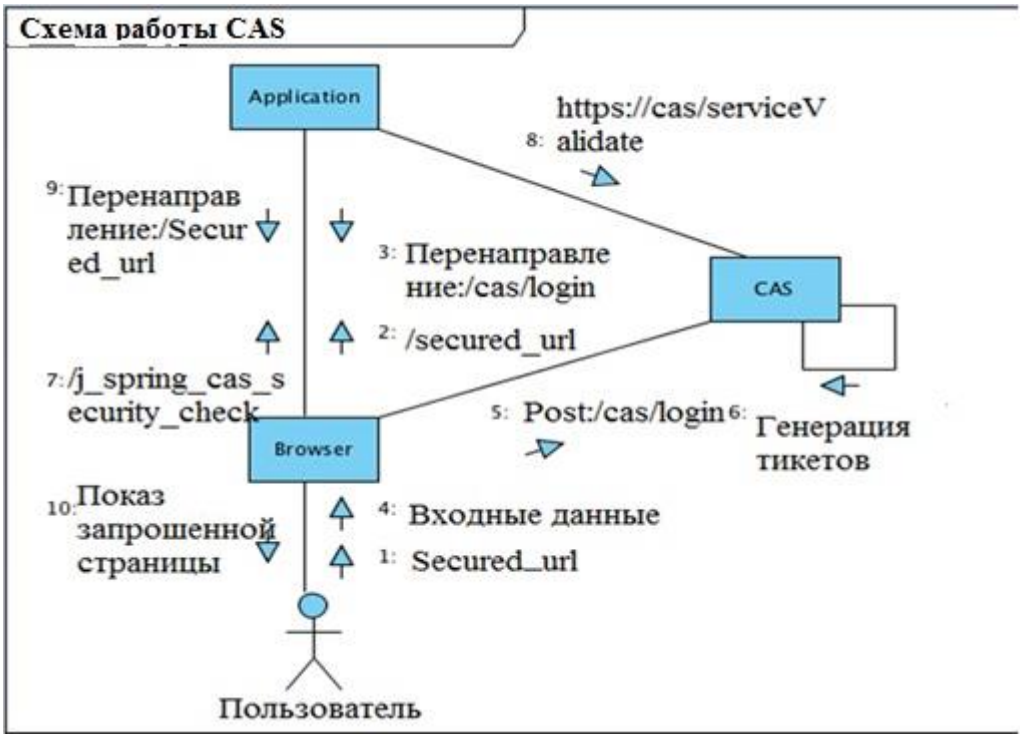

Рисунок 2.1 – Алгоритм работы CAS

Глава 3 Методология создания сервера единой аутентификации зачетной книжки

3.1 Настройка веб – сервиса

В этой главе будет подробна расписана установка и настройка ПО для ОС Ubuntu 14.04. Установка программ будет происходить через командную строку.

Сначала необходимо установить "Tomcat". Tomcat – это контейнер

сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Tomcat позволяет запускать веб – приложения, содержит ряд программ для само конфигурирования. Для установки Tomcat необходимо ввести в терминале "sudo apt – get install tomcat7", что продемонстрировано на рис. 3.1.

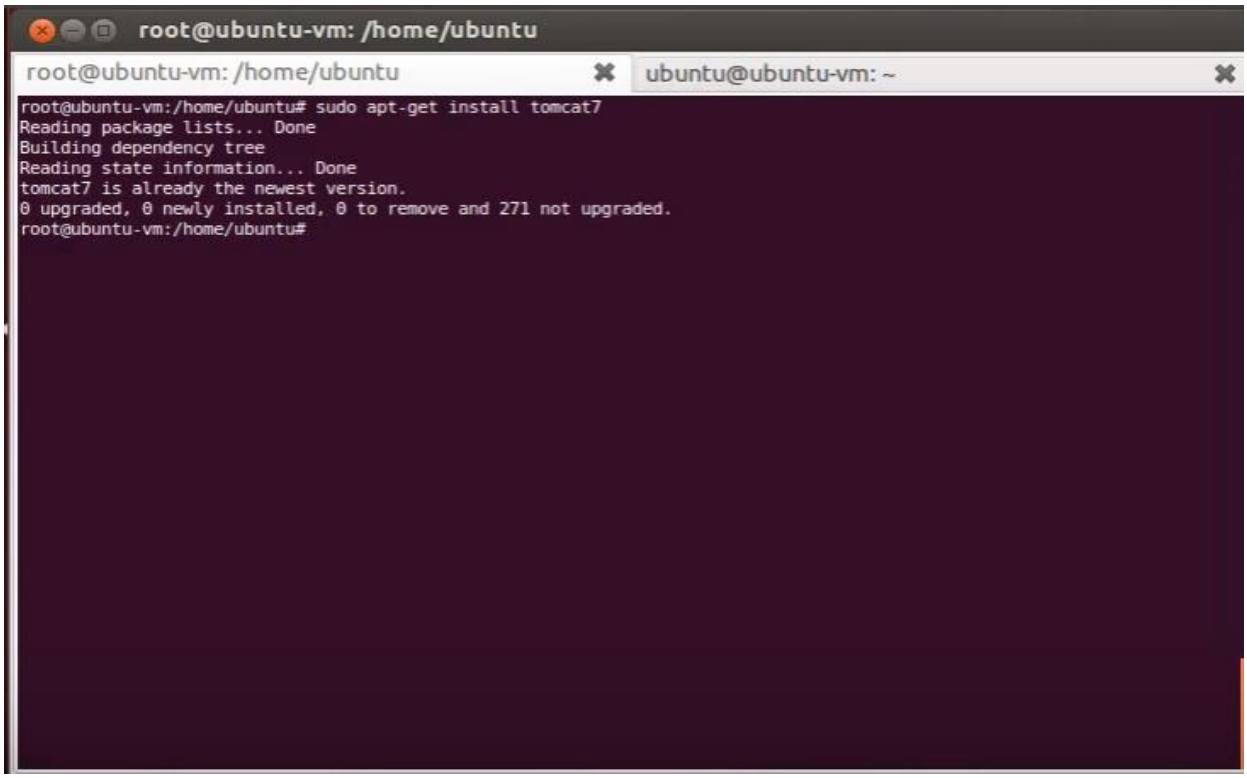

Рисунок 3.1 – Результат ввода команды "sudo apt – get install tomcat7" в консоль операционной системы Linux

Затем нужно создать keystore файл. Keystore используется для хранения собственных приватных ключей и сертификатов сервера или клиента. Для аутентификации клиента и сервера устанавливающих SSL соединение требуются приватные ключи и сертификаты. Если используется односторонняя аутентификация, то keystore нужен только на серверной стороне. При двусторонней аутентификации и клиент и сервер обмениваются сертификатами, соответственно и у сервера, и у клиента должен быть keystore

с парой приватный ключ/публичный ключ + сертификат. Т.е. иными словами Keystore используется для хранения ключей и сертификатов, использующихся для идентификации владельца ключа (клиента или сервера). Для работы с keystore в java дистрибутиве есть специальная утилита keytool. Для создания keystore файла нужно ввести в терминал команду "sudo keytool –keysize 4096 –genkey –alias tomcat –keyalg RSA –keystore CAS.keystore" что продемонстрировано на рис. 3.2.

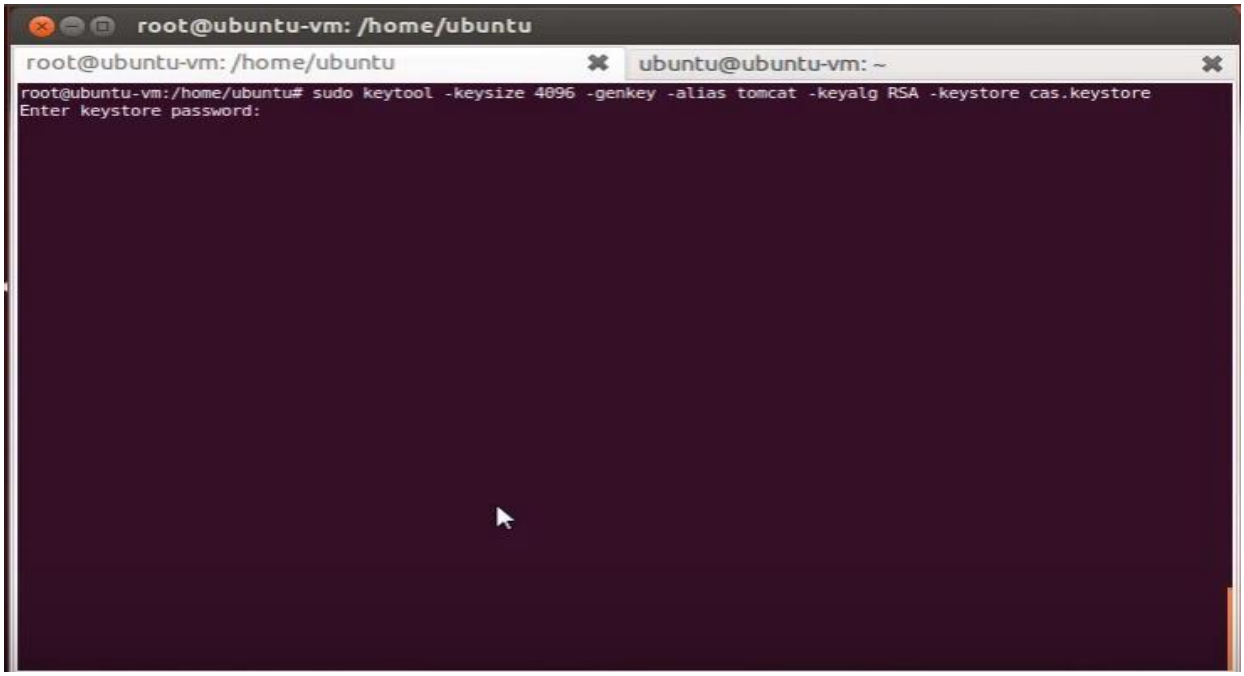

Рисунок 3.2 – Результат ввода команды sudo keytool –keysize 4096 –genkey –alias tomcat –keyalg RSA –keystore CAS.keystore" в консоль операционной системы Linux

После создания keystore файла необходимо перенести его в каталог, где установлен tomcat. Для этого необходимо ввести в терминале "mv CAS.keystore /etc/tomcat7". Результат команды продемонстрирован на рис. 3.3.

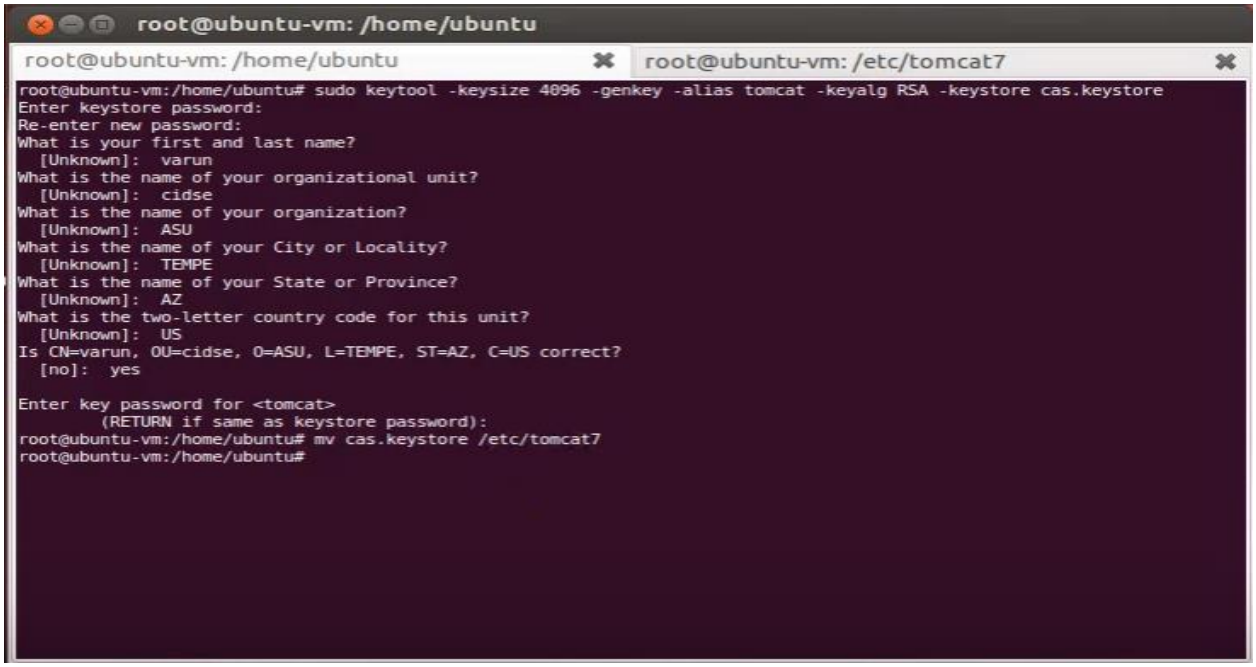

Рисунок 3.3 – Ввод команды "sudo gedit /etc/tomcat7/server.xml"

После этого необходимо настроить установленный ранее веб – сервер "Tomcat". Для этого нужно открыть и отредактировать файл конфигурации. Чтобы сделать это необходимо ввести в терминале команду "sudo gedit /etc/tomcat7/server.xml" см. рис. 3.4.

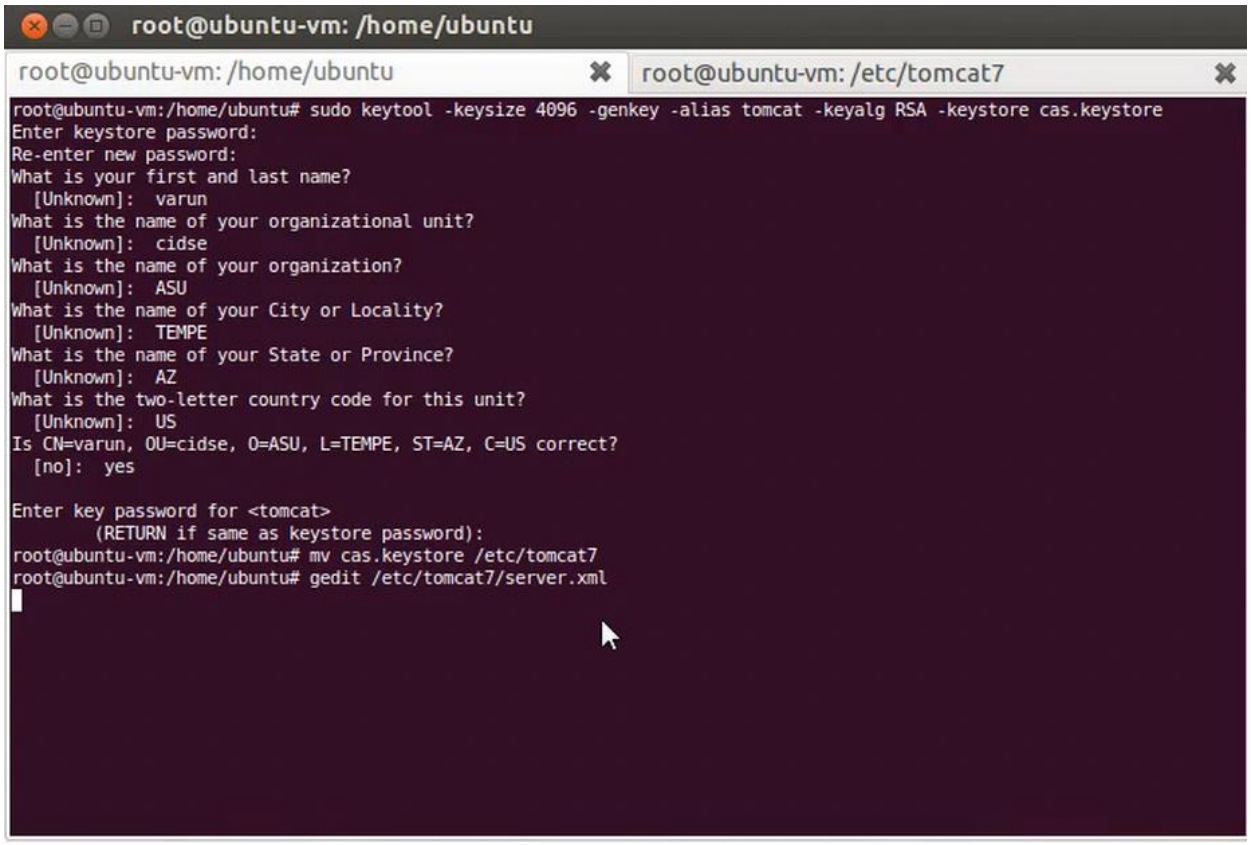

Рисунок 3.4 – Процесс обеспечения доступа к файлу server.xml

В открывшемся окне необходимо найти элемент "Connector" и заменить его на "<Connector port = "8443" protocol = "HTTP/1.1" SSLEnabled = "true" maxThreads = "150" scheme = "https" secure = "true" clientAuth = "false" sslProtocol = "TLS" keystoreFile = "/etc/tomcat7/CAS.keystore" keystorePass = "123456". Процесс изменения значений продемонстрирован на рис. 3.5.

