Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»

Кафедра «Программного обеспечения автоматизированных систем»

УТВЕРЖДАЮ:
Ректор
_____Н.В. Дубив
«31» августа 2020 г.

Рабочая программа учебной дисциплины

ОБЛАЧНЫЕ ТЕХНОЛОГИИ И РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

образовательной программы высшего образования –
программы магистратуры
09.04.04 Программная инженерия
направленность
Методы и алгоритмы интеллектуальной обработки данных
в информационно-вычислительных системах

формы обучения – очная

Курган 2020

Рабочая программа дисциплины «Облачные технологии и разработка мобильных приложений» составлена в соответствии с учебными планами программы магистратуры «Программная инженерия» (Методы и алгоритмы интеллектуальной обработки данных в информационно-вычислительных системах) очной формы обучения, утвержденными 28.08.2020 г.

Рабочая программа одобрена на заседании кафедры Программного обеспечения автоматизированных систем 29 08.2020 года, протокол № 1.

Рабочую программу составил:

Доцент кафедры
«Программное обеспечение
автоматизированных систем»
к.т.н., доцент _____ А.М. Семахин

Заведующий кафедрой
«Программное обеспечение
автоматизированных систем»
к.т.н., доцент _____ В.К. Волк

Согласовано:

Начальник
Управления
образовательной деятельности _____ С.Н. Синицын

Специалист
по учебно-методической работе
Учебно-методического отдела _____ Г.В. Казанкова

1. ОБЪЕМ ДИСЦИПЛИНЫ

Всего: 5 зачетных единиц трудоемкости (180 академических часов)

Очная форма обучения

Вид учебной работы	На всю дисциплину	Семестр
		4
Аудиторные занятия (контактная работа с преподавателем), всего часов в том числе:	**72**	**72**
Лекции	24	24
Лабораторные работы	48	48
Аудиторные занятия в интерактивной форме, часов	-	-
Самостоятельная работа, всего часов в том числе:	**108**	**108**
Подготовка к экзамену	27	27
Другие виды самостоятельной работы (самостоятельное изучение тем (разделов) дисциплины)	81	81
Вид промежуточной аттестации	**экзамен**	**экзамен**
Общая трудоемкость дисциплины и трудоемкость по семестрам, часов	**180**	**180**

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина «Облачные технологии и разработка мобильных приложений» включена в модуль «Технологии распределённой обработки данных», дисциплины (модули) по выбору 2 (ДВ.2) части, формируемой участниками образовательных отношений, блока 1 учебного плана.

Изучение дисциплины базируется на результатах обучения, сформированных при изучении следующих дисциплин:

- Искусственные нейронные сети и глубокое обучение.
- Методологии и технологии информационно-вычислительных систем.
- Системный анализ, управление и принятие решений.
- Структуры и алгоритмы обработки данных.
- Архитектуры информационно-вычислительных систем.

Результаты изучения дисциплины используются при освоении профильных дисциплин, включенных в модули «Анализ данных и машинное обучение», «Информационно-вычислительные системы», «Технологии распределённой обработки данных» и «Прикладные задачи интеллектуального анализа данных».

3. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

Основная цель изучения дисциплины «Облачные технологии и разработка мобильных приложений» – формирование теоретических знаний описания хода создания и практического применения облачных технологий в разработке мобильных приложений.

Задачи дисциплины:

1) изучение:

–облачная и нативная облачная архитектура;

– стиль конфигурации двенадцатифакторных приложений:

– приёмы комплексного тестирования облачных приложений;

– миграции приложения в облако;

–микросервисы;

– платформа Pivotal Cloud Foundry;

– интеграции данных;

– сервис-брокеры.

2) практическое освоение:

– фреймфорк Spring Boot 2.0;

– языки программирования Kotlin, Java;

Компетенции, формируемые в результате освоения дисциплины:

– способен разрабатывать и использовать программное обеспечение для моделирования, анализа, распознавания и обработки информации, в том числе – в системах искусственного интеллекта (ПК-3);

– способен проектировать архитектуры высокопроизводительных программных систем и проводить оценку их производительности (ПК-4).

В результате освоения дисциплины обучающийся должен демонстрировать следующие ***результаты обучения***:

Должен знать:

– программное обеспечение для моделирования, анализа, распознавания и обработки информации, в том числе – в системах искусственного интеллекта (ПК-3);

– методику проектирования архитектуры высокопроизводительных программных систем и проводить оценку их производительности (ПК-4).

Должен уметь:

– применять программное обеспечение для моделирования, анализа, распознавания и обработки информации, в том числе – в системах искусственного интеллекта (ПК-3);

– применять методику проектирования архитектуры высокопроизводительных программных систем и проводить оценку их производительности (ПК-4).

Должен владеть:

– навыками работы с программным обеспечением для моделирования, анализа, распознавания и обработки информации, в том числе – в системах искусственного интеллекта (ПК-3);

– навыками работы с методикой проектирования архитектуры высокопроизводительных программных систем и проводить оценку их производительности (ПК-4).

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1 Учебно-тематический план. Очная форма обучения. Семестр 4

Рубеж	Номер раздела, темы	Наименование раздела, темы	Количество часов контактной работы с преподавателем	
			Лекции	Практические занятия
Рубеж 1	1	Приложения, оптимизированные для работы в облачной среде	2	8
	2	Проект Spring Boot и платформа Pivotal Cloud Foundry	2	
	3	Миграция приложения в облако	2	
	4	Создание HTTP-и RESTful-сервисов с помощью Spring	2	12
	5	Сервисы маршрутизации Cloud Foundry и пограничные сервисы	2	
	6	Способы управления данными в среде Spring с помощью Spring Data	2	
		Рубежный контроль №1	-	2
Рубеж 2	7	Методы интеграции распределённых сервисов и данных в среде Spring	2	12
	8	Способы приёма и обработки данных с помощью микросервисов	2	
	9	Способы управления состоянием в распределённых системах	2	
	10	Методы создания систем, поддерживающих отслеживаемость и оперативное реагирование	2	12
	11	Способы создания сервис-брокеров для платформы Cloud Foundry	2	
	12	Организационные причины использования Cloud Foundry и архитектура микросервисов	2	
		Рубежный контроль №2	-	2
		Всего:	**24**	**48**

4.2 Содержание лекционных занятий

Тема 1. Приложения, оптимизированные для работы в облачной среде

Технология AWS. Масштабируемость. Надёжность. Адаптивность. Компания Netflix. Параметры облака, характеризуемые различными уровнями абстракции. Основная концепция облачных вычислений: предоставление управленческих услуг для управления виртуализированной вычислительной инфраструктуры (Infrastructure as a Service, IaaS). Два подхода, влияющие на скорость архитектурных изменений: микросервисы и облако. Микросервисы: создание системы из небольших распределённых компонентов. Облако: снижение затрат и усилий, необходимых для управления инфраструктурой. Два преимущества перехода к архитектуре распределённых систем в облаке: адаптивность и надёжность. Недостатки монолитного приложения виртуальной машины Java (Java Virtual Machine, JVM): снижение темпов работы из-за необходимости координировать вносимые изменения, невысокая надёжность. Netflix OOS. Принципы создания приложений. Методология 12 факторов: кодовая база, зависимости, конфигурация, вспомогательные сервисы, сборка, выпуск, практическое применение, процессы, привязка портов, многопоточное выполнение, утилизируемость, функциональная совместимость разработки и практического применения, ведение регистрационных записей, процессы администрирования.

