**2.1 Эскизное проектирование управляющего программного обеспечения**

Сформулируем требования, предъявляемые к системе управления конфигурацией:

* Применение интерпретируемого языка программирования.
* Возможность удаленного управления целевыми машинами.
* Возможность группового применения конфигураций.
* Возможность решения большое число задач администрирования.
* Осуществление контроля над процессом применения конфигурации и состояния управляемых узлов.
* Наличие внутреннего планировщика для изменения конфигурации в соответствии с некоторым планом, расписанием.
* Наличие удобно интерфейса администрирования

В ходе сравнения существующих программ управления конфигурациями программный продукт Ansible был выбран в качестве основы для будущей системы. С его помощью осуществляется мониторинг состояния узлов, формирование скриптов конфигурации и контроль выполнения этих скриптов. Ansible обеспечивает групповое применение конфигураций. Этот программный продукт обладает большим количеством модулей, которые охватывают почти все задачи администрирования. Для решения каких-либо специфичных задач есть возможность создания собственных модулей. ПО написано на языке Python, следовательно, для его работы на управляющей машине должен быть установлен интерпретатор языка Python. Ansible для изменения конфигурации формирует python-скрипт, который при помощи протокола SSH применяется на удаленной машине.

Для доступа к управляемым узлам на сервере должен быть установлен SSH-клиент. В состав Astra Linux входит пакет OpenSSH, обеспечивающий функционал, как сервера, так и клиента. Для соединения с клиентскими машинами в интерпретатор должен быть дополнительно установлен модуль Paramiko. Данный модуль предоставляет удобный интерфейс для взаимодействия с OpenSSH из Python.

Для выполнения скрипта на клиентской машине необходим интерпретатор языка Python версии не менее 2.4. Для возможности подключения к управляемому узлу, на нем должен быть установлен SSH-сервер. В качестве сервера будет использоваться OpenSSH. Для подключения Ansible к управляемым узлам необходимо произвести настройку аутентификации по открытому ключу.

Для выполнения задач в соответствии с некоторым планом необходим планировщик задач. В данной системе будет применяться планировщик cron, так как в составе Astra Linux нет других альтернатив для выбора.

**Выбор технологии для построения приложения администрирования системы.**

Приложение администрирования системы управления конфигурацией может быть реализовано различными методами. Проведем выбор архитектуры приложения и языка программирования.

Для построения приложения администрирования возможно несколько подходов:

1. Разработка клиентского и серверного приложений.
2. Разработка приложения на основе веб-технологий.

Веб-приложение - клиент-серверное приложение, в котором клиентом выступает браузер, а сервером — веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Веб-приложения являются частным случаем клиент-серверных приложений.

Преимущества веб-приложений:

1. Простота развертывания и обновления
2. Мобильность (доступ к приложению можно осуществлять с любого компьютера в сети)
3. Веб-приложения не требуют установки на компьютер пользователя объемного программного обеспечения. Для полноценной работы нужен только браузер и доступ к серверу.

Из недостатков можно выделить необходимость наличия отдельного сервера приложений, малое быстродействие и отклик интерфейса, ограничение использования возможностей операционной системы.

Исходя из того, что данная система управления конфигурациями не является системой реального времени, а, значит, не накладывается жестких ограничений по времени отклика приложения, а так же, требуется возможность доступа к системе множеству пользователей, приложение для администрирования системы будет разрабатываться на основе веб-технологий.

**Выбор языка программирования**

Приложение для администрирования системы управления конфигурацией может быть реализовано на различных языках программирования на базе серверов приложений и систем управления базами данных. Проведем выбор языка программирования из доступных в ОС Atsra Linux SE и удовлетворяющих требованиям.

Требования по сертификации и ограничения на запуск приложений без цифровой подписи делают невозможным создание приложений на компилируемом языке программирования, таком как C++ и ему подобные.

Под эти ограничения не попадают языки и программные платформы, позволяющие исполнять промежуточный байт-код, например Java/Java VM и C#/Mono. Однако проведенный анализ показал неприменимость этих языков и платформ для создания распределенного клиент-серверного приложения администрирования под ОС Astra Linux SE.

