



Федеральное государственное образовательное бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)

Кафедра информационных технологий
Факультет информационных технологий и анализа больших данных

Документ подписан усиленной неквалифицированной электронной подписью.

Организация: «ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ
РОССИЙСКОЙ ФЕДЕРАЦИИ»

Сертификат: TJkuRPC4RuBNxYn06+P+gJ4u2Om81LkV

Утверждено: Проректор по учебной и методической работе, Е.А. Каменева

Дата: 09.02.2026 г.

А.В. Поляков
Микросервисная архитектура

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки
01.03.02 - Прикладная математика и информатика,
Образовательная программа «Прикладное машинное обучение»
Профиль:
«Прикладное машинное обучение»

Рекомендовано

*Факультет информационных технологий и анализа больших данных
(протокол № 09 от 16.06.2026 г.)*

Одобрено

*Кафедра информационных технологий
(протокол № 04 от 25.05.2026 г.)*



Содержание

1. Наименование дисциплины	3
2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	3
3. Место дисциплины в структуре образовательной программы	6
4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	6
5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий	7
5.1. Содержание дисциплины	7
5.2. Учебно – тематический план	9
5.3. Содержание семинаров, практических занятий	10
6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине	12
6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	12
6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю	15
7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	16
8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	6
9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	9
10. Методические указания для обучающихся по освоению дисциплины	31
11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)	33
12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	33



1. Наименование дисциплины

Микросервисная архитектура

2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине

Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
ПКП-7	Способность создавать прикладные программные компоненты и интегрировать в них методы анализа данных, машинного обучения и искусственного интеллекта	Владеет инструментальными средствами разработки прикладных программных продуктов: языками программирования, библиотеками, фреймворками	Знать: экосистему языков для микросервисной разработки: Go, Java/Kotlin (Spring Boot), Python (FastAPI), Node.js — области применения и компромиссы; фреймворки для построения микросервисов: Spring Boot, FastAPI, gRPC; клиентские библиотеки для взаимодействия между сервисами; инструменты управления зависимостями и сборки: Maven/Gradle, pip/Poetry, npm; принципы семантического версионирования библиотек; инструменты контейнеризации и оркестрации: Docker, Docker Compose, основы Kubernetes; принципы упаковки сервисов в образы. Уметь: реализовывать микросервис с REST/gRPC API на выбранном стеке: структурировать код, подключать зависимости, настраивать конфигурацию через переменные окружения; упаковывать сервис в Docker-образ: писать многоэтапный Dockerfile, оптимизировать размер образа, публиковать в реестр; подбирать и подключать библиотеки для типовых задач: ORM, HTTP-клиенты, сериализа-



Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
			ция, кэширование; оценивать зрелость и лицензионную чистоту зависимостей; разворачивать локальную среду из нескольких сервисов через Docker Compose; отлаживать межсервисное взаимодействие.
		Владеет архитектурными принципами и паттернами создания прикладных программных продуктов на различных платформах	Знать: принципы проектирования: SOLID, DRY, KISS, 12-Factor App — применительно к микросервисной архитектуре; паттерны микросервисов: API Gateway, BFF, Saga (хореография/оркестрация), Outbox, CQRS, Event Sourcing, Sidecar; принципы декомпозиции на сервисы: Domain-Driven Design, Bounded Context, принципы высокой связности и слабого сцепления; паттерны обеспечения отказоустойчивости: Circuit Breaker, Retry, Bulkhead, Timeout; паттерны обнаружения сервисов (Service Discovery). Уметь: декомпозировать предметную область на Bounded Context и проектировать границы микросервисов, минимизируя межсервисную связность; выбирать и применять паттерны для конкретных задач: реализовывать Saga для распределённых транзакций, Outbox для надёжной публикации событий; проектировать API Gateway и BFF: определять ответственности шлюза, маршрутизацию, агрегацию ответов под разных клиентов; применять принципы 12-Factor App при проектировании: разделять конфигурацию, логи, процессы; обеспечивать stateless-поведение сервисов.



Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
		Демонстрирует понимание побочных процессов, сопровождающих прикладную разработку программного обеспечения в промышленных условиях и владеет соответствующим инструментарием	Знать: практики наблюдаемости (observability): метрики (Prometheus), трассировка (OpenTelemetry, Jaeger), логирование (ELK/Loki) — three pillars of observability; CI/CD-практики и инструменты: Git Flow / Trunk-Based Development, GitHub Actions / GitLab CI, принципы Pipeline as Code; управление конфигурацией и секретами: внешние конфиги (Consul, ConfigMap), хранилища секретов (Vault, Kubernetes Secrets), принципы ротации; стратегии деплоя в промышленных условиях: Blue/Green, Canary, Rolling Update; управление даунтаймом и откат релизов. Уметь: инструментировать сервис для наблюдаемости: экспортировать метрики в Prometheus, добавлять spans для распределённой трассировки, формировать структурированные логи; строить и поддерживать CI/CD-пайплайн: этапы lint → test → build → deploy; настраивать условия выкатки и автоматический откат по результатам проверок; диагностировать инциденты в распределённой системе: использовать трейсы и логи для локализации причины сбоя по запросу, пронизывающему несколько сервисов; реализовывать безопасное хранение и передачу конфигурации: выносить секреты из кода, настраивать передачу через переменные окружения или внешнее хранилище.



3. Место дисциплины в структуре образовательной программы

Дисциплина «Микросервисная архитектура» относится к «Модулю "разработке распределенных приложений"».

4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся

Очная форма обучения

Вид учебной работы по дисциплине	Всего (в з/е и часах)	Семестр 7 (в часах)
Общая трудоемкость дисциплины	3/108	108
Контактная работа – Аудиторные занятия	50	50
<i>Лекции</i>	<i>16</i>	<i>16</i>
<i>Семинары, практические занятия</i>	<i>34</i>	<i>34</i>
<i>Самостоятельная работа</i>	<i>58</i>	<i>58</i>
Вид текущего контроля	Контрольная работа	Контрольная работа
Вид промежуточной аттестации	Зачет	Зачет



5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий

5.1. Содержание дисциплины

Тема 1. Введение в микросервисную архитектуру

Понятие микросервиса и микросервисной архитектуры. Свойства, характерные для микросервисной архитектуры. Компонентное представление через сервисы. Гетерогенность. Куб масштабирования и микросервисы. Микросервисы как разновидность модульности. Микросервисная архитектура для FTGO. Сравнение микросервисной и сервис-ориентированной архитектур. Достоинства и недостатки микросервисной архитектуры. Язык шаблонов микросервисной архитектуры. Организация и процессы. Переход на микросервисы.

Тема 2. Стратегии декомпозиции

Определение микросервисной архитектуры приложения. Определение системных операций. Разбиение на сервисы по бизнес-возможностям. Разбиение на сервисы по проблемным областям. Декомпозиция программного продукта на микросервисы. Рекомендации по декомпозиции. Трудности при разбиении приложения на сервисы. Определение API сервисов.

Тема 3. Межпроцессное взаимодействие в микросервисной архитектуре

Обзор межпроцессного взаимодействия в микросервисной архитектуре. Стили взаимодействия. Описание API в микросервисной архитектуре. Взаимодействие на основе удаленного вызова процедур. Использование REST. Использование gRPC. Обнаружение сервисов. Взаимодействие с помощью асинхронного обмена сообщениями. Использование асинхронного обмена сообщениями для улучшения доступности.

Тема 4. Управление транзакциями в микросервисной архитектуре

Управление транзакциями с помощью повествований. Микросервисная архитектура и необходимость в распределенных транзакциях. Проблемы с распределенными транзакциями. Координация повествований.

Тема 5. Проектирование микросервисов и систем на основе микросервисной архитектуры



Шаблоны организации бизнес-логики. Изучение основ проектирования микросервисов, декомпозиции проекта на микросервисы, организации взаимодействия между компонентами сложных систем.