Тема 2. Проект Spring Boot и платформа Pivotal Cloud Foundry

Проект с открытым кодом Spring Initializer, инструмент Spring Boot. Проекты Maven и Gradle. Среда IDE Spring Tool Suite (STS), основанная на Eclipse. Установка Spring Tool Suite (STS). Создание нового проекта с помощью Spring Initializer. Руководства по Spring. Конфигурация. Платформа Cloud Foundry (Platform as a Service, PaaS). Реализации Cloud Foundry. Платформа Pivotal Web Services: повышение производительности за счёт автоматизации. Проект Pivotal Reactor 3.0. Reactor API. Reactive Streams. Контроль обратного потока (backpressure). Стиль конфигурации двенадцатифакторных приложений. Класс PropertyPlaceholderConfigurer. Абстракция Environment и @Value. Профили. Конфигурация Bootiful. Сервер конфигурации Spring Cloud. Клиенты Spring Cloud Config. Обновляемая конфигурация. Комплексное тестирование приложений. ориентированных на выполнение в облаке. Компонентный состав теста. Тестирование в Spring Boot. Тестовые срезы. Имитация, используемая в тестах. Работа с Servlet Container в @SpringBootTest. Аннотации тестирования: @JsonTest, @WebMvcTest, @DataJpaTest, @RestClientTest. Сквозное тестирование. Тестирование распределённых систем. Тестирование контрактов, ориентированных на потребителя. Проект Spring Cloud Contract.

Тема 3. Миграция приложения в облако

Перемещение существующих приложений в облако. Инструмент Cloud Foundry: интерфейс командной строки (CLI). Оригинальные сборочные пакеты (buildpacks). Заказные (подстраиваемые) сборочные пакеты. Приложения

в контейнере. Реструктуризация для перемещения приложения в облако. Обращение к опорным сервисам. Достижение паритета сервисов с помощью Spring. Вызовы удалённых процедур. HTTP-сессии со Spring Session. Сервис сообщений Java. Распределенные транзакции, использующие протокол X/Open XA и JTA. Облачные файловые системы. Управление идентификацией.

Тема 4. Создание HTTP-и RESTful-сервисов с помощью Spring

Протокол REST (Representational State Transfer), применяемый в веб-программировании, поддерживаемый API. Модель зрелости Леонарда Ричардсона. Уровень 0: трясина POX. Уровень 1: ресурсы. Уровень 2: HTTP-операции. Уровень 3: средства управления гипермедиа (Hypermedia controls). Простые REST API, создаваемые с помощью Spring MVC. Согласование содержимого. Чтение и запись двоичных данных. Google Protocol Buffers. Обработка ошибок. Гипермедиа. Управление версиями API. Документирование REST API. Клиентская сторона. REST-клиенты для специализированного исследования и взаимодействия. Шаблон RestTemplate.

Тема 5. Сервисы маршрутизации Cloud Foundry и пограничные сервисы

Маршрутизация в облачной среде: используется программирование, система DNS не подходит. Абстракция DiscoveryClient. Упрощает взаимодействие с реестром и перечисление всех зарегистрированных экземпляров. Библиотека Netflix Ribbon – библиотека балансировки нагрузки на стороне клиента. Стратегии: циклический перебор, балансировка нагрузки на основе распределения по возможности предоставления более объёмных ответов, возможности расширения. Сервисы маршрутизации Cloud Foundry. Пограничные сервисы. Сервис приветствий. Простой пограничный сервис. Библиотека Netflix Feign – библиотека, сводящая клиентов сервиса к простому определению интерфейса и к ряду соглашений. Netflix Feign интегрируется с Spring Cloud. Фильтрация и проксирование с использованием микропрокси-компонента Netflix Zuul. CorsFilter – стандартный фильтр Filter. Типы Zuul-фильтров: предварительные фильтры (pre), фильтры маршрутизации (routing), последующие фильтры (post), фильтры ошибки (error). Обеспечение безопасности в пограничной зоне. Стандарт OAuth 1.0, 1.0a, 2.0. Клиент. Владелец ресурса. Сервер ресурсов, Сервер авторизации. Приложения на стороне сервиса. Одностраничные приложения на HTML и JavaScript. Приложения без пользователей. Доверенные клиенты. Среда Spring Security. Среда Spring Cloud Security. Сервер авторизации Spring Security OAuth. Защита сервера ресурсов приветствий. Создание одностраничного приложения, защищённого OAuth.

Тема 6. Способы управления данными в среде Spring с помощью Spring Data

Моделирование данных. Система управления реляционными базами данных (СУРБД). База данных NoSQL. Проект с открытым кодом Spring Data. Структура приложения Spring Data. Конструирование пакетов Java для данных предметной области. Начало работы с доступом к данным СУРБД на

JDBC. Поддержка имеющейся в Spring технологии JDBC. Сервисы данных проекта Spring Data: Spring Data JPA (MySQL), Spring Data MongoDB, Spring Data Neo4j, Spring Data Redis. Spring Data JPA. Сервис учётных записей Account. Описание предметной области сервиса учётных записей Account с помощью JPA. Проведение аудита с помощью JPA. Комплексные тесты. Spring Data MongoDB. Сервис заказов Order. Классы документов, применяемые при использовании MongoDB. Аудит с использованием MongoDB. Spring Data Neo4j. сервисInventory. Конфигурирование Neo4j. Моделирование графов данных с помощью Neo4j. Проект Spring Data Redis: строки, списки, наборы, хеши, отсортированные наборы, битовые массивы и HyperLogLogs.

Тема 7. Методы интеграции распределённых сервисов и данных в среде Spring

Рассылка сообщений. Уведомления о событиях. Передача состояния за счёт переноса события. Порождение события. Архитектуры, управляемые событиями со Spring Integration. Конечные точки рассылки сообщений. Поставщики сообщений, шаблон конкурирующих потребителей и порождение событий. Распространение типа «публикация-подписка». Распространение от точки к точке. Spring Cloud Stream. Производитель потока. Потребитель потока.

Тема 8. Способы приёма и обработки данных с помощью микросервисов

Пакетные рабочие нагрузки. Пакетная обработка – одновременная программная обработка пакетов входных данных. Среда Spring Batch для поддержки обработки больших объёмов записей. Регистрационно-трассировочные инструменты, управление транзакциями, статистика обработки заданий, перезапуск и пропуск заданий, управление ресурсами. Диспетчеризация. Удалённое разделение задания Spring Batch на части с помощью рассылки сообщений. Управление задачами. Интеграция с рабочим потоком, ориентированная на процесс. Распределение с помощью рассылки сообщений.

Тема 9. Способы управления состоянием в распределённых системах

Теорема CAP Theorem: распределённая система может иметь не более двух из трёх востребованных свойств (согласованность, высокая доступность, допуск к сетевым разделам). Распределённые транзакции. Изоляция сбоев и постепенное снижение качественных характеристик. Сага-шаблон. CQRS (Command Query Responsibility Segregation) – разделение ответственности на команды и запросы. API жалоб. API статистики жалоб. Среда потока данных Spring Cloud Data Flow. Потоки. REST API. Знакомство с клиентами Data Flow. Запросы: регистрация задач и потоковых приложений, определение потока и задач, запуск потоков и задач, опрос статуса запущенных потоков и задач. Инструментальная панель. Оболочка Spring Cloud Data Flow. DataFlowTemplate.

Тема 10. Методы создания систем, поддерживающих отслеживаемость и оперативное реагирование

Аспекты создания систем: создание системы, обладающей масштабируемостью и справляющуюся с бизнес-требованиями, создание системы, справляющуюся с неожиданностями в своей работе. Операции двенадцати факторов. Отслеживаемость. Сравнение отслеживаемости и частоты получения данных при внедрении и извлечении. Получение текущего состояния приложения с помощью Spring Boot Actuator. Показатели. Взаимосвязанные представления показателей. Размерности показателей. Показатели доставки из приложения Spring Boot. Идентификация сервиса с помощью конечной точки/info. Проверки работоспособности. Контрольные события. Ведение журнала приложения. Определение характера выходных регистрационных данных. Определение уровней регистрации. Распределённая трассировка. Spring Cloud Sleuth. OpenZipkin. Отслеживание других платформ и технологий. Информационные панели. Отслеживание нижестоящих сервисов с помощью Hystrix Dashboard. Spring Boot Admin от команды Codecentric. Информационная панель Ordina Microservices Dashboard. AppsManager платформы Pivotal Cloud Foundry. Восстановление работоспособности.

