Федеральное государственное автономное образовательное учреждение
высшего образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Московский институт электроники и математики им. А.Н. Тихонова

Проект 19106

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ
на разработку информационной системы
**Mekstack: Приватное Облако**

УТВЕРЖДАЮ

Рыбаков Петр Владимирович
Руководитель направления

Подпись

1 ноября 2023 г.

СОГЛАСОВАНО

Емельяненко Максим Владимирович
Руководитель проекта

Подпись

1 ноября 2023 г.

Москва, 2023

# Содержание

# 1. Актуальность проекта

В настоящее время большинство проектов требуют наличия специализированной виртуальной инфраструктуры, предоставляемой учебным заведением. Вопреки очевидным преимуществам, которые дает применение виртуальных вычислительных машин, это вызывает возникновение нового ряда требований со стороны пользователей, удовлетворение которых является целью данного проекта.

В рамках образовательного процесса студентов МИЭМ был проведен анализ, в результате которого были выявлены следующие требования к инфраструктуре и функционалу информационной системы (ИС), предоставляющей виртуальную инфраструктуру:

1. Разработка и внедрение интуитивно понятной и оперативной системы документации с функцией поиска, способной максимально раскрыть возможности частного облачного сервиса и обеспечить поддержку пользователей в освоении разнообразных методик облачных решений, включая наглядные примеры применения.
2. Создание высокодоступной системы, обладающей функционалом автоматического восстановления после сбоев (в том числе после отключения электроэнергии на территории всего кампуса) для обеспечения непрерывности работы облачных сервисов.
3. Интеграция современных технологических решений, используемых в передовых IT-компаниях, для предоставления студентам актуального практического опыта.
4. Реализация механизмов, позволяющих пользователям самостоятельно решать возникающие проблемы с минимальной зависимостью от службы технической поддержки и снижающих нагрузку на персонал, ответственный за обработку запросов.
5. Обеспечение комплексного мониторинга состояния виртуальных машин и облачных сервисов с возможностью настройки пользовательских метрик и системы оповещений для своевременного реагирования на инциденты.
6. Разработка интерфейса или системы управления, которая дает возможность пользователям самостоятельно управлять ресурсами в облаке без вмешательства сторонних специалистов.
7. Внедрение возможности подключения к виртуальной инфраструктуре с использованием доменной авторизации Высшей школы экономики.
8. Предоставление функционала для самостоятельного создания клиентов OpenID с целью настройки доменной авторизации в собственных информационных системах.
9. Реализация возможности публикации разработанных информационных систем в сети Интернет.

10. Обеспечение доступа к оперативной технической поддержке для решения возникающих вопросов и отладки проблем.

**Преподавателям МИЭМ нужно:**

1. Обеспечить функцию автоматизации стандартных операций для проведения лабораторных работ, включая автоматизированное создание виртуальных машин, но не ограничиваясь им.

**Сотрудникам МИЭМ требуется:**

1. Реализация функции контроля использования и/или недостачного использования запрошенных ресурсов информационных технологий.
2. Создание системы регистрации событий, которая позволит фиксировать все действия и изменения в виртуальной среде, что станет основой для проведения аудита и будет способствовать эффективной диагностике проблем.
3. Внедрение системы оперативного реагирования на инциденты информационной безопасности.
4. Организация студенческой инфраструктуры, функционирующей в изоляции от критически важных элементов инфраструктуры Высшей школы экономики.
5. Способ снижения нагрузки на персонал, отвечающий за обслуживание виртуальных машин.

# 2. Назначение результатов проекта

1. Автоматизация процесса обработки заявок для сокращения времени их рассмотрения и исключения необходимости ручного ввода данных.
2. Внедрение системы самообслуживания для пользователей, позволяющей им самостоятельно управлять выделенными ресурсами, что сократит нагрузку на персонал техподдержки.
3. Создание комплексной системы мониторинга, обеспечивающей постоянный контроль за состоянием виртуальных машин, облачной инфраструктуры и информационных систем как сервисных, так и пользовательских.
4. Реализация функции логирования всех значимых событий для обеспечения возможности аудита и упрощения диагностики проблем.
5. Упрощение эксплуатации облачной инфраструктуры за счет разработки инструкций по работе с виртуальными машинами и облачными сервисами.
6. Улучшение пользовательского интерфейса с целью сделать его более интуитивно понятным и удобным для пользователей.

7. Интеграция с системами Московского института электроники и математики и личным кабинетом Высшей школы экономики для создания единого информационного пространства.
8. Оптимизация использования ресурсов путем идентификации и обработки заброшенных виртуальных машин и дисков.

