МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных технологий и прикладной математики

УТВЕРЖДАЮ:
Проректор по учебной работе,
качеству образования – первый
проректор
_____Хагуров Т.А.
подпись
« 29 » августа 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1. О.33 Облачные технологии и бэкэндразработка

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Профиль Искусственный интеллект и аналитика данных

Форма обучения очная

Квалификация бакалавр

Краснодар 2025

Рабочая программа дисциплины «Облачные технологии и бэкэндразработка» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Программу составил(и):
А.А. Полупанов, доцент каф. ИТ, канд. техн., наук, доцент

Рабочая программа дисциплины утверждена на заседании кафедры информационных технологий протокол № 1 от «26» августа 2025г.

Заведующий кафедрой Подколзин В.В.

Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики
протокол № 01 «28» августа 2025 г.
Председатель УМК факультета Коваленко А.В.

подпись

Рецензенты:
Мостовой Евгений Викторович, генеральный директор ООО «Портал-Юг», e-mail: mostovoy@portal-yug.ru

Луценко Евгений Вениаминович, доктор экономических наук, кандидат технических наук, профессор кафедры компьютерных технологий и систем Федерального государственного бюджетное образовательное учреждение высшего образования «Кубанский государственный аграрный университет имени И.Т. Трубилина», e-mail: prof.lutsenko@gmail.com

1 Цели и задачи изучения дисциплины (модуля)

1.1 Цель освоения дисциплины

Цель: Формирование у студентов систематизированных знаний, практических умений и навыков в области проектирования, разработки и развертывания серверной части веб-приложений с использованием современных облачных платформ.

1.2 Задачи дисциплины

- Изучить фундаментальные принципы бэкенд-разработки, архитектурные паттерны и протоколы взаимодействия.
- Освоить практические навыки создания RESTful API и работы с базами данных.
- Сформировать понимание концепции «Инфраструктура как код» (IaC) и научиться применять её на практике.
- Изучить основы контейнеризации приложений с помощью Docker и оркестрации с помощью Kubernetes.
- Освоить принципы работы основных облачных провайдеров (на примере одного или нескольких: Yandex Cloud, AWS, Google Cloud, Azure) и научиться развертывать в них бэкенд-приложения.
- Изучить основы обеспечения безопасности, мониторинга и масштабирования облачных приложений.

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Облачные технологии и бэкэндразработка» относится к обязательной части учебного плана.

Дисциплина относится к вариативной части профессионального цикла. Для успешного освоения дисциплины студенты должны знать основы:

- Программирования (предпочтительно Python, Node.js или Go).
- Структур данных и алгоритмов.
- Основ компьютерных сетей и операционных систем (Linux).

1.4 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Роль 1: Data Analyst (Аналитик данных)

Задачи:

1. Статистический анализ, визуализация данных, предварительная обработка.
2. Создание прогнозных моделей
3. Построение аналитических моделей для поддержки бизнес-решений.

Роль 2: MLOps (Специалист по эксплуатации ИИ)

Задачи:

1. DevOps для ML.
2. Автоматизация, мониторинг ML-систем.
3. Операционное управление жизненным циклом ML-моделей.

Роль 3: AI PM (Менеджер проектов ИИ)

Задачи:

1. Управление ИИ-проектами от идеи до внедрения
2. Анализ бизнес-требований и постановка задач
3. Оценка эффективности и ROI ИИ-решений

Изучение данной учебной дисциплины направлено на формирование у обучающихся следующих компетенций: ОПК-3.2; ОПК-4.1; ОПК-4.2; SS-2.1; SS-2.2

Код и наименование индикатора*	Результаты обучения по дисциплине
ОПК-3 Способен понимать и применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения	
ОПК-3.2 Ориентируется в современных положениях и концепциях прикладного и системного программного обеспечения, архитектуры компьютеров и сетей (в том числе и глобальных), технологии создания и сопровождения программных продуктов и программных комплексов	**Знать:**. Жизненный цикл ПО и основы проектной документации. Требования ИБ на этапах разработки (хранение секретов, валидация данных) **Уметь:**. Использовать полученную информацию для проектирования и реализации бэкенд-сервисов. Применять лучшие практики безопасности при разработке **Владеть:** Навыками работы с Git и CI/CD для коллективной разработки и развертывания.
SS-2 Способен осуществлять свою трудовую деятельность с учётом необходимости эффективной коммуникации и взаимодействия в рамках коллективной проектной работы в сфере ИИ	
SS-2.1 Эффективно коммуницирует с участниками проектной команды при планировании, реализации и анализе результатов работы	Способен формулировать собственное понимание задач и уточнять его у других
SS-2.2 Учитывает профессиональные и ролевые особенности коллег при совместной разработке технических решений и представлении результатов	Способен кратко объяснить, какую функцию выполняет ИИ-система, какие требования необходимо учесть при формировании обучающей выборки и т.п.

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 3 зач. ед. (108 час.), их распределение по видам работ представлено в таблице

Вид учебной работы		**Всего часов**	**Семестры (часы)**
			7
Контактная работа, в том числе:		**54,3**	**54,3**
Аудиторные занятия (всего):		**50**	**50**
Занятия лекционного типа		**16**	16
Лабораторные занятия		**34**	34
Занятия семинарского типа (семинары, практические занятия)			
Иная контактная работа:		**2,3**	**2,3**
Контроль самостоятельной работы (КСР)		**2**	2
Промежуточная аттестация (ИКР)		**0,3**	0,3
Самостоятельная работа, в том числе:		**20**	**20**
Проработка учебного (теоретического) материала		**12**	12
Выполнение индивидуальных заданий (типовой расчет)		**8**	8
Подготовка к текущему контролю			
Контроль:		**35,7**	**35,7**
Подготовка к экзамену		**35,7**	35,7
Общая трудоемкость	**час.**	**108**	**108**
	в том числе контактная работа	**54,3**	**54,3**

	зач. ед	**3**	**3**

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.
Разделы (темы) дисциплины, изучаемые в 7 семестре

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
1.	Введение в Go. Основы синтаксиса и типы.	10	2		4	4
2.	Стандартная библиотека Go, работа с сетью и конкурентностью	12	2		6	4
3.	Создание веб-сервисов на Go: REST, gRPC, фреймворки	12	2		6	4
4.	Работа с данными: БД, миграции, кеширование	12	4		4	4
5.	Облачная инфраструктура: Docker, мониторинг, TLS	24	6		14	4
ИТОГО по разделам дисциплины		**70**	**16**		**34**	**20**
Контроль самостоятельной работы (КСР)		2				
Промежуточная аттестация (ИКР)		0,3				
Подготовка к текущему контролю		35,7				
Общая трудоемкость по дисциплине		**108**				