Тема 6. Разработка микросервисов

Изучение инструментов и подходов к разработке микросервисов, настройка взаимодействия между микросервисами. Реализация хранилища событий. Совместное использование повествований и порождения событий. Шаблоны внешних API. Проблемы с проектированием внешних API. Реализация API-шлюза. Разработка безопасных сервисов. Проектирование конфигурируемых сервисов. Развертывание сервисов с помощью пакетов для отдельных языков. Развертывание сервисов в виде виртуальных машин. Развертывание сервисов в виде контейнеров. Бессерверное развертывание сервисов. Стороннее ПО для разработки и обслуживания микросервисов. Изучение Docker, Gateway, Continuous delivery. изучение CI/CD, изучение систем мониторинга. AMQP. Socket. HTTP. HTTPS. Mercury. FTP.

Тема 7. Тестирование микросервисов

Стратегии тестирования микросервисных архитектур. Обзор методик тестирования. Трудности тестирования микросервисов. Процесс развертывания. Написание модульных тестов для сервиса. Написание интеграционных тестов. Разработка компонентных тестов. Написание сквозных тестов.



5.2. Учебно – тематический план

Очная форма обучения

п/ п	Наименование тем (разделов) дисциплины	Трудоемкость в часах				
		Всего	Контактная работа - Аудиторная работа			Самостоятельная работа
			Общая, в т.ч.:	Лекции	Семинары, практические занятия	
1	Введение в микросервисную архитектуру	10	4	2	2	6
2	Стратегии декомпозиции	10	4	2	2	6
3	Межпроцессное взаимодействие в микросервисной архитектуре	12	4	2	2	8
4	Управление транзакциями в микросервисной архитектуре	10	4	2	2	6
5	Проектирование микросервисов и систем на основе микросервисной архитектуры	20	12	2	10	8
6	Разработка микросервисов	32	16	4	12	16
7	Тестирование микросервисов	14	6	2	4	8
В целом по дисциплине		108	50	16	34	58



5.3. Содержание семинаров, практических занятий

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Введение в микросервисную архитектуру	Понятие микросервиса и микросервисной архитектуры.	Занятия в интерактивной форме в виде дискуссий.
Стратегии декомпозиции	Декомпозиция программного продукта на микросервисы.	Интерактивная форма, практикум по решению задач по тематике занятия и коллективное обсуждение решений.
Межпроцессное взаимодействие в микросервисной архитектуре	Обзор межпроцессного взаимодействия в микросервисной архитектуре.	Занятия в интерактивной форме в виде дискуссий.
Управление транзакциями в микросервисной архитектуре	Микросервисная архитектура и необходимость в распределенных транзакциях.	Занятия в интерактивной форме в виде дискуссий.
Проектирование микросервисов и систем на основе микросервисной архитектуры	Построение архитектуры программного продукта с применением стороннего ПО. Разработка простейшего микросервиса. Создание Gateway для программного продукта на основе микросервисной архитектуры. Обмен информацией между микросервисами по протоколу HTTP. Очередь сообщений как способ асинхронного взаимодействия между микросервисами. Логирование в микросервисах. Организация непрерывной доставки для микросервисов. Авторизация в программных продуктах на основе микросервисной архитектуры.	Интерактивная форма, практикум по решению задач по тематике занятия и коллективное обсуждение решений.



Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Разработка микросервисов	Разработка микросервиса. AMQP. Socket. HTTP. HTTPS. Mercury. FTP. Тестирование ПО с микросервисной архитектурой. Docker. Gateway. Continuous delivery.	Интерактивная форма, практикум по решению задач по тематике занятия и коллективное обсуждение решений.
Тестирование микросервисов	Тестирование ПО с микросервисной архитектурой.	Интерактивная форма, практикум по решению задач по тематике занятия и коллективное обсуждение решений.