# 3. Цель разработки

1. Создание автоматизированного, конфигурируемого и расширяемого кластера серверов на основе проекта OpenStack Kayobe.
2. Создание системы централизованного управления и мониторинга ресурсов.
3. Интеграция с личным кабинетом ВШЭ.
4. Подготовка подробной документации (документация администратора, архитектурная документация в формате Docs as a Code) и пользовательских мануалов, настройка Elasticsearch для поиска по документации.
5. Создание VPN сервиса для подключения студентов к сети облака.
6. Создание ИС для выделения доменов и создания tls-сертификатов пользователям.
7. Интеграция с Ceph SDS для получения высокоэффективного, устойчивого к отказам и самовосстанавливающегося хранилища данных.
8. Адаптация и внедрение OpenStack Swift, системы хранения, оптимизированной для работы с неструктурированными данными.
9. Адаптация и внедрение NFSaaS (Manilla), чтобы пользователи облака могли создавать и управлять файловыми шарами (например, NFS или CIFS) без необходимости управлять низлежащими серверами файлового хранения или NAS (network-attached storage) устройствами.
10. Адаптация и внедрение DBaas (Trove), чтобы пользователи могли развертывать и управлять базами данных без необходимости заботиться о поддержке и управлении физическими серверами.
11. Разработка прокси-сервиса для кэширования и предоставления общего доступа к артефактам, который обеспечит оптимизацию доступа к зависимостям и библиотекам для ускорения процессов сборки и развертывания.
12. Разработать бота на базе ChatGPT для умной коммуникации с пользователем.
13. Реализовать сервисную сетку (service mesh) на базе Istio внутри кластера Kubernetes для обеспечения управления трафиком, безопасности, и наблюдаемости ИС с целью повышения надёжности и управляемости микросервисной архитектуры.
14. Предоставить централизованный, защищённый сервис для хранения, управления и контроля доступа к секретам и конфиденциальной информации, используя HashiCorp Vault как сервис в облачной инфраструктуре.
15. Реализовать облачный сервис удостоверяющего центра (IdP) для центра-

лизованного управления идентификацией и аутентификацией пользователей с использованием интеграции с системой Vault.
16. Автоматизировать процесс создания и обновления образов операционных систем для использования в облачной инфраструктуре, обеспечивая постоянную актуализацию безопасности и оптимизацию ресурсов.
17. Разработать и реализовать комплексные сценарии восстановления после сбоев (disaster recovery), гарантирующие минимальное время простоя и минимальную потерю данных в случае катастрофических событий или сбоев инфраструктуры.
18. Внедрить системы обнаружения и предотвращения вторжений для мониторинга и защиты облачной инфраструктуры от неавторизованного доступа, атак и уязвимостей.
19. Развернуть систему SIEM для анализа событий безопасности в реальном времени и управления инцидентами, повышая общую безопасность и соответствие нормативным требованиям.
20. Предоставить управляемый сервис Kubernetes, позволяющий пользователям развертывать, управлять и масштабировать контейнеризованные приложения с полной поддержкой и автоматизацией задач инфраструктуры.
21. Разработать и интегрировать модуль удостоверяющего центра для Keystone, который позволит упростить и централизовать доменную аутентификацию и авторизацию пользователей в Mekstack.

# 4. Требования к результатам проекта

## 4.1. Интеграция авторизации с личным кабинетом

Нужно разработать и интегрировать модуль удостоверяющего центра для Keystone, который позволит упростить и централизовать аутентификацию и авторизацию пользователей в Mekstack.

**Триггер начала работ:**

Успешная имплементация Apache Kafka в личном кабинете МИЭМ.

**Требования к сервису:**

Сервис должен создавать и изменять пользователей и проекты в Keystone аналогично их сущностям в кабинете ВШЭ.

**Задачи:**

1. Разработать пользовательский модуль аутентификации Keystone, который будет использовать данные проектов пользователя из OpenID.
2. Написать патч для Keystone.
3. Написать пайплайн сборки и публикации образов.
4. Написать пайплайн для периодического слияния нашей версии Keystone с основной кодовой базой Keystone.
5. Написать юнит-тесты для патча.
6. Настроить логирование для нового модуля аутентификации.
7. Реализовать канареечное развертывание для поэтапного внедрения модуля.
8. Разработать пайплайн для сборки образов Keystone с новым модулем.
9. Автоматизировать публикацию образов в реестр.
10. Обеспечить версионирование для возможности отката и ведения аудита.
11. Интегрировать пайплайн с существующими процессами CI/CD.
12. Провести аудит безопасности нового модуля аутентификации.
13. Устранить уязвимости, обнаруженные в ходе аудита.
14. Гарантировать соответствие соответствующим стандартам безопасности и лучшим практикам.
15. Разработать стратегию внедрения нового модуля в существующие среды Keystone.
16. Создать инструмент миграции для пользователей, использующих предыдущий метод аутентификации.
17. Протестировать процесс миграции в контролируемых условиях.

## 4.2. Документация облака для разработчиков

Наиболее возможно полная документация функциональности облака с узконаправленной информацией для разработчиков.

**Для этого должна быть следующая документация:**

1. Документация архитектуры всех сервисов
2. Документация архитектуры облака
3. Схема сетевых потоков (L3)
4. Схема сетевой связанности (L2)
5. Схема расположения оборудования в стойках (L1)

## 4.3. Документация облака для пользователей

Документация, доступная по аресу docs.cloud.hse.ru.

**Сервис с документацией должен удовлетворять следующим требованиям:**

1. Описание самых популярных сценариев работы с облаком
2. Инструкции по использованию компонентов облака для продвинутых пользователей, которым нужен узконаправленный функционал платформы.
3. Работа по технологии Docs as a Code.
4. Ссылки на другие сервисы Mekstack и на техподдержку в Zulip.
5. Автоматическая сборка и публикация страниц документации на изменения в системе контроля версий.
6. Возможность пользователей предлагать изменения в документацию при помощи систем контроля версий.
7. Интегрированный поиск OpenSearch.
8. Интегрированная система обратной связи.

## 4.4. Сервис публикации пользовательских HTTP серверов (NATaaS)

Создание ИС, предназначенного для выделения доменов и создания TLS сертификатов пользователям на базе Envoy, который регистрирует все изменения. Данная реализация поможет существенно снизить временные затраты по сравнению с выполнением аналогичных действий вручную.