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

2.3 Содержание разделов (тем) дисциплины

2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
1	Введение в Go. Основы синтаксиса и типы.	Синтаксис Go, статическая типизация, структуры, интерфейсы, основы конкурентности (горутины)	Вопросы к экзамену 1-5
2	Стандартная библиотека Go, работа с сетью и конкурентностью	Обзор stdlib: net/http, context, sync, io. Модель конкурентности: каналы, мьютексы, WaitGroup. Основы сетевых протоколов (L1-L7 OSI)	Вопросы к экзамену 6-10
3	Создание веб-сервисов на Go: REST, gRPC, фреймворки	REST-архитектура, gRPC как альтернатива, фреймворки (Echo, Gin), middleware, валидация. Отказоустойчивость систем	Вопросы к экзамену 11-15
4	Работа с данными: БД, миграции, кеширование	Работа с PostgreSQL, использование ORM (GORM), миграции, кеширование (Redis), основы NoSQL. Логирование приложений	Вопросы к экзамену 16-20
5	Облачная инфраструктура: Docker, мониторинг, TLS	Контейнеризация (Docker, Docker Compose), мониторинг (Prometheus, Grafana), TLS/ACME-сертификаты, P2P (WebRTC), интеграция LLM API (DeepSeek)	Вопросы к экзамену 21-26

2.3.2 Занятия семинарского типа / лабораторные занятия

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
1	Введение в Go. Основы синтаксиса и типы.	1. Основы синтаксиса Go: типы, структуры, интерфейсы	ЛР
2	Стандартная библиотека Go, работа с сетью и конкурентностью	2. Р*абота с горутинами и каналами. Синхронизация* 3. Создание HTTP-сервера с использованием стандартной библиотеки».	ЛР
3	Создание веб-сервисов на Go: REST, gRPC, фреймворки	4. *Реализация REST API с Middleware (логирование, аутентификация)* 5. Разработка gRPC-сервиса с использованием Protocol Buffers	ЛР
4	Работа с данными: БД, миграции, кеширование	6. *Интеграция с PostgreSQL: подключение, миграции, выполнение запросов*	ЛР
5	Облачная инфраструктура: Docker, мониторинг, TLS	7. *онтейнеризация приложения: написание Dockerfile, сборка образа*. 8. *Настройка Docker Compose для запуска приложения с БД и кешем* 9. *Внедрение системы логирования и мониторинга (Prometheus) в приложение* 10. Настройка TLS-сертификатов с помощью ACME (Traefik или Caddy)	ЛР
6		11. Интеграция LLM API (на примере DeepSeek): создание чат-сервиса. 12. Основы WebRTC: создание простого P2P-соединения между клиентами.	ЛР

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.4 Примерная тематика курсовых работ (проектов) – не предусмотрено

2.4 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Вид СРС	Перечень учебно-методического обеспечения дисциплины по выполнению самостоятельной работы
1	Проработка и повторение лекционного материала, материала учебной и научной литературы, подготовка к семинарским занятиям	Методические указания для подготовки к лекционным и семинарским занятиям, утвержденные на заседании УМК ФКТиПМ ФГБОУ ВО «КубГУ», протокол №1 от 28.08.2025 г
2	Подготовка к текущему контролю	Методические указания для подготовки к лекционным и семинарским занятиям, утвержденные на заседании УМК ФКТиПМ ФГБОУ ВО «КубГУ», протокол №1 от 28.08.2025 г

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ограниченными возможностями здоровья (ОВЗ) предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа,

– в форме аудиофайла,

– в печатной форме на языке Брайля.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа,

– в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

3. Образовательные технологии, применяемые при освоении дисциплины (модуля)

В соответствии с требованиями ФГОС программа дисциплины предусматривает использование в учебном процессе следующих образовательные технологии: чтение лекций с использованием мультимедийных технологий; лабораторные занятия.

С точки зрения применяемых методов используются как традиционные информационно-объяснительные лекции, так и интерактивная подача материала с мультимедийной системой. Компьютерные технологии в данном случае обеспечивают возможность разнопланового отображения алгоритмов и демонстрационного материала. Такое сочетание позволяет оптимально использовать отведенное время и раскрывать логику и содержание дисциплины.

Лабораторное занятие позволяет научить студента применять теоретические знания при решении и исследовании конкретных задач. Лабораторные занятия проводятся в компьютерных классах. Подход разбора конкретных ситуаций широко используется как преподавателем, так и студентами при проведении анализа результатов самостоятельной работы. Это обусловлено тем, что в процессе исследования часто встречаются задачи, для которых единых подходов не существует. Каждая конкретная задача при своем исследовании имеет множество подходов, а это требует разбора и оценки целой совокупности конкретных ситуаций.

– Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

– Информационно-коммуникационные технологии (ИКТ) - расширяют рамки образовательного процесса, повышая его практическую направленность, способствуют интенсификации самостоятельной работы учащихся и повышению познавательной активности. В рамках ИКТ выделяются 2 вида технологий:

– Технология использования компьютерных программ – позволяет эффективно дополнить процесс обучения языку на всех уровнях.

– Интернет-технологии – предоставляют широкие возможности для поиска информации, разработки научных проектов, ведения научных исследований.

– проектная технология - индивидуальная или коллективная деятельность по отбору, распределению и систематизации материала по определенной теме, в результате которой составляется проект;

– анализ конкретных ситуаций - анализ реальных проблемных ситуаций, имевших место в соответствующей области профессиональной деятельности, и поиск вариантов лучших решений;

– развитие критического мышления – образовательная деятельность, направленная на развитие у студентов разумного, рефлексивного мышления, способного выдвинуть новые идеи и увидеть новые возможности.

Подход разбора конкретных задач и ситуаций широко используется как преподавателем, так и студентами во время лекций, лабораторных занятий и анализа результатов самостоятельной работы. Это обусловлено тем, что при решении каждой

конкретной задачи имеется, как правило, несколько методов, а это требует разбора и оценки целой совокупности конкретных ситуаций.

4. Оценочные и методические материалы

4.1 Оценочные средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Безопасность информационных систем».

Оценочные средства включает контрольные материалы для проведения **текущего контроля** в форме тестовых заданий, разноуровневых заданий, типовых расчетов и **промежуточной аттестации** в форме вопросов и заданий к экзамену.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Структура оценочных средств для текущей и промежуточной аттестации

Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства	
		Текущий контроль	Промежуточная аттестация
1. Введение в Go. Основы синтаксиса и типы	ОПК-3.2; SS-2.1; SS-2.2	- Лабораторная работа 1: "Основы синтаксиса" - Тестирование по типам данных - Практический квиз по структурам и интерфейсам	Теоретический вопрос на экзамене (вопросы 1-5)
2. Стандартная библиотека Go, работа с сетью и конкурентностью	ОПК-3.2; SS-2.1; SS-2.2	- Лабораторная работа 2: "Горутины и каналы" - Лабораторная работа 3: "HTTP-сервер"	Теоретический вопрос + практическая задача на экзамене (вопросы 6-10)