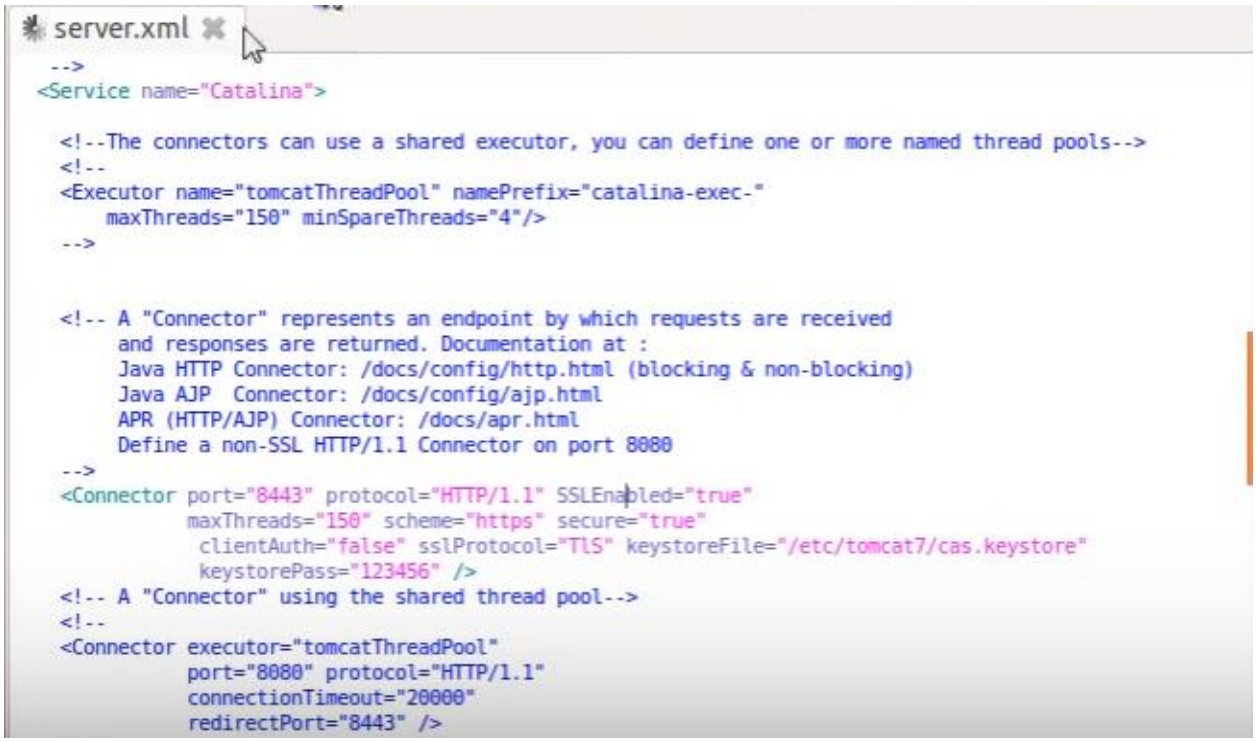

Рисунок 3.5 – Внутреннее строение файла server.xml

После настройки веб – сервера необходимо скачать CAS – server. Для этого в терминале необходимо ввести команду "wget http://downloads.jasig.org/CAS/CAS – server – 3.5.2.1 – release.zip" см. рис. 3.6.

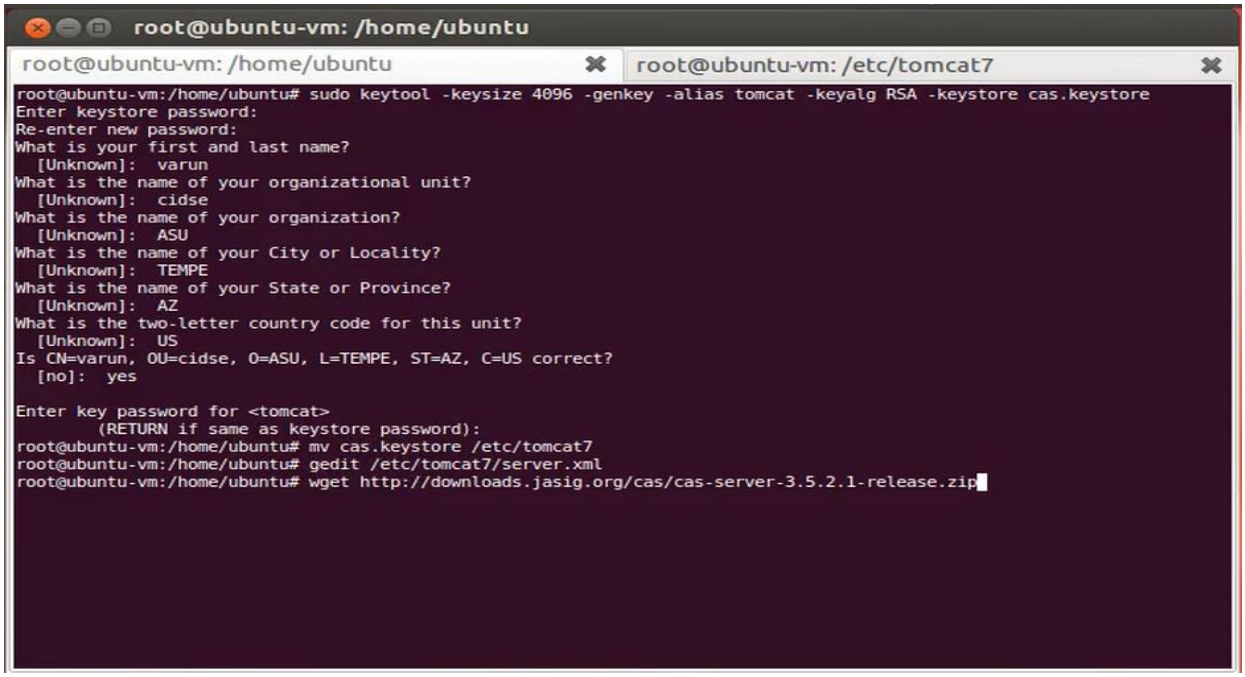

Рисунок 3.6 – Процесс загрузки CAS – server

После этого нужно разархивировать скачанный архив и переместить все файлы в папку webapps, которая находится в каталоге где установлен веб – сервер tomcat. Процесс разархивации представлен на рис. 3.7.

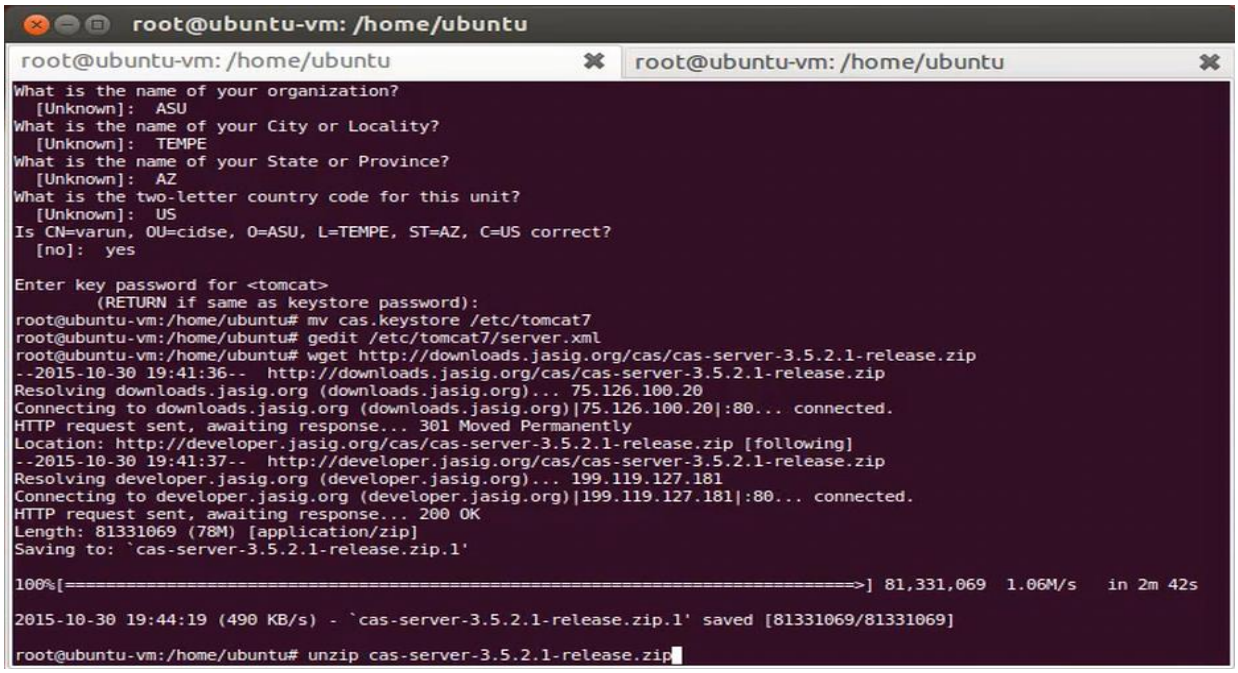

Рисунок 3.7 – Перемещение скачанных файлов в папку webapps

После этого нужно перезапустить веб – сервер tomcat. Сделать это можно с помощью команды "/etc/init.d/tomcat7 restart" см. рис. 3.8.

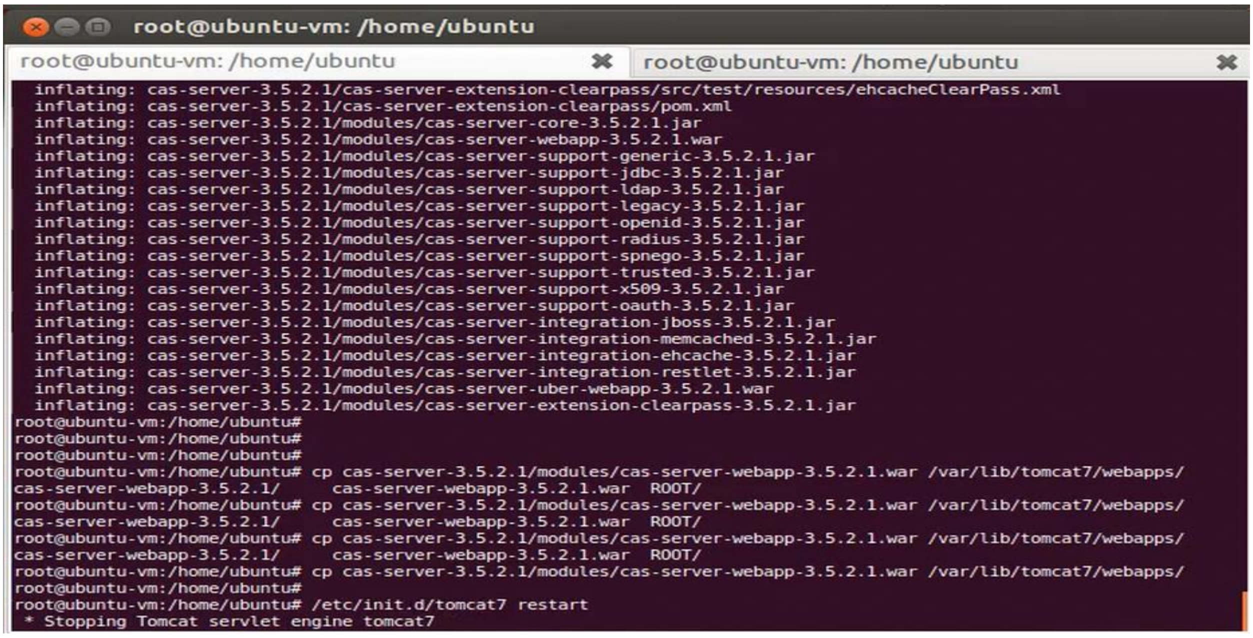

Рисунок 3.8 – Процесс перезапуска сервера tomcat7

Для следующего шага понадобится использовать утилиту iptables. Iptables — утилита командной строки, является стандартным интерфейсом управления работой межсетевого экрана (брандмауэра) Netfilter для ядер Linux, начиная с версии 2.4. С её помощью администраторы создают и изменяют правила, управляющие фильтрацией и перенаправлением пакетов. Для работы с семейством протоколов IPv6 существует отдельная версия утилиты — Ip6tables. Для использования утилиты Iptables требуются привилегии суперпользователя (root). Нужно перенаправить все входящие пакеты в порт 443 на порт 8443, для этого в терминале необходимо ввести команду "iptables –A PREROUTING –t nat –I eth0 –p tcp – – dport 443 – j REDIRECT – – to – port 8443". Результат выполнения на рис. 3.9.[14-20]

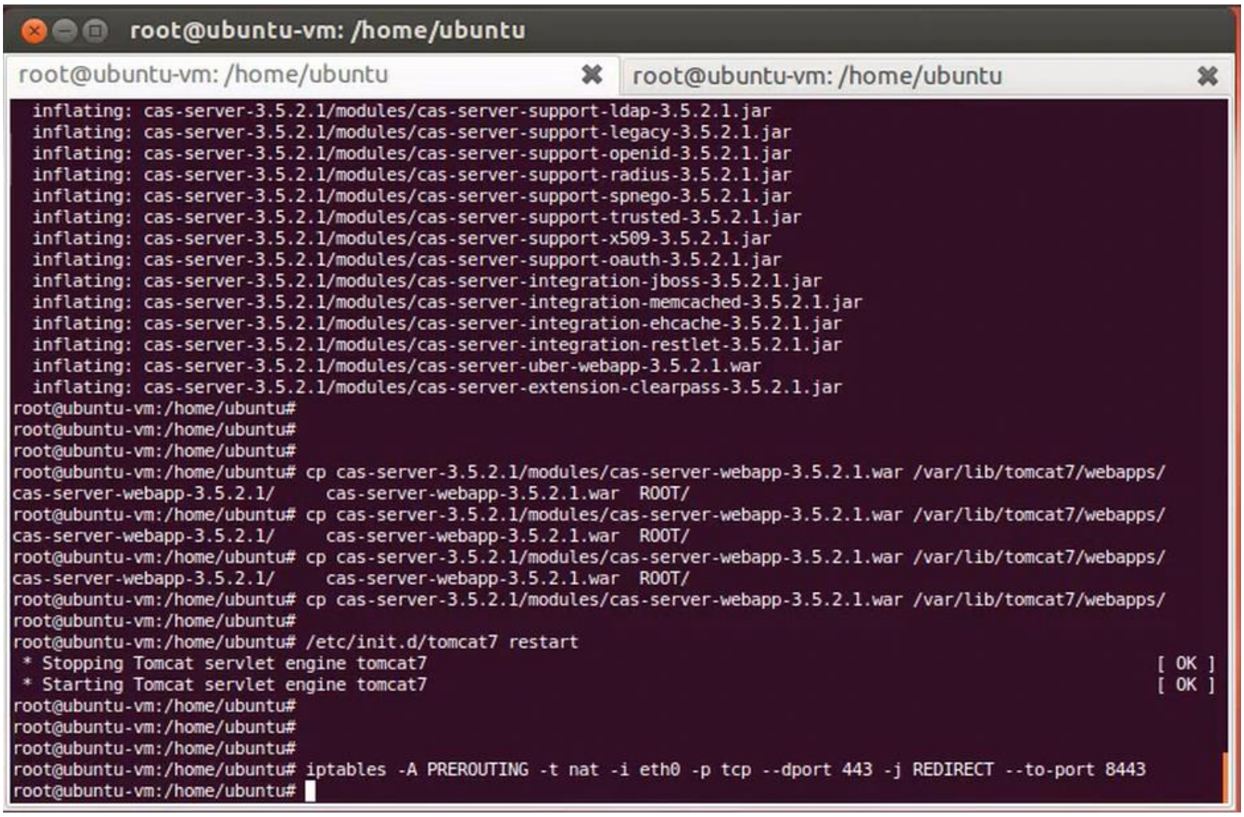

Рисунок 3.9 – Перенаправление входящих пакетов на порт 8443

После всех проделанных действий нужно открыть окно браузера и перейти по адресу "https://localhost:8443/CAS – server – webapp – 3.5.2.1". В открывшемся окне будет форма авторизации CAS. Их можно увидеть на рис.3.10 и 3.11.