В OC Astra Linux SE возможно создание и выполнение интерпретируемых программ, выполняемых в изолированной среде. При таком подходе код программы представляет собой обычный текстовый файл (сценарий) построчно выполняемый интерпретатором. Возможности у такого способа выполнения программы ограничены по сравнению с программой, запускаемой непосредственно ОС, однако такой подход позволяет избежать потенциальных проблем с безопасностью и нарушением целостности ОС.

В ОС Astra Linux SE имеются следующие интерпретаторы, позволяющие создавать веб-приложения:

* Perl;
* PHP;
* Ruby;
* Python.

2.3.1 Обзор возможностей языка Perl по созданию веб-приложений в ОС Astra Linux SE

Perl – высокоуровневый интерпретируемый динамический язык программирования общего назначения, созданный Ларри Уоллом, лингвистом по образованию. Название языка представляет собой аббревиатуру, которая расшифровывается как Practical Extraction and Report Language – "практический язык для извлечения данных и составления отчётов".

Основной особенностью языка считаются его богатые возможности для работы с текстом, в том числе работа с регулярными выражениями, встроенная в синтаксис. Перл унаследовал много свойств от языков С, AWK, скриптовых языков командных оболочек UNIX.

2.3.2 Обзор возможностей языка PHP по созданию веб-приложений в ОС Astra Linux SE

PHP (Hypertext PreProcessor, препроцессор гипертекста) – язык программирования, исполняемый на стороне веб-сервера, спроектированный Расмусом Лердорфом (Rasmus Lerdorf) в качестве инструмента создания динамических и интерактивных веб-сайтов.

Этот язык оказался достаточно гибким и мощным, поэтому приобрел большую популярность и используется в проектах любого масштаба: от простого блога до крупнейших веб-приложений в Интернете [15].

В области программирования для Сети PHP – один из самых распространенных скриптовых языков (наряду с JSP, Perl и языками, используемыми в ASP.NET [10, 14]) благодаря своей простоте, скорости выполнения, богатой функциональности, кроссплатформенности и распространению исходных кодов на основе лицензии PHP.

Популярность в области построения веб-приложений определяется наличием большого набора встроенных средств для разработки веб-приложений, основными из которых являются:

* автоматическое извлечение POST и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
* файловые функции эффективно обрабатывают как локальные, так и удалённые файлы;
* автоматическая отправка HTTP-заголовков;
* работа с сессиями;
* обработка файлов, загружаемых на сервер;
* работа с HTTP заголовками и HTTP авторизацией;
* работа с удалёнными файлами и сокетами.

Преимущества PHP:

* является свободным программным обеспечением, распространяемым под особой лицензией (PHP license);
* легок в освоении на всех этапах;
* поддерживается большим сообществом пользователей и разработчиков;
* имеет развитую поддержку баз данных;
* имеется огромное количество библиотек и расширений языка;
* может использоваться в изолированной среде;
* предлагает нативные средства организации веб-сессий, программный интерфейс расширений;
* может быть развёрнут почти на любом сервере;
* портирован под большое количество аппаратных платформ и операционных систем.

Недостатки PHP:

* не подходит для создания десктопных приложений или системных компонентов;
* имеет слабые средства для работы с исключениями;
* глобальные параметры конфигурации влияют на базовый синтаксис языка, что затрудняет настройку сервера и разворачивание приложений;
* объекты передаются по значению, а не по ссылке, как это делается в большинстве других языков;
* веб-приложения, написанные на PHP, зачастую имеют проблемы с безопасностью.

Скрипты, написанные на языке PHP, обычно хранятся в файлах с расширением .php, которые содержат в себе смесь обычных HTML-тэгов со специальной разметкой: открывающим тэгом <?php и закрывающим ?>.

С целью упрощения создания и развертывания web-приложений на языке PHP были разработаны специализированные программные платформы. Наиболее популярными из них являются Laravel, Yii, Symfony, Zend, Joomla FMS. Для разделения решаемых задач между программистом и дизайнером существуют так называемые шаблонизаторы – специальные фильтры, позволяющие дизайнеру на простом языке описать как должна выглядеть страница, а программисту – какие данные должны быть на странице. Большинство фреймворков поддерживают концепцию MVC и позволяют разрабатывать распределенные приложения.