		- Код-ревью конкурентных решений	
3. Создание веб-сервисов на Go: REST, gRPC, фреймворки	ОПК-3.2; SS-2.1; SS-2.2	- Лабораторная работа 4: "REST API с Middleware" - Лабораторная работа 5: "gRPC-сервис" - Защита проекта API	Теоретический вопрос + анализ архитектурного решения (вопросы 11-15)
4. Работа с данными: БД, миграции, кеширование	ОПК-3.2; SS-2.1; SS-2.2	- Лабораторная работа 6: "Интеграция с PostgreSQL" - Тестирование знаний SQL-инъекций - Практикум по транзакциям	Теоретический вопрос + проектирование схемы БД (вопросы 16-20)
5. Облачная инфраструктура: Docker, мониторинг, TLS	**ОПК-3.2; SS-2.1; SS-2.2**	- Лабораторная работа 7: "Контейнеризация" - Лабораторная работа 8: "Docker Compose" - Лабораторная работа 9: "Мониторинг" - Лабораторная работа 10: "TLS"	Комплексный практический кейс (вопросы 21-26)
6. Комплексный проект	**ОПК-3.2; SS-2.1; SS-2.2**	- Этап 1: Проектирование архитектуры - Этап 2: Реализация core-функциональности - Этап 3: Интеграция и развертывание - Код-ревью проекта	Защита кейса

Показатели, критерии и шкала оценки сформированных компетенций

№ п/п	Код и наименование индикатора	Результаты обучения	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
Соответствие освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **удовлетворительно /зачтено**)				
<u>на пороговом уровне</u>:				
2.	**ОПК-3.2**	**Знать:** Основные этапы жизненного цикла ПО. Назначение Git. **Уметь:** Создавать простые REST-эндпоинты. Выполнять базовые операции в Git. **Владеть:** Навыками работы с Git на уровне пользователя.	Лаб. 3, 4, 6	Вопросы 12, 16
3.	**ОПК-3.2**	**Знать:** Понятия "алгоритмическая сложность", "структуры данных". **Уметь:** Формализовать простую задачу и выбрать структуру данных из готовых вариантов. **Владеть:** Навыками анализа простых алгоритмов.	Лаб. 1, 2	Вопросы 1, 7
4.	**ОПК-3.2**	**Знать:** Базовый синтаксис Go. Простые идиомы языка. **Уметь:** Писать простой рабочий код по образцу. **Владеть:** Навыками написания простейших программ на Go.	Лаб. 1, 3, 4	Вопросы 1, 11
5.	**SS-2.1**	Способен сформулировать вопрос по задаче и понять ответ коллеги.	Лаб. 4, 6	-

6.	**SS-2.2**	Способен объяснить назначение своего модуля в проекте.	Лаб. 4	-
Соответствие освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **хорошо /зачтено**)				
<u>на базовом уровне</u>:				
2.	**ОПК-3.2**	**Знать:** Принципы CI/CD. Основы безопасной разработки (OWASP). **Уметь:** Настраивать пайплайн сборки. Реализовывать базовые механизмы аутентификации. **Владеть:** Навыками работы в команде над одним репозиторием.	Лаб. 4, 5, 7, 8	Вопросы 14, 18
3.	**ОПК-3.2**	**Знать:** Сложность основных операций со структурами данных. Паттерны конкурентного программирования в Go. **Уметь:** Выбирать оптимальные структуры данных для типовых задач бэкенда. Оценивать сложность алгоритмов. **Владеть:** Навыками оптимизации запросов и алгоритмов.	Лаб. 2, 6	Вопросы 8, 19
4.	**ОПК-3.2**	**Знать:** Стандарты написания кода в Go. Принципы тестирования. **Уметь:** Писать идоматичный код. Создавать модульные тесты. Использовать линтеры. **Владеть:** Навыками рефакторинга кода.	Лаб. 3, 4, 5, 6	Вопросы 12, 17
5.	**SS-2.1**	Способен ясно формулировать технические задачи и координировать действия в команде.	Лаб. 5, 8	Защита кейса
6.	**SS-2.2**	Способен объяснить архитектурные решения и взаимодействие компонентов системы.	Лаб. 8, 9	Защита кейса
Соответствие освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **отлично /зачтено**)				
<u>на продвинутом уровне:</u>				
2.	**ОПК-3.2**	**Знать:** Принципы проектирования отказоустойчивых и масштабируемых систем. **Уметь:** Проектировать и развертывать полноценные cloud-native приложения. **Владеть:** Навыками построения полноценного CI/CD пайплайна для сложного проекта.	Лаб. 8, 9, 10	Вопросы 22, 23, 30
3.	**ОПК-3.2**	**Знать:** Специализированные алгоритмы и структуры данных для распределенных систем. **Уметь:** Проектировать и оптимизировать алгоритмы для высоконагруженных систем. **Владеть:** Навыками профилирования и сложной оптимизации производительности.	Лаб. 11, 12	Вопросы 27, 29
4.	**ОПК-3.2**	**Знать:** Паттерны проектирования enterprise-уровня. Принципы DDD, Clean Architecture. **Уметь:** Создавать код производственного качества, готовый к масштабированию. **Владеть:** Навыками создания поддерживаемого и тестируемого кода в команде.	Все лабораторные	Вопросы 25, 28, 30
5.	**SS-2.1**	Способен проводить код-ревью, аргументировать решения и разрешать технические конфликты.	Лаб. 11, 12	Защита проекта
6.	**SS-2.2**	Способен проектировать и объяснять архитектуру сложных распределенных систем с ИИ-компонентами.	Лаб. 11, 12	Защита проект

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Структура лабораторных работ

Лабораторная работа 1: «Основы синтаксиса Go: типы, структуры, интерфейсы»

- Создание структур данных для доменной модели
- Реализация интерфейсов Stringer и error
- Работа с тестами в Go

Лабораторная работа 2: «Работа с горутинами и каналами. Синхронизация»

- Создание concurrent-сканера портов
- Реализация воркер-пула для обработки задач
- Синхронизация с помощью sync.WaitGroup и mutex

Лабораторная работа 3: «Создание HTTP-сервера с использованием стандартной библиотеки»

- Настройка маршрутизации http.ServeMux
- Реализация обработчиков для эндпоинтов
- Добавление middleware для логирования

Лабораторная работа 4: «Реализация REST API с Middleware (логирование, аутентификация)»

- Разработка CRUD API для сущности
- Добавление JWT-аутентификации
- Создание цепочки middleware

Лабораторная работа 5: «Разработка gRPC-сервиса с использованием Protocol Buffers»

- Определение proto-файлов
- Генерация кода и реализация сервера
- Создание gRPC-клиента

Лабораторная работа 6: «Интеграция с PostgreSQL: подключение, миграции, выполнение запросов»

- Настройка подключения к БД
- Создание миграций с помощью goose
- Реализация репозиториев с SQL-запросами

Лабораторная работа 7: «Контейнеризация приложения: написание Dockerfile, сборка образа»

- Создание многоступенчатого Dockerfile
- Оптимизация размера образа
- Настройка health-check

Лабораторная работа 8: «Настройка Docker Compose для запуска приложения с БД и кешем»

- Создание docker-compose.yml
- Настройка сети и томов
- Конфигурирование зависимостей сервисов

Лабораторная работа 9: «Внедрение системы логирования и мониторинга (Prometheus) в приложение»

- Добавление метрик Prometheus
- Настройка сбора метрик
- Визуализация в Grafana

Лабораторная работа 10: «Настройка TLS-сертификатов с помощью ACME (Traefik или Caddy)»

- Конфигурация reverse proxy
- Автоматическое получение сертификатов
- Настройка HTTPS

Лабораторная работа 11: «Интеграция LLM API (на примере DeepSeek): создание чат-сервиса»