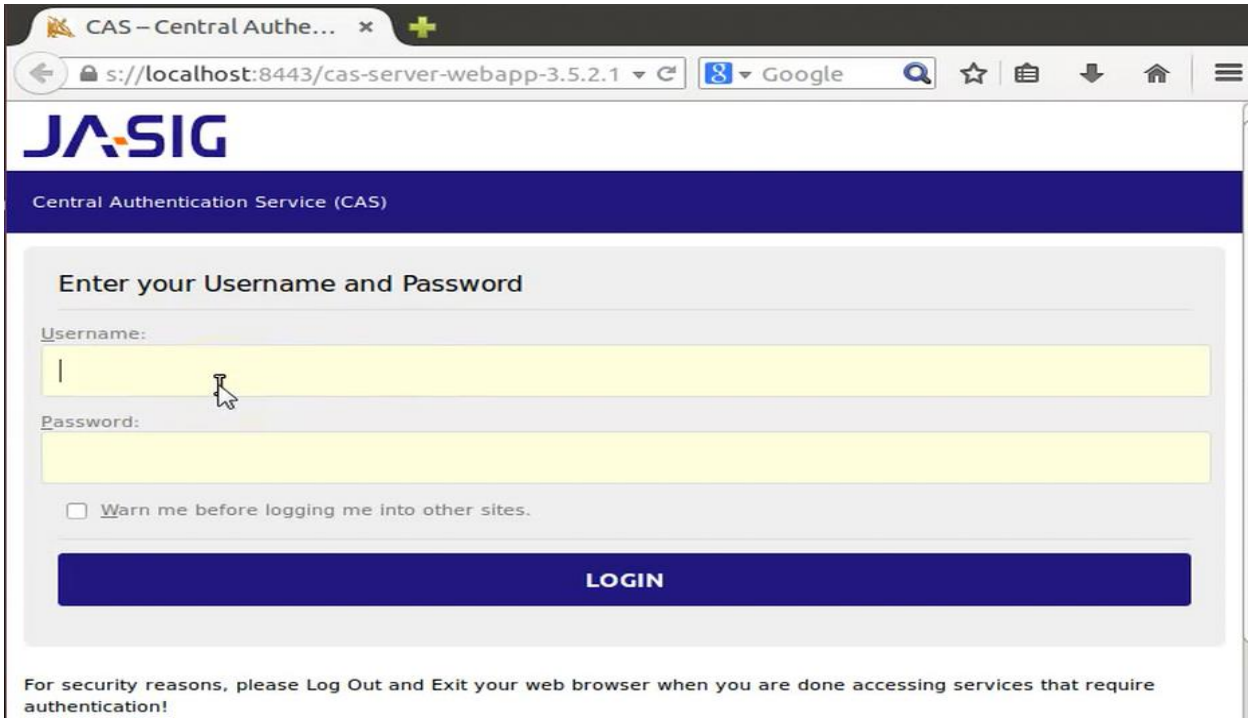

Рисунок 3.10 – Окно ввода логина – пароля

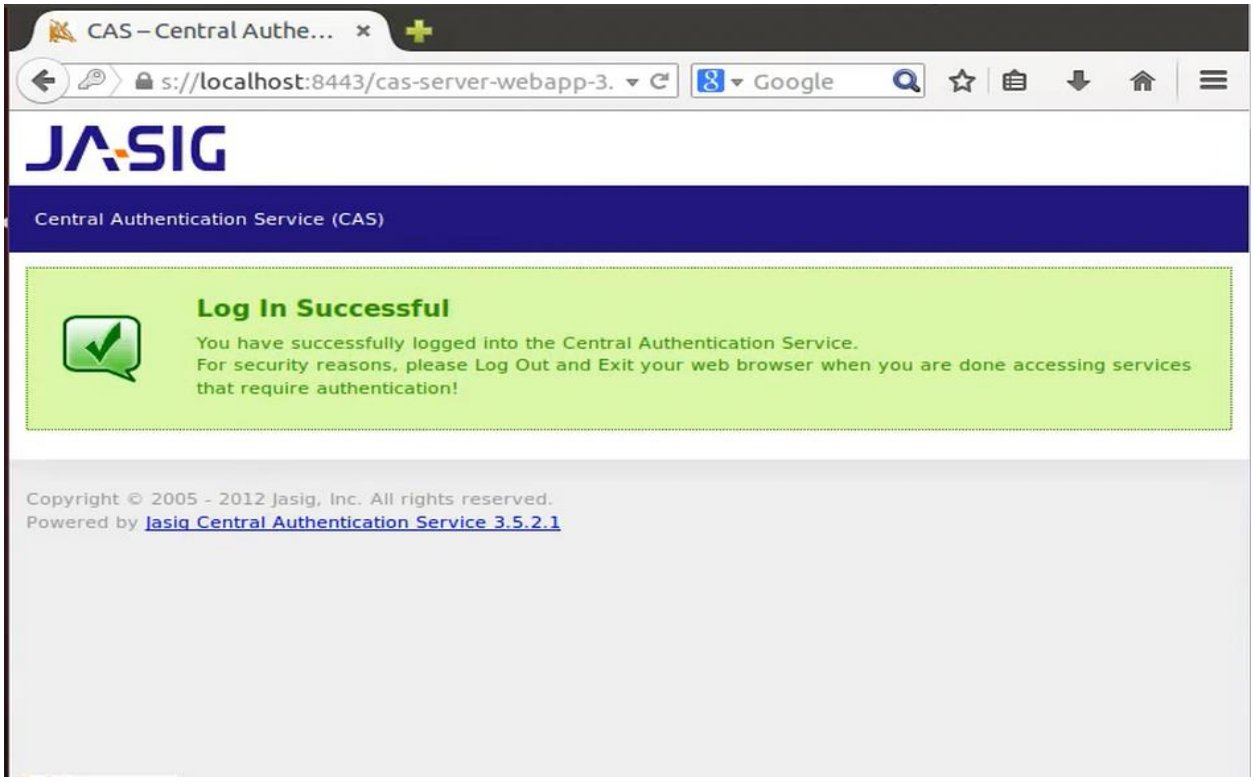

Рисунок 3.11 – Результат успешной аутентификации на сервере CAS

3.2 Настройка Jasig CAS

Первым делом, нам нужно установить и развернуть LDAP сервер. Для этого нам необходимо прописать команду "apt – get iinstall slapd ldap – utils"

Процесс ввода представлен на рис. 3.12.

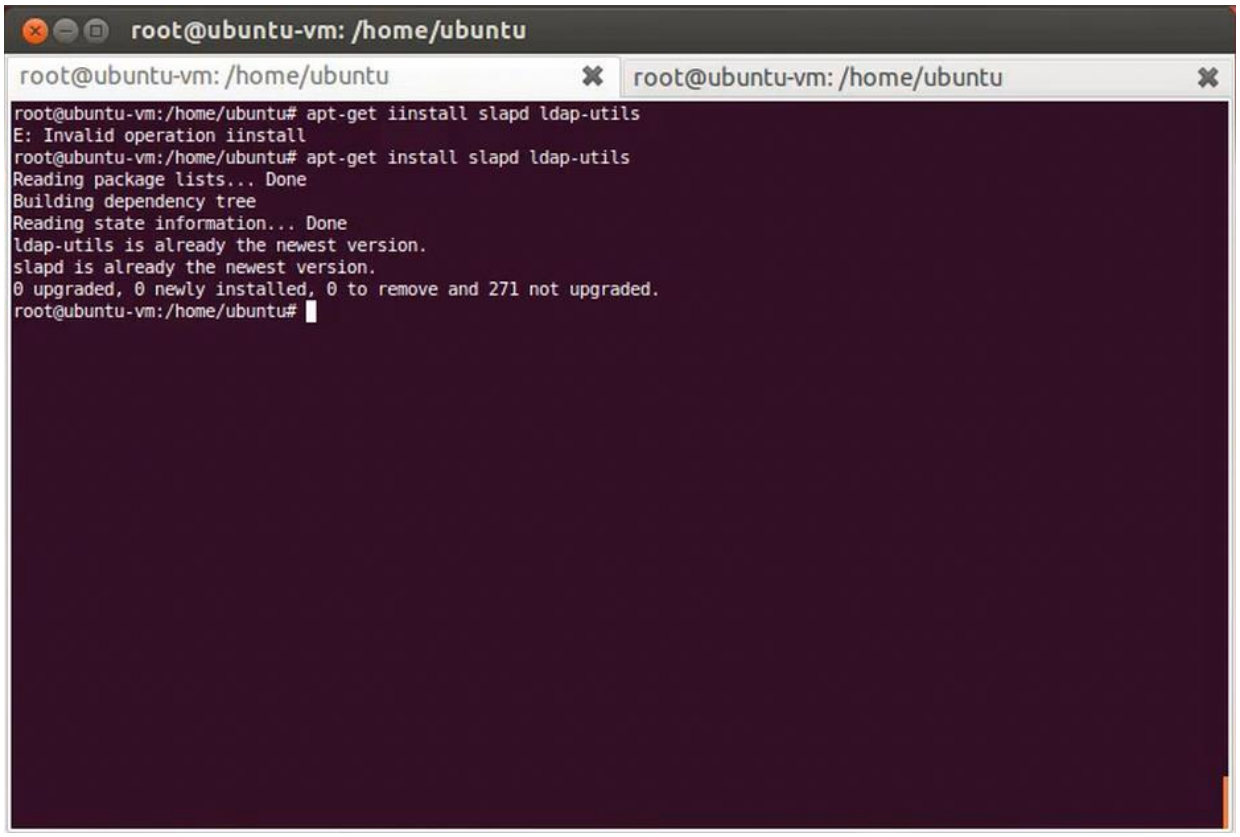

Рисунок 3.12 – процесс установки и развертки Ldap

Затем приступим к ее настройке. Для этого введем команду "dpkg – reconfigure slapd". Процесс ввода команды представлен на рис. 3.13.

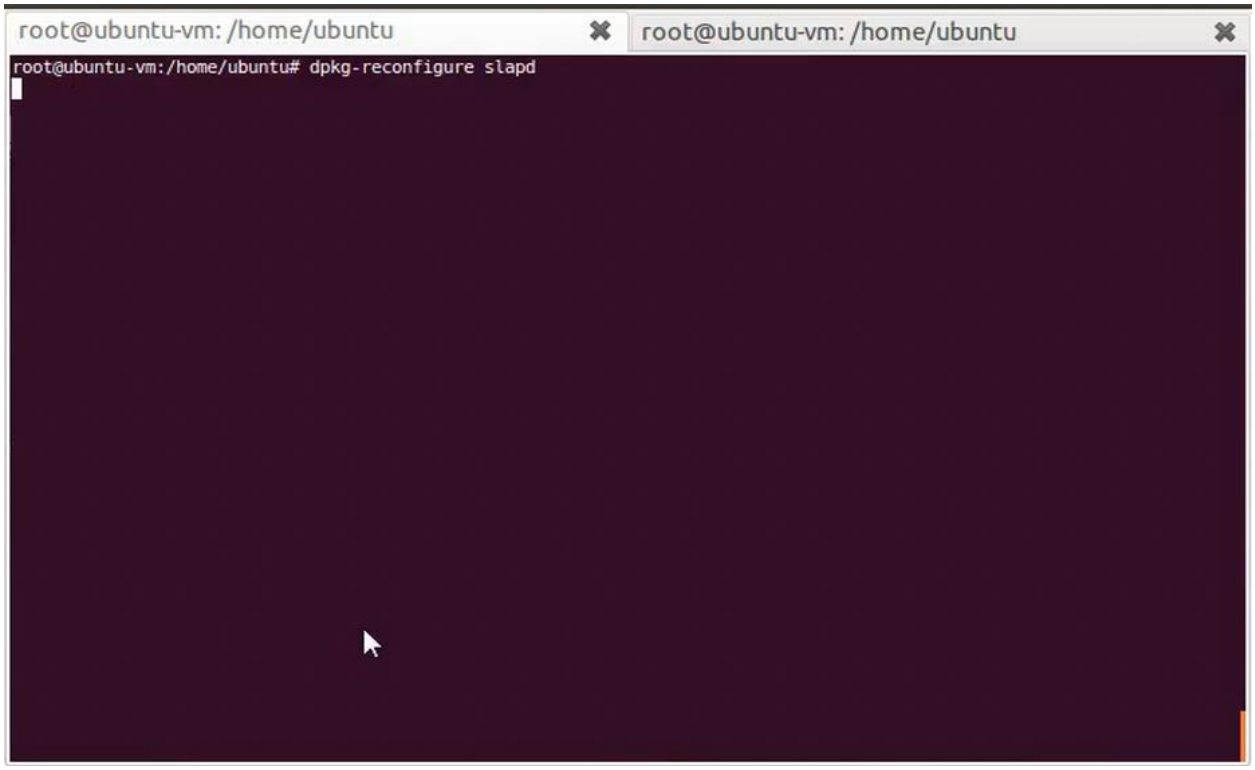

Рисунок 3.13 – Ввод команды на реконфигурацию ldap в консоль Linux

Во всплывшем окне нажмем подтверждение. Окно подтверждения можно увидеть на рис.3.14.

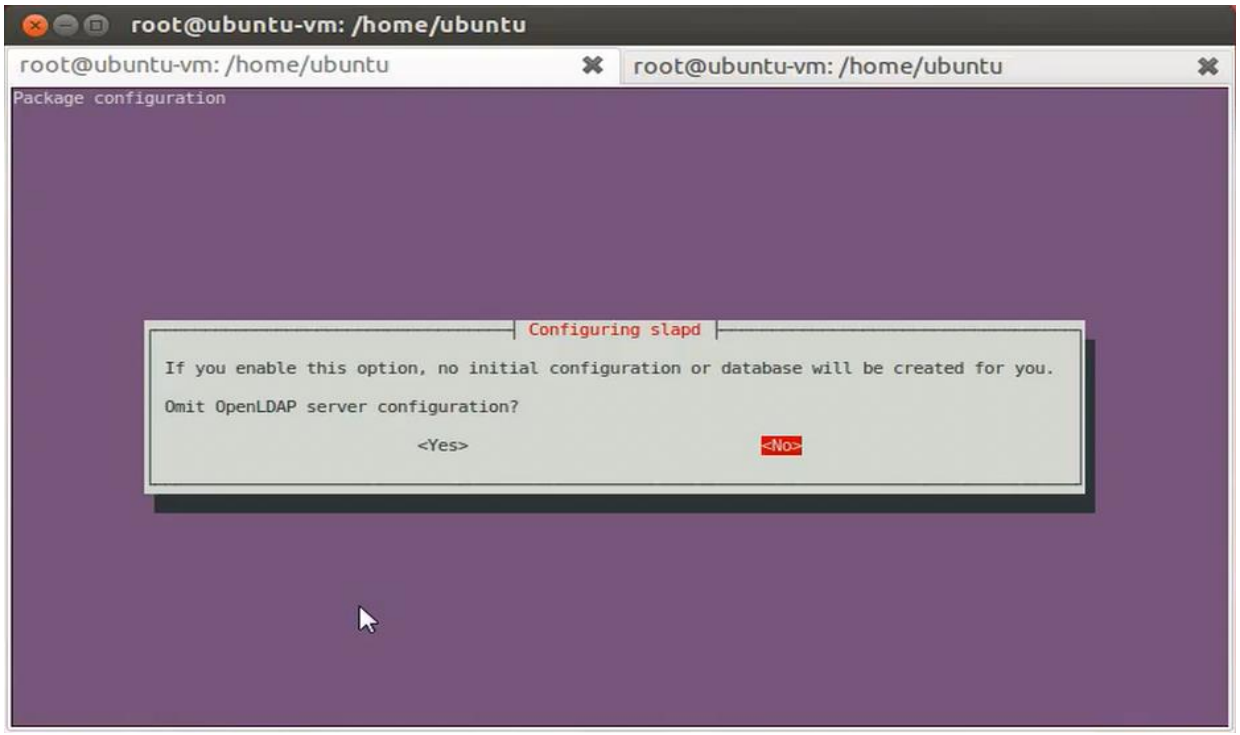

Рисунок 3.14 – Запрос на подтверждение реконфигурации ldap

Задаем имя домена. Продемонстрировано на рис. 3.15.