**Ключевые функции NATaaS**

1. **Перенаправление запросов (Request Forwarding):** Основная функция обратного прокси, обработка входящих HTTP запросов и направление их к соответствующим бэкенд-серверам.
2. **Балансировка нагрузки (Load Balancing):** Распределение входящего сетевого трафика между группой бэкенд-серверов для предотвращения перегрузки отдельных серверов.
3. **TLS Termination:** Расшифровка входящих TLS соединений на прокси и пересылка трафика на бэкенд-серверы внутри сети без шифрования для снижения вычислительной нагрузки на бэкенд-серверы.
4. **Мультиплексирование соединений:** Снижение нагрузки на бэкенд-серверы за счет повторного использования соединений для нескольких клиентских запросов.
5. **HTMX SPA и gRPC HTTP/2:** Должна быть обеспечена поддержка HTMX SPA и gRPC HTTP2 для взаимодействия с бекендами и сервисом обнаружения сервисов

**Функции для оптимизация производительности**

1. **Кэширование (Caching):** Хранение копий часто запрашиваемых ресурсов для снижения нагрузки на сервер и улучшения времени отклика.
2. **Сжатие (Compression):** Сжатие исходящего контента для ускорения передачи данных клиенту.
3. **Поддержка HTTP/3 и gRPC:** Поддержка современных протоколов для эффективной коммуникации.

**Функции безопасности**

1. **Web Application Firewall (WAF):** Защита от общих уязвимостей веб-приложений и атак, таких как SQL-инъекция и межсайтовый скриптинг (XSS).
2. **Защита от DDoS-атак:** Снижение эффекта распределенных атак типа ”отказ в обслуживании”с автоматизированной фильтрацией трафика.
3. **API Throttling и Rate Limiting:** Предотвращение злоупотреблений и управление трафиком путем ограничения числа запросов от пользователя за определенный временной промежуток.
4. **Фильтрация IP-адресов (IP Filtering):** Создание белых и черных списков IP-адресов или диапазонов для контроля доступа к бэкенд-сервисам.

**Интеграция и настройка**

1. **Поддержка пользовательских доменных имен:** Возможность для клиентов использовать свои собственные доменные имена для услуги обратного прокси.
2. **Маршрутизация на основе пути (Path-Based Routing):** Маршрутизация запросов на основе пути или имени хоста в запросе.
3. **Модификация заголовка хоста (Host Header Modifications):** Изменение заголовка хоста в HTTP-запросах при их прохождении через прокси.

**Мониторинг и управление**

1. **Логирование и мониторинг:** Подробные журналы доступа и реальный мониторинг производительности прокси.
2. **Оповещения и уведомления:** Автоматизированные уведомления о проблемах, таких как простои сервера или высокая задержка ответов.
3. **Аналитика и отчетность:** Предоставление данных о моделях трафика, поведении пользователей и производительности API.

**Инструменты для разработчиков**

1. **Интеграция с CI/CD:** Инструменты для интеграции с системами непрерывной интеграции и непрерывной доставки.

**Масштабируемость и надежность**

1. **Автоматическое масштабирование (Auto Scaling):** Автоматическая корректировка числа экземпляров обратного прокси-сервиса в зависимости от нагрузки.
2. **Высокая доступность (High Availability):** Разработка системы с учетом отказоустойчивости и обеспечение доступности ИС через резервирование и дублирование.
3. **Корректное переключение при отказе (Graceful Failover):** В случае сбоя сервера автоматическое перенаправление трафика на здоровые серверы без потери запросов.

**Интеграционные задачи**

1. **Интеграция с Fail2ban:** Должна быть обеспечена интеграция с Fail2ban для автоматического блокирования подозрительных IP адресов и предотвращения несанкционированного доступа.
2. **Интеграция с SIEM:** Должны быть настроены механизмы для сбора, анализа и сохранения логов в системе SIEM для обеспечения возможности мониторинга безопасности и управления инцидентами.
3. **Архитектура Service Discovery:** Должна быть реализована архитектура обнаружения ИС для автоматического обновления маршрутов и конфигураций в реальном времени, исключая Single Point of Failure.
4. **TLS авторегистрация:** Должна быть предусмотрена функция автоматической регистрации и обновления TLS сертификатов для пользовательских доменов.
5. **Добавление пользовательских доменов:** Должна быть возможность для пользователей добавлять собственные доменные имена в сервис.
6. **Лизинг поддоменов:** Должна быть предоставлена функциональность лизинга наших поддоменов пользователям с возможностью управления и настройки.
7. **Управление правилами форвардинга:** В веб-интерфейсе должна быть предусмотрена функция управления правилами форвардинга доменов пользователей.
8. **Интеграция с OpenStack:** Должна быть реализована интеграция с OpenStack для автоматического форвардинга трафика на инфраструктуру проектов.
9. **OpenID авторизация:** Должна быть внедрена система доменной OpenID авторизации с использованием edu.hse.ru.

10. **Интеграция с Grafana:** Должна быть предусмотрена интеграция iframe с Grafana для отображения информации о статусе и нагрузке поддоменов пользователя.

## 4.5. Сервис авторизации и аутентификации пользователей (Hashicorp Vault)

Должен быть реализован облачный сервис удостоверяющего центра (IdP) для централизованного управления идентификацией и аутентификацией пользователей с использованием интеграции с системой Vault. У пользователей должна быть возможность использовать Vault IdP в своих сервисах для осуществления доменной авторизации.