- Настройка клиента для API
- Реализация чат-интерфейса
- Обработка потоковых ответов

Лабораторная работа 12: «Основы WebRTC: создание простого P2P-соединения между клиентами»

- Настройка сигнального сервера
- Установка P2P-соединения
- Передача данных между клиентами

Типовые варианты кейсов

Кейс 1: «Разработка высоконагруженного сервиса сокращения URL»

Исходные данные:
Старая система сокращения URL не справляется с нагрузкой 10,000 RPS. Требуется разработать новый сервис на Go с характеристиками:

- Сокращение URL с генерацией короткой ссылки (6 символов)
- Редирект с подсчетом переходов
- 99.9% доступности
- Хранение данных о 100 млн+ ссылок

Порядок решения кейса 1:

Этап 1: Анализ и проектирование (ОПК-3.2, ОПК-4.1)
1.1. Проанализировать требования к нагрузке и определить необходимые ресурсы
1.2. Выбрать алгоритм генерации коротких ссылок (base62, snowflake ID)
1.3. Определить архитектуру: монолит vs микросервисы
1.4. Выбрать СУБД: PostgreSQL vs Redis vs комбинированное решение
1.5. Спроектировать схему БД и API endpoints

Этап 2: Разработка ядра приложения (ОПК-4.1, ОПК-4.2)
2.1. Реализовать алгоритм генерации коротких ссылок

go

```
func GenerateShortURL() string {
    // реализация base62 кодирования
}
```

2.2. Создать структуры данных для хранения URL и статистики
2.3. Реализовать репозиторий для работы с БД
2.4. Написать HTTP handlers для создания и редиректа URL

Этап 3: Оптимизация производительности (ОПК-4.1, ОПК-4.2)
3.1. Добавить кеширование редиректов в Redis
3.2. Реализовать пакетную вставку для счетчиков переходов
3.3. Оптимизировать SQL-запросы и добавить индексы
3.4. Настроить connection pooling к БД

Этап 4: Обеспечение надежности (ОПК-3.2)
4.1. Добавить health checks и метрики Prometheus
4.2. Реализовать circuit breaker для внешних зависимостей
4.3. Настроить structured logging
4.4. Добавить rate limiting для API

Этап 5: Развертывание и мониторинг (ОПК-3.2)
5.1. Создать Dockerfile и docker-compose.yml
5.2. Настроить CI/CD пайплайн
5.3. Развернуть в облаке с балансировщиком нагрузки
5.4. Настроить мониторинг и алертинг

Кейс 2: «Миграция монолита на микросервисную архитектуру»

Исходные данные:
Существующий монолит интернет-магазина не справляется с нагрузкой. Требуется:

- Выделить сервисы: Пользователи, Заказы, Товары, Платежи
- Обеспечить согласованность данных между сервисами
- Сохранить возможность отката изменений

Порядок решения кейса 2:

Этап 1: Анализ доменной области (ОПК-3.2, SS-2.1)
1.1. Провести анализ существующего кода и выделить bounded contexts
1.2. Определить межсервисные взаимодействия и контракты API
1.3. Составить картуру сервисов и их ответственности
1.4. Спроектировать схему базы данных для каждого сервиса

Этап 2: Разработка сервиса Заказов (ОПК-4.1, ОПК-4.2)
2.1. Реализовать gRPC API для управления заказами

protobuf

```
service OrderService {
  rpc CreateOrder(CreateOrderRequest) returns (Order);
  rpc GetOrder(GetOrderRequest) returns (Order);
}
```

2.2. Реализовать паттерн Saga для управления распределенными транзакциями
2.3. Добавить Outbox pattern для надежной отправки событий
2.4. Настроить миграции БД с помощью goose

Этап 3: Реализация межсервисной коммуникации (ОПК-3.2)
3.1. Настроить service discovery с Consul
3.2. Реализовать асинхронную коммуникацию через RabbitMQ/Kafka
3.3. Добавить retry logic и fallback механизмы
3.4. Реализовать distributed tracing с Jaeger

Этап 4: Безопасность и аутентификация (ОПК-3.2)
4.1. Реализовать JWT-based аутентификацию
4.2. Настроить межсервисную аутентификацию с mTLS
4.3. Добавить валидацию входных данных во всех сервисах
4.4. Настроить RBAC для API

Этап 5: Контейнеризация и оркестрация (ОПК-3.2)
5.1. Создать Dockerfile для каждого сервиса
5.2. Написать Kubernetes manifests с health checks
5.3. Настроить Ingress для маршрутизации трафика
5.4. Реализовать canary deployment стратегию

Кейс 3: «Реализация Real-time чата с WebRTC и AI-помощником»

Исходные данные:
Требуется разработать видеочат с возможностью:

- P2P видеосвязи между пользователями
- Текстового чата с AI-ассистентом (DeepSeek API)
- Записи сессий и аналитики

Порядок решения кейса 3:

Этап 1: Сигнальный сервер WebRTC (ОПК-4.2)
1.1. Реализовать WebSocket сервер для обмена SDP offer/answer
1.2. Создать механизм ICE candidate exchange
1.3. Реализовать комнаты для групповых звонков

Этап 2: Интеграция AI-ассистента (ОПК-4.2)
2.1. Настроить клиент для DeepSeek API
2.2. Реализовать обработку потоковых ответов

go

```
func (a *AIAssistant) StreamChat(sessionID string, message string) {
    // обработка потокового ответа от LLM
}
```

2.3. Добавить контекст диалога и историю сообщений
2.4. Реализовать механизм прерывания ответа ассистента

Этап 3: Бэкенд для управления сессиями (ОПК-4.1)
3.1. Разработать систему аутентификации пользователей
3.2. Реализовать хранение истории чатов в PostgreSQL
3.3. Добавить сервис аналитики и метрик использования
3.4. Реализовать модерацию контента

Этап 4: Обеспечение производительности (ОПК-3.2)
4.1. Настроить TURN сервер для NAT traversal
4.2. Реализовать кеширование частых запросов к AI API
4.3. Добавить горизонтальное масштабирование для сигнального сервера
4.4. Настроить мониторинг качества видеосвязи

Этап 5: Безопасность и развертывание (ОПК-3.2)
5.1. Реализовать end-to-end шифрование для текстового чата
5.2. Настроить HTTPS и WSS для всех соединений
5.3. Создать docker-compose для локальной разработки
5.4. Развернуть в облаке с балансировкой нагрузки

Зачетно-экзаменационные материалы для промежуточной аттестации
Примерный перечень вопросов для экзамена

	Вопросы экзамена	**Перечень компетенций (части компетенции**
1	Типы данных в Go: Чем отличаются array и slice? В каких случаях использовать каждый из них?	ОПК-3.2; SS-2.1; SS-2.2
2	Структуры и интерфейсы: Что такое interface в Go? Как работает механизм implicit implementation? Приведите пример использования интерфейса error.	ОПК-3.2; SS-2.1; SS-2.2
3	Горутины: Что такое горутина? Чем отличается от потока ОС? Как ограничить количество одновременно выполняемых горутин?	ОПК-3.2; SS-2.1; SS-2.2