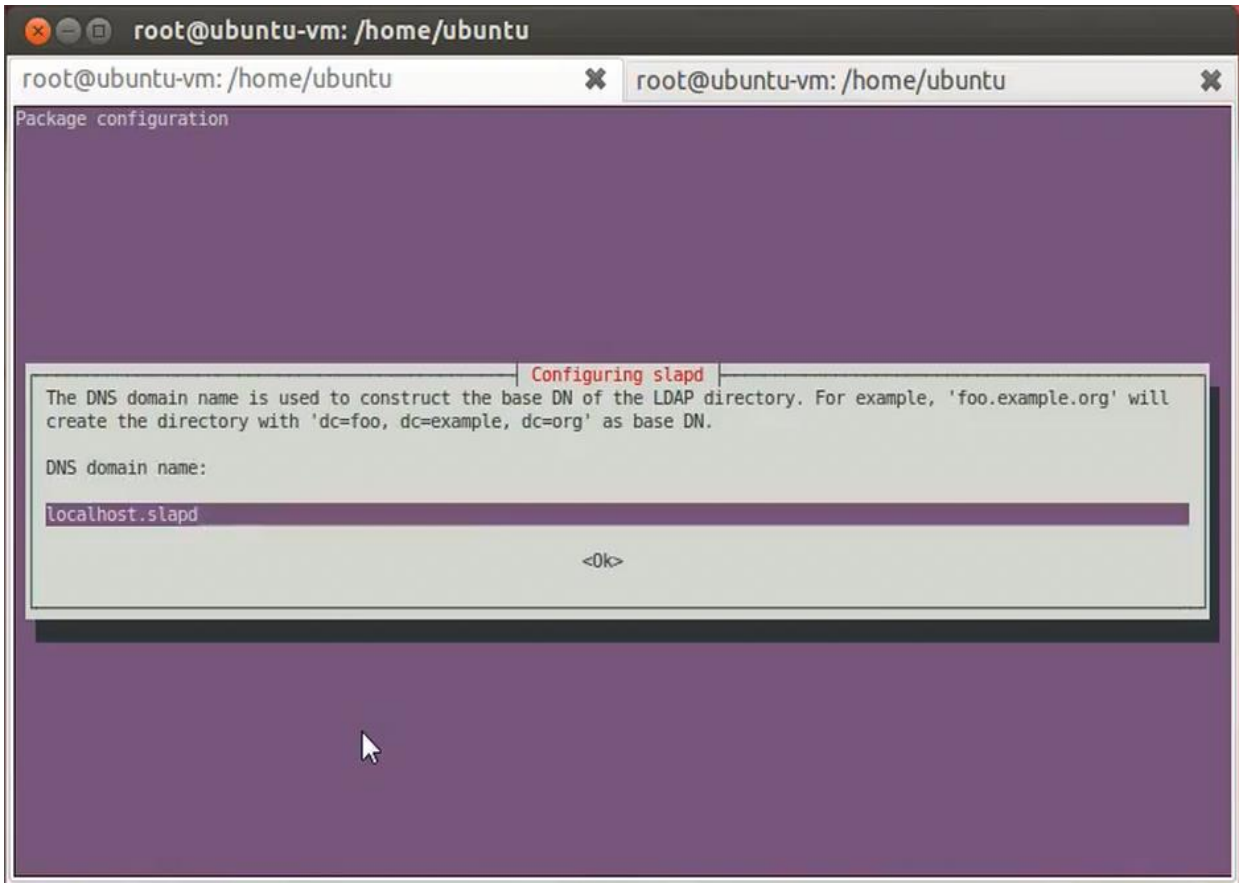

Рисунок 3.15 – Выбор имени домена

После ввода всех паролей нам требуется выбрать какую из разновидностей баз данных выбрать. Выбор предоставлен из двух разновидностей, поддерживаемых LDAP. В связи с тем, что настройка BDB достаточно трудозатратна воспользуемся HDB. Выбор представлен на рис. 3.16.

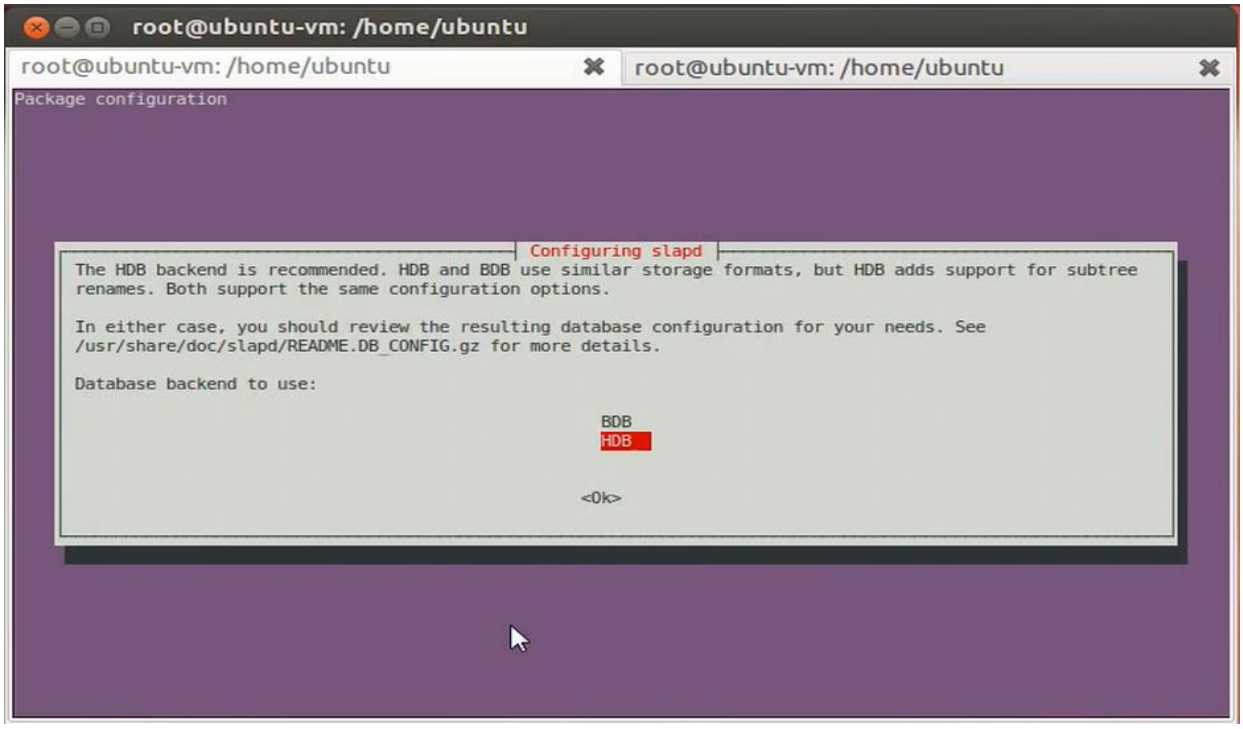

Рисунок 3.16 – Выбор базы данных для использования

Требуется подтвердить перенос базы. Подтверждение на рис. 3.17.

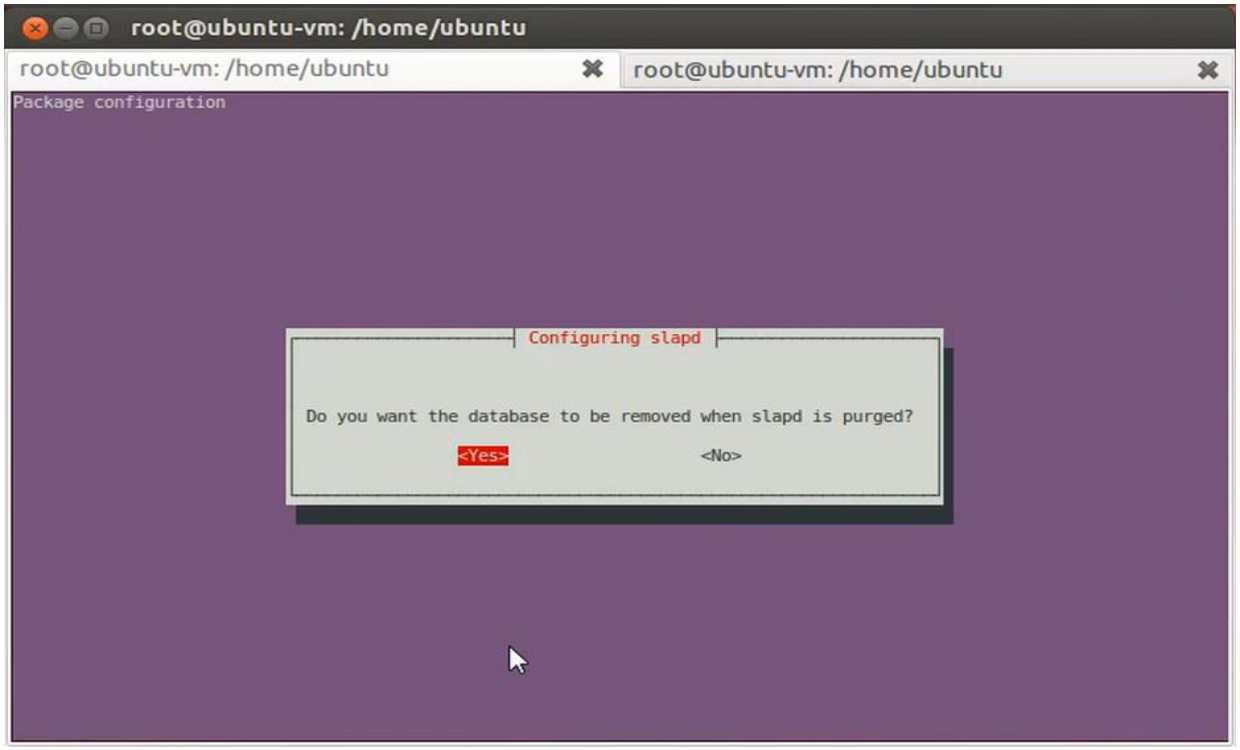

Рисунок 3.17 – Подтверждение переноса базы

Поскольку, старая баз данных является пустой – удаляем ее см. рис. 3.18.

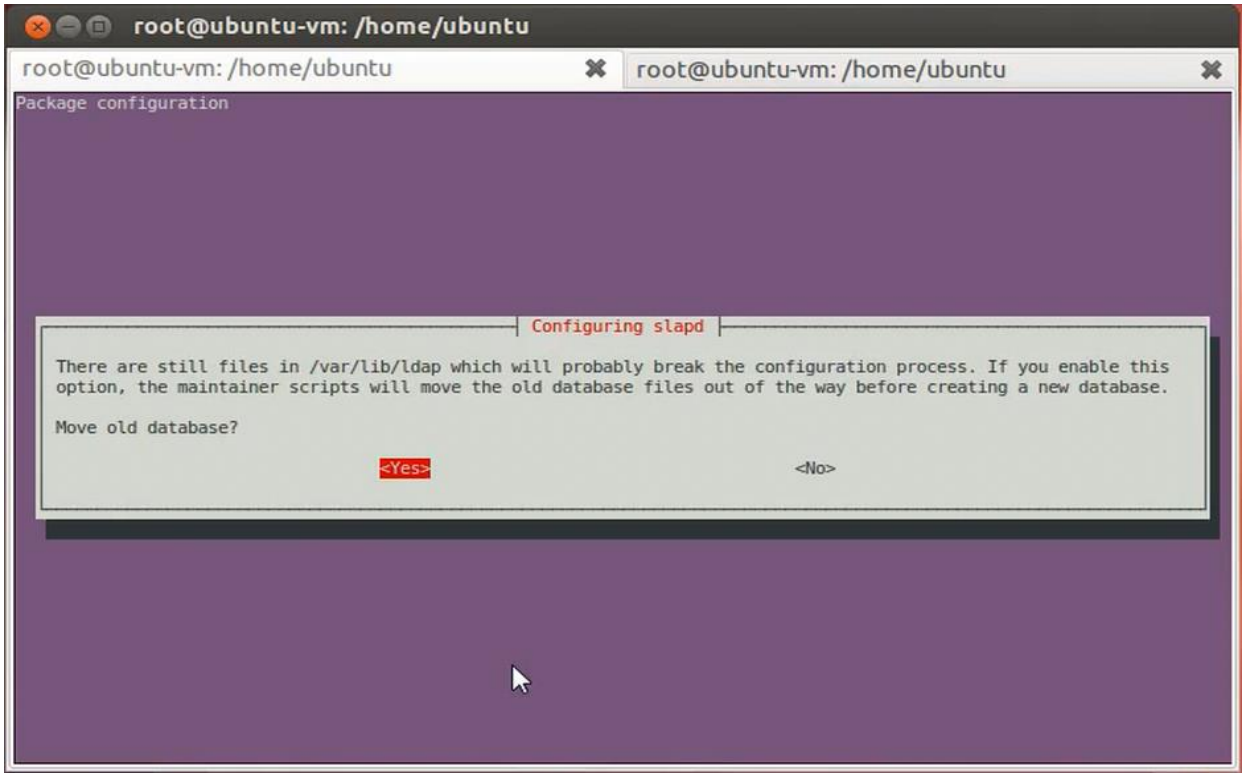

Рисунок 3.18 – Удаление старой базы данных

Повторяем реконфигурирование. Показано на рис. 3.19.

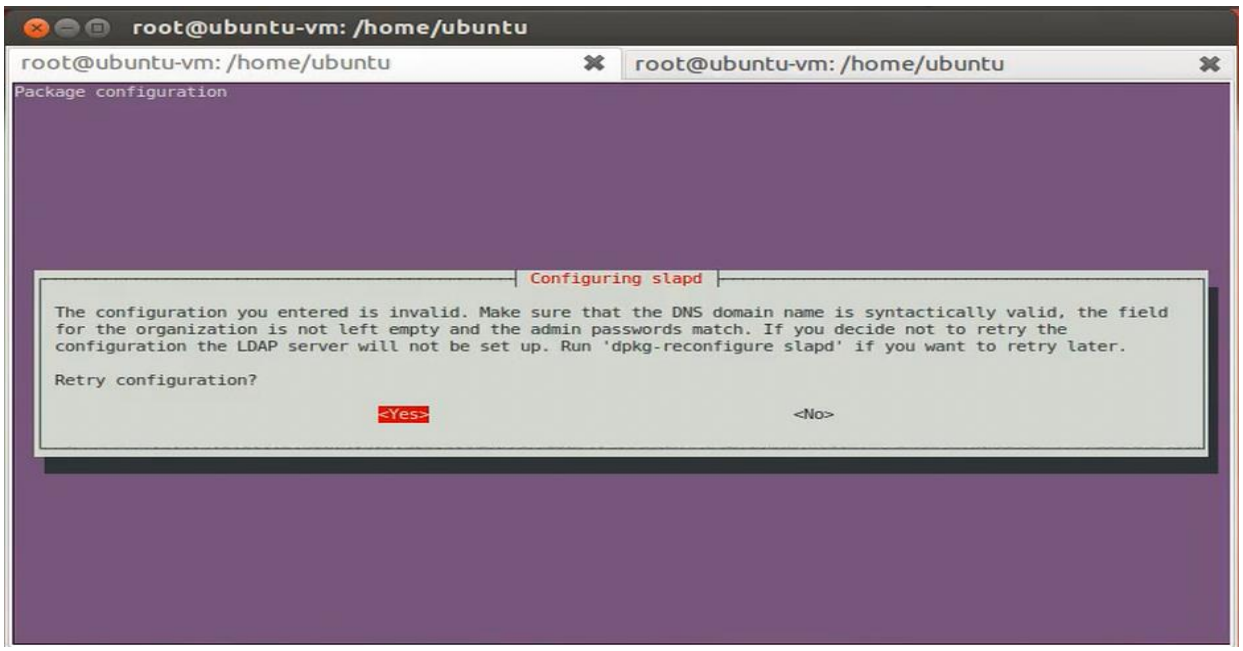

Рисунок 3.19 – Повторение процедуры реконфигурирования

После реконфигурации нам необходимо перезапустить службу. Перезапуск службы показан на рис. 3.20.