**Для этого должны быть реализованы следующие функции:**

1. Синхронизация данных с кабинетом пользователя для актуализации информации о проектах пользователей.
2. Шифрование всех данных для обеспечения безопасности конфиденциальной информации.
3. Обеспечение высокой доступности ИС для непрерывной работы пользователей.
4. Использование алгоритма RAFT для репликации данных, гарантирующего их надежность и согласованность.
5. Предоставление пользователям доступа к API для интеграции с внешними приложениями и ИС.
6. Реализация системы единого входа (SSO) с использованием доменной OpenID авторизации по auth.hse.ru для упрощения процесса входа.
7. Возможность создания пользователями своих OpenID клиенты для авторизации в различных ИС.
8. Регулярное создание бекапов данных для обеспечения их сохранности.
9. Тестирование процедур восстановления из бекапов для проверки надежности соответствующих механизмов.
10. Внедрение практик непрерывной интеграции и непрерывного развертывания (CI/CD) для ускорения процессов разработки и обеспечения качества.
11. Использование Gitops для автоматизации и управления инфраструктурой через git.
12. Развертывание ИС в Kubernetes для улучшения управляемости и масштабируемости.
13. Обеспечение безопасности ИС и выполнение обязательств по обеспечению безопасности (Security Assurance).
14. Создание обучающих материалов и руководств для пользователей для повышения их осведомленности и умения пользоваться сервисом.

15. Автоматическое масштабирование ИС с использованием инструментов Kubernetes HPA (Horizontal Pod Autoscaler) для адаптации к изменяющейся нагрузке.
16. Внедрение сервис-мешей, таких как Istio, для управления трафиком, повышения отказоустойчивости и масштабируемости ИС.

## 4.6. Monitoring

Помимо разработки новых ИС также нужно осуществлять мониторинг существующих.

**Для этого нужно будет выполнить следующие задачи:**

1. Настройка и интеграция Grafana для визуализации данных мониторинга.
2. Настройка Prometheus для сбора метрик с различных ИС.
3. Интеграция с OpenStack для мониторинга его ключевых компонентов.
4. Настройка системы Alerting для создания и управления алертами, включая прогнозирующие сбои.
5. Для SNEEDaaS мониторить следующие метрики:
    5.1. “Здоровье” балансировщика нагрузки.
    5.2. Доступ из публичной сети.
    5.3. Доступ из внешней сети.
    5.4. Задержку в сети.
6. Для DNoS-HA мониторить следующие метрики:
    6.1. “Здоровье” балансировщика нагрузки.
    6.2. Доступ из публичной сети.
    6.3. Задержку в сети.
    6.4. Состояние основных компонентов OpenStack.
    6.5. Срок действия сертификатов.
7. Для VPNaaS мониторить следующие метрики:
    7.1. Производительность ИС.
    7.2. Количество свободных IP-адресов.
8. Для Vault мониторить следующие метрики:
    8.1. Статус ИС.
    8.2. Количество пользователей.
    8.3. Частоту входа в систему.
    8.4. Синхронизацию заданий маппинга.
    8.5. Дату последнего запуска задания.
    8.6. Время выполнения задания.

## 4.7. Monitoring as a Service

У пользователя должна быть возможность мониторинга состояния его виртуальной инфраструктуры.

**Для этого нужно реализовать следующие функции:**

1. Мониторинг в реальном времени для непрерывного контроля за системами и приложениями.
2. Сбор показателей производительности для анализа и оптимизации работы систем.
3. Настраиваемые информационные панели, позволяющие пользователям создавать дашборды согласно их потребностям.
4. Настраиваемые оповещения для своевременного реагирования на критические события или проблемы.
5. Хранение исторических данных в состоянии для анализа тенденций и паттернов поведения системы.
6. Обнаружение аномалий для раннего выявления нестандартного поведения или потенциальных проблем.
7. Мониторинг безопасности для контроля за угрозами и уязвимостями системы.
8. Отчетность по политикам и соответствию для проверки соответствия виртуальных машин организационным политикам и создания отчетов для аудита соответствия.
9. Масштабируемая архитектура, для своевременной адаптации сервиса к изменяющимся требованиям без потери производительности.
10. Механизмы отказоустойчивости для поддержания работы ИС мониторинга в случае сбоев системы.
11. Интеграция с системами обнаружения вторжений для повышения уровня безопасности мониторинга.
12. Настраиваемые интервалы мониторинга, позволяющие пользователям определять частоту проверок для их виртуальных машин в соответствии с их потребностями.
13. Портал самообслуживания для добавления и управления виртуальными машинами пользователями без необходимости прямого вмешательства IT-персонала.
14. API для автоматизации задач мониторинга и интеграции с другими системами, такими как инструменты предоставления услуг или оркестрирования.

## 4.8. Кеширующее зеркало часто используемых артефактов

Часто используемые артефакты должны кэшироваться для уменьшения времени работы с ними при использовании ресурсов из публичных репозиториев.

**Для этого реализовать следующие требования:**

1. Кэширование артефактов для ускорения доступа к часто запрашиваемым элементам.
2. Управление шардами для распределения данных и оптимизации производительности.
3. Контроль доступа для обеспечения безопасности и соответствия политикам.
4. Функциональность поиска для быстрого нахождения артефактов внутри хранилища.
5. Высокая доступность для обеспечения непрерывности доступа к артефактам.
6. Масштабируемость для адаптации к изменяющимся объемам нагрузки.
7. Безопасность хранения и передачи артефактов для предотвращения несанкционированного доступа и атак.
8. Управление репозиториями для артефактов различных типов, таких как образы Docker, пакеты Maven, пакеты npm и другие.
9. Разрешение зависимостей для автоматического разрешения и загрузки зависимости, необходимые первичным артефактам.
10. Прокси-функциональность к другим внешним репозиториям артефактов, позволяя доступ к ним, как если бы они были локальными.
11. Неизменяемое кэширование для гарантии того, что однажды кэшированный артефакт не изменяется.
12. Проверка контрольных сумм для подтверждения целостности артефакта через проверку контрольных сумм для предотвращения использования поврежденных или подделанных двоичных файлов.
13. Оптимизация пропускной способности для уменьшения использования внешнего трафика за счет обслуживания артефактов из локального кэша по возможности.
14. Регулярные проверки консистентности для обеспечения синхронизации кэшированных артефактов с источниками восходящего потока.
15. Поддержка маркировки и простановки тегов артефактов для упрощения организации и извлечения.