Тема 11. Способы создания сервис-брокеров для платформы Cloud Foundry

Сервис-брокер. Каталог сервисов. Вид со стороны платформы. Cloud Controller. API сервис-брокер. Реализация сервис-брокера с помощью Cloud Foundry Service Broker. Простой сервис-брокер Amazon S3. Управление экземплярами сервиса. Привязки сервисов. Обеспечение безопасности сервис-брокера. Развёртывание. Выпуск с помощью BOSH. Выпуск с помощью Cloud Foundry. Регистрация сервис-брокера Amazon S3. Создание экземпляров сервиса Amazon S3. Клиентское приложение S3.

Тема 12. Организационные причины использования Cloud Foundry и архитектура микросервисов

Непрерывная поставка в Amazon. Конвейер поставки. Этапы: сборка и блочное тестирование, комплексное тестирование, выпуск и развёртывание. Тестирование. Непрерывная поставка для микросервисов. Инструменты. Concourse. Непрерывно поставляемые микросервисы. Установка Concourse. Основная конструкция конвейера. Компонент Maven с присвоенной версией. Развёртывание на платформе Cloud Foundry. Тестирование контрактов, ориентированных на потребителя.

4.3 Практические занятия. Очная форма обучения. Семестр 4

Номер раздела, темы	Наименование раздела, Темы	Наименование практической работы	Норматив времени, час.
1	Приложения, оптимизированные для работы в облачной среде	Разработка и реализация серверной части интернет-мессенджера с помощью фреймворка Spring Boot 2.0	8
2	Проект Spring Boot и платформа Pivotal Cloud Foundry		
3	Миграция приложения в облако		
4	Создание HTTP-и REST-ful-сервисов с помощью Spring	Разработка и создание клиентской части интернет-мессенджера с помощью фреймворка Spring Boot 2.0	12
5	Сервисы маршрутизации Cloud Foundry и пограничные сервисы		
6	Способы управления данными в среде Spring с помощью Spring Data	Рубежный контроль №1	2
7	Методы интеграции распределённых сервисов и данных в среде Spring	Разработка и создание серверной части веб-приложения Place Reviwer на основе платформы Spring	12
8	Способы приёма и обработки данных с помощью микросервисов		
9	Способы управления состоянием в распределённых системах		
10	Методы создания систем, поддерживающих отслеживаемость и оперативное реагирование	Разработка и создание клиентской части веб-приложения Place Reviwer на основе платформы Spring	12
11	Способы создания сервис-брокеров для платформы Cloud Foundry		
12	Организационные причины использования Cloud Foundry и архитектура микросервисов	Рубежный контроль №2	2
			48

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Лекционный курс основывается на методе обучения, использующем технологию, при которой обучающиеся конспектируют теоретический материал, участвуют в опросах и дискуссиях. В этом случае задействованы зрительная, слуховая, моторная и ассоциативная виды памяти.

При прослушивании лекций рекомендуется в конспекте отмечать все важные моменты, на которых заостряет внимание преподаватель, в частности те, которые направлены на качественное выполнение соответствующей лабораторной работы.

Преподавателем запланировано использование при чтении лекций технологии учебной дискуссии. Поэтому рекомендуется фиксировать для себя интересные моменты с целью их активного обсуждения на дискуссии в конце лекции.

Залогом качественного выполнения лабораторных работ является самостоятельная подготовка к ним накануне путем повторения материалов лекций. Рекомендуется подготовить вопросы по неясным моментам и обсудить их с преподавателем в начале занятия.

Лабораторные работы выполняются с применением фреймворка Spring Boot 2.0, языков программирования Kotlin, Java, интегрированной среды программирования Spring Tool Suite и новых версий этих программных продуктов.

Преподавателем запланировано применение на практических занятиях технологий развивающейся кооперации, коллективного взаимодействия, разбора конкретных ситуаций.

Для текущего контроля успеваемости по очной обучения преподавателем используется балльно-рейтинговая система контроля и оценки академической активности. Поэтому настоятельно рекомендуется тщательно прорабатывать материал дисциплины при самостоятельной работе, участвовать во всех формах обсуждения и взаимодействия, как на лекциях, так и на практических занятиях в целях лучшего освоения материала и получения высокой оценки по результатам освоения дисциплины.

Выполнение самостоятельной работы подразумевает самостоятельное изучение разделов дисциплины, подготовку к практическим занятиям, к рубежным контролям, подготовку к экзамену.

Рекомендуемая трудоемкость самостоятельной работы для очной формы обучения представлена в таблице:

Рекомендуемый режим самостоятельной работы
Очная форма

Наименование вида самостоятельной работы	Рекомендуемая трудоемкость, акад. час.
	4 семестр
Самостоятельное изучение тем дисциплины:	**33**
Внутреннее устройство и возможности Spring Boot 2.0	8
Работа с WebFlex и Reactive Data	8
Создание приложений Spring Integration и Spring Cloud Stream	8
Расширение возможностей Spring Boot	9
Подготовка к практическим занятиям (по 2 часа на каждое занятие)	**44**
Подготовка к рубежным контролям (по 2 часа на каждый рубеж)	**4**
Подготовка к экзамену	**27**
Всего:	**108**

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

6.1. Перечень оценочных средств

1. Балльно-рейтинговая система контроля и оценки академической активности обучающихся в КГУ (для очной формы обучения).
2. Отчёты обучающихся по лабораторным работам.
3. Банк заданий к рубежным контролям № 1, № 2 (для очной формы обучения).
4 Банк заданий к экзамену.

6.2. Система балльно-рейтинговой оценки работы обучающихся по дисциплине

Очная форма обучения

№	Наименование	Содержание					
1	Распределение баллов за семестры по видам учебной работы, сроки сдачи учебной ра-	Распределение баллов, 4 семестр					
		Вид учебной работы:	Посещение лекций	Выполнение и защита отчетов по лабораторным работам	Рубежный контроль №1	Рубежный контроль №2	Экзамен

	боты *(доводятся до сведения обучающихся на первом учебном занятии)*	Балльная оценка:	1б*12=12 б	12б*4=48б	5	5	30
2	Критерий пересчета баллов в традиционную оценку по итогам работы в семестре и экзамена	60 и менее баллов – неудовлетворительно; 61…73 – удовлетворительно; 74… 90 – хорошо; 91…100 – отлично.					
3	Критерии допуска к промежуточной аттестации, возможности получения автоматического экзамена по дисциплине, возможность получения бонусных баллов	Для допуска к промежуточной аттестации (экзамену) обучающийся должен набрать не менее 50 баллов, выполнить все лабораторные работы и сдать рубежные контроли. Для получения автоматического экзамена с оценкой удовлетворительно обучающемуся необходимо набрать 68 баллов. По согласованию с преподавателем обучающемуся, набравшему минимум 68 баллов, могут быть добавлены дополнительные (бонусные) баллы за активность на консультациях, активное участие в научной и методической работе, оригинальность принятых решений в ходе выполнения лабораторных работ, за участие в значимых учебных и внеучебных мероприятиях кафедры и выставляется оценка хорошо или отлично автоматически.					
6	Формы и виды учебной работы для неуспевающих (восстановившихся на курсе обучения) обучающихся для получения недостающих баллов в конце семестра	В случае если к промежуточной аттестации (экзамену) набрана сумма менее 50 баллов, обучающемуся необходимо набрать недостающее количество баллов за счет выполнения дополнительных заданий, до конца последней (зачетной) недели семестра. При этом необходимо проработать материал всех пропущенных лабораторных работ для экзамена. Формы дополнительных заданий (назначаются преподавателем): - выполнение и защита пропущенной лабораторной работы для экзамена (при невозможности дополнительного проведения практического занятия преподаватель устанавливает форму дополнительного задания по тематике пропущенной лабораторной работы для экзамена самостоятельно) – до 8 баллов. Ликвидация академических задолженностей, возникших из-за разности в учебных планах при переводе или восстановлении, проводится путем выполнения дополнительных заданий, форма и объем которых определяется преподавателем.					