4	Каналы: Для чего используются буферизированные и небуферизированные каналы? Что произойдет при отправке в закрытый канал?	ОПК-3.2; SS-2.1; SS-2.2
5	Синхронизация: Когда использовать `sync.Mutex` vs `sync.RWMutex`? Что такое race condition и как ее обнаружить?	ОПК-3.2; SS-2.1; SS-2.2
6	HTTP-сервер: Как создать HTTP-сервер с помощью стандартной библиотеки? Что такое `http.Handler` и `http.HandlerFunc`?	ОПК-3.2; SS-2.1; SS-2.2
7	REST API: Какие принципы REST нарушены в этом эндпоинте: `GET /api/deleteUser/123`? Как исправить?	ОПК-3.2; SS-2.1; SS-2.2
8	Middleware: Как работает цепочка middleware? Напишите пример middleware для логирования времени выполнения запроса.	ОПК-3.2; SS-2.1; SS-2.2
9	gRPC vs REST: В каких случаях выбрать gRPC, а в каких REST? Какие преимущества у Protocol Buffers перед JSON?	ОПК-3.2; SS-2.1; SS-2.2
10	Аутентификация: Чем отличается JWT от сессионной аутентификации? Как безопасно хранить JWT-токены на клиенте?	ОПК-3.2; SS-2.1; SS-2.2
11	PostgreSQL: Как организовать подключение к БД из Go-приложения? Что такое connection pool и как им управлять?	ОПК-3.2; SS-2.1; SS-2.2
12	Миграции: Зачем нужны миграции БД? Опишите процесс отката неудачной миграции с использованием goose.	ОПК-3.2; SS-2.1; SS-2.2
13	SQL-инъекции: Как предотвратить SQL-инъекции в Go-приложении? Покажите на примере использования prepared statements.	ОПК-3.2; SS-2.1; SS-2.2
14	Транзакции: В каких случаях необходимо использовать транзакции? Как обеспечить атомарность нескольких операций с БД?	ОПК-3.2; SS-2.1; SS-2.2
15	Кеширование: Какие стратегии кеширования вы знаете? Как организовать инвалидацию кеша?	ОПК-3.2; SS-2.1; SS-2.2
16	Dockerfile: Что такое многоступенчатая сборка и зачем она нужна? Как уменьшить размер итогового образа для Go-приложения?	ОПК-3.2; SS-2.1; SS-2.2
17	Docker Compose: Как организовать сетевую связь между контейнерами приложения и БД? Что такое health checks?	ОПК-3.2; SS-2.1; SS-2.2
18	Мониторинг: Какие типы метрик собирает Prometheus? Как добавить кастомную метрику в Go-приложение?	ОПК-3.2; SS-2.1; SS-2.2
19	Логирование: Чем structured logging отличается от обычного? Какие преимущества дает использование JSON-формата для логов?	ОПК-3.2; SS-2.1; SS-2.2
20	TLS/HTTPS: Как автоматически получать и обновлять TLS-сертификаты с помощью Let's Encrypt? В чем преимущество использования Caddy?	ОПК-3.2; SS-2.1; SS-2.2
21	Безопасность: Какие основные угрозы безопасности для веб-приложений и как их mitigate в Go?	ОПК-3.2; SS-2.1; SS-2.2
22	Оптимизация: Ваше приложение начало тормозить при увеличении нагрузки. Опишите пошаговый план диагностики и оптимизации.	ОПК-3.2; SS-2.1; SS-2.2
23	Архитектура: Как превратить монолитное приложение в микросервисную архитектуру? Какие сложности могут возникнуть?	ОПК-3.2; SS-2.1; SS-2.2
24	Масштабирование: Какие метрики необходимо отслеживать в продакшн-среде? Как настроить алертинг при аномальном поведении системы?	ОПК-3.2; SS-2.1; SS-2.2
25	Интеграция: Как организовать обработку потоковых ответов от внешних API (например, LLM API) в вашем сервисе?	ОПК-3.2; SS-2.1; SS-2.2
26	WebRTC: Какую роль играет сигнальный сервер в установлении P2P-соединения? Какие протоколы используются для обмена метаданными?	ОПК-3.2; SS-2.1; SS-2.2
27	Обработка ошибок: Как правильно обрабатывать временные сбои при обращении к внешним API (retry logic, circuit breaker)?	ОПК-3.2; SS-2.1; SS-2.2
28	Безопасность: Обнаружена уязвимость в одной из библиотек вашего приложения. Каков порядок действий?	ОПК-3.2; SS-2.1; SS-2.2

29	Масштабирование: Как превратить ваше монолитное приложение в микросервисную архитектуру? Какие сложности могут возникнуть?	ОПК-3.2; SS-2.1; SS-2.2
30	Мониторинг: Какие метрики необходимо отслеживать в продакшн-среде? Как настроить алертинг при аномальном поведении системы?	ОПК-3.2; SS-2.1; SS-2.2

4.2 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Методические рекомендации, определяющие процедуры оценивания лабораторных работ:

Оценка зависит от:

- **Качества реализации** (работоспособность кода, эффективность)
- **Анализа результатов** (интерпретация, сравнение методов)
- **Оформления и отчетности** (четкость, соответствие стандартам)

Уровень	Требования
Теоретическая подготовка (20%)	
Пороговый (3/5)	Приведены базовые формулы, но без глубокого объяснения.
Базовый (4/5)	Дано обоснование выбранных методов, ссылки на источники.
Продвинутый (5/5)	Сравнение альтернативных подходов, критика ограничений.
Практическая реализация (40%)	
Пороговый (3/5)	Код работает, но без обработки ошибок и оптимизации.
Базовый (4/5)	Чистый код с комментариями, тестами и визуализацией.
Продвинутый (5/5)	Использование векторных операций, сравнение нескольких методов.
Анализ результатов (30%)	
Пороговый (3/5)	Простые графики без интерпретации.
Базовый (4/5)	Сравнение с теоретическими ожиданиями (например, сходимость градиентного спуска).
Продвинутый (5/5)	Анализ влияния гиперпараметров, статистические выводы.
Оформление отчета (10%)	
Пороговый (3/5)	Есть введение, код и выводы.
Базовый (4/5)	Структурированный отчет с графиками и таблицами.

Продвинутый (5/5)	LaTeX-документ или Jupyter Notebook с интерактивными визуализациями.

Критерии оценки кейсов:

Архитектурные решения (30%) - обоснованность выбора технологий
Качество кода (25%) - соответствие стандартам, тестирование
Производительность (20%) - оптимизация, обработка нагрузки
Безопасность (15%) - защита данных, предотвращение уязвимостей
Документация (10%) - описание решения, deployment guide

Каждый кейс позволяет проверить несколько компетенций в комплексе, имитируя реальные рабочие задачи.

4.3 Методические указания по организации лабораторных работ по дисциплине

«Облачные технологии и бэкэндразработка»

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Цель лабораторного практикума:
Формирование системных практических навыков проектирования, разработки и развертывания бэкенд-приложений с использованием современных облачных платформ и технологий.