6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине

6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы

Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
Введение в микросервисную архитектуру	Переход на микросервисы.	Работа с учебной литературой. Разбор вопросов по теме занятия.
Стратегии декомпозиции	Определение API сервисов.	Работа с учебной литературой. Разбор вопросов по теме занятия.
Межпроцессное взаимодействие в микросервисной архитектуре	Использование асинхронного обмена сообщениями для улучшения доступности.	Работа с учебной литературой. Разбор вопросов по теме занятия.
Управление транзакциями в микросервисной архитектуре	Координация повествований.	Работа с учебной литературой. Разбор вопросов по теме занятия.
Проектирование микросервисов и систем на основе микросервисной архитектуры	Организации взаимодействия между компонентами сложных систем.	Работа с учебной литературой. Разбор вопросов по теме занятия.
Разработка микросервисов	Развертывание сервисов в виде виртуальных машин. Развертывание сервисов в виде контейнеров. Бессерверное развертывание сервисов.	Работа с учебной литературой. Разбор вопросов по теме занятия.



Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
Тестирование микросервисов	Написание сквозных тестов.	Работа с учебной литературой. Разбор вопросов по теме занятия.



6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю

Примерный вариант контрольной работы

Требуется реализовать систему, состоящую из нескольких взаимодействующих друг с другом сервисов.

Общие требования:

1. Каждый сервис имеет свое собственное хранилище, если оно ему нужно. Для учебных целей можно использовать один instance базы данных, но каждый сервис работает только со своей логической базой. Запросы между базами запрещены.
2. Для межсервисного взаимодействия использовать HTTP (придерживаться RESTful). Допускается использовать и другие протоколы, например, gRPC, но это требуется согласовать с преподавателем.
3. Выделить Gateway Service как единую точку входа и межсервисной коммуникации. Горизонтальные запросы между сервисами делать нельзя.
4. Код хранить на Github, для сборки использовать Github Actions.
5. Gateway Service должен запускаться на порту 8080, остальные сервисы запускать на портах 8050, 8060, 8070.
6. Каждый сервис должен быть завернут в docker.
7. В docker-compose.yml прописать сборку и запуск docker контейнеров.
8. В classroom.yml дописать шаги на сборку и прогон unit-тестов.

Пример системы:

1. Система предоставляет пользователю возможность поиска и покупки билетов. При покупке билетов пользователю начисляются баллы, которые он может использовать для оплаты.

Покупка билета:

1. Запрос к Flight Service для проверки, что такой рейс существует. Если Flight Service недоступен, то запрос завершается с ошибкой.
2. Выполняется запрос к Ticket Service на создание записи о билете. Если сервис недоступен, то запрос завершается с ошибкой.
3. Если при покупке билета указан флаг paidFromBalance, то в Bonus Service выполняется запрос на списание бонусов. Иначе, выполнится запрос на пополнение бонусного счета на 10% от стоимости заказа. В любом случае в Bonus Service будет создана запись в таблице privilege_history.
4. Если запрос к Bonus Service завершился неудачей (500 ошибка или сервис недоступен), то выполняется откат операции создания заказа в Ticket Service.

Возврат билета:

1. Выполняется запрос к Ticket Service для обновления статуса билета. Если этот сервис недоступен, то весь запрос завершается ошибкой.



2. После этого выполняется запрос к Bonus Service на откат изменений в бонусном счете. Если этот сервис недоступен, то пользователю все равно отдается информация, что операция завершилась успешно, а на Gateway Service запрос ставится в очередь и повторяется пока не завершится успехом (timeout 10 секунд).

Дополнительная информация

Критерии балльной оценки различных форм текущего контроля успеваемости содержатся в соответствующих методических рекомендациях Кафедры информационных технологий Факультета информационных технологий и анализа больших данных.



7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

Перечень компетенций с указанием индикаторов их достижения в процессе освоения образовательной программы содержится в разделе 2. «Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине».

