

Министерство цифрового развития, связи и массовых коммуникаций РФ
Уральский технический институт связи и информатики (филиал) ФГБОУ ВО
"Сибирский государственный университет телекоммуникаций и информатики" в
г. Екатеринбурге
(УрТИСИ СибГУТИ)

КАФЕДРА
ИСТ

ОТЧЕТ

По дисциплине «Сетевое программирование»
Практическое занятие № 9
«Микро-сервисная архитектура»

Выполнил: студент гр. ПЕ-126
Камков Д.А.

Проверил: Ст.преп.,
Бурумбаев Д.И.

Ассистент:

Екатеринбург, 2024

1 Цель работы:

1.1 Закрепление знаний по теме «Микро-сервисная архитектура».

2 Подготовка к работе:

2.1 Изучить теоретический материал по теме «Микро-сервисная архитектура».

3 Задание:

3.1 Ответить письменно на вопросы тестового задания.

Каликов
Смородина
Тест 1, 2, 3
3 (зачтено)
Ф. Бурумбаев 0.4

4. Контрольные вопросы:

4.1 Что такое микро-сервисы и в чем их отличие от монолитной архитектуры?

Микро-сервисы – это подход к разработке программного обеспечения, при котором приложение разбивается на небольшие автономные сервисы, каждый из которых выполняет определенную функцию. Отличие от монолитной архитектуры заключается в том, что в монолите весь функционал приложения находится в одном монолите, в то время как в микро-сервисах функционал разделен на отдельные сервисы.

4.2 Какие преимущества предоставляет микро-сервисная архитектура в сравнении с монолитной?

- Легкость масштабирования и обновления отдельных сервисов
- Улучшенную отказоустойчивость и надежность
- Упрощенное развертывание и управление сервисами
- Улучшенную гибкость и возможность использования различных технологий для каждого сервиса

4.3 Какие основные принципы следует учитывать при проектировании микро-сервисной архитектуры?

- Отделение функциональности на независимые сервисы
- Каждый сервис должен быть автономным и масштабируемым
- Использование стандартизированных интерфейсов для взаимодействия между сервисами
- Мониторинг и управление сервисами для обеспечения высокой доступности

4.4 Какие вызовы возникают при разработке и масштабировании микро-сервисных систем?

- Управление сложностью межсервисного взаимодействия
- Обеспечение безопасности и целостности данных при распределенной архитектуре
- Мониторинг и отладка распределенных систем
- Управление версиями и согласованностью между сервисами

4.5 Какие технологии можно использовать для реализации связи между микро-сервисами?

- HTTP/REST API
- Message Queues (например, RabbitMQ, Kafka)
- gRPC
- Service Mesh (например, Istio)

4.6 Какие компоненты обычно включаются в микро-сервисную архитектуру?

- Сервисы (независимые компоненты, реализующие определенную функциональность)
- API Gateway (для маршрутизации запросов к различным сервисам)
- Service Registry (для регистрации и обнаружения сервисов)
- Load Balancer (для распределения нагрузки между сервисами)
- Мониторинг и логирование (для отслеживания работы сервисов и выявления проблем)

4.7 Как микро-сервисы обеспечивают автономность и независимость отдельных компонентов системы?

Микро-сервисы обеспечивают автономность и независимость отдельных компонентов системы путем разделения функциональности на отдельные сервисы, каждый из которых может быть разработан, развернут и масштабирован независимо от других. Это позволяет изменять и обновлять отдельные компоненты без влияния на работу других сервисов.

4.8 Какие подходы к обеспечению безопасности можно использовать в микро-сервисной архитектуре?

Для обеспечения безопасности в микро-сервисной архитектуре можно использовать следующие подходы:

- Использование аутентификации и авторизации для контроля доступа к сервисам.
- Реализация шифрования данных для защиты конфиденциальной информации.
- Применение механизмов мониторинга и аудита для обнаружения и реагирования на потенциальные угрозы.

4.9 Каким образом контейнеризация помогает в развертывании и управлении микро-сервисами?

Контейнеризация помогает в развертывании и управлении микро-сервисами путем упаковки каждого сервиса в отдельный контейнер, который содержит все необходимые зависимости и настройки. Это упрощает процесс развертывания, масштабирования и управления сервисами, а также обеспечивает изоляцию и надежность работы приложений.

4.10 Как можно обеспечить высокую доступность и отказоустойчивость в микро-сервисной архитектуре?

Для обеспечения высокой доступности и отказоустойчивости в микро-сервисной архитектуре можно использовать следующие методы:

- Распределение нагрузки между сервисами для предотвращения единой точки отказа.
- Использование механизмов репликации и балансировки нагрузки для обеспечения непрерывной работы при отказе отдельных компонентов.
- Реализация мониторинга и автоматического восстановления для быстрого обнаружения и устранения проблем.