1.2. Задачи:

- Освоить полный цикл разработки бэкенд-сервисов
- Приобрести навыки работы с облачными платформами (Yandex Cloud/AWS)
- Освоить принципы контейнеризации и оркестрации
- Научиться реализовывать системы мониторинга и безопасности

1.3. Форма организации:

- Индивидуальное выполнение работ
- Поэтапное усложнение заданий
- Обязательная защита каждой работы

2. ОРГАНИЗАЦИЯ ВЫПОЛНЕНИЯ РАБОТ

2.1. Подготовительный этап:

bash

 # Настройка окружения
- Установка Go 1.21+, Docker, Docker Compose
- Регистрация в облачной платформе (Yandex Cloud)
- Создание GitHub/GitLab репозитория

- Настройка IDE (VS Code с Go плагинами)

2.2. Общие требования ко всем работам:

- Код должен соответствовать Code Style выбранного языка
- Обязательное использование системы контроля версий
- Наличие документации (README.md)
- Написание unit-тестов для ключевых функций

2.3. Процесс выполнения:

1. Изучение теоретического материала
2. Получение задания у преподавателя
3. Поэтапная реализация
4. Тестирование и отладка
5. Code review с преподавателем
6. Защита работы

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО КОНКРЕТНЫМ РАБОТАМ

Лаб. работа 1: «Разработка базового HTTP-сервера»

go

```go
// Цель: освоить создание REST API на Go
package main

import (
    "net/http"
    "encoding/json"
)

func healthHandler(w http.ResponseWriter, r *http.Request) {
    json.NewEncoder(w).Encode(map[string]string{"status": "ok"})
}

func main() {
    http.HandleFunc("/health", healthHandler)
    http.ListenAndServe(":8080", nil)
}
```

Критерии оценки: Корректность обработчиков, структура проекта, обработка ошибок.

Лаб. работа 2: «Реализация аутентификации и авторизации»

go

```
// Цель: освоить JWT и middleware
func authMiddleware(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        token := r.Header.Get("Authorization")
        // валидация JWT токена
        if !isValidToken(token) {
            http.Error(w, "Unauthorized", http.StatusUnauthorized)
            return
        }
        next.ServeHTTP(w, r)
    })
}
```

Требования: Безопасное хранение паролей, refresh tokens, ролевая модель.

Лаб. работа 3: «Интеграция с PostgreSQL»

sql

```
-- Миграция для создания таблицы пользователей
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);
```

go

```
// Репозиторий для работы с БД
type UserRepository struct {
    db *sql.DB
}

func (r *UserRepository) CreateUser(user *User) error {
    // использование prepared statements
}
```

Методические указания: Использование миграций, транзакции, connection pooling.

Лаб. работа 4: «Контейнеризация приложения»

dockerfile

```
# Многоступенчатая сборка
FROM golang:1.21-alpine AS builder
WORKDIR /app
COPY go.mod go.sum ./
RUN go mod download
```

```
COPY . .
RUN CGO_ENABLED=0 go build -o main .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /app/main .
COPY --from=builder /app/config.yaml .
EXPOSE 8080
CMD ["./main"]
```

Критерии оценки: Оптимизация размера образа, безопасность, multi-stage build.

Лаб. работа 5: «Развертывание в облаке»

yaml

```yaml
# docker-compose.yml для локальной разработки
version: '3.8'
services:
  app:
    build: .
    ports:
      - "8080:8080"
    environment:
      - DATABASE_URL=postgresql://user:pass@db:5432/app
    depends_on:
      - db

  db:
    image: postgres:15
    environment:
      - POSTGRES_DB=app
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=pass
```

Требования: Настройка облачной инфраструктуры, безопасность доступа, мониторинг.

Лаб. работа 6: «Внедрение мониторинга»

go

```go
// Интеграция Prometheus метрик
func init() {
    prometheus.MustRegister(requestCount)
}
```

```
func metricsMiddleware(next http.Handler) http.Handler {
   return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
      requestCount.WithLabelValues(r.Method, r.URL.Path).Inc()
      next.ServeHTTP(w, r)
   })
}
```

Методические указания: Настройка Grafana, сбор бизнес-метрик, алертинг.

Лаб. работа 7: «Реализация кеширования»

go

```
// Интеграция Redis для кеширования
type Cache struct {
   client *redis.Client
}

func (c *Cache) Get(key string) ([]byte, error) {
   return c.client.Get(key).Bytes()
}

func (c *Cache) Set(key string, value []byte, expiration time.Duration) error {
   return c.client.Set(key, value, expiration).Err()
}
```

Требования: Стратегии инвалидации, обработка сбоев кеша.

4. ТРЕБОВАНИЯ К ОТЧЕТНОСТИ

4.1. Структура отчета:

- Титульный лист
- Цель и задачи работы
- Описание реализованной функциональности
- Листинги ключевых фрагментов кода

- Результаты тестирования
- Выводы и возникшие трудности

4.2. Критерии оценки:

- **Отлично (5):** Полное выполнение + дополнительные улучшения + качественные тесты
- **Хорошо (4):** Полное выполнение основных требований
- **Удовлетворительно (3):** Выполнение базовых требований с незначительными ошибками
- **Неудовлетворительно (2):** Критические ошибки или невыполнение требований

4.3. Сроки сдачи:

- Каждая работа выполняется в течение 2 академических недель
- Защита проводится индивидуально с демонстрацией функциональности
- Возможна одна повторная защита после исправления замечаний

5. РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ

5.1. Работа с кодом:

- Соблюдение принципов SOLID и Clean Architecture
- Регулярные коммиты с осмысленными сообщениями
- Использование линтеров и форматеров кода
- Написание модульных и интеграционных тестов

5.2. Безопасность:

- Проверка зависимостей на уязвимости (govulncheck)
- Хранение секретов в environment variables
- Валидация всех входных данных
- Использование HTTPS в продакшене

5.3. Документирование:

- Четкие инструкции по запуску в README.md
- Описание API endpoints (OpenAPI/Swagger)
- Комментарии для сложной бизнес-логики

markdown

```
# Пример структуры README.md
## Описание проекта
## Требования
## Установка и запуск
## API Endpoints
## Тестирование
```

6. ФОРМИРОВАНИЕ ИТОГОВОЙ ОЦЕНКИ

6.1. Веса лабораторных работ:

- Лабораторные работы 1-2: 20%
- Лабораторные работы 3-4: 30%
- Лабораторные работы 5-7: 40%
- Активность и качество код-ревью: 10%

6.2. Дополнительные баллы:

- Реализация дополнительного функционала: +1 балл
- Качественная документация: +1 балл
- Высокое покрытие тестами: +1 балл

6.3. Штрафные баллы:

- Нарушение сроков сдачи: -1 балл
- Несоответствие code style: -1 балл
- Отсутствие тестов: -2 балла

7. ТЕХНИКА БЕЗОПАСНОСТИ

- Не размещать секреты в публичных репозиториях
- Использовать .env файлы для конфигурации
- Регулярно обновлять зависимости
- Проверять Docker образы на уязвимости

Методические указания обеспечивают последовательное освоение современных технологий бэкенд-разработки и облачной инфраструктуры.

Соответствие лабораторных работ и индикаторов компетенций

В таблице ниже представлено, как каждая лабораторная работа способствует формированию заявленных компетенций.