Типовые контрольные задания или иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний

Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
ПКП-7. Способность создавать прикладные программные компоненты и интегрировать в них методы анализа данных, машинного обучения и искусствен-	Владеет инструментальными средствами разработки прикладных программных продуктов: языками программирования, библиотеками, фреймворками	Знать: экосистему языков для микросервисной разработки: Go, Java/Kotlin (Spring Boot), Python (FastAPI), Node.js — области применения и компромиссы; фреймворки для построения микросервисов: Spring Boot, FastAPI, gRPC; клиентские библиотеки	Задание 1 Реализовать микросервис управления пользователями (CRUD) с REST API на заданном фреймворке; подключить ORM, реализовать валидацию входных данных, вернуть корректные HTTP-статусы. Задание 2 Написать многоэтапный Dockerfile для существующего сервиса, минимизировав размер итогового образа; объяснить сделанные решения (базовый образ, кэширование слоёв, права пользователя). Задание 3 Составить Docker Compose для трёх сервисов (API, БД, брокер сообщений); проверить взаимодействие, описать зависимости через depends_on и health-check. Задание 4



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
ного интеллекта		для взаимодействия между сервисами; инструменты управления зависимостями и сборки: Maven/Gradle, pip/Poetry, npm; принципы семантического версионирования библиотек; инструменты контейнеризации и оркестрации: Docker, Docker Compose, основы Kubernetes; принципы упаковки сервисов в образы. Уметь: реализовывать микросервис с REST/gRPC API на выбранном стеке:	Выбрать и обосновать стек для нового микросервиса по заданным нефункциональным требованиям (нагрузка, latency, команда); сравнить не менее двух альтернатив.



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		структурировать код, подключать зависимости, настраивать конфигурацию через переменные окружения; упаковывать сервис в Docker-образ: писать многоэтапный Dockerfile, оптимизировать размер образа, публиковать в реестр; подбирать и подключать библиотеки для типовых задач: ORM, HTTP-клиенты, сериализация, кэширование; оценивать зрелость и лицензионную чистоту зависимостей; разворачи-	



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		вать локальную среду из нескольких сервисов через Docker Compose; отлаживать межсервисное взаимодействие.	
	Владеет архитектурными принципами и паттернами создания прикладных программных продуктов на различных платформах	Знать: принципы проектирования: SOLID, DRY, KISS, 12-Factor App — применительно к микросервисной архитектуре; паттерны микросервисов: API Gateway, BFF, Saga (хореография/оркестрация), Outbox, CQRS, Event Sourcing, Sidecar; принципы декомпозиции на сервисы:	Задание 1 По описанию предметной области (например, доставка еды) выделить Bounded Context, предложить декомпозицию на микросервисы и обосновать границы с точки зрения связности и сцепления. Задание 2 Спроектировать реализацию паттерна Saga (хореография) для сценария «Оформление заказа»: описать события, обработчики и компенсирующие транзакции при сбое. Задание 3 Реализовать Circuit Breaker при вызове внешнего сервиса: настроить пороги срабатывания, состояния (closed/open/half-open) и fallback-поведение; продемонстрировать работу на тестовом стенде. Задание 4 Провести ревью предложенной архитектуры на соответствие принципам 12-Factor App: указать нарушения (жёстко заданные конфиги, stateful-логика и т.д.) и предложить исправления.



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		Domain-Driven Design, Bounded Context, принципы высокой связности и слабого сцепления; паттерны обеспечения отказоустойчивости: Circuit Breaker, Retry, Bulkhead, Timeout; паттерны обнаружения сервисов (Service Discovery). Уметь: декомпонировать предметную область на Bounded Context и проектировать границы микросервисов, минимизируя межсервисную связность; выбирать	



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		и применять паттерны для конкретных задач: реализовать Saga для распределённых транзакций, Outbox для надёжной публикации событий; проектировать API Gateway и BFF: определять ответственности шлюза, маршрутизацию, агрегацию ответов под разных клиентов; применять принципы 12-Factor App при проектировании: разделять конфигурацию, логи, процессы; обеспечи-	