2.3.3 Обзор возможностей языка Ruby по созданию веб-приложений в ОС Astra Linux SE

Ruby – динамический императивный объектно-ориентированный язык программирования, разработанный Юкихиро Матсумото. Ruby был создан под влиянием таких языков, как Perl, Eiffel и Smalltalk.

Он характеризуется динамической типизацией и автоматическим управлением памятью. Язык Ruby используется в веб-разработке в составе открытого веб-фреймворка Rails, чаще называемого Ruby on Rails (RoR). Он представляет собой программный каркас, написанный на языке программирования Ruby [16] и является открытым ПО. Ruby on Rails предоставляет архитектурный образец "модель-вид-контроллер" для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером базы данных.

Ruby on Rails может работать с любым веб-сервером, поддерживающим FastCGI. В качестве сервера базы данных поддерживаются MySQL, Firebird, PostgreSQL, IBM DB2, Oracle и Microsoft SQL Server.

Преимущества Ruby:

* открытая разработка;
* работает на многих платформах;
* может внедряться в HTTP-разметку;
* относится к языкам программирования сверхвысокого уровня (VHLL), то есть обладает высоким уровнем абстракции и предметным подходом в реализации алгоритмов;
* реализует концептуально чистую объектно-ориентированную парадигму;
* предоставляет продвинутые методы манипуляции строками и текстом;
* легко интегрирует в свои программы высокопроизводительные серверы баз данных;
* благодаря VHLL программы на Ruby хорошо масштабируются и легко сопровождаются;
* простой и чистый синтаксис значительно облегчает программистам первые шаги в обучении этому языку;
* имеется простой программный интерфейс для создания многопоточных приложений;
* имеет продвинутые средства для работы с массивами;
* возможности языка можно расширить при помощи библиотек, написанных на C или Ruby;
* зарезервированные слова могут являться идентификаторами, если это не создаёт неоднозначности для парсера;
* дополнительные возможности для обеспечения безопасности;
* имеет встроенный отладчик.

Недостатки Ruby:

* сложен в освоении;
* недостаточность информационных ресурсов и поддержки;
* более низкая производительность по сравнению со другими языками, применяемыми в веб-разработке;
* относительно медленно разрабатывается и развивается.

Файлы программ, написанных на Ruby, имеют расширение .rb. Эти файлы можно запустить как скрипты командной оболочки при помощи интерпретатора.

2.3.4 Обзор возможностей языка Python по созданию веб-приложений в ОС Astra Linux SE

Как и Ruby, Python имеет целью приблизить синтаксис реальной программы, написанной на нем, к описывающему задачу псевдокоду, что позволяет программисту уменьшить объем программы. Идея создания данного языка возникла в конце 1980-х и была реализована Гвидо ван Россумом.

Python – мультипарадигмальный язык программирования: он позволяет совмещать процедурный подход к написанию кода с объектно-ориентированным и функциональным.

Преимущества Python:

* открытая разработка;
* прост в освоении, особенно на начальном этапе;
* особенности синтаксиса стимулируют программиста писать хорошо читаемый код;
* предоставляет средства быстрого прототипирования и динамической семантики;
* хорошая информационная поддержка;
* множество полезных библиотек и расширений языка можно легко использовать в своих проектах благодаря предельно унифицированному механизму импорта и программным интерфейсам;
* механизмы модульности хорошо продуманы и могут быть легко использованы.

Недостатки Python:

* слабая поддержка многопоточности;
* меньшая производительность по сравнению с основными виртуальными машинами Java.

Интерпретатор Python можно использовать как для запуска скриптов, так и в режиме интерактивной оболочки.

Исследование возможностей интерпретаторов и программных платформ показало, что наиболее развитым интерпретируемым языком общего назначения является Python. Программная платформа Django, написанная на этом языке хорошо подходит для написания веб-приложений.