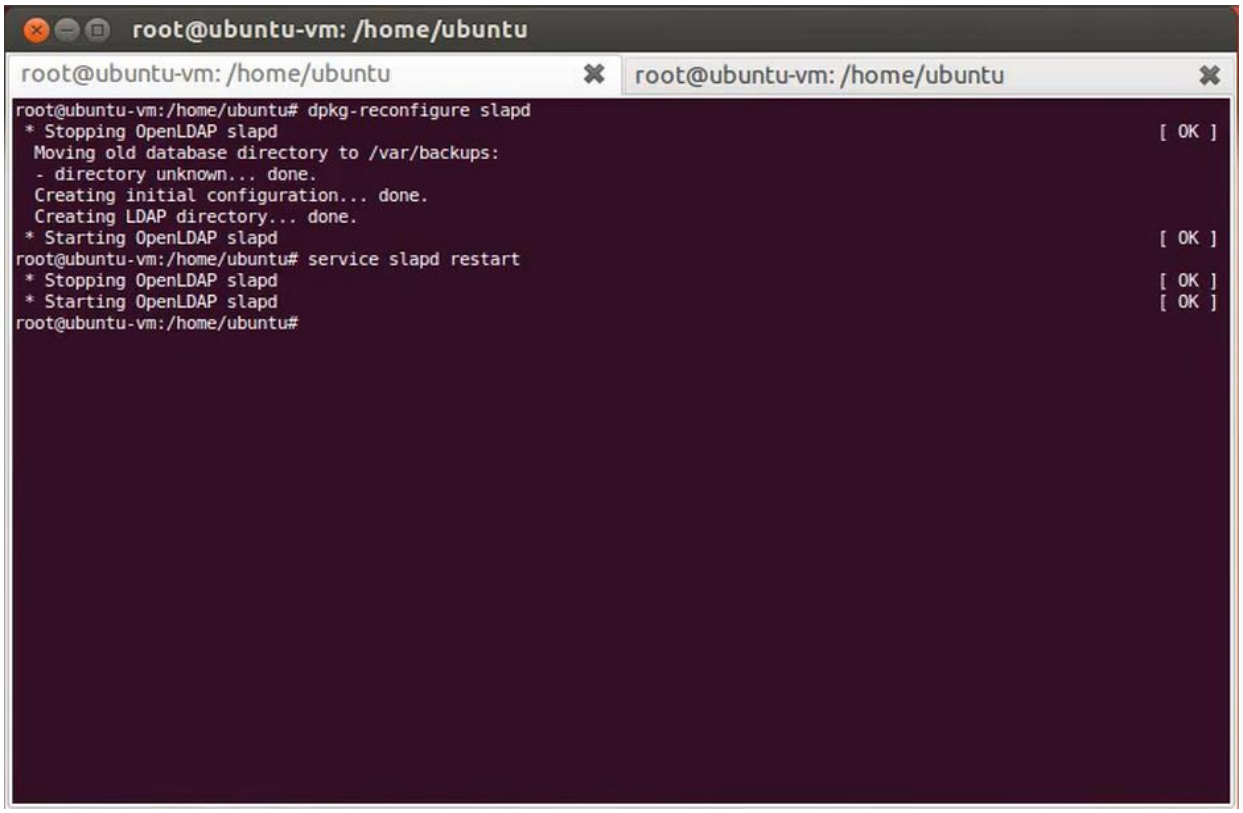

Рисунок 3.20 – Перезапуск служб

Убедимся, что перезапуск прошел успешно и все данные введены корректно. Для этого используем команду “ldapsearch –x –b dc=localhost, dc=slapd”. Использование команды демонстрируется на рис. 3.21. Команда позволит нам посмотреть подключения в которых задействована ldap, а так же покажет работает ли она в данный момент.

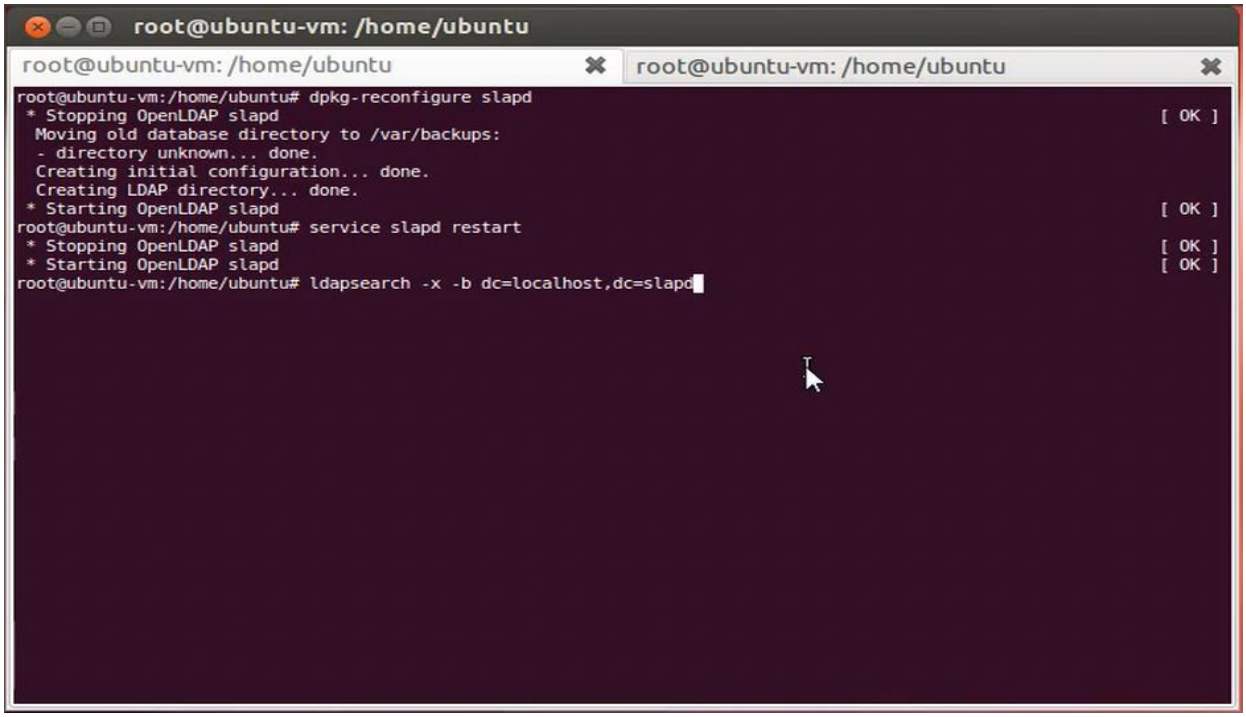

Рисунок 3.21 – Проверка данных на корректность

Войдем в Ldif – файл. Процесс входа показан на рис. 3.22.

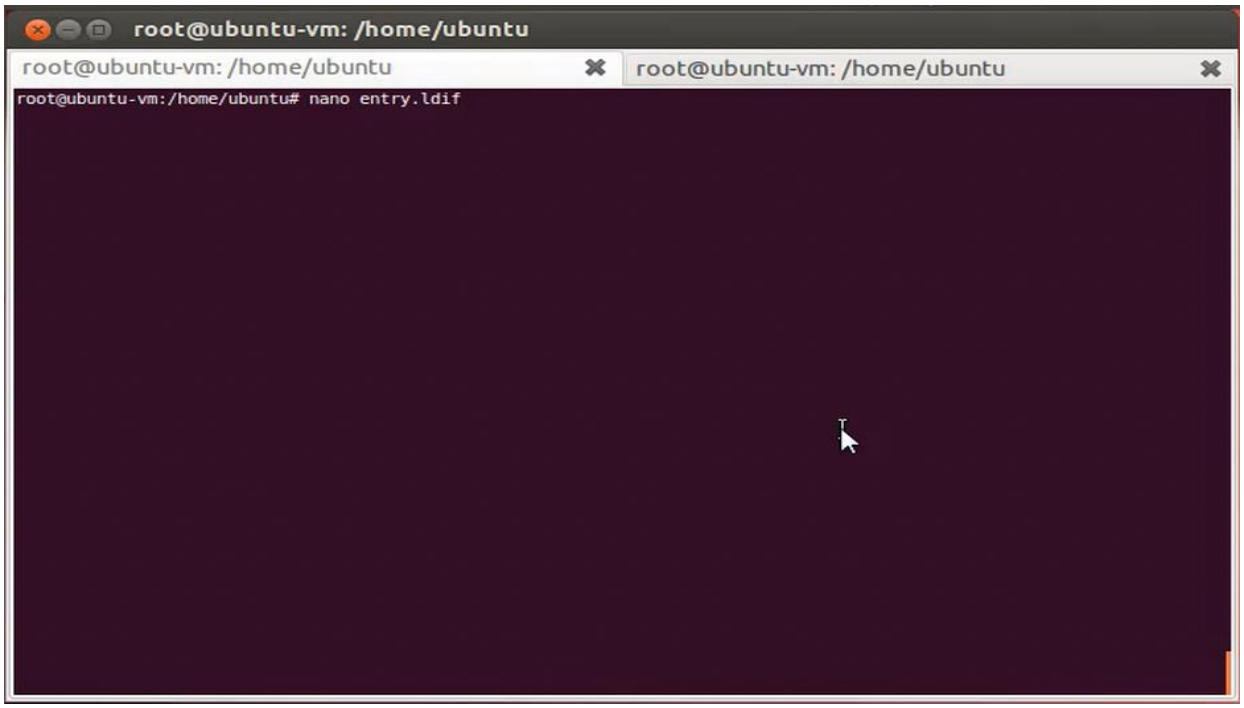

Рисунок 3.22 – Вход в ldif – файл.

Проверяем данные на соответствие ожидаемым значениям.

Содержимое ldif – файла продемонстрировано на рис. 3.23, 3.24.

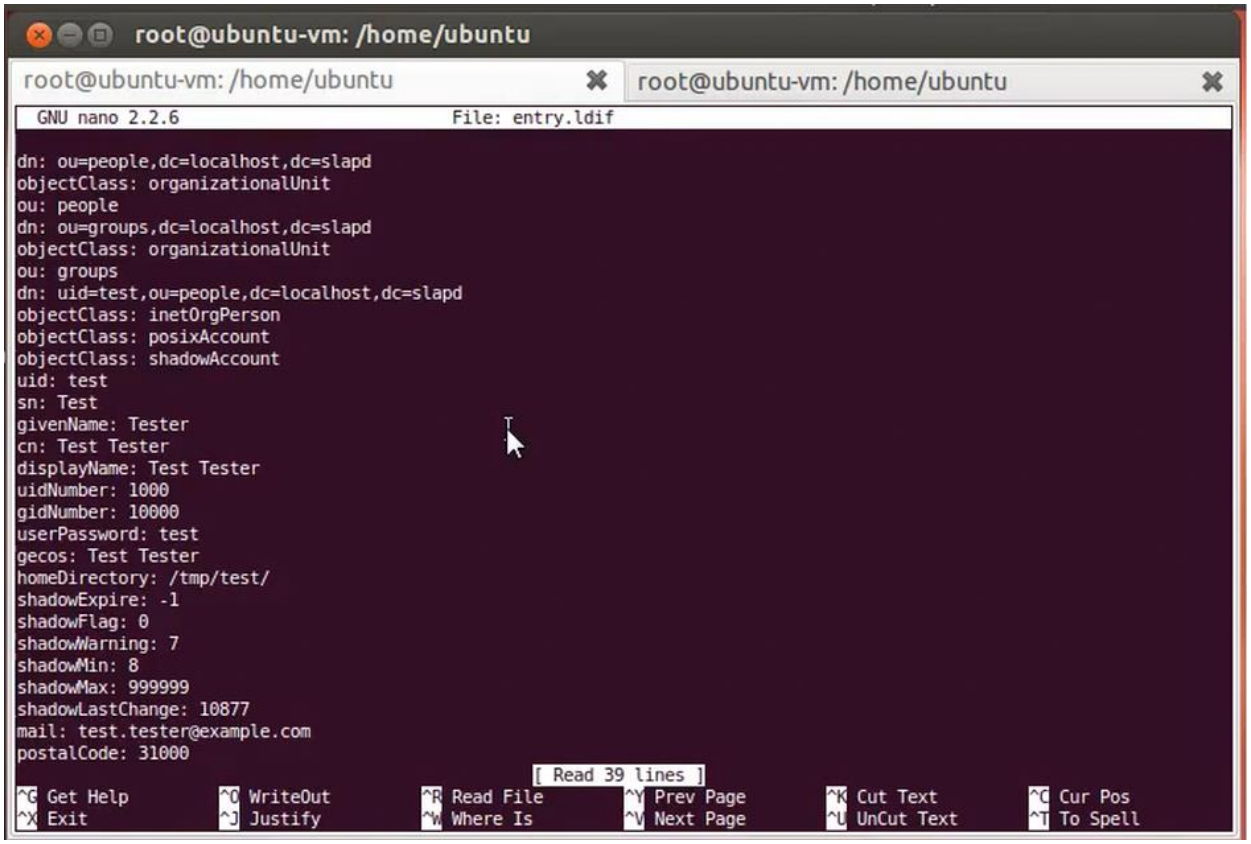

Рисунок 3.23 – Проверка первого каталога на корректность

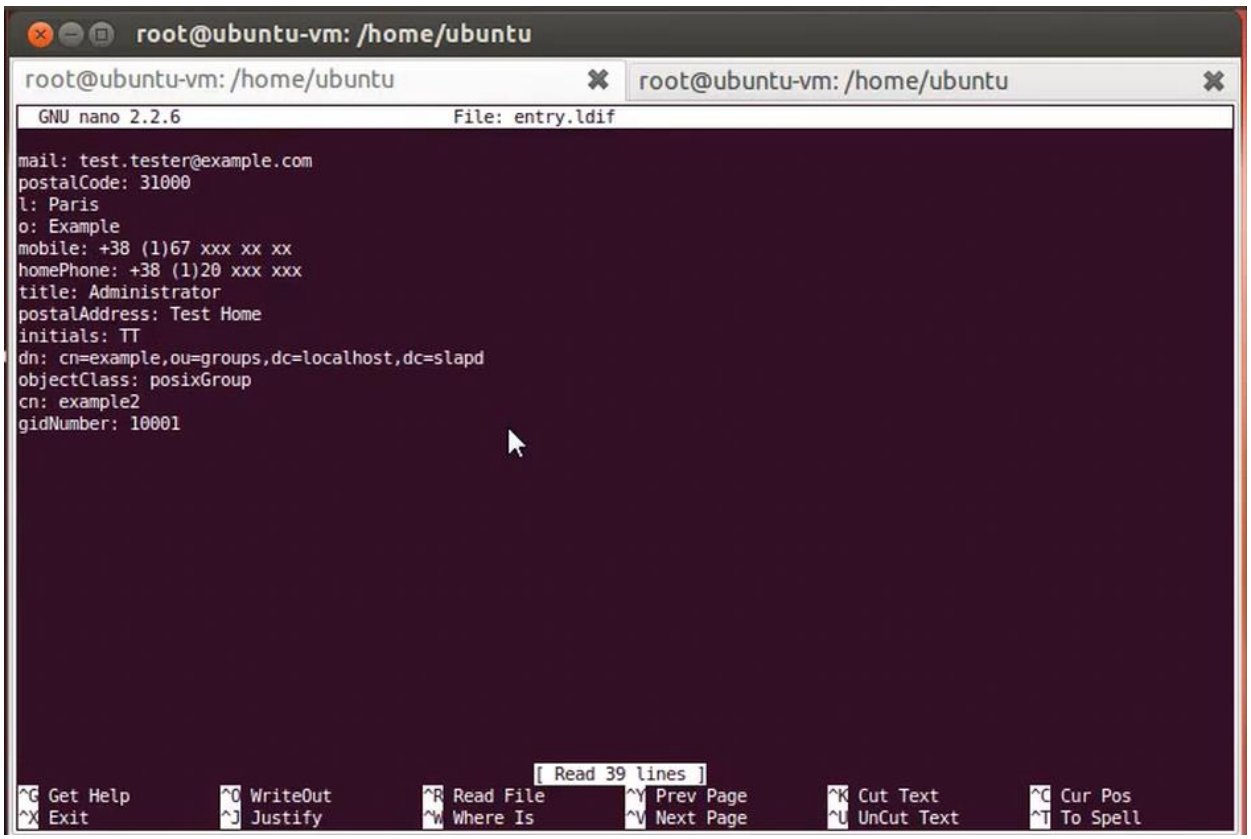

Рисунок 3.24 – Проверка второго каталога на корректность

Перезапускаем сервис. Процедура перезапуска на рис. 3.25.