6.3. Процедура оценивания результатов освоения дисциплины

Рубежные контроли проводятся в виде ответов на вопросы в письменной форме. Экзамен проводится в виде ответов на вопросы билета в устной форме.

Перед проведением рубежного контроля преподаватель прорабатывает с обучающимися основной материал соответствующих разделов дисциплины в форме краткой лекции-дискуссии.

На подготовку к рубежному контролю обучающемуся отводится 2 часа самостоятельной работы. На выполнение тестовых заданий рубежных контролей обучающемуся отводится 2 часа на практических занятиях.

Варианты заданий для рубежных контролей № 1, № 2 состоят из 20 вопросов. Для определения баллов при проверке рубежных контролей используются интервальные оценки, представленные в таблице

Количество правильных ответов	1-5	6-8	9-11	12-14	15-17	18-20
Количество баллов	0	1	2	3	4	5

Преподаватель оценивает в баллах результаты рубежного контроля каждого обучающегося по количеству правильных ответов и заносит в ведомость учета текущей успеваемости.

Билет экзамена содержит 2 вопроса. Вопросы к экзамену доводятся до обучающегося на последней лекции в семестре. На подготовку ответа обучающему отводится 1 астрономический час.

Результаты текущего контроля успеваемости и экзамена заносятся преподавателем в экзаменационную ведомость, которая сдается в организационный отдел института в день экзамена, а также выставляются в зачетную книжку обучающего.

6.4. Примеры оценочных средств для рубежных контролей и экзамена

6.4.1 Примерные задания для рубежного контроля №1 Очная форма обучения, 4 семестр

ВАРИАНТ 1_1

1 Что понимается под облачными технологиями?

1 Технологии использования серверных ресурсов с одновременным запуском большого количества виртуальных серверов, зависимых друг от друга.

2 Технологии использования локальных ресурсов с одновременным запуском большого количества виртуальных серверов, независимо друг от друга.

*3 Технологии использования серверных ресурсов с одновременным запуском большого количества виртуальных серверов, независимо друг от друга.

4 Технологии использования локальных ресурсов с одновременным запуском большого количества виртуальных серверов, зависимых друг от друга.

2Что понимается под облаком?

1 Малое количество высокопроизводительных серверов, на которых запущены виртуальные машины (серверы), общие для каждого пользователя.

*2 Большое количество высокопроизводительных серверов, на которых запущены виртуальные машины (серверы), свои для каждого пользователя.

3 Большое количество малопроизводительных серверов, на которых запущены виртуальные машины (серверы), свои для каждого пользователя.

4 Малое количество высокопроизводительных серверов, на которых запущены виртуальные машины (серверы), общие для каждого пользователя.

3 Какое облако называется частным (private cloud)?

*1 Инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также клиентами и подрядчиками данной организации.

2 Инфраструктура, предназначенная для свободного использования широкой публикой. Может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации). Физически существует в юрисдикции владельца — поставщика услуг.

3 Комбинация из двух или более различных облачных инфраструктур (частных, публичных или общественных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями передачи данных и приложений.

4 Вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи (например, миссии, требований безопасности, политики, и соответствия различным требованиям).

4 Какое облако называется публичным (public cloud)?

1 Вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи (например, миссии, требований безопасности, политики, и соответствия различным требованиям).

2 Комбинация из двух или более различных облачных инфраструктур (частных, публичных или общественных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями передачи данных и приложений.

*3 Инфраструктура, предназначенная для свободного использования широкой публикой. Может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации). Физически существует в юрисдикции владельца — поставщика услуг.

4 Инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также клиентами и подрядчиками данной организации.

5 Какое облако называется гибридным (hybrid cloud)?

1 Вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи

(например, миссии, требований безопасности, политики, и соответствия различным требованиям).

*2 Комбинация из двух или более различных облачных инфраструктур (частных, публичных или общественных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями передачи данных и приложений.

3 Инфраструктура, предназначенная для свободного использования широкой публикой. Может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации). Физически существует в юрисдикции владельца — поставщика услуг.

4 Инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также клиентами и подрядчиками данной организации.

6 Какое облако называется общественным (communitycloud)?

1 Комбинация из двух или более различных облачных инфраструктур (частных, публичных или общественных), остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями передачи данных и приложений.

2 Инфраструктура, предназначенная для свободного использования широкой публикой. Может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций (или какой-либо их комбинации). Физически существует в юрисдикции владельца — поставщика услуг.

3 Инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей (например, подразделений одной организации), возможно также клиентами и подрядчиками данной организации.

*4 Вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи (например, миссии, требований безопасности, политики, и соответствия различным требованиям).

7 Что такое программное обеспечение как услуга (SaaS, Software-as-a-Service)?

1 Модель, когда потребителю предоставляется возможность использования облачной инфраструктуры для размещения базового программного обеспечения для последующего размещения на нём новых или существующих приложений (собственных, разработанных на заказ или приобретённых тиражируемых приложений).

*2 Модель, в которой потребителю предоставляется возможность использования прикладного программного обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских устройств или посредством тонкого клиента, например, из браузера (например, веб-почта) или интерфейс программы.

3 Инфраструктура как услуга, например, виртуальные серверы и виртуальная сеть; клиент может устанавливать любое программное обеспечение и приложения.

4 Модель, в которой потребителю предоставляется возможность использования диаграммы классов обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских и серверных устройств или посредством тонкого клиента или толстого клиента, например, из браузера (например, веб-почта) или интерфейс программы.

8 Что такое платформа как услуга (PaaS, Platform-as-a-Service)?

1 Модель, в которой потребителю предоставляется возможность использования прикладного программного обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских устройств или посредством тонкого клиента, например, из браузера (например, веб-почта) или интерфейс программы.

2 Модель, в которой потребителю предоставляется возможность использования диаграммы классов обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских и серверных устройств или посредством тонкого клиента или толстого клиента, например, из браузера (например, веб-почта) или интерфейс программы.

*3 Модель, когда потребителю предоставляется возможность использования облачной инфраструктуры для размещения базового программного обеспечения для последующего размещения на нём новых или существующих приложений (собственных, разработанных на заказ или приобретённых тиражируемых приложений).

4 Инфраструктура как услуга, например, виртуальные серверы и виртуальная сеть; клиент может устанавливать любое программное обеспечение и приложения.

9 Что такое инфраструктура как услуга (IaaS, Infrastructure as a Service)?

1 Модель, в которой потребителю предоставляется возможность использования прикладного программного обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских устройств или посредством тонкого клиента, например, из браузера (например, веб-почта) или интерфейс программы.

2 Модель, когда потребителю предоставляется возможность использования облачной инфраструктуры для размещения базового программного обеспечения для последующего размещения на нём новых или существующих приложений (собственных, разработанных на заказ или приобретённых тиражируемых приложений).

3 Модель, в которой потребителю предоставляется возможность использования диаграммы классов обеспечения провайдера, работающего в облачной инфраструктуре и доступного из различных клиентских и серверных устройств или посредством тонкого клиента или толстого клиента, например, из браузера (например, веб-почта) или интерфейс программы.

*4 Инфраструктура как услуга, например, виртуальные серверы и виртуальная сеть; клиент может устанавливать любое программное обеспечение и приложения.

10 Что понимается под методологией 12 факторов?

*1 Набор принципов создания приложений, составленный создателями облачной платформы Heroku

2 Набор принципов создания приложений, составленный создателями облачной платформы Xeroku

3 Набор принципов создания приложений, составленный создателями облачной платформы Heroky

4 Набор принципов создания приложений, составленный создателями облачной платформы Xeroky

11 Какие факторы относятся к факторам методологии 12 факторов?