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		вать stateless-поведение сервисов.	
	Демонстрирует понимание побочных процессов, сопровождающих прикладную разработку программного обеспечения в промышленных условиях и владеет соответствующим инструментарием	Знать: практики наблюдаемости (observability): метрики (Prometheus), трассировка (OpenTelemetry, Jaeger), логирование (ELK/Loki) — three pillars of observability; CI/CD-практики и инструменты: Git Flow / Trunk-Based Development, GitHub Actions / GitLab CI, принципы Pipeline as Code; управление конфигурацией и секретами: внешние конфиги (Consul,	Задание 1 Добавить в существующий микросервис инструментарий: настроить экспорт метрик (latency, error rate, throughput) в Prometheus и собрать дашборд в Grafana с порогами алертинга. Задание 2 По предоставленному набору трейсов и логов (Jaeger + Loki) воспроизвести и локализовать причину задержки запроса в цепочке из трёх сервисов; описать предполагаемое исправление. Задание 3 Описать и реализовать CI/CD-пайплайн для микросервиса: этапы сборки, тестирования, публикации образа и Canary-деплой с автоматическим откатом при превышении порога ошибок. Задание 4 Провести аудит репозитория на наличие секретов и небезопасных конфигураций; перенести чувствительные данные в безопасное хранилище и обновить пайплайн с учётом изменений.



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		ConfigMap), хранилища секретов (Vault, Kubernetes Secrets), принципы ротации; стратегии деплоя в промышленных условиях: Blue/Green, Canary, Rolling Update; управление даунтаймом и откат релизов. Уметь: инструментировать сервис для наблюдаемости: экспортировать метрики в Prometheus, добавлять spans для распределённой трассировки, формировать структурированные логи; строить и под-	



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		держивать CI/CD-пайплайн: этапы lint → test → build → deploy; настраивать условия выкатки и автоматический откат по результатам проверок; диагностировать инциденты в распределённой системе: использовать трейсы и логи для локализации причины сбоя по запросу, предоставляющему несколько сервисов; реализовывать безопасное хранение и передачу конфигурации: выносить секреты из кода, настраивать пе-	



Наименование компетенции	Наименование индикаторов достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
		редачу через переменные окружения или внешнее хранилище.	



Примеры практико-ориентированных заданий

Задание 1.

Студенту предлагается предметная область (например, система управления заказами), для которой необходимо выделить набор микросервисов, описать их интерфейсы и реализовать взаимодействие между сервисами с использованием REST или асинхронного обмена сообщениями.

Результат выполнения:

- схема микросервисной архитектуры;
- описание API микросервисов;
- реализованные сервисы с настроенным межсервисным взаимодействием.

Задание 2.

На основе готового набора микросервисов студент выполняет контейнеризацию компонентов и организует их развертывание в тестовой среде с использованием оркестратора контейнеров.

Результат выполнения:

- конфигурационные файлы контейнеризации и развертывания;
- развернутое микросервисное приложение;
- отчет о процессе развертывания и проверке работоспособности.

Примеры вопросов для подготовки к промежуточной аттестации

Примерные вопросы для подготовки к зачету

1. Понятие микросервиса и микросервисной архитектуры
2. Свойства, характерные для микросервисной архитектуры
3. Компонентное представление через сервисы
4. Гетерогенность
5. Куб масштабирования и микросервисы
6. Микросервисы как разновидность модульности
7. Микросервисная архитектура для FTGO
8. Сравнение микросервисной и сервис-ориентированной архитектур
9. Достоинства и недостатки микросервисной архитектуры
10. Язык шаблонов микросервисной архитектуры
11. Организация и процессы
12. Переход на микросервисы
13. Определение микросервисной архитектуры приложения
14. Определение системных операций
15. Разбиение на сервисы по бизнес-возможностям