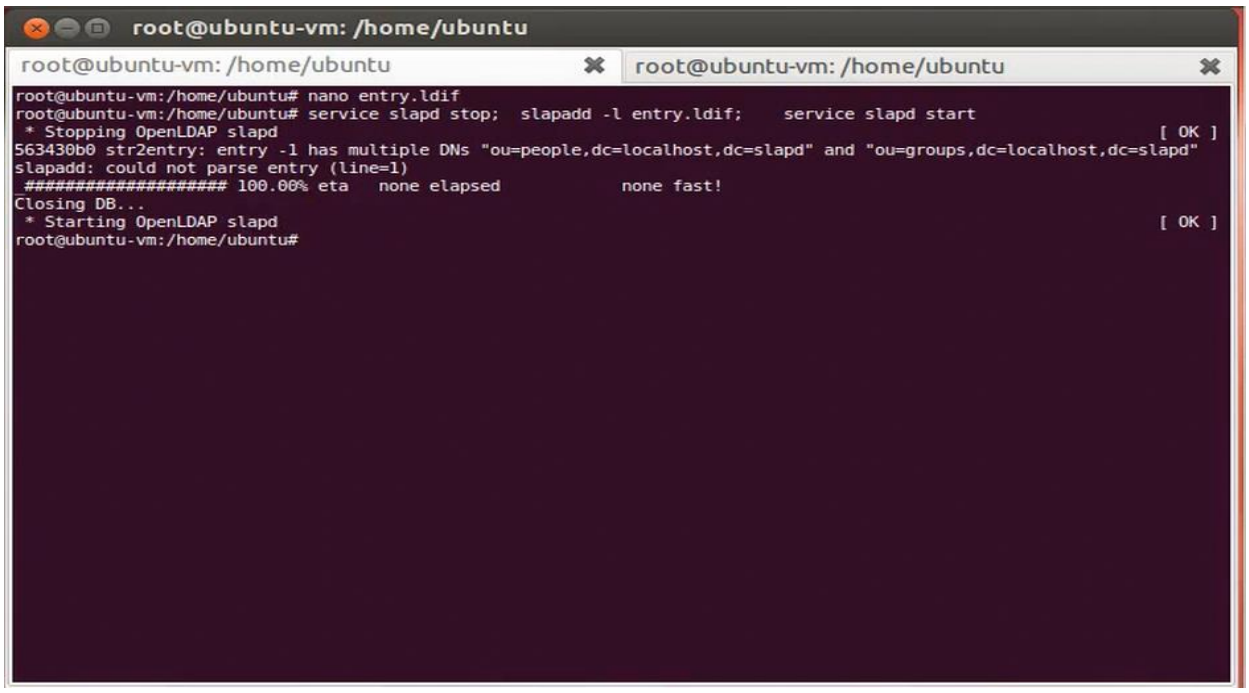

Рисунок 3.25 – Перезапуск сервиса

Еще раз проверяем успешность введенных команд. Процесс показан на рисунке 3.26.

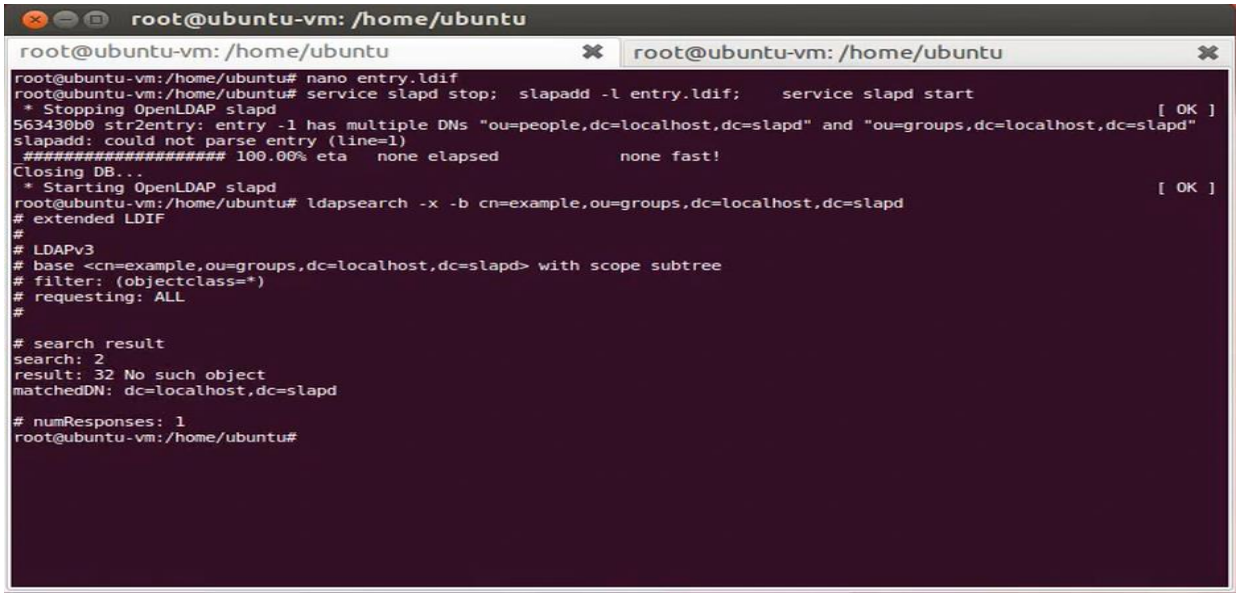

Рисунок 3.26 – дополнительная проверка корректности введенных

команд

Далее нам необходим Apache 2 сервер. Apache 2 сервер является одним из наиболее распространенных и надежных серверов для системы Linux. Ставим и запускаем его. Процесс установки Apache – сервера на рис. 3.27.

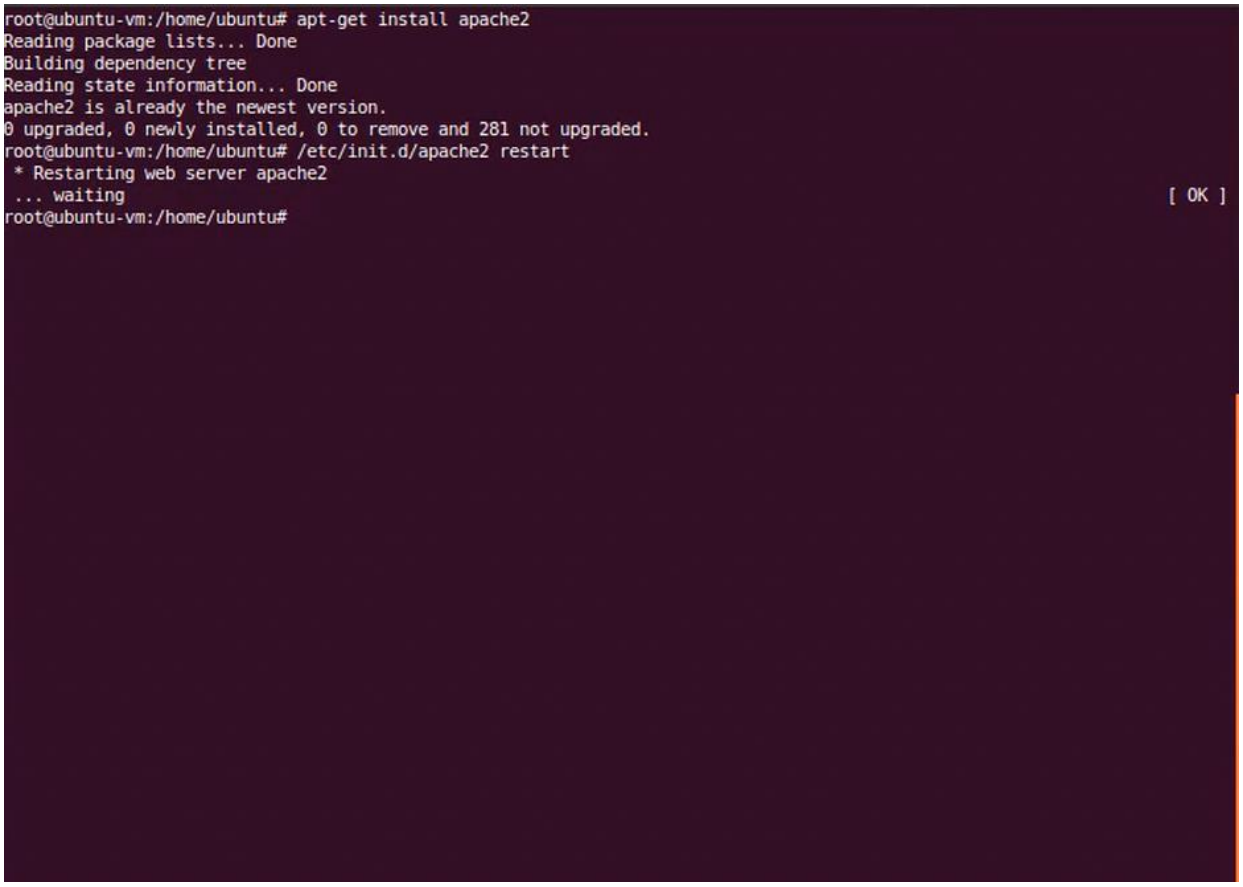

Рисунок 3.27 – Установка Apache2

Далее поставим mysql. MySql – РСУБД, поддерживающаяся корпорацией Oracle является непревзойденным стандартом в своей области. Процесс установки на рис. 3.28.

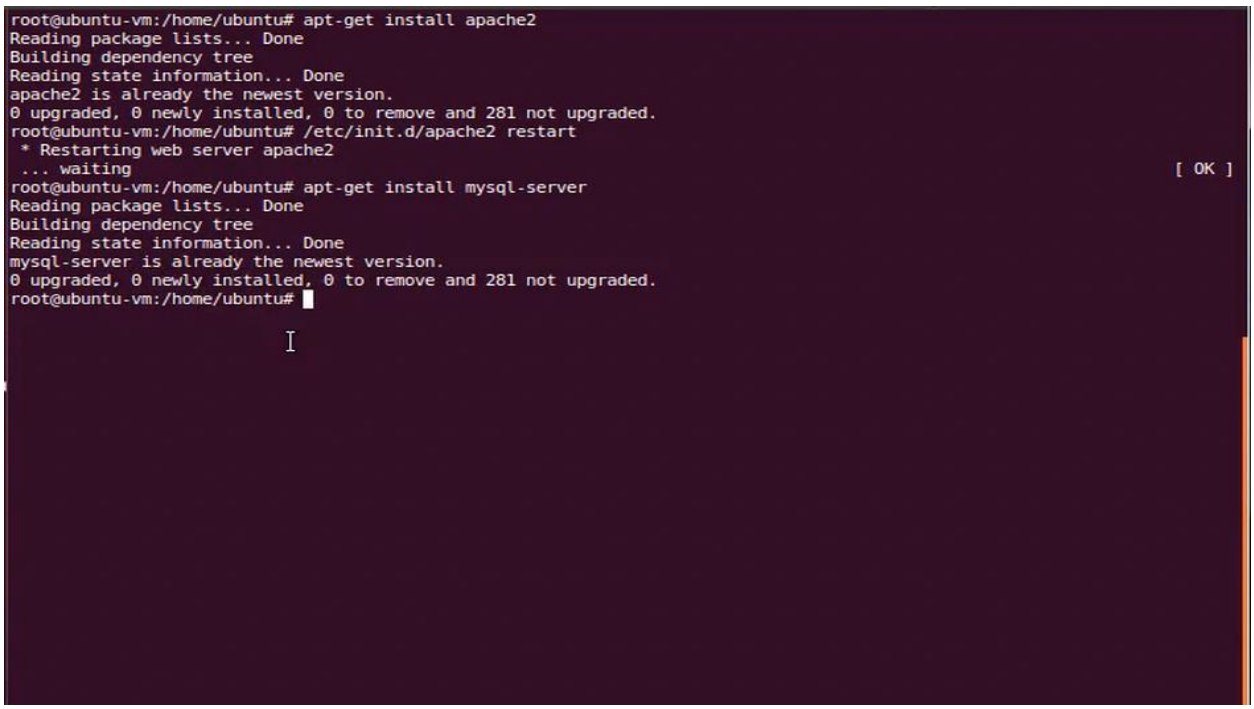

Рисунок 3.28 – процесс установки Mysql

Далее зайдем в MySql монитор. На рисунке 3.29 представлена внутренняя структура MySql монитора.

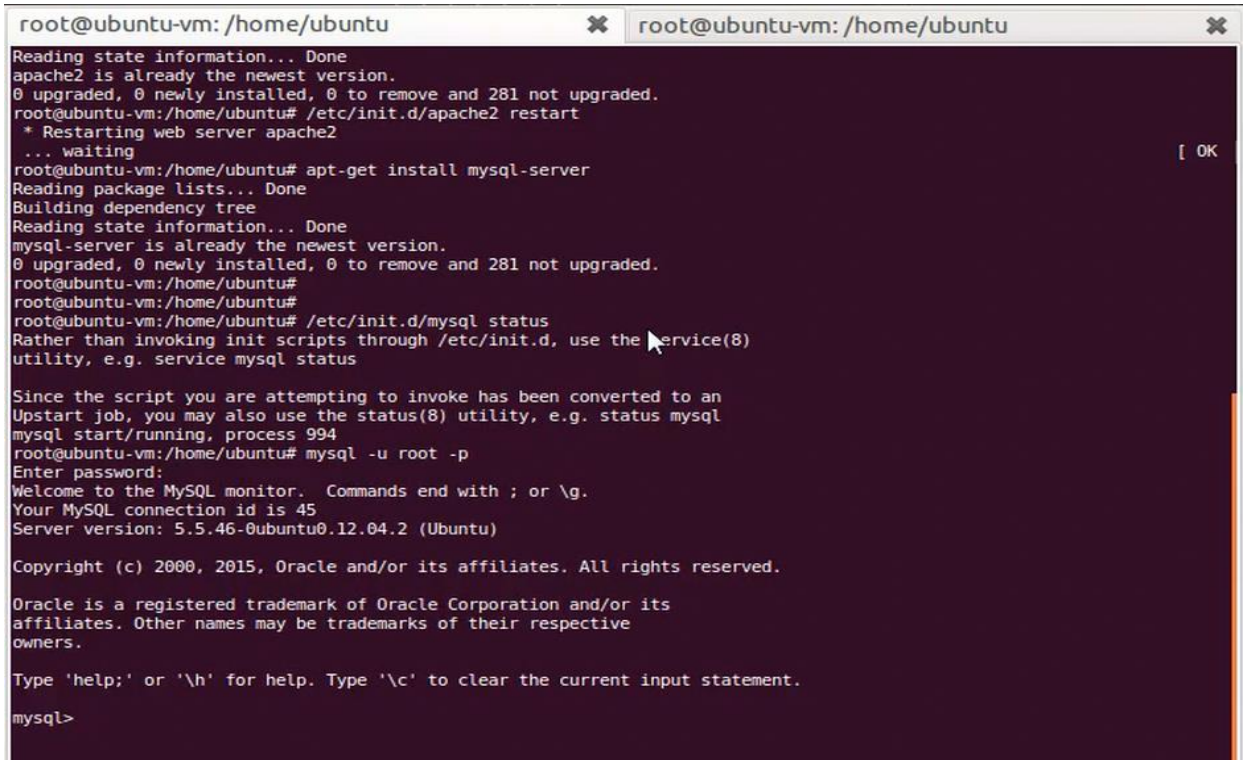

Рисунок 3.29 – Внутренняя структура утилиты MySql монитор

Устанавливаем пакет php5 – mySql. Установка MySql на рис. 3.30.