**Выбор СУБД**

Для работы веб-приложения и системы в целом необходимо хранилище данных. В роли хранилища будет использоваться реляционная СУБД, потому, что они наиболее распространены, в большинстве языков программирования присутствуют инструменты для доступа к ним и большинство ориентированы на работу с реляционными базами данных.

Для работы макета не требуется большой объем данных и высокая производительность СУБД. При этом, необходима легкость в переносе существующей базы данных. Данным преимуществом обладают встраиваемы СУБД.

Встраиваемые СУБД - архитектура систем управления базами данных, когда СУБД представляет собой программный модуль, подключаемый прикладной программой. Экземпляр базы данных представляет собой файл определенного формата.

В состав Astra Linux входит только встраиваемая СУБД SQLite. Данная СУБД хорошо зарекомендовала себя среди разработчиков программного обеспечения, большинство языков программирования предоставляют инструменты для работы с ней.

**Сервер приложений**

Единственным сервером приложений, доступным в ОС Astra Linux SE, является web-сервер Apache. Основными достоинствами Apache считаются надежность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках, поддерживает.

Одним из недостатков сервера чаще всего отмечают избыточный функционал подключенных по умолчанию модулей. Данный недостаток можно устранить путем правки конфигурационных файлов и комментирования строк, подключающих не требуемые модули.

**Структурная схема**

В соответствии с требованиями и проведенным сравнением средств управления конфигурацией предлагается структурная схема системы управления конфигурацией операционной системы Astra Linux SE.

Управляющий сервер состоит из:

* Интерпретатора языка Python версии не менее 2.7. Он необходим для работы веб-приложения и программы Ansible.
* Пакета OpenSSH для связи с клиентскими машинами по протоколу SSH.
* Веб-сервера Apache, который необходим для работы веб-приложения.
* Базы данных SQLite.
* Планировщика cron, для выполнения задач по расписанию.
* Модуля django для работы веб-приложения.
* Веб-приложения

Веб-приложение необходимо для предоставления пользователю интерфейса для взаимодействия с системой управления конфигурацией и для взаимодействия с другими элементами системы. Оно состоит из:

* Модуля работы с планировщиком
* Модуля редактирования шаблонов
* Модуль управления задачами
* Модуль мониторинга состояния узлов

Ansible осуществляет конфигурацию управляемых узлов при помощи формируемых скриптов на языке Python. Для их выполнения на управляемых узлах должен быть установлен интерпретатор языка Python версии не менее 2.7.

Для возможности подключения к узлу необходимо развернуть на нем SSH-сервер и создать пользователя с административными правами. После этого требуется настроить аутентификацию подключений по SSH при помощи открытого ключа.

**Основные процессы, проходящие в системе.**

Можно выделить 4 основных сценария работы системы управления конфигурацией:

1. Формирование шаблонов для решения основных задач администрирования;
2. Добавление задач конфигурирования;
3. Выполнение задачи планировщиком;
4. Мониторинг состояния управляемых узлов.

Пользователю необходимо сформировать шаблоны для решения различных задач администрирования, например, создание пользователей, изменение прав доступа к файлам и т.д. Процесс формирования шаблона представлен на диаграмме 2. При помощи веб-интерфейса редактируется текст шаблона, указывается категория решаемой при помощи шаблона задачи, определяются переменные параметры шаблона, такие как имена, пароли пользователей, имена групп, имена файлов и т.д. После окончания редактирования шаблон сохраняется в базе данных.

Процесс конфигурирования управляемых узлов заключается в выполнении задач. Задача – это набор конфигураций, которые необходимо выполнить для приведения управляемых узлов в требуемое состояние. Процесс добавления задачи представлен на диаграмме 3. Пользователь выбирает из списка доступных действий необходимые, определяется время и дату выполнения задачи, устанавливает значения параметров шаблона. После окончания редактирования формируется плейбук задачи. Плейбук – это сценарий конфигурирования, написанный на языке YAML. Задаче присваивается идентификатор, и плейбук сохраняется в базе данных. В список задач планировщика добавляется задача на выполнение служебного скрипта.