1 Mods

2 Nogs

*3 Logs

4 Sogs

12 Что понимается под Spring Boot?

1 Набор настроенных модулей, работающие в Spring Feign и упрощающие конфигурацию приложения, написанного на Spring.

*2 Набор настроенных модулей, работающие в Spring Framework и упрощающие конфигурацию приложения, написанного на Spring.

3 Набор настроенных модулей, работающие в Spring Frame и упрощающие конфигурацию приложения, написанного на Spring.

4 Набор настроенных модулей, работающие в Spring Flight и упрощающие конфигурацию приложения, написанного на Spring.

13 Что понимается под Pivotal Cloud Foundry?

1 Платформа для облачных систем, выступающая в качестве некоего слоя абстракции для виртуальной среды, позволяющая создать унифицированную площадку, на которой можно запускать любые приложения с привязкой к конкретному облаку или гипервизору.

2 Платформа для обычных систем, выступающая в качестве некоего слоя абстракции для виртуальной среды, позволяющая создать унифицированную площадку, на которой можно запускать любые приложения без привязки к конкретному облаку или гипервизору.

3 Платформа для обычных систем, выступающая в качестве некоего слоя абстракции для виртуальной среды, позволяющая создать унифицированную площадку, на которой можно запускать любые приложения с привязкой к конкретному облаку или гипервизору.

*4 Платформа для облачных систем, выступающая в качестве некоего слоя абстракции для виртуальной среды, позволяющая создать унифицированную площадку, на которой можно запускать любые приложения без привязки к конкретному облаку или гипервизору.

14 Какие преимущества Cloud Foundry?

*1 Переносимость приложений

*2 Обеспечение вертикального и горизонтального масштабирования.

*3 Поддержка различных поставщиков IaaS.

4 Статическая маршрутизация

15 Что понимается под блочным стилем тестирования?

1 Тест, которому в ходе выполнения требуется доступ к контексту приложения Spring (ApplicationContext).

2 Тест, которому в ходе выполнения не требуется доступ к тексту приложения Spring (ApplicationText).

*3 Тест, которому в ходе выполнения не требуется доступ к контексту приложения Spring (ApplicationContext).

4 Тест, которому в ходе выполнения требуется доступ к контексту приложения Spring (ApplicationText).

16 Что понимается под комплексным стилем тестирования?

1 Тест, которому в ходе выполнения не требуется доступ к контексту приложения Spring (ApplicationContext).

2 Тест, которому в ходе выполнения не требуется доступ к тексту приложения Spring (ApplicationText).

3 Тест, которому в ходе выполнения требуется доступ к контексту приложения Spring (ApplicationText).

*4 Тест, которому в ходе выполнения требуется доступ к контексту приложения Spring (ApplicationContext).

17 Платформа Pivotal Cloud Foundry предоставляет долговременную файловую систему?

1 Да

*2 Нет

18 Какая правильная формулировка закона Постела (принцип надёжности)?

1 Реализации сервиса должны быть либеральным в том, что они производят, но консервативными в отношении того, что принимают от других.

2 Реализации сервиса должны быть консервативными в том, что они производят, и консервативными в отношении того, что принимают от других.

*3 Реализации сервиса должны быть консервативными в том, что они производят, но либеральными в отношении того, что принимают от других.

4 Реализации сервиса должны быть либеральными в том, что они производят, но либеральными в отношении того, что принимают от других.

19 Сколько уровней включает модель зрелости Леонарда Ричардсона?

1 1

2 2

3 3

*4 4

20 Какая библиотека от компании Netflix сводит получение клиентов сервиса к простому определению интерфейса и к ряду соглашений?

1 Netfix Ribbon

2 Netfix Zuul
*3Netflix Feign
4 Netflix Gini

6.4.2 Примерные задания для рубежного контроля №2
Очная форма обучения, 4 семестр

ВАРИАНТ 2_1

1 Что представляет собой проект Spring Data?

1 Комплексный проект с закрытым кодом, предоставляющий абстракцию для взаимодействия с хранилищем данных наряду с сохранением особых черт её модели базы данных.

*2 Комплексный проект с открытым кодом, предоставляющий абстракцию для взаимодействия с хранилищем данных наряду с сохранением особых черт её модели базы данных.

3 Комплексный проект с открытым кодом, предоставляющий абстракцию для взаимодействия с хранилищем данных наряду с не сохранением особых черт её модели базы данных.

4 Комплексный проект с закрытым кодом, предоставляющий абстракцию для взаимодействия с хранилищем данных наряду с не сохранением особых черт её модели базы данных.

2 Какое правильное определение класса предметной области?

1 Базовый класс, работающий в качестве модели для данных предметной области, состоящий из набора открытых полей и предоставляющий содержимое, используя геттеры и сеттеры, в зависимости от конструкции модели этой области.

2 Базовый класс, работающий в качестве модели для данных предметной области, состоящий из набора закрытых полей и предоставляющий содержимое, используя теттеры и сеттеры, в зависимости от конструкции модели этой области.

*3 Базовый класс, работающий в качестве модели для данных предметной области, состоящий из набора закрытых полей и предоставляющий содержимое, используя геттеры и сеттеры, в зависимости от конструкции модели этой области.

4 Базовый класс, работающий в качестве модели для данных предметной области, состоящий из набора открытых полей и предоставляющий содержимое, используя теттеры и сеттеры, в зависимости от конструкции модели этой области.

3 Какое правильное программное описание основного класса предметной области, представляющей модель User?

1 *private class User {*
private Song id;
private String firstName;
private String lastName;

```
        private String email;
}
2 public class User {
        private Long id;
        private Text firstName;
        private Text lastName;
        private Text email;
}
3 public class User {
        public Long id;
        public String firstName;
        public String lastName;
        public String email;
}
*4 public class User {
        private Long id;
        private String firstName;
        private String lastName;
        private String email;
}
```

4 Какие правильные сервисы данных проекта Spring Data?

*1 Spring Data JPA (MySQL)

*2 Spring Data MongoDB

*3 Spring Data Neo4j

4 Spring Data JRA

5 Что означает JPA?

*1 Java Persistence API – спецификация, описание которой представлено в качестве части JSR 220, принадлежащей спецификации JCP (Java Community Process), предоставляющая абстракции и реализации для ORM-технологий конкретных производителей таких как Hibernate и DataNucleas.

2 Java Persistence API – спецификация, описание которой представлено в качестве части JSR 320, принадлежащей спецификации JCP (Java Community Process), предоставляющая абстракции и реализации для ORM-технологий конкретных производителей таких как Hibernate и DataNucleas.

3 Java Persistence API – спецификация, описание которой представлено в качестве части JSR 420, принадлежащей спецификации JCP (Java Community Process), предоставляющая абстракции и реализации для ORM-технологий конкретных производителей таких как Hibernate и DataNucleas.

4 Java Persistence API – спецификация, описание которой представлено в качестве части JSR 120, принадлежащей спецификации JCP (Java Community Process), предоставляющая абстракции и реализации для ORM-

технологий конкретных производителей таких как Hibernate и Data-Nucleas.

6 Проект Spring Data Neo4j позволяет управлять данными на основе хранилища для Neo4j, популярной графовой базы данных NoSQL?

*1 Да.

2 Нет.

7 Какие структуры данных поддерживает Redis?

1 Массивы

*2 Хеши

3 Деки

4 Деревья

8 Проект Spring Data Redis позволяет среде Spring интегировать компонент для Redis-хранилища на основе использования пар «ключ–значение»?

*1 Да

2 Нет

9 Какое предназначение Spring Integration?

1 Обеспечение легкого обмена сообщениями в приложениях на базе Spring и поддержка интеграции с внутренними системами с помощью декларативных адаптеров, обеспечивающих более высокий уровень абстракции по сравнению с поддержкой Spring для удаленного взаимодействия, обмена сообщениями и планирования.