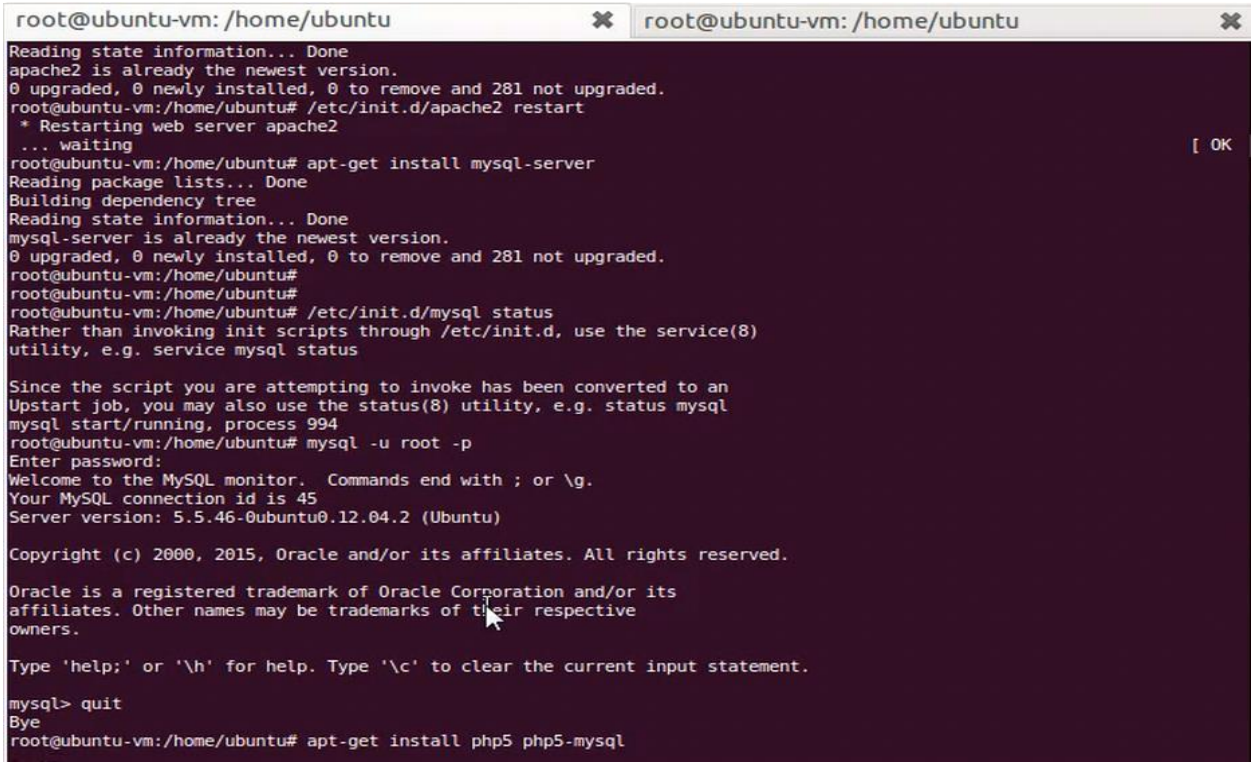

Рисунок 3.30 – устанавливаем пакет php5 – mySql

Проверяем успешность соединения с базой данных mySql см. рис. 3.31.

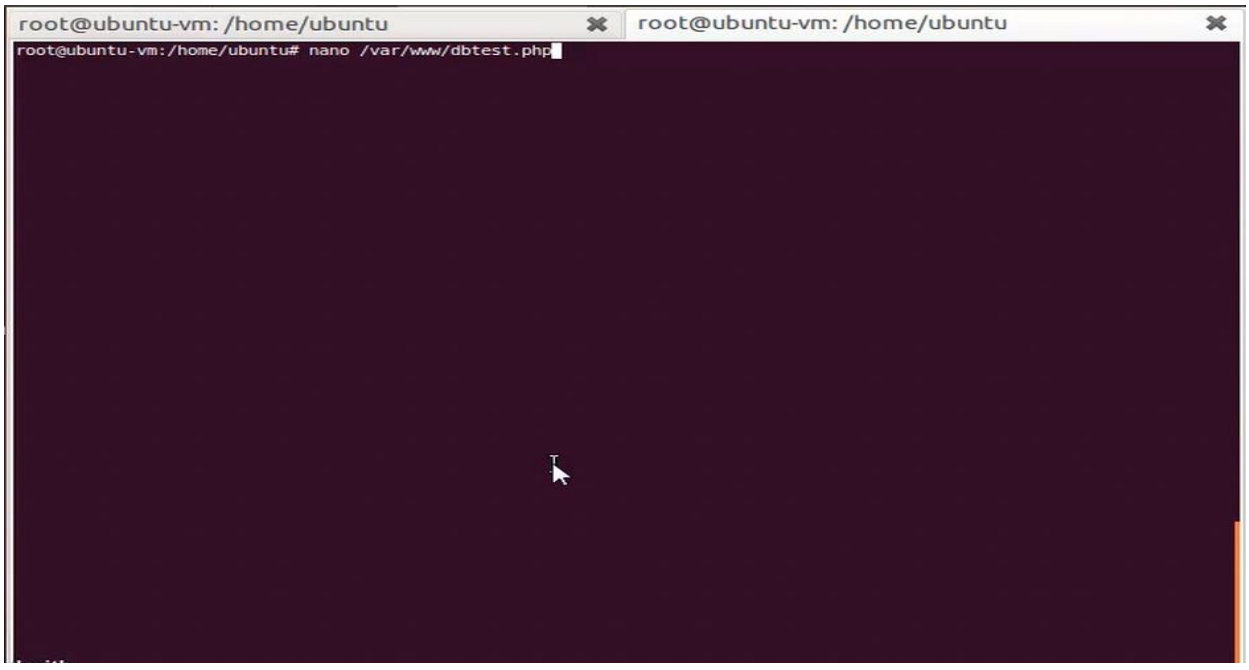

Рисунок 3.31 – Проверка успешности соединения с базой данных mySql

Соединение установлено успешно. Успешность соединения демонстрируется на рис. 3.32.

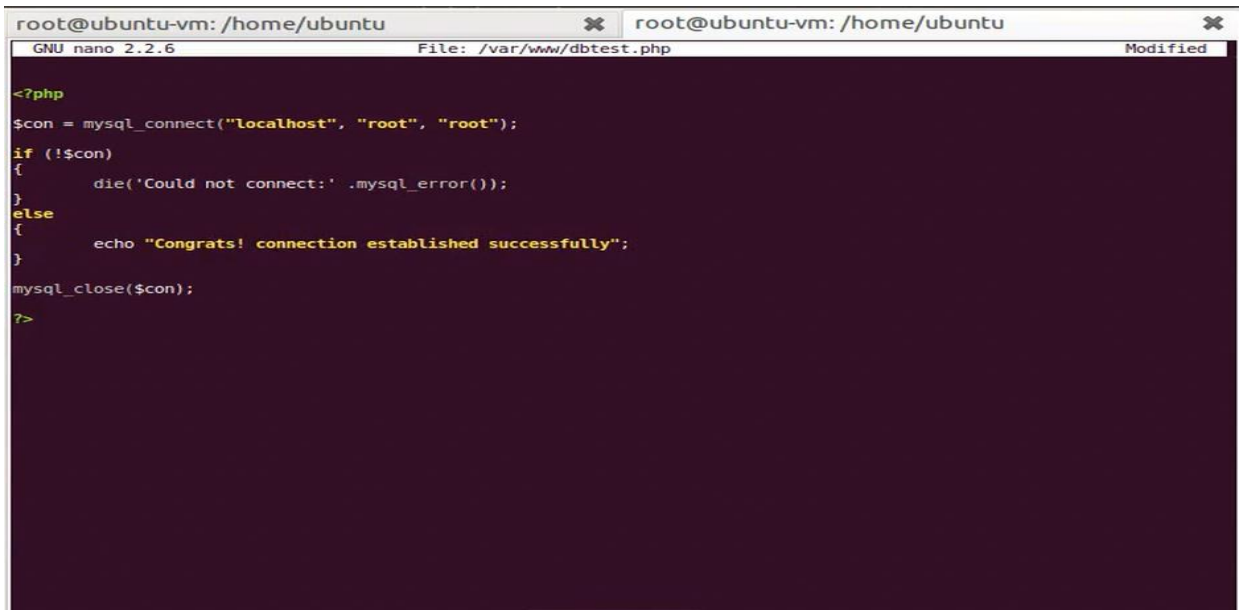

Рисунок 3.32 – Демонстрация результатов соединения

После всех этих процедур устанавливаем PHP – Pear. Процесс установки на рис. 3.33. Pear – это структурированная библиотека исходного кода. Ее основная цель – предоставить компоненты многократного использования.

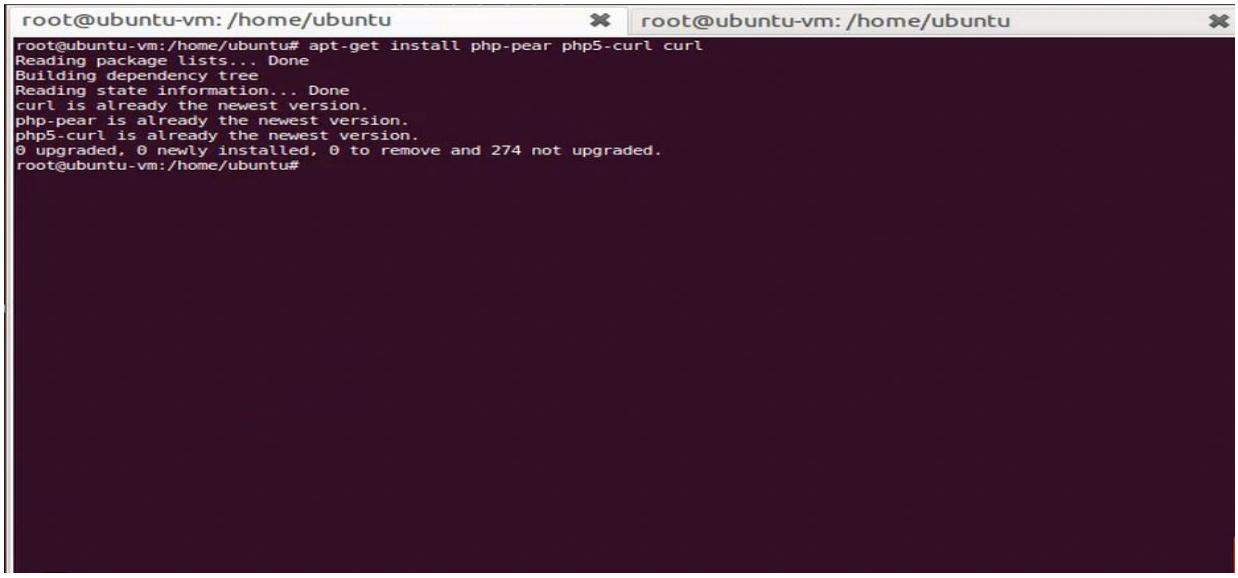

Рисунок 3.33 – Процесс установки phph – pear

Устанавливааем CAS – клиент. Установка показана на рис. 3.34.

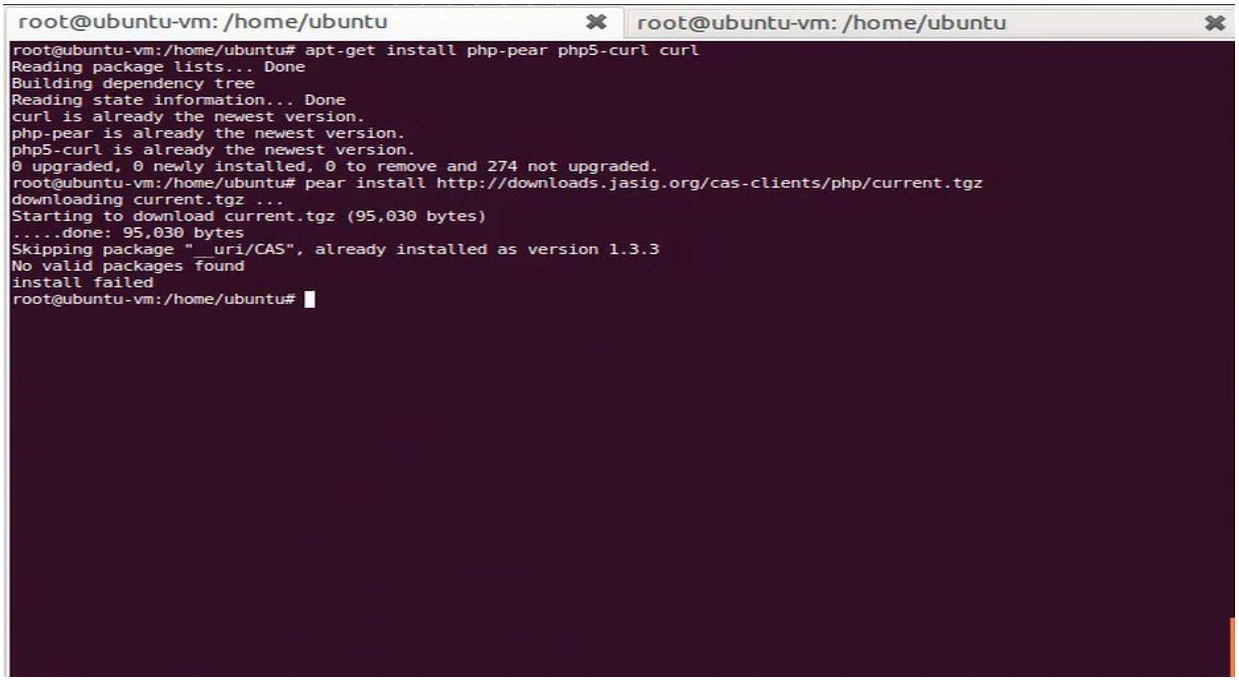

Рисунок 3.34 – Процесс установки CAS – клиента.

Логинимся в CAS. Процедура входа в CAS рис. 3.35.

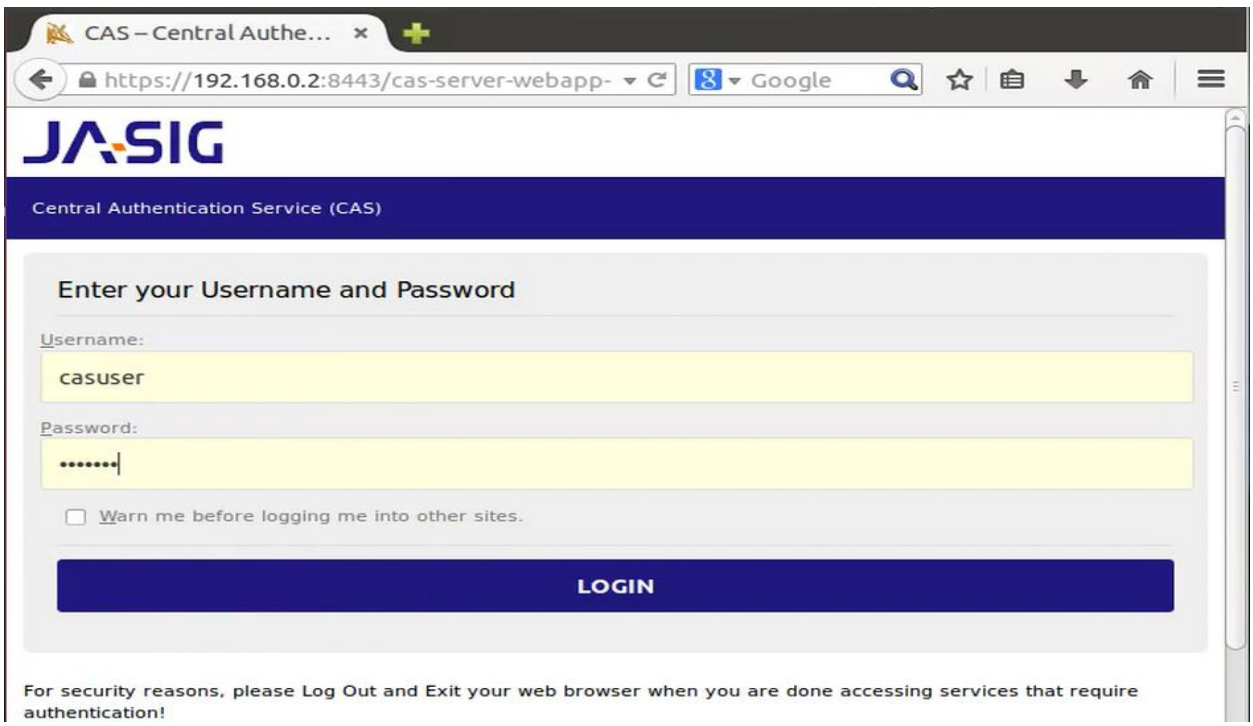

Рисунок 3.35 – Процесс входа в CAS

Вход успешен. Успешный вход демонстрируется на рис. 3.36.