16. Разбиение на сервисы по проблемным областям
17. Декомпозиция программного продукта на микросервисы
18. Рекомендации по декомпозиции
19. Трудности при разбиении приложения на сервисы
20. Определение API сервисов
21. Обзор межпроцессного взаимодействия в микросервисной архитектуре
22. Стили взаимодействия
23. Описание API в микросервисной архитектуре
24. Взаимодействие на основе удаленного вызова процедур
25. Использование REST
26. Использование gRPC
27. Обнаружение сервисов
28. Взаимодействие с помощью асинхронного обмена сообщениями
29. Использование асинхронного обмена сообщениями для улучшения доступности
30. Управление транзакциями с помощью повествований
31. Микросервисная архитектура и необходимость в распределенных транзакциях
32. Проблемы с распределенными транзакциями
33. Координация повествований
34. Шаблоны организации бизнес-логики
35. Изучение основ проектирования микросервисов
36. Декомпозиции проекта на микросервисы
37. Организации взаимодействия между компонентами сложных систем
38. Изучение инструментов и подходов к разработке микросервисов
39. Настройка взаимодействия между микросервисами
40. Реализация хранилища событий
41. Совместное использование повествований и порождения событий
42. Шаблоны внешних API
43. Проблемы с проектированием внешних API
44. Реализация API-шлюза
45. Разработка безопасных сервисов
46. Проектирование конфигурируемых сервисов
47. Развертывание сервисов с помощью пакетов для отдельных языков
48. Развертывание сервисов в виде виртуальных машин
49. Развертывание сервисов в виде контейнеров
50. Бессерверное развертывание сервисов
51. Стороннее ПО для разработки и обслуживания микросервисов



52. Изучение Docker, Gateway, Continuous delivery
53. Изучение CI/CD, изучение систем мониторинга
54. AMQP. Socket. HTTP. HTTPS. Mercury. FTP
55. Стратегии тестирования микросервисных архитектур
56. Обзор методик тестирования
57. Трудности тестирования микросервисов
58. Процесс развертывания
59. Написание модульных тестов для сервиса
60. Написание интеграционных тестов
61. Разработка компонентных тестов
62. Написание сквозных тестов



8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины

Основная литература:

1. Водяхо, А. И. Архитектурные решения информационных систем [Электронный ресурс] / Водяхо А. И., Выговский Л. С., Дубенецкий В. А., Цехановский В. В. 3-е изд., стер. Санкт-Петербург : Лань, 2022 356 с. Книга из коллекции Лань - Информатика <https://e.lanbook.com/book/254624> ISBN 978-5-507-44710-7. [БИК ID: RU-LAN-BOOK-254624]
2. Гусев, К. В. Системная и программная инженерия [Электронный ресурс] : методические указания по выполнению практических работ / Гусев К. В., Воронцов Ю. А., Михайлова Е. К., входят: Москва : РТУ МИРЭА, 2021 30 с. Книга из коллекции РТУ МИРЭА - Информатика <https://e.lanbook.com/book/182487>. [БИК ID: RU-LAN-BOOK-182487]
3. Миронов, А. Н. Системная и программная инженерия [Электронный ресурс] / Миронов А. Н., Воронцов Ю. А., Михайлова Е. К., Трушин С. М. Москва : РТУ МИРЭА, 2022 129 с. Книга из коллекции РТУ МИРЭА - Психология. Педагогика <https://e.lanbook.com/book/310997>. [БИК ID: RU-LAN-BOOK-310997]

Дополнительная литература:

1. Баланов, А. Н. Построение микросервисной архитектуры и разработка высоконагруженных приложений [Электронный ресурс] : учебное пособие для вузов / Баланов А. Н. 2-е изд., стер. Санкт-Петербург : Лань, 2025 244 с. Книга из коллекции Лань - Информатика <https://e.lanbook.com/book/456920> ISBN 978-5-507-52652-9. [БИК ID: RU-LAN-BOOK-456920]
2. Андрианова, Е. Г. Проектная практика [Электронный ресурс] : учебно-методическое / Андрианова Е. Г., Полторак А. В. Москва : РТУ МИРЭА, 2021 166 с. Книга из коллекции РТУ МИРЭА - Информатика <https://e.lanbook.com/book/218432>. [БИК ID: RU-LAN-BOOK-218432]