## 4.9. Бот техподдержки

У пользователей должна быть возможность получить быструю техподдрежку.

**Для этого нужно:**

1. Разработать бота на базе ChatGPT для умной коммуникации с пользователем
2. Предоставить доступ боту к
   2.1. API облака с правами чтения
   2.2. Утилитам для сетевой диагностики: ping, ssh, dig
   2.3. Документации и поиску по документации
   2.4. Примерам Terraform и Ansible конфигурации
3. Реализовать возможность реконфигурации бота без редеплоя
4. Должна быть разработана база ответов на частые вопросы

## 4.10. K8s Service mesh (Istio)

Взаимодействие по сети должно быть защищено для безопасности данных пользователей.

**Для этого нужно реализовать следующее:**

1. Поддержка протокола HTTP/3 для обеспечения современного и эффективного веб-трафика.
2. Реализация взаимного TLS (mTLS) для обеспечения безопасности коммуникаций в gRPC, включая аутентификацию и шифрование.
3. Возможность перенаправления трафика для управления запросами и маршрутизации.
4. Внедрение мониторинга с использованием sidecar-контейнеров для сбора метрик и мониторинга ИС без изменения кода приложений.
5. Интеграция с системами SIEM и IDS через sidecar-контейнеры для обеспечения безопасности и мониторинга угроз в режиме реального времени.
6. Интеграция Fail2ban или аналогичного механизма для защиты от брутфорс-атак и других методов автоматизированных угроз.

## 4.11. Сервис хранения секретов пользователей (Hashicorp Vault)

У пользователя должна быть возможность хранить свои секреты в защищенном хранилище.

**Нужно реализовать следующее:**

1. Использование алгоритма RAFT для обеспечения надежной и согласованной репликации данных.

2. Генерация динамических секретов по требованию с ограниченным временем жизни для доступа к системам, ИС или инструментам.
3. Автоматический или ручной отзыв секретов и учетных данных.
4. Надежное хранение секретов с шифрованием статичных данных и при передаче данных.
5. Поддержка версионирования ключей и значений с возможностью отката к предыдущим версиям.
6. Подробное ведение журналов действий, выполняемых в Vault, для аудита и контроля соответствия требованиям.
7. Предоставление услуг шифрования как ИС напрямую из Vault.
8. Механизмы аренды и продления срока действия секретов с автоматическим обновлением и отзывом.
9. Поддержка высокой доступности Vault с конфигурацией в нескольких центрах обработки данных.
10. Способность масштабирования для удовлетворения потребностей и обработки большого числа запросов.
11. Поддержка резервных узлов для обслуживания запросов только на чтение и замены основных узлов при их отказе.
12. Взаимодействие с Vault через HTTP API для интеграции с различными приложениями и ИС.
13. Расширение функционала Vault с помощью пользовательских плагинов или использование плагинов, разработанных сообществом.
14. Возможность расширения Vault новыми бэкендами секретов и методами аутентификации.
15. Мониторинг состояния и производительности Vault в реальном времени.
16. Интеграция с инструментами мониторинга, такими как Prometheus, Grafana, Splunk для получения аналитических данных и визуализации.
17. Защита экземпляра Vault с помощью механизма опечатывания, требующего кворум ключей для доступа к хранимым секретам.
18. Инструменты и процедуры для резервного копирования и восстановления Vault в случае катастрофы.

## 4.12. Автоматическая сборка облачных образов ОС

Нужно реализовать автоматизацию процесса создания и обновления образов операционных систем для использования в облачной инфраструктуре, обеспечивая постоянную актуализацию безопасности и оптимизацию ресурсов.

**Для этого должны быть реализованы следующие функции:**

1. Разработать процедуры тестирования для образов ОС, чтобы гарантировать их функциональность и стабильность.

2. Реализовать систему создания и удаления образов, позволяющую управлять жизненным циклом образа от разработки до отката.
3. Обеспечить настройку локальных зеркал в образах для ускорения процесса установки и обновления пакетов.
4. Включить функционал настройки таймзоны в образах, чтобы они соответствовали локальным настройкам пользователя.
5. Настроить процесс сборки для поддержки операционных систем Ubuntu, Debian и Rocky.
6. Интегрировать добавление агентов мониторинга и Gopnik в образы для наблюдения за состоянием и производительностью ОС на базе образов в облаке.

## 4.13. Ceph SDS

Должна быть реализована интеграция с Ceph SDS для получения высокоэффективного, устойчивого к отказам и самовосстанавливающегося хранилища данных.