Компетенция	Соответствующие лабораторные работы	Обоснование
ОПК-5.1 Применяет современные языки программирования и технологии для решения математических и вычислительных задач	**ЛР 1, 2, 3, 7** - ЛР 1: Формализация обработки HTTP-запросов - ЛР 2: Разработка алгоритмов аутентификации - ЛР 3: Проектирование структуры БД, оптимизация запросов - ЛР 7: Реализация алгоритмов кеширования	Работы требуют анализа сложности алгоритмов, выбора оптимальных структур данных, формализации бизнес-логики
ОПК-6.1 Разрабатывает эффективные алгоритмы и реализует их в виде программного кода с учетом временной и пространственной сложности	**ЛР 1, 2, 3, 4, 6, 7** - ЛР 1-3: Написание production-ready кода на Go - ЛР 4: Создание Dockerfile по best practices - ЛР 6: Интеграция мониторинга в код приложения - ЛР 7: Реализация кеширования с обработкой ошибок	Все работы требуют написания чистого, тестируемого кода, соответствующего стандартам языка и готового к интеграции в реальные системы
SS-2.1 Эффективно коммуницирует с участниками проектной команды	**ЛР 5, 6** - ЛР 5: Документирование процесса развертывания - ЛР 6: Описание метрик для команды разработки - Все ЛР: Защита работ, объяснение решений	Формирование навыков технической коммуникации, документирования решений, взаимодействия при код-ревью
SS-2.2 Учитывает профессиональные особенности при совместной разработке	**ЛР 4, 5** - ЛР 4: Создание воспроизводимой среды для команды - ЛР 5: Настройка	Разработка решений, учитывающих потребности разных участников команды (разработчики, DevOps, тестировщики)

Компетенция	Соответствующие лабораторные работы	Обоснование
	инфраструктуры для коллективной работы - Все ЛР: Следование принятым в команде стандартам	

Итоговый контроль

- Все задания выполнены в срок.
- Отчет защищен (студент может объяснить любую часть кода).
- Результаты воспроизводимы (запуск на другом компьютере дает тот же вывод).

Преподаватель может использовать этот чек-лист при проверке работ, а также рекомендовать студентам применять его для самоконтроля. Этот подход минимизирует ошибки и обеспечит системное освоение компетенций.

5. Перечень учебной литературы, информационных ресурсов и технологий

5.1 Учебная литература:

1. Шелудько, В. М. Основы программирования на языке высокого уровня Python : учебное пособие : [16+] / В. М. Шелудько. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2017. – 147 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=500056 (дата обращения: 20.11.2024). – Библиогр. в кн. – ISBN 978-5-9275-2649-9. – Текст : электронный.
2. Баланов, А. Н. Облачные технологии : учебное пособие для вузов / А. Н. Баланов. — 2-е изд., стер. — Санкт-Петербург : Лань, 2025. — 204 с. — ISBN 978-5-507-53005-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/464192 (дата обращения: 12.11.2025).
3. Золкин, А. Л. Математические модели автоматов и формальных языков в сфере искусственного интеллекта : учебное пособие для вузов / А. Л. Золкин. — Санкт-Петербург : Лань, 2025. — 160 с. — ISBN 978-5-507-52513-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/494984 (дата обращения: 12.11.2025).
4. Борзунов, С. В. Языки программирования. Python: решение сложных задач / С. В. Борзунов, С. Д. Кургалин. — Санкт-Петербург : Лань, 2023. — 192 с. — ISBN 978-5-507-45923-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/319394 (дата обращения: 20.11.2024). — Режим доступа: для авториз. пользователей.
5. Букунов, С. В. Разработка приложений с графическим пользовательским интерфейсом на языке Python / С. В. Букунов, О. В. Букунова. — Санкт-Петербург : Лань, 2023. — 88 с. — ISBN 978-5-507-45191-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/292856 (дата обращения: 20.11.2024). — Режим доступа: для авториз. пользователей.
6. Рощин, Сергей Михайлович. Современные интернет-**технологии**: семь главных трендов / Сергей Рощин. - 3-е изд. - Москва : Издательско-торговая корпорация "Дашков и К°", 2023. - 123 с. : ил. - Библиогр. в конце гл. - ISBN 978-5-394-05510-2 : 133 р. - Текст : непосредственный. URL - http://megapro.kubsu.ru/MegaPro/UserEntry?Action=Link_FindDoc&id=278432&idb=0
7. Марк, С. Программирование на Go. Разработка приложений XXI века : учебное пособие / С. Марк ; перевод с английского А. Н. Киселёв. — Москва : ДМК Пресс, 2013. — 580 с. — ISBN 978-5-94074-854-0. — Текст : электронный // Лань : электронно-

библиотечная система. — URL: https://e.lanbook.com/book/69944 (дата обращения: 12.11.2025).
8. Sun, X., Li, J., Kovalenko, A.V., Feng, W., Ou, Y. Integrating Reinforcement Learning and Learning From Demonstrations to Learn Nonprehensile Manipulation //IEEE Transactions on Automation Science and Engineering, 2023, 20(3), 1735–1744, DOI: 10.1109/TASE.2022.3185071, Q1
9. Petukhova, A.V.; Kovalenko, A.V.; Ovsyannikova, A.V. Algorithm for Optimization of Inverse Problem Modeling in Fuzzy Cognitive Maps. Mathematics 2022, 10, 3452. DOI: 10.3390/math10193452, Q1
10. Kirillova, E.; Kovalenko, A.; Urtenov, M. Study of the Current–Voltage Characteristics of Membrane Systems Using Neural Networks. AppliedMath 2025, 5, 10. https://doi.org/10.3390/appliedmath5010010
11. Polykovskiy, Daniil, et al. "Molecular sets (MOSES): a benchmarking platform for molecular generation models." Frontiers in pharmacology 11 (2020): 565644.
12. Khrabrov, Kuzma, et al. "$\nabla^ 2$ DFT: A Universal Quantum Chemistry Dataset of Drug-Like Molecules and a Benchmark for Neural Network Potentials." Advances in Neural Information Processing Systems 37 (2024): 36869-36889.

5.2. Периодические издания и конференции (A*):

1. IEEE Transactions on Big Data – научные статьи по обработке больших данных.
2. Journal of Big Data (SpringerOpen) – открытый журнал с исследованиями в области Big Data.
3. Big Data Research (Elsevier) – публикации по анализу, управлению и визуализации данных.
4. Data Science Journal (CODATA) – междисциплинарные исследования данных.
5. ACM Transactions on Knowledge Discovery from Data (TKDD) – методы извлечения знаний из больших данных.
6. https://openreview.net/forum?id=FMMF1a9ifL
7. https://openreview.net/forum?id=ElUrNM9U8c#discussion
8. https://openreview.net/forum?id=JoO6mtCLHD
9. https://aclanthology.org/2024.findings-emnlp.760/
10. https://aclanthology.org/2020.coling-main.588/
11. https://link.springer.com/chapter/10.1007/978-3-030-72113-8_30
12. https://link.springer.com/chapter/10.1007/978-3-031-42448-9_10
13. https://aclanthology.org/2024.findings-naacl.288/