2 Обеспечение легкого обмена сообщениями в приложениях на базе Spring и поддержка интеграции с внешними системами с помощью декларативных адаптеров, обеспечивающих более низкий уровень абстракции по сравнению с поддержкой Spring для удаленного взаимодействия, обмена сообщениями и планирования.

*3 Обеспечение легкого обмена сообщениями в приложениях на базе Spring и поддержка интеграции с внешними системами с помощью декларативных адаптеров, обеспечивающих более высокий уровень абстракции по сравнению с поддержкой Spring для удаленного взаимодействия, обмена сообщениями и планирования.

4 Обеспечение легкого обмена сообщениями в приложениях на базе Spring и поддержка интеграции с внутренними системами с помощью декларативных операций, обеспечивающих более высокий уровень абстракции по сравнению с поддержкой Spring для удаленного взаимодействия, обмена сообщениями и планирования.

10 Какая основная цель использования Spring Integration?

1 Предоставление простой модели для построения корпоративных интеграционных решений при не сохранении разделения проблем, которое необходимо для создания поддерживаемого и тестируемого кода.

*2 Предоставление простой модели для построения корпоративных интеграционных решений при сохранении разделения проблем, которое необходимо для создания поддерживаемого и тестируемого кода.

3 Предоставление сложной модели для построения корпоративных интеграционных решений при сохранении разделения проблем, которое необходимо для создания поддерживаемого и тестируемого кода.

4 Предоставление сложной модели для построения корпоративных интеграционных решений при не сохранении разделения проблем, которое необходимо для создания поддерживаемого и тестируемого кода.

11 Какое предназначение фреймворка Spring Cloud Stream?

*1 Для создания масштабируемых событийно-управляемых микросервисов, связанных с общими системами обмена сообщениями, с предоставлением гибкой модели программирования, построенной на идиомах Spring и лучших практиках, включая поддержку постоянной семантики pub/sub, групп потребителей и разделов с отслеживанием состояния.

2 Для создания масштабируемых событийно-управляемых микросервисов, связанных с общими системами обмена сообщениями, с предоставлением гибкой модели программирования, построенной на идиомах Spring и лучших практиках, включая поддержку постоянного синтаксиса pub/sub, групп потребителей и разделов с отслеживанием состояния.

3 Для создания масштабируемых событийно-управляемых микросервисов, не связанных с общими системами обмена сообщениями, с предоставлением гибкой модели программирования, построенной на идиомах Spring и лучших практиках, включая поддержку постоянной семантики pub/sub, групп потребителей и разделов с отслеживанием состояния.

4 Для создания масштабируемых событийно-управляемых микросервисов, связанных с общими системами обмена сообщениями, с предоставлением гибкой модели программирования, построенной на идиомах Spring и лучших практиках, включая поддержку постоянной семантики rub/tub, групп потребителей и разделов с отслеживанием состояния.

12 Какие основные блоки Spring Cloud Stream?

1 Иерархические узлы: элементы, обеспечивающие взаимодействие в сложной структуре.

*2 Целевые связующие: компоненты, ответственные за обеспечение интеграции с внешними системами обмена сообщениями.

*3 Привязки назначения: мост между внешними системами обмена сообщениями и кодом приложения (производитель/потребитель), предоставляемым конечным пользователем.

*4 Сообщение: каноническая структура данных, используемая производителями и потребителями для связи с целевыми связующими (и, следовательно, другими приложениями через внешние системы обмена сообщениями).

13 Какие реализации binder поддерживает Spring Cloud Stream?

*1 RabbitMQ

2 RabbitCloud

*3 Amazon Kinesis

*4 Google PubSub

14 Пакетная обработка – одновременная программная обработка пакетов входных данных?

*1 Да

2 Нет

15 Какое предназначение фреймворка Spring Batch?

1 Spring Batch предоставляет многократно используемые функции, необходимые для обработки небольших объемов записей, включая ведение журнала/трассировку, управление транзакциями, статистику обработки заданий, перезапуск заданий, пропуск и управление ресурсами, продвинутые технические услуги и функции, которые позволят выполнять чрезвычайно объемные и высокопроизводительные пакетные задания с помощью методов оптимизации и секционирования.

2 Spring Batch предоставляет многократно используемые функции, необходимые для обработки больших объемов записей, включая ведение журнала/трассировку, управление транзакциями, статистику обработки заданий, перезапуск заданий, пропуск и управление ресурсами, продвинутые технические услуги и функции, которые позволят выполнять чрезвычайно объемные и высокопроизводительные пакетные задания с помощью методов нечёткого множества.

3 Spring Batch предоставляет многократно используемые функции, необходимые для обработки больших объемов записей, включая ведение журнала/трассировку, управление транзакциями, статистику обработки заданий, перезапуск заданий, пропуск и управление ресурсами, продвинутые технические услуги и функции, которые позволят выполнять чрезвычайно объемные и малопроизводительные пакетные задания с помощью методов экспертных оценок.

*4 Spring Batch предоставляет многократно используемые функции, необходимые для обработки больших объемов записей, включая ведение журнала/трассировку, управление транзакциями, статистику обработки заданий, перезапуск заданий, пропуск и управление ресурсами, продвинутые технические услуги и функции, которые позволят выполнять чрезвычайно объемные и высокопроизводительные пакетные задания с помощью методов оптимизации и секционирования.

16 Какие характеристики фреймворка Spring Batch?

*1 Управление транзакциями

*2 Обработка на основе блоков

*3 Декларативный ввод-вывод

4 Агрегация

17 Что понимается под CQRS?

1 CQRS – подход проектирования программного обеспечения, при котором код, изменяющий состояние, не отделяется от кода, просто читающего это состояние. Подобное разделение может быть логическим и основываться на разных уровнях или оно может быть физическим и включать разные звенья (tiers), или уровни. В основе этого подхода лежит принцип Command-query separation (CQS).

2 CQRS – подход проектирования программного обеспечения, при котором код, изменяющий состояние, отделяется от кода, просто читающего это состояние. Подобное разделение может быть синтаксическим и основываться на разных уровнях или оно может быть физическим и включать разные звенья (tiers), или уровни. В основе этого подхода лежит принцип Command-query separation (CQS).

3 CQRS – подход проектирования программного обеспечения, при котором код, изменяющий состояние, отделяется от кода, просто читающего это состояние. Подобное разделение может быть синтаксическим и основываться на разных уровнях или оно может быть семантическим и включать разные звенья (tiers), или уровни. В основе этого подхода лежит принцип Command-query separation (CQS).

*4 CQRS – подход проектирования программного обеспечения, при котором код, изменяющий состояние, отделяется от кода, просто читающего это состояние. Подобное разделение может быть логическим и основываться на разных уровнях или оно может быть физическим и включать разные звенья (tiers), или уровни. В основе этого подхода лежит принцип Command-query separation (CQS).

18 Какая основная идея CQS?

1 Основная идея CQS – в объекте методы могут быть двух типов:

Queries: Методы возвращают результат, изменяя состояние объекта.
Commands: Методы не изменяют состояние объекта, не возвращая значение.

*2 Основная идея CQS – в объекте методы могут быть двух типов:

Queries: Методы возвращают результат, не изменяя состояние объекта.
Commands: Методы изменяют состояние объекта, не возвращая значение.

3 Основная идея CQS – в объекте методы могут быть двух типов:

Queries: Методы возвращают результат, не изменяя состояние объекта.
Commands: Методы изменяют состояние объекта, возвращая значение.

4 Основная идея CQS – в объекте методы могут быть двух типов:

Queries: Методы возвращают результат, изменяя состояние объекта.
Commands: Методы изменяют состояние объекта, возвращая значение.

19 Какой правильный программный код класса User с одним методом IsEmailValid?