**Для этого должны быть реализованы следующие требования:**

1. Поддержка CephFS для предоставления услуг файлового хранилища через сервис общего файлового хранилища OpenStack (Manila).
2. Реализация механизмов для обеспечения высокой доступности ИС.
3. Поддержка масштабируемости ИС для обработки увеличивающегося объема данных и запросов.
4. Внедрение Thin provisioning для эффективного распределения хранилища.
5. Реализация функции создания снапшотов и клонирования данных.
6. Поддержка автоматической репликации данных для повышения надежности хранения.
7. Реализация возможности автоматического или ручного перемещения данных между различными уровнями хранилища (SSD, HDD) в зависимости от моделей доступа.
8. Поддержка шифрования хранилища для неиспользуемых данных, что повышает безопасность конфиденциальных данных.
9. Введение управления квотами для контроля использования ресурсов хранения.
10. Средства для мониторинга производительности хранения.
11. Поддержка асинхронной репликации на вторичные сайты для целей восстановления после сбоев.

## 4.14. Система объектного хранения (Swift)

Должна быть реализована адаптация OpenStack Swift, система хранения, оптимизированная для работы с неструктурированными данными.

**Для этого должны быть реализованы следующие требования:**

1. Автоматическая репликация данных.
2. Возможность горизонтального масштабирования путём добавления новых узлов.
3. Обеспечение отсутствия единой точки отказа в архитектуре системы.
4. Интеграция с системой удостоверений OpenStack (Keystone) для аутентификации и контроля доступа.
5. Предоставление RESTful и S3 API для хранения и извлечения файлов с использованием стандартных HTTP-вызовов.
6. Поддержка высокой степени параллелизма обработки запросов.
7. Механизмы кэширования для ускорения чтения данных.
8. Возможность хранения и управления большими объектами путём их сегментации.
9. Возможность назначения пользовательских метаданных объектам и контейнерам.
10. Поддержка установки квот для контроля использования хранилища.
11. Интеграция Swift с различными компонентами OpenStack и предоставление доступа через SDK на разных языках программирования.
12. Поддержка использования ИС в качестве хранилища для целей резервного копирования и архивации.
13. Предоставление каждому проекту в Mekstack своей изолированной среды Swift для конфиденциальности и безопасности.
14. Интеграция с телеметрическими службами Mekstack для мониторинга и оповещения.
15. Генерация подробных журналов активности для интеграции с внешними инструментами управления логами.

## 4.15. Сервис предоставления хранилищ NFS (Manilla)

Должна быть реализована адаптация NFSaaS (Manilla), чтобы пользователи облака могли создавать и управлять файловыми шарами (например, NFS или CIFS) без необходимости управлять нижележащими серверами файлового хранения или NAS (network-attached storage) устройствами

**Для этого должны быть реализованы следующие требования:**

1. Поддержка NFS, SMB/CIFS и других популярных протоколов файлового обмена.

2. Возможность динамического создания и выделения файловых ресурсов по требованию пользователя без ручного вмешательства.
3. Обеспечение избыточной конфигурации (redundantly configured) хранения для поддержания доступности в случае сбоев оборудования или сети.
4. Автоматическое перемещение данных между различными уровнями хранения на основе моделей доступа для оптимизации стоимости и производительности.
5. Интеграция дедупликации данных для уменьшения объёма хранения путём удаления повторяющихся данных.
6. Сжатие данных в реальном времени для увеличения эффективной ёмкости хранения и снижения сетевой нагрузки.
7. Надёжное шифрование данных в покое и в процессе передачи для защиты конфиденциальной информации.
8. Возможность планирования и создания снапшотов и резервных копий по требованию для защиты данных и восстановления на определённый момент времени.
9. Подробные журналы всех действий для аудита соответствия и мониторинга безопасности.
10. API для интеграции с другими облачными ИС, приложениями и инструментами автоматизации.
11. Возможность устанавливать квоты на пользователя или группу для контроля потребления хранения.
12. Интеграция со службами каталогов, такими как LDAP или Active Directory для аутентификации пользователей.
13. Удобный веб-интерфейс для управления файловыми ресурсами, мониторинга использования и настройки параметров.
14. Периодические проверки здоровья и предсказывать анализ возможных отказов для обеспечения целостности системы.
15. Политики управления жизненным циклом данных, архивацией и хранением автоматически.
16. Детальные отчёты о потреблении хранения, активности пользователей и показателях производительности.
17. Настраиваемая система оповещений для различных событий, таких как превышение квот, проблемы со здоровьем системы или попытки несанкционированного доступа.

## 4.16. Сервис управления базами данных (DBaaS)

Должна быть реализована адаптация DBaas (Trove), чтобы пользователи могли развертывать и управлять базами данных без необходимости заботиться о поддержке и управлении физическими серверами

**Для этого должны быть реализованы следующие функции:**

1. Поддержка различных баз данных, таких как MySQL, PostgreSQL, MongoDB, Redis и Cassandra.
2. Настройка автоматических резервных копирований для защиты данных с возможностью планирования задач.
3. Возможность восстановления баз данных из резервных копий пользователями.
4. Поддержка настройки репликации баз данных для повышения производительности и увеличения избыточности данных.
5. Предложение конфигураций для высокой доступности через механизмы репликации и кластеризации.
6. Возможность пользователей устанавливать и изменять параметры конфигурации для экземпляров баз данных.
7. Интеграция с телеметрическими службами Mekstack для предоставления метрик мониторинга производительности баз данных.
8. Предоставление предопределенных конфигураций баз данных для упрощения развертывания.
9. Возможность применения правил брандмауэра к экземплярам баз данных для обеспечения безопасности.
10. Возможность создания снимков баз данных для использования в будущем или для целей резервного копирования.
11. Предоставление RESTful API для всей функциональности, что позволяет автоматизацию и интеграцию с другими службами или приложениями.
12. Поддержка кластеризации для некоторых типов баз данных, позволяющая горизонтальное масштабирование для обработки больших нагрузок.
13. Использование OpenStack Cinder для хранения баз данных для обеспечения надежного и последовательного управления хранилищем.
14. Возможность изменения размера экземпляров баз данных на лету для корректировки ресурсов без значительного простоя.
15. Интеграция с ИС ведения журнала запросов для отслеживания запросов к базе данных в целях безопасности и оптимизации.
16. Поддержка различных версий хранилищ данных, позволяющая пользователям выбирать необходимые им версии.