Планировщик осуществляет запуск скриптов на выполнение в соответствии с расписанием. Процесс выполнения задачи представлен на диаграмме 4. При срабатывании события на выполнение задачи в планировщике, исполняется служебный скрипт, которые получает из базы данных необходимый плейбук и передает его на исполнение в Ansible. После выполнения конфигурирования скрипт записывает результат выполнения в базу данных.

Процесс мониторинга управляемых узлов осуществляется при помощи утилиты Ping. Данный процесс представлен на диаграмме 5. При возникновении необходимости проверить состояние узлов, веб-приложение получает из базы список узлов, запускает проверку достижимости узлов. Скрипт на языке Python последовательно осуществляет проверку достижимости узлов. Результаты проверки заносятся в базу. По запросу пользователя веб-приложение отображает статус управляемых узлов.

**Модель угроз**

Определим возможные угрозы безопасности для создаваемой системы.

В системе управления конфигурациями выделяются следующие активы:

* Управляющий сервер
* Управляемые узлы
* Коммуникации

**Возможные нарушители**

Определим возможные виды нарушителей. По отношению к системе могут действовать два типа нарушителей: внутренние и внешние.

Внешние нарушители – лица, не имеющие права доступа к информационной системе, ее отдельным компонентам и реализующие угрозы безопасности информации из-за границ информационной системы; Внутренние нарушители – лица, имеющие право постоянного или разового доступа к информационной системе, ее отдельным компонентам.

Внутренними нарушителями в данной системе являются: администратор системы управления конфигурациями, пользователь системы управления конфигурациями, пользователь управляемого узла.

Внешними нарушителями могут являться представители специальных служб иностранных государств, террористических, экстремистских группировок, преступные группы, конкурирующие организации, разработчики, производители, поставщики программных, технических и программно-технических средств, а также бывшие администраторы и пользователи системы.

**Возможные угрозы**

Исходя, из структурной схемы, система может быть подвержена следующим угрозам:

1. Компрометация управляющего сервера.
2. Перехват данных, передаваемых по SSH-соединению.
3. Подмена управляемого узла.

Компрометацию управляющего сервера можно осуществить только при несанкционированном доступе к нему. В этом случае злоумышленник получает прямой доступ к управляющему серверу. Ему становится известна информация об административных паролях управляемых узлов и о ключе для аутентификации SSH-соединения. Так же, злоумышленник получает доступ к хранилищу конфигураций и может исказить действующие шаблоны конфигураций. Эту угрозу может реализовать администратор системы управления конфигурации.

Перехват данных, передаваемых по SSH-соединению, реализуется при помощи атаки «Человек посередине». При реализации данной угрозы злоумышленник может получить ключи для аутентификации SSH-соединения. Появляется возможность искажения передаваемых сообщений, навязывания ложных сообщений. Данная угроза может реализовываться внешними и внутренними нарушителями.

Подмена управляемого узла осуществляется путем подмены ip-адреса компьютера в сети на адрес атакуемого управляемого узла. В случае реализации данной угрозы управляемый узел становится недоступен для управления и может возникнуть ситуация, когда конфигурация узла может не соответствовать необходимым требованиям. Данная угроза может быть реализована пользователем управляемого узла и пользователем системы.

При рассмотрении модели угроз определены следующие допущения. Администратор и пользователи системы являются доверенными и реализуемые ими угрозы не рассматриваются. Считаем, что вся наша система развернута внутри контролируемой зоны, а каналы, выходящие за ее пределы, зашифрованы, и возможность злоумышленника подключиться к коммуникациям отсутствует, следовательно, угроза перехвата трафика со стороны внешнего нарушителя отсутствует. Будем считать, что квалификация пользователей управляемых узлов недостаточна для проведения атаки «Человек посередине», но способны нецеленаправленно провести подмену ip-адреса управляемого узла.

Таким образом, с учетом допущений, в системе могут действовать только внутренние нарушители – пользователи управляемых узлов. Единственная реализуемая угроза – подмена управляемого узла из-за неквалифицированных действий пользователей.