1
```
public class User
{
    public string Email { get; private set; }
    public bool IsEmailValid(string email)
    {
    string isMatch = Regex.IsMatch("email pattern", email);
        if (isMatch)
    {
        Email = email; // Command
    }
    return isMatch; // Query
    }
```

```
}
2 public class User
{
        private string Email { get; private set; }
        private bool IsEmailValid(string email)
        {
        bool isMatch = Regex.IsMatch("email pattern", email);
              if (isMatch)
        {
              Email = email; // Command
        }
        return isMatch; // Query
        }
}
*3 public class User
{
        public string Email { get; private set; }
        public bool IsEmailValid(string email)
        {
        bool isMatch = Regex.IsMatch("email pattern", email);
              if (isMatch)
        {
              Email = email; // Command
        }
        return isMatch; // Query
        }
}
4 public class User
{
        public string Email { get; private set; }
        public string IsEmailValid(string email)
        {
        bool isMatch = Regex.IsMatch("email pattern", email);
              if (isMatch)
        {
              Email = email; // Command
        }
        return isMatch; // Query
        }
}
```

20 Какой правильный программный код после применения принципа CQS и разделения методов на Command и Query?

```
*1 public class User
{
        public string Email { get; private set; }
```

```
        public bool IsEmailValid(string email) // Query
        {
                return Regex.IsMatch("email pattern", email);
        }
                public void ChangeEmail(string email) // Command
        {
                if (IsEmailValid(email) == false)
                throw new ArgumentOutOfRangeException(email);
        Email = email;
        }
}
2 public class User
{
        public string Email { get; private set; }
        public bool IsEmailValid(string email) // Query
        {
                return Regex.IsMatch("email pattern", email);
        }
                public void ChangeEmail(string email) // Command
        {
                if (IsEmailValid(email) = false)
                throw new ArgumentOutOfRangeException(email);
        Email = email;
        }
}
3 public class User
{
        public string Email { get; private set; }
        public bool IsEmailValid(string email) // Query
        {
                return Regex.IsMatch("email pattern", email);
        }
                public void ChangeEmail(string email) // Command
        {
                if (IsEmailValid(email) != false)
                throw new ArgumentOutOfRangeException(email);
        Email = email;
        }
}
4 public class User
{
        public string Email { get; public set; }
        public bool IsEmailValid(string email) // Query
        {
                return Regex.IsMatch("email pattern", email);
```

```
    }
        public void ChangeEmail(string email) // Command
    {
        if (IsEmailValid(email) == false)
        throw new ArgumentOutOfRangeException(email);
    Email = email;
    }
}
```

6.4.3 Примерный перечень вопросов для экзамена

1 Технология AWS. Основные концепции облачных вычислений. Преимущества разработки приложений, оптимизированных для работы в облачной среде.

2 Принципы создания облачных приложений. Методология 12 факторов.

3 Проект с открытым кодом Spring Initializer. Инструмент Spring Boot. Среда IDE Spring Tool Suite.

4 Платформы Cloud Foundry, Pivotal Web Services. Проект Pivotal Reactor 3.0. Reactive Streams.

5 Стиль конфигурации двенадцати факторных приложений. Класс PropertyPlaceholderConfigurer. Абстракции. Профили.

6 Комплексное тестирование облачных приложений. Компонентный состав теста. Тестирование в Spring Boot. Тестовые срезы.

7 Имитация, используемая в тестах. Сквозное тестирование. Тестирование распределённых систем.

8 Перемещение приложения в облако. Инструмент Cloud Foundry. Заказные и сборочные пакеты.

9 Протокол REST. Модель зрелости Леонарда Ричардсона. Простые REST API, создаваемые с помощью Spring MVC.

10 Библиотека Netflix Ribbon. Стратегии. Сервисы маршрутизации Cloud Foundry.

11 Пограничные сервисы. Библиотека Netflix Feign. Фильтрация и проксирование с использованием Netflix Zuul. Типы Zuul фильтров.

12 Моделирование данных. База данных NoSQL. Проект с открытым кодом Spring Data. Структура приложения Spring Data.

13 Сервисы данных проекта Spring Data: Spring Data JPA (MySQL), Spring Data MongoDB, Sping Data Neo4j, Spring Data Redis.

14 Методы интеграции распределённых сервисов и данных в среде Spring.

15 Пакетные рабочие нагрузки. Среда Spring Batch. Диспетчеризация. Удалённое разделение задания Spring Batch на части с помощью рассылки сообщений.

16 Теорема CAP Theorem. Распределённые транзакции. SQRS (Разделение ответственности на команды и запросы).

17 Среда потока данных Spring Cloud Data Flow. Клиенты DataFlow. Оболочка Spring Cloud Data Flow.

18 Аспекты создания систем. Операции двенадцати факторов. Получение текущего состояния приложения с помощью Spring Boot Actuator. Показатели.

19 Распределённая трассировка. Инструмент Spring Cloud Sleuth. Отслеживание других платформ и технологий. Отслеживание нижестоящих сервисов с помощью Hystrix Dashboard. Инструмент Spring Boot Admin от команды Codecentric.

20 Сервис-брокер. Каталог сервисов. Реализация сервис-брокера с помощью Cloud Foundry ServiceBroker.

21 Простой сервис-брокер Amazon S3. Управление экземплярами сервиса. Регистрация сервис-брокераAmazon S3. Создание экземпляра сервиса Amazon S3. Клиентское приложение S3.

22 Организационные причины использования Cloud Foundry и архитектура микросервисов.

6.5. Фонд оценочных средств

Полный банк заданий для текущего, рубежных контролей и промежуточной аттестации по дисциплине, показатели, критерии, шкалы оценивания компетенций, методические материалы, определяющие процедуры оценивания образовательных результатов, приведены в учебно-методическом комплексе дисциплины.

7. ОСНОВНАЯ И ДОПОЛНИТЕЛЬНАЯ УЧЕБНАЯ ЛИТЕРАТУРА

7.1. Основная учебная литература

1 Клашанов, Ф. К. Вычислительные системы и сети, облачные технологии : учебно-методическое пособие / Ф. К. Клашанов. — Москва : МИСИ – МГСУ, 2020. — 40 с. — ISBN 978-5-7264-2187-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/145093.

2 Карр, Н. Великий переход: что готовит революция облачных технологий / Н. Карр ; перевод с английского А. Баранова. — Москва : Манн, Иванов и Фербер, 2014. — 272 с. — ISBN 978-5-91657-892-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/62379.

3 Васильев, Н. П. Введение в гибридные технологии разработки мобильных приложений : учебное пособие для вузов / Н. П. Васильев, А. М. Заяц. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 160 с. — ISBN 978-5-8114-8181-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/173103.

4 Соколова, В. В. Разработка мобильных приложений : учебное пособие / В. В. Соколова. — Томск : ТПУ, 2014. — 176 с. — ISBN 978-5-4387-0369-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/82830.

5 Губарев, В. В. Введение в облачные вычисления и технологии : учеб. пособие / Губарев В. В. – Новосибирск : Изд-во НГТУ, 2013. – 48 с. – ISBN 978-5-7782-2252-6. – Текст : электронный // ЭБС "Консультант студента" : [сайт]. – URL : https://www.studentlibrary.ru/book/ISBN9785778222526.html.

6 Льюис, Ш. , Данн М. Нативная разработка мобильных приложений / Льюис Ш. , Данн М. , пер. с англ. А. Н. Киселева. – Москва : ДМК Пресс, 2020. – 376 с. – ISBN 978-5-97060-845-6. – Текст : электронный // ЭБС "Консультант студента" : [сайт]. – URL : https://www.studentlibrary.ru/book/ISBN9785970608456.html.

7 Сомон, П. Волшебство Kotlin : практическое руководство / П. Сомон ; пер. с анг. А. Н. Киселева. – Москва : ДМК Пресс, 2020. – 536 с. – ISBN 978-5-97060-801-2. – Текст : электронный. – URL: https://znanium.com/catalog/product/1094968.