## 4.17. Disaster Recovery сценарии

Должны быть реализованы комплексные сценарии восстановления после сбоев (disaster recovery), гарантирующие минимальное время простоя и потерь данных в случае катастрофических событий или сбоев инфраструктуры

**Для этого нужно сделать следующее:**

1. Разработка и проведение тестирования планов восстановления после сбоев в продуктовом окружении для проверки их надежности.

## 4.18. Managed K8s

Нужно предоставить управляемый сервис Kubernetes, позволяющий пользователям развертывать, управлять и масштабировать контейнеризованные приложения с полной поддержкой и автоматизацией задач инфраструктуры

**Для этого нужно сделать следующее:**

1. Предоставить пользователям возможность развертывания k8s кластеров минимальным количеством действий через пользовательский интерфейс.
2. Обеспечить возможность деплоя k8s кластеров через API для автоматизации процесса и интеграции с внешними системами.
3. Оптимизировать процесс установки k8s для работы с оборудованием, имеющим ограниченный объем оперативной памяти, для повышения эффективности использования ресурсов.
4. Внедрить систему мониторинга для отслеживания состояния кластеров k8s и ресурсов железа.
5. Настроить систему алертинга, которая будет уведомлять администраторов о проблемах и событиях в реальном времени.

## 4.19. Сервис отслеживания сверхутилизации ресурсов (Generic Over-Provisioning Nuker and Inspection Kit) (GOPNIK)

Должен быть разработан сервис для определения и устранения неэффективно используемых ресурсов, например обнаружения выделенной виртуальной машине оперативной памяти, которую не используют.

**Для этого должны быть реализованы следующие функции:**

1. Отслеживание использования ресурсов
    1.1. Мониторинг в реальном времени: Должен быть сделан мониторинг использования ЦП, памяти, хранения и ввода/вывода в реальном времени.
    1.2. Границы использования: Должны быть установлены границы для определения ”недоиспользования”на основе процентных соотношений или конкретных метрик.

1.3. Простановка тегов ресурсов: Должны быть проставлены теги для категоризации и мониторинга ресурсов по проекту, отделу или пользователю.

2. Обнаружение аномалий
    2.1. Анализ паттернов использования: Должны быть реализованы модели машинного обучения для изучения нормальных паттернов использования и обнаружения аномалий.
    2.2. Обнаружение неактивных ресурсов: Должны быть идентифицированы ресурсы с низкой или нулевой активностью за настраиваемый период времени.
3. Мониторинг активности пользователей
    3.1. Отслеживание входов: Должны быть отслежены входы пользователей в облачную среду и активные сессии.
    3.2. События развертывания и удаления: Должны быть отслежены события развертывания или удаления ресурсов пользователями.
4. Управление затратами
    4.1. Анализ затрат: Должны быть рассчитаны затраты на недоиспользованные ресурсы.
    4.2. Отчеты Showback и Chargeback: Должны быть созданы отчеты, показывающие распределение затрат между различными отделами или проектами при использовании облака.
5. Система уведомлений
    5.1. Оповещение: Должны быть отправлены уведомления пользователям или администраторам при недоиспользовании их ресурсов.
    5.2. Пользовательские правила уведомлений: Должна быть предоставлена возможность настройки условий, при которых срабатывают уведомления.
6. Применение политик
    6.1. Определение политик: Администраторы должны иметь возможность определять политики для обращения с недоиспользованными ресурсами.
    6.2. Автоматическое устранение: Должны быть реализованы автоматические действия, такие как остановка или освобождение ресурсов в соответствии с политиками.
7. Отчетность и панели управления
    7.1. Отчеты об использовании: Должны быть сгенерированы подробные отчеты об использовании ресурсов и случаях их занятия.

7.2. Интерактивные панели управления: Должны быть предоставлены панели управления с фильтрами для различных параметров для визуализации использования ресурсов.

8. Портал самообслуживания
    8.1. Пользовательский интерфейс: Должен быть предложен удобный HTMX веб-интерфейс с поддержкой gRPC и HTTP/2 для пользователей для проверки их использования ресурсов.
    8.2. Самоустранение: Пользователи должны иметь возможность самостоятельно устранять проблемы с занятием ресурсов через портал.
9. Масштабируемость и производительность
    9.1. Масштабируемая архитектура: Сервис должен быть спроектирован таким образом, чтобы обрабатывать увеличивающееся количество ресурсов без снижения производительности.
    9.2. Минимальное влияние на производительность: Должно быть обеспечено, что процессы мониторинга и анализа оказывают минимальное воздействие на производительность облака.

## 4.20. Системы обнаружения и предотвращения вторжений (IDS/IPS)

Должны быть внедрены системы обнаружения и предотвращения вторжений для мониторинга и защиты облачной инфраструктуры от известных атак и эксплуатации уязвимостей.