Нормативные правовые акты:

-

9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/> (<http://library.fa.ru/files/elibfa.pdf>)



2. Электронно-библиотечная система BOOK.RU <http://www.book.ru>
3. Электронно-библиотечная система "Университетская библиотека ОНЛАЙН"
<http://biblioclub.ru/>
4. Научная электронная библиотека eLibrary.ru <http://elibrary.ru>
5. Информационно-образовательный портал Финуниверситета: <https://org.fa.ru>
6. Электронно-библиотечная система Znanium <http://www.znanium.com>
7. Образовательная платформа "ЮРАЙТ" <https://urait.ru/>
8. • Электронно-библиотечная система издательства Проспект
<http://ebs.prospekt.org/books>
9. Электронно-библиотечная система издательства "Лань" <https://e.lanbook.com/>
10. СПАРК <https://spark-interfax.ru/>



10. Методические указания для обучающихся по освоению дисциплины

Основные этапы работы студента по дисциплине **«Микросервисная архитектура»**:

1. Предварительная ориентировка в подлежащем изучению учебном материале по программе.
2. Ознакомление с рекомендованной учебной литературой.
3. Слушание и конспектирование лекций, а также выполнение других видов учебной работы.
4. Планирование самостоятельной работы.
5. Обобщение и систематизация информации, почерпнутой из лекций и прочитанной литературы.
6. Выполнение контрольной работы.

Студентам при подготовке контрольной работы следует использовать нормативные документы Финансового университета, Методические рекомендации по планированию и организации внеаудиторной самостоятельной работы студентов по образовательным программам бакалавриата и магистратуры в Финансовом университете, утвержденные приказом Финуниверситета от 11.05.2021 г. № 1040/о (см. сайт Финансового Университета: на главной странице раздел "Наш университет" → "Единая правовая база Финуниверситета" → "Организация учебного процесса" → "Нормативные документы по самостоятельной работе").

Рекомендации по работе с учебным материалом:

1. Осознавайте наличный уровень полученных вами знаний.
2. В ситуации непонимания нужно выявить тот первичный уровень и факторы непонимания, которые стали препятствием понимания последующего.
3. Задавайте сами себе вопросы и пытайтесь ответить на них.

Рекомендации по работе на лекции и с лекционным материалом:

1. Основная задача на лекции – осмысление излагаемого в ней материала. Для этого необходимо слушать лекцию с самого начала, не упуская общих, ориентирующих в материале рассуждений и установок лектора.
2. Ведение записей на лекции важно и полезно для лучшего осмысливания материала, для сохранения информации, с целью ее дальнейшего использования.
3. Для облегчения записи рекомендуется применять сокращения повторяющихся терминов или хорошо известных понятий.

Рекомендации по работе с литературой:

1. Если возникли затруднения при разыскивании материала, по какому-либо конкретному вопросу, следует обратиться к предметному указателю, напечатанному,



как правило, в конце каждого литературного источника.

2. Предметный указатель – это алфавитный список основных научных понятий (терминов), содержание которых раскрыто в книге, рядом с термином стоят числа, обозначающие номера страниц, на которых изложен материал, относящийся к данному понятию.



11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем

Комплект лицензионного программного обеспечения:

1. Антивирус Kaspersky
2. Комплект свободно распространяемого программного обеспечения
3. Пакет офисных программ

Современные профессиональные базы данных и информационные справочные системы

1. Информационно-правовая система «Гарант»
2. Информационно-правовая система «Консультант Плюс»

Сертифицированные программные и аппаратные средства защиты информации

1. не предусмотрены

12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

1. **Компьютерный класс** для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенный оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), персональные компьютеры, набор демонстрационного оборудования (проектор, экран)