8 Лонг Джош, Бастани Кеннет Java в облаке. Spring Boot, Spring Cloud, Cloud foundry. – Спб.: Питер, 2019. – 624 с.

9 Фелиппе Гутьерес Spring Boot 2: лучшие практики для профессионалов. – СПб.: Питер, 2021. – 464 с.

7.2. Дополнительная учебная литература

1 ОБЛАЧНЫЕ ТЕХНОЛОГИИ В РОССИЙСКИХ БАНКАХ. Результаты исследования 2015 – Москва :НАФИ, 2015. – 6 с. – Текст : электронный. - URL: https://znanium.com/catalog/product/953779.

2 Жемеров, Д. Kotlin в действии / Д. Жемеров, С. Исакова ; перевод с английского А. Н. Киселев. — Москва : ДМК Пресс, 2018. — 402 с. — ISBN 978-5-97060-497-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/112926.

3 Хабитуев, Б. В. Программирование на языке Java: практикум : учебное пособие / Б. В. Хабитуев. — Улан-Удэ : БГУ, 2020. — 94 с. — ISBN 978-5-9793-1548-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/171791.

4 Аделекан И. Kotlin: программирование на примерах: Пер с англ. – СПб.: БХВ-петербург, 2020. – 432 с.

8. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

1. Семахин А.М. Облачные технологии и разработка мобильных приложений. Методические указания к выполнению лабораторных работ для студентов направления подготовки 09.04.04 «Программная инженерия». Курган, КГУ, 2021. – 56 с. (электронный).

4. Семахин А.М. Облачные технологии в разработке мобильных приложений: учебное пособие. – Курган : Изд-во КГУ, 2021 – 82 с. (электронный).

9. РЕСУРСЫ СЕТИ «ИНТЕРНЕТ», НЕОБХОДИМЫЕ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1. Федеральный портал «Российское образование» URL: http://www.edu.ru/
2. Сайт дистанционного обучения в НОУ «ИНТУИТ». URL: http://www.intuit.ru/

10. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И ИНФОРМАЦИОННЫЕ СПРАВОЧНЫЕ СИСТЕМЫ

При чтении лекций используются слайдовые презентации.

Минимальные требования к операционной системе и программному обеспечению компьютера, используемого при показе слайдовых презентаций: Windows, Foxit Reader.

11. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

11.1 Техническое обеспечение

№	Наименование	Использование
1	Комплект: ноутбук, медиа-проектор, экран	Для демонстрации иллюстративного материала при чтении лекций.
2	Персональный компьютер стандартной комплектации	Используется в качестве инструмента и объекта исследования при выполнении практических работ.

11.2 Программное обеспечение

№	Наименование	Использование
1	Операционная система Windows 10	Управление устройствами компьютерной системы и обеспечение удобного интерфейса для работы.

2	Интегрированная среда программирования Spring Tool Suite, фреймворк Spring Boot 2	Формализация алгоритмов решения задач при выполнении лабораторных работ
3	Языки программирования Kotlin, Java	Формализация алгоритмов решения задач при выполнении лабораторных работ

ПРИЛОЖЕНИЕ 1

Аннотация к рабочей программе дисциплины

«Облачные технологии и разработка мобильных приложений»

образовательной программы высшего образования –
программы магистратуры

09.04.04 Программная инженерия
направленность
Методы и алгоритмы интеллектуальной обработки данных в информационно-вычислительных системах

формы обучения – очная

Трудоемкость освоения дисциплины – 5 зач. ед. (180 акад. часа)

Содержание дисциплины

Технология AWS. Масштабируемость. Надёжность. Адаптивность. Компания Netflix. Параметры облака, характеризуемые различными уровнями абстракции. Основная концепция облачных вычислений: предоставление управленческих услуг для управления виртуализированной вычислительной инфраструктуры (Infrastructure as a Service, IaaS).

Проект с открытым кодом Spring Initializr, инструмент Spring Boot. Проекты Maven и Gradle. Среда IDE Spring Tool Suite (STS), основанная на Eclipse. Установка Spring Tool Suite (STS). Создание нового проекта с помощью Spring Initializr. Руководства по Spring. Конфигурация. Платформа Cloud Foundry (Platform as a Service, PaaS). Реализации Cloud Foundry. Платформа Pivotal Web Services: повышение производительности за счёт автоматизации. Проект Pivotal Reactor 3.0. Reactor API. Reactive Streams. Контроль обратного потока (backpressure). Стиль конфигурации двенадцатифакторных приложений. Класс PropertyPlaceholderConfigurer.

Перемещение существующих приложений в облако. Инструмент Cloud Foundry: интерфейс командной строки (CLI). Оригинальные сборочные пакеты (buildpacks). Заказные (подстраиваемые) сборочные пакеты. Приложения в контейнере. Реструктуризация для перемещения приложения в облако. Обращение к опорным сервисам.

Протокол REST (Representational State Transfer), применяемый в веб-программировании, поддерживаемый API. Модель зрелости Леонарда Ричардсона. Уровень 0: трясина POX. Уровень 1: ресурсы. Уровень 2: HTTP-операции. Уровень 3: средства управления гипермедиа (Hypermedia controls).

Простые REST API, создаваемые с помощью Spring MVC. Согласование содержимого. Чтение и запись двоичных данных. Google Protocol Buffers. Обработка ошибок. Гипермедиа.

Маршрутизация в облачной среде: используется программирование, система DNS не подходит. Абстракция DiscoveryClient. Упрощает взаимодействие с реестром и перечисление всех зарегистрированных экземпляров.

Моделирование данных. Система управления реляционными базами данных (СУРБД). База данных NoSQL. Проект с открытым кодом Spring Data. Структура приложения Spring Data.

Рассылка сообщений. Уведомления о событиях. Передача состояния за счёт переноса события. Порождение события. Архитектуры, управляемые событиями со Spring Integration. Конечные точки рассылки сообщений. Поставщики сообщений, шаблон конкурирующих потребителей и порождение событий.

Пакетные рабочие нагрузки. Пакетная обработка – одновременная программная обработка пакетов входных данных. Среда Spring Batch для поддержки обработки больших объёмов записей.

Теорема CAP Theorem: распределённая система может иметь не более двух из трёх востребованных свойств (согласованность, высокая доступность, допуск к сетевым разделам). Распределённые транзакции. Изоляция сбоев и постепенное снижение качественных характеристик. Сага-шаблон. CQRS (Command Query Responsibility Segregation) – разделение ответственности на команды и запросы. API жалоб. API статистики жалоб. Среда потока данных Spring Cloud Data Flow. Потоки. REST API.

Аспекты создания систем: создание системы, обладающей масштабируемостью и справляющуюся с бизнес-требованиями, создание системы, справляющуюся с неожиданностями в своей работе. Операции двенадцати факторов. Отслеживаемость. Проверки работоспособности. Контрольные события. Ведение журнала приложения. Определение характера выходных регистрационных данных. Определение уровней регистрации.

Сервис-брокер. Каталог сервисов. Вид со стороны платформы. Cloud Controller. API сервис-брокер. Реализация сервис-брокера с помощью Cloud Foundry Service Broker. Простой сервис-брокер Amazon S3. Управление экземплярами сервиса. Привязки сервисов. Обеспечение безопасности сервис-брокера. Развёртывание. Выпуск с помощью BOSH. Выпуск с помощью Cloud Foundry. Регистрация сервис-брокера

Непрерывная поставка в Amazon. Конвейер поставки. Этапы: сборка и блочное тестирование, комплексное тестирование, выпуск и развёртывание. Тестирование. Непрерывная поставка для микросервисов. Инструменты. Concourse. Непрерывно поставляемые микросервисы. Установка Concourse. Основная конструкция конвейера. Компонент Maven с присвоенной версией. Развёртывание на платформе Cloud Foundry. Тестирование контрактов, ориентированных на потребителя.