**Для этого нужно сделать следующее:**

1. Реализовать систему обнаружения аномалий сетевого трафика, способную идентифицировать необычные паттерны и потенциальные угрозы в реальном времени.
2. Интегрировать систему с доступными базами данных TI для сигнатурного анализа трафика с целью обнаружения известных типов атак и вредоносных действий на основе известных сигнатур.
3. Предоставить возможности визуализации и составления отчетов для наглядного представления состояния сетевой безопасности и истории инцидентов.

## 4.21. Управление событиями и информацией о безопасности (SIEM)

Установить систему SIEM для анализа и управления событиями безопасности в реальном, создания корреляционных событий, повышающее общее

понимание о состоянии безопасности инфраструктуры, а также устранения угроз информационной безопасности.

**Для этого нужно сделать следующее:**

1. Обеспечить сбор событий информационных систем, включая логи операционных систем и журналы аутентификаций.
2. Реализовать интеграцию ИС SIEM с средствами защиты информации.
3. Предусмотреть нормализацию и агрегацию данных из различных источников для унификации формата и упрощения анализа.
4. Внедрить корреляционные события для выявления связанных действий и потенциальных угроз, которые невозможно обнаружить при одиночном анализе.
5. Разработать механизм обогащения информации на основе внешних репутационных источников для обогащения контекста событий.
6. Обеспечить хранение данных событий на заданный период времени.
7. Реализация оповещения операторов ИС о созданных корреляционных событиях.
8. Разработать процедуры реагирования на инциденты (incident response) на основе детектируемых событий.
9. Интегрировать сервис SIEM с системами автоматической сборки (autobuilds) для отслеживания изменений и потенциальных уязвимостей в процессах разработки и развертывания.

## 4.22. Задачи менеджеров продукта

Для успешной реализации продукта менеджеры должны выполнять следующие задачи:

1. Подготовка материалов для прохождения контроля проектным офисом.
2. Работа с пользователями (сбор обратной связи, CusDev).
3. Продвижение проекта, общение с потенциальными партнерами/инвесторами.
4. Интеграция с другими проектами МИЭМ.
5. Подготовка пользовательской документации.
6. Согласование финансирования.
7. Ведение переговоров в рамках вуза.
8. Контроль промежуточных точек проекта.
9. Проработка UX.

# 5. Календарный план

| Задание | Период работы | Исполнители |
|---|---|---|
| Monitoring | 31.10.2023 - 31.01.2024 | Services, SRE |
| Monitoring as a Service | 16.11.2023 - 29.04.2024 | Monitoring, SRE |
| NAT as a service | 16.10.2023 - 22.01.2024 | Services, SRE, Kubernetes, Frontend |
| Документация функциональности облака | 30.11.2023 - 29.04.2024 | ТехДок, SRE, Frontend |
| Caching Artifact Proxy | 16.10.2023 - 5.02.2024 | SRE |
| Сервис хранения секретов пользователей | 13.11.2023 - 25.12.2023 | SRE |
| Бот техподдержки | 11.10.2023 - 5.02.2024 | Services |
| GOPNIK | 20.11.2023 - 29.04.2024 | Services, SRE, Monitoring |
| Disaster Recovery сценарии | 8.02.2024 - 5.03.2024 | SRE |
| Автоматическая сборка облачных образов ОС | 11.12.2023 - 05.02.2024 | SRE |
| Ceph SDS | 20.11.2023 - 20.03.2024 | SRE |
| Object storage (Swift) | 01.01.2024 - 05.02.2024 | SRE |
| NFSaaS (Manilla) | 22.01.2024 - 23.02.2024 | SRE |
| DBaaS (Trove) | 25.01.2024 - 29.03.2024 | SRE |
| Managed K8s | 01.02.2024 - 01.04.2024 | SRE |
| IDS/IPS | 30.11.2023 - 29.04.2024 | InfoSec |
| SIEM | 30.11.2023 - 29.04.2024 | InfoSec |
| Подготовка материалов для прохождения контроля проектным офисом | 20.10.2023 - 8.11.2023 | Product Managers |

| | | |
|---|---|---|
| Проработка UX | 01.12.2023 - 01.01.2024 | Product Managers |
| Написание юзер гайдов | 01.12.2023 - 01.01.2024 | Product Managers |
| Согласование финансирования | 01.01.2024 - 01.02.2024 | Product Managers |
| Работа с пользователями | 01.02.2024 - 01.03.2024 | Product Managers |
| Интеграция с другими проектами | 01.02.2024 - 01.03.2024 | Product Managers |
| Продвижение проекта, общение с потенциальными партнерами/инвесторами | 01.03.2024 - 01.04.2024 | Product Managers |

# 6. Требования к отчетным материалам

**К разработанному решению должна быть написана сопутствующая документация**

1. Пользовательская документация
2. Документация разработчика
3. Отчет о проделанной работе
4. Документация из раздела “Документация функциональности облака”

Документация должна быть выполнена на русском языке и соответствовать отраслевым стандартам.

**Требования к пользовательской документации**

1. Описание самых популярных сценариев работы с облаком
2. Инструкции по использованию компонентов облака для продвинутых пользователей, которым нужен узконаправленный функционал платформы.

**Требования к документации разработчика**

1. Документация должна быть актуальной и содержать последнюю информацию о проекте, включая изменения и обновления.
2. Документация должна включать описание всех основных функций, а также детальную информацию о каждой функции, включая ее входные и выходные данные, алгоритмы, используемые при ее выполнении, любые ограничения и требования.
3. Документация должна быть организована логически и иметь хорошую структуру, чтобы пользователи могли быстро найти нужную информацию.
4. Документация должна быть размещена в доступном месте, чтобы другие разработчики и пользователи могли получить к ней доступ.