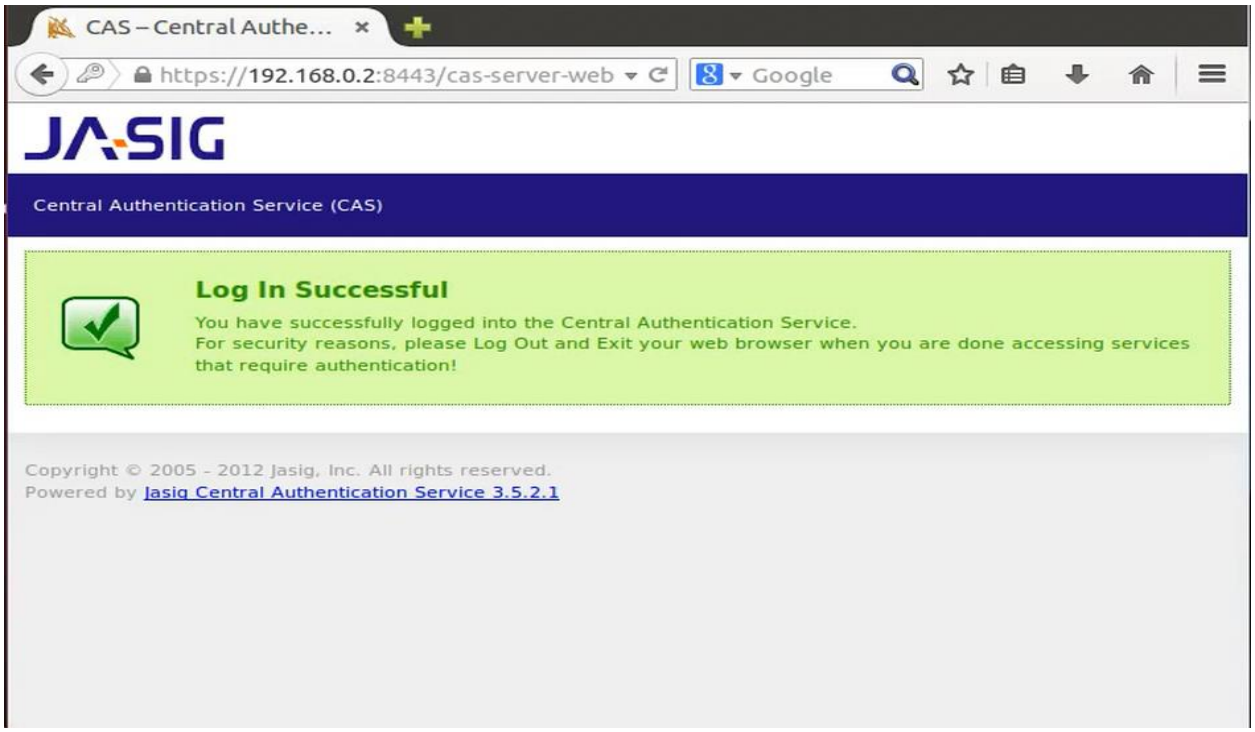

Рисунок 3.36 – Подтверждение корректности выполненных действий

Теперь нам нужно подключить надстройку, позволяющую воспользоваться сервером аутентификации.

Необходимо создать php файл, который будет отвечтаь за выполнение основных функций нашего сервера.

Загружаем настройки с центрального файла конфигурации

```
require_once 'config.php';
```

Подгружаем необходимые библиотеки

```
require_once $phpcas_path . '/CAS.php';
```

Запускаем phpCAS

```
phpCAS::client(CAS_VERSION_2_0, $cas_host, $cas_port, $cas_context);
```

Запускаем принудительную аутентификацию через phpCAS

```
phpCAS::forceAuthentication();
```

Инициализируем выход по запросу

```
if (isset($_REQUEST['logout'])) {
```

```
phpCAS::logout();
```

Проверяем, корректность настройки. Для этого используем данный код

```
<html>
  <head>
    <title>phpCAS simple client</title>
  </head>
  <body>
    <h1>Successfull Authentication!</h1>
    <?php require 'script_info.php' ?>
    <p>the user's login is <b><?php echo phpCAS::getUser(); ?></b>.</p>
    <p>phpCAS version is <b><?php echo phpCAS::getVersion(); ?></b>.</p>
    <p><a href="?logout=">Logout</a></p>
  </body>
</html>
```

При получении сообщения «Successful Authentication» настройка считается выполненной корректно.

Заключение

Целью данной выпускной квалификационной работы была разработка и внедрение системы единой аутентификации для Тольяттинского Государственного Университета.

Для достижения поставленной цели были проанализированы и подробно изучены основные технологии и решения, используемые при организации систем единого входа.

Решение было разработано на базе Jasig CAS с применением таких технологий, как LDAP, Apache 2, MySQL.

В результате внедрения удалось значительно повысить эффективность и производительность труда, сократив затраты на многочисленные переходы между сервисами. Так же была устранена проблема множества различных идентификаторов для одного человека.

Список литературы

1. Веллинг Л. Разработка веб-приложений с помощью PHP и MySQL/ Л. Веллинг, Л. Томсон - М.: Вильямс, 2011 – 848 с.

2. Васвани В. Разработка веб-приложений на PHP/ В. Васвани – СПБ.: Питер, 2012. – 432 с.

3. Грофф Р. Д. SQL. Полное руководство. 3-е издание/ Р.Д. Грофф, П.Н. Вайнберг, Э. Д. Оппель – М.: Вильямс, 2015. – 959 с.

4. Дейт К. SQL и реляционная теория. Как грамотно писать код на SQL/ К. Дж. Дейт. – М.: Символ плюс, 2011 – 474 с.

5. Кайт Т. Oracle для профессионалов: архитектура и методики программирования. 3-е изд./ Т. Кайт, Д. Кун. – М.: Вильямс, 2016. – 960 с.

6. Колесниченко Д. PHP и MySQL. Разработка Web-приложений/Д. Колесниченко. – СПБ.: БХВ-Петербург, 2013. – 543 с.

7. Коробко И. PowerShell как средство автоматического администрирования/ И. Коробко – М.: ДМК Пресс, 2015. – 224 с.

8. Котеров Д. PHP 5/ Д. Котеров – СПБ.: БХВ-Петербург, 2011. – 180 с.

9. Роббинс А. Linux. Программирование в примерах/ А. Роббинс – М.: КУДИЦ-Пресс, 2011. – 256 с.

10. Скляр Д. PHP. Рецепты программирования/ Д. Скляр, Р. Трахтенберг – СПб.: Питер, 2015. – 784 с.

11. Зандастра М. PHP. Объекты, шаблоны и методики программирования/ М. Зандстра - М.: Вильямс, 2015. – 576 с.

12. Харингтон Д. PHP. Трюки/ Д. Харингтон. – СПБ.: Питер, 2011. – 445 с.

13. Чижиков Д. Методология внедрения Microsoft Active Directory/ Д. Чижиков - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2011. – 168 с.

14. Янк К. PHP и MySQL. От новичка к профессионалу/ К. Янк – М.: Эксмо, 2013. – 384 с.

15. Anita Sobe Single Sign-On in IMS-based IPTV Systems: Towards the interworking of the Generic Bootstrapping Architecture and Liberty Alliance Identity Federation. 1-st edition, National Security Agency, 2011. – 470 c.

16. Duffy Amy, Single Sign-On 169 Success Secrets - 169 Most Asked Questions on Single Sign-On - What You Need to Know. – 1-st edition, HISTORY INK BOOKS, 2014 – 520 c.

17. Gerard Blokdijk, Single Sign-On - Simple Steps to Win, Insights and Opportunities for Maxing Out Success. – 1-st edition, Complete Publishing, 2015 – 215 c.

18. Saroj Subramanian, Oracle EBS 12.2 Single Sign On with Oracle Access Manager 11.1.2.3. – 1-st edition, Saroj Subramanian , 2015. – 420 c.

19. Sonia Bui, Single Sign-on Solution for MYSEA Services. 1-st edition, National Security Agency, 2011. – 340 c.

20. phpCAS examples// https://wiki.jasig.org [Электронный ресурс]: https://wiki.jasig.org/display/CASC/phpCAS+examples

Приложение А

Смена языка в клиенте CAS

```
      * @category Authentication
       * @package  PhpCAS
       * @author   Joachim Fritschi <jfritschi@freenet.de>
       * @author   Adam Franco <afranco@middlebury.edu>
       * @license  http://www.apache.org/licenses/LICENSE-2.0  Apache
License 2.0
       * @link     https://wiki.jasig.org/display/CASC/phpCAS
       */

      // Load the settings from the central config file
      require_once 'config.php';
      // Load the CAS lib
      require_once $phpcas_path . '/CAS.php';

      // Enable debugging
      phpCAS::setDebug();
      // Enable verbose error messages. Disable in production!
      phpCAS::setVerbose(true);

      // Initialize phpCAS
      phpCAS::client(CAS_VERSION_2_0, $cas_host, $cas_port,
$cas_context);

      // For production use set the CA certificate that is the issuer
of the cert
      // on the CAS server and uncomment the line below
      // phpCAS::setCasServerCACert($cas_server_ca_cert_path);

      // For quick testing you can disable SSL validation of the CAS
server.
      // THIS SETTING IS NOT RECOMMENDED FOR PRODUCTION.
```

```
      // VALIDATING THE CAS SERVER IS CRUCIAL TO THE SECURITY OF THE
CAS PROTOCOL!
      phpCAS::setNoCasServerValidation();

      // set the language to french
      phpCAS::setLang(PHPCAS_LANG_FRENCH);

      // force CAS authentication
      phpCAS::forceAuthentication();

      // at this step, the user has been authenticated by the CAS
server
      // and the user's login name can be read with phpCAS::getUser().

      // moreover, a PGT was retrieved from the CAS server that will
      // permit to gain accesses to new services.

      // for this test, simply print that the authentication was
successfull
      ?>
      <html>
        <head>
          <title>Exemple d'internationalisation de phpCAS</title>
        </head>
        <body>
          <h1>Authentification r&eacute;ussie !</h1>
          <?php require 'script_info.php' ?>
          <p>L'utilisateur connect&eacute; est <b><?php echo
phpCAS::getUser(); ?></b>.</p>
          <p>La version de phpCAS est <b><?php echo
phpCAS::getVersion(); ?></b>.</p>
        </body>
      </html>
```

Приложение Б

Изменение стиля на страницах с ошибкой

```
      * @category Authentication
       * @package  PhpCAS
       * @author   Joachim Fritschi <jfritschi@freenet.de>
       * @author   Adam Franco <afranco@middlebury.edu>
       * @license  http://www.apache.org/licenses/LICENSE-2.0  Apache
License 2.0
       * @link     https://wiki.jasig.org/display/CASC/phpCAS
       */

      // Load the settings from the central config file
      require_once 'config.php';
      // Load the CAS lib
      require_once $phpcas_path . '/CAS.php';

      // Enable debugging
      phpCAS::setDebug();
      // Enable verbose error messages. Disable in production!
      phpCAS::setVerbose(true);

      // Initialize phpCAS
      phpCAS::client(CAS_VERSION_2_0, $cas_host, $cas_port,
$cas_context);

      // For production use set the CA certificate that is the issuer
of the cert
      // on the CAS server and uncomment the line below
      // phpCAS::setCasServerCACert($cas_server_ca_cert_path);

      // For quick testing you can disable SSL validation of the CAS
server.
      // THIS SETTING IS NOT RECOMMENDED FOR PRODUCTION.
```

```
    // VALIDATING THE CAS SERVER IS CRUCIAL TO THE SECURITY OF THE
CAS PROTOCOL!
    phpCAS::setNoCasServerValidation();

    // customize HTML output
    phpCAS::setHTMLHeader(
        '<html>
      <head>
        <title>__TITLE__</title>
      </head>
      <body>
      <h1>__TITLE__</h1>'
    );
    phpCAS::setHTMLFooter(
        '<hr>
        <address>
          phpCAS __PHPCAS_VERSION__,
          CAS __CAS_VERSION__ (__SERVER_BASE_URL__)
        </address>
      </body>
    </html>'
    );

    // force CAS authentication
    phpCAS::forceAuthentication();

    // at this step, the user has been authenticated by the CAS
server
    // and the user's login name can be read with phpCAS::getUser().

    // for this test, simply print that the authentication was
successfull
    ?>
    <html>
```

```
        <head>
          <title>phpCAS simple client with HTML output
customization</title>
        </head>
        <body>
          <h1>Successfull Authentication!</h1>
          <?php require 'script_info.php' ?>
          <p>the user's login is <b><?php echo phpCAS::getUser();
?></b>.</p>
          <p>phpCAS version is <b><?php echo phpCAS::getVersion();
?></b>.</p>
        </body>
      </html>
```

Приложение В

Использование proxy – сервера

```
      * @category Authentication
       * @package  PhpCAS
       * @author   Joachim Fritschi <jfritschi@freenet.de>
       * @author   Adam Franco <afranco@middlebury.edu>
       * @license  http://www.apache.org/licenses/LICENSE-2.0  Apache
License 2.0
       * @link     https://wiki.jasig.org/display/CASC/phpCAS
       */

      // Load the settings from the central config file
      require_once 'config.php';
      // Load the CAS lib
      require_once $phpcas_path . '/CAS.php';

      // Enable debugging
      phpCAS::setDebug();
      // Enable verbose error messages. Disable in production!
```

```
      phpCAS::setVerbose(true);

      // Initialize phpCAS
      phpCAS::proxy(CAS_VERSION_2_0, $cas_host, $cas_port,
$cas_context);

      // For production use set the CA certificate that is the issuer
of the cert
      // on the CAS server and uncomment the line below
      // phpCAS::setCasServerCACert($cas_server_ca_cert_path);

      // For quick testing you can disable SSL validation of the CAS
server.
      // THIS SETTING IS NOT RECOMMENDED FOR PRODUCTION.
      // VALIDATING THE CAS SERVER IS CRUCIAL TO THE SECURITY OF THE
CAS PROTOCOL!
      phpCAS::setNoCasServerValidation();

      // force CAS authentication
      phpCAS::forceAuthentication();

      // at this step, the user has been authenticated by the CAS
server
      // and the user's login name can be read with phpCAS::getUser().

      // moreover, a PGT was retrieved from the CAS server that will
      // permit to gain accesses to new services.

      ?>
      <html>
        <head>
          <title>phpCAS proxied proxy example (with sessioning)</title>
          <link rel="stylesheet" type='text/css' href='example.css'/>
        </head>
```

```
        <body>
          <h1>phpCAS proxied proxy example (with sessioning)</h1>
          <?php require 'script_info.php' ?>
          <p>the user's login is <b><?php echo phpCAS::getUser();
?></b>.</p>
          <h2>Response from service <?php echo $serviceUrl; ?></h2>
      <?php
      flush();
      // call a service and change the color depending on the result
      if (phpCAS::serviceWeb($serviceUrl, $err_code, $output)) {
          echo '<div class="success">';
      } else {
          echo '<div class="error">';
      }
      echo $output;
      echo '</div>';
                                                                    ?>
        </body>
```

Приложение Г

Организация выхода «одной кнопкой»

```
      * @package  PhpCAS
       * @author   Joachim Fritschi <jfritschi@freenet.de>
       * @author   Adam Franco <afranco@middlebury.edu>
       * @license  http://www.apache.org/licenses/LICENSE-2.0  Apache
License 2.0
       * @link     https://wiki.jasig.org/display/CASC/phpCAS
       */
```

```
	// Load the settings from the central config file
	require_once 'config.php';
	// Load the CAS lib
	require_once $phpcas_path . '/CAS.php';

	// Enable debugging
	phpCAS::setDebug();
	// Enable verbose error messages. Disable in production!
	phpCAS::setVerbose(true);

	// Initialize phpCAS
	phpCAS::client(SAML_VERSION_1_1, $cas_host, $cas_port,
$cas_context);

	// For production use set the CA certificate that is the issuer
of the cert
	// on the CAS server and uncomment the line below
	phpCAS::setCasServerCACert($cas_server_ca_cert_path);

	// For quick testing you can disable SSL validation of the CAS
server.
	// THIS SETTING IS NOT RECOMMENDED FOR PRODUCTION.
	// VALIDATING THE CAS SERVER IS CRUCIAL TO THE SECURITY OF THE
CAS PROTOCOL!
	// phpCAS::setNoCasServerValidation();

	// Handle SAML logout requests that emanate from the CAS host
exclusively.
	// Failure to restrict SAML logout requests to authorized hosts
could
	// allow denial of service attacks where at the least the server
is
	// tied up parsing bogus XML messages.
	phpCAS::handleLogoutRequests(true, $cas_real_hosts);
```

```
// Force CAS authentication on any page that includes this file
phpCAS::forceAuthentication();

// Some small code triggered by the logout button
if (isset($_REQUEST['logout'])) {
    phpCAS::logout();
}
?>
<html>
  <head>
    <title>Advanced SAML 1.1 example</title>
  </head>
  <body>
<h2>Advanced SAML 1.1 example</h2>
<?php require 'script_info.php' ?>


Authentication succeeded for user
<strong><?php echo phpCAS::getUser(); ?></strong>.


<h3>User Attributes</h3>
<ul>
<?php
foreach (phpCAS::getAttributes() as $key => $value) {
    if (is_array($value)) {
        echo '<li>', $key, ':<ol>';
        foreach ($value as $item) {
            echo '<li><strong>', $item, '</strong></li>';
        }
        echo '</ol></li>';
    } else {
        echo '<li>', $key, ': <strong>', $value, '</strong></li>'
```

```
. PHP_EOL;
          }
      }
          ?>
      </ul>
      <p><a href="?logout=">Logout</a></p>
      </body>
      </html>
```