5.3. Интернет-ресурсы, в том числе современные профессиональные базы данных и информационные справочные системы

Электронно-библиотечные системы (ЭБС):

1. ЭБС «ЮРАЙТ» https://urait.ru/
2. ЭБС «УНИВЕРСИТЕТСКАЯ БИБЛИОТЕКА ОНЛАЙН» http://www.biblioclub.ru/
3. ЭБС «BOOK.ru» https://www.book.ru
4. ЭБС «ZNANIUM.COM» www.znanium.com
5. ЭБС «ЛАНЬ» https://e.lanbook.com

Профессиональные базы данных

1. Scopus http://www.scopus.com/
2. ScienceDirect https://www.sciencedirect.com/
3. Журналы издательства Wiley https://onlinelibrary.wiley.com/
4. Научная электронная библиотека (НЭБ) http://www.elibrary.ru/
5. Полнотекстовые архивы ведущих западных научных журналов на Российской платформе научных журналов НЭИКОН http://archive.neicon.ru

6. Национальная электронная библиотека (доступ к Электронной библиотеке диссертаций Российской государственной библиотеки (РГБ) https://rusneb.ru/
7. Springer eBooks (i.e. 2020 eBook collections): https://link.springer.com/
8. "Лекториум ТВ" http://www.lektorium.tv/
9. Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Информационные справочные системы

1. Консультант Плюс - справочная правовая система (доступ по локальной сети с компьютеров библиотеки)

Ресурсы свободного доступа

1. КиберЛенинка http://cyberleninka.ru/;
2. Американская патентная база данных http://www.uspto.gov/patft/
3. Министерство науки и высшего образования Российской Федерации https://www.minobrnauki.gov.ru/;
4. Федеральный портал "Российское образование" http://www.edu.ru/;
5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru/;
6. Единая коллекция цифровых образовательных ресурсов http://school-collection.edu.ru/ .
7. Проект Государственного института русского языка имени А.С. Пушкина "Образование на русском" https://pushkininstitute.ru/;
8. Справочно-информационный портал "Русский язык" http://gramota.ru/;
9. Служба тематических толковых словарей http://www.glossary.ru/;
10. Словари и энциклопедии http://dic.academic.ru/;
11. Образовательный портал "Учеба" http://www.ucheba.com/;
12. Законопроект "Об образовании в Российской Федерации". Вопросы и ответы http://xn--273--84d1f.xn--p1ai/voprosy_i_otvety

Собственные электронные образовательные и информационные ресурсы КубГУ

1. Электронный каталог Научной библиотеки КубГУ http://megapro.kubsu.ru/MegaPro/Web
2. Электронная библиотека трудов ученых КубГУ http://megapro.kubsu.ru/MegaPro/UserEntry?Action=ToDb&idb=6
3. Среда модульного динамического обучения http://moodle.kubsu.ru
4. База учебных планов, учебно-методических комплексов, публикаций и конференций http://infoneeds.kubsu.ru/
5. Библиотека информационных ресурсов кафедры информационных образовательных технологий http://mschool.kubsu.ru;
6. Электронный архив документов КубГУ http://docspace.kubsu.ru/
7. Электронные образовательные ресурсы кафедры информационных систем и технологий в образовании КубГУ и научно-методического журнала "ШКОЛЬНЫЕ ГОДЫ" http://icdau.kubsu.ru/

5.4 Перечень информационно-коммуникационных технологий

1. Облачные платформы и сервисы

AWS/GCP/Azure/YandexCloud/cloud.ru – облачные вычисления для использования LLM

2. Системы управления версиями и коллаборации

Git/GitHub/GitLab – контроль версий кода и совместная разработка

3. Система управления обучением

Moodle – сдача работ

5.5 Перечень лицензионного и свободно распространяемого программного обеспечения

Свободное ПО (Open Source)

Go (Golang) - Язык программирования

Инструменты для разработки ИИ-систем:

SWI Prolog – реализация логических и экспертных систем

Java – язык программирования общего назначения с мощной экосистемой, активно используемый в разработке ИИ-систем, особенно в корпоративной среде и образовании

Python – основной язык программирования для ИИ-приложений

Библиотеки для работы с знаниями и логикой:

networkx – построение и анализ семантических сетей

rdflib – работа с RDF и онтологиями

OWLready2 – загрузка и модификация онтологий в формате OWL

Системы управления знаниями:

Protégé – инструмент для создания онтологий

Apache Jena – платформа для работы с RDF, RDFS и OWL

Neo4j – графовая СУБД для представления знаний в виде графов

Инструменты для нечеткой логики и агентов:

jFuzzyLogic – библиотека нечеткой логики на Java

scikit-fuzzy – библиотека нечеткой логики на Python

simpful – реализация нечетких систем на Python

Визуализация и анализ:

Graphviz – построение графов и диаграмм (семантические сети, деревья вывода)

Matplotlib / Plotly – визуализация результатов работы ИИ-моделей

Streamlit / Gradio – создание веб-интерфейсов для экспертных систем и систем поддержки принятия решений

СУБД и хранение знаний:

SQLite / PostgreSQL – хранение структурированных данных и знаний

MongoDB – хранение неструктурированных данных и графов знаний

Neo4J – обработка RDF-данных

6. Методические указания для обучающихся по освоению дисциплины (модуля)

По курсу предусмотрено проведение лекционных занятий, на которых дается основной систематизированный материал. В ходе лекционных занятий разбираются элементы теории и практики дискретной математики, приводятся примеры решения задач, проводится анализ наиболее распространенных ошибок. После прослушивания лекции рекомендуется выполнить упражнения, приводимые в аудитории для самостоятельной работы.

По курсу предусмотрено проведение лабораторных занятий, на которых дается прикладной систематизированный материал. В ходе занятий разбираются методы решений задач по темам. После занятия рекомендуется выполнить упражнения, приводимые для самостоятельной работы.

При самостоятельной работе студентов необходимо изучить литературу, приведенную в перечнях выше, для осмысления вводимых понятий, анализа предложенных подходов и методов дискретной математики. При решении новой задачи студент должен уметь выбрать метод решения и его обоснование.

Важнейшим этапом курса является самостоятельная работа по дисциплине. В процессе самостоятельной работы студент приобретает навыки работы с дискретными объектами.

Используются активные, инновационные образовательные технологии, которые способствуют развитию общекультурных, общепрофессиональных компетенций и профессиональных компетенций обучающихся:

- проблемное обучение;
- разноуровневое обучение;
- проектные методы обучения;
- исследовательские методы в обучении;
- обучение в сотрудничестве (командная, групповая работа);
- информационно- коммуникационные технологии.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Учебно-методическим обеспечением курсовой работы студентов являются:

1. учебная литература;
2. нормативные документы ВУЗа;
3. методические разработки для студентов.

Самостоятельная работа студентов включает:

– оформление итогового отчета (пояснительной записки).
– анализ нормативно-методической базы организации;
– анализ научных публикации по заранее определённой теме;
– анализ и обработку информации;
– работу с научной, учебной и методической литературой,
– работа с конспектами лекций, ЭБС.

Для самостоятельной работы представляется аудитория с компьютером и доступом в Интернет, к электронной библиотеке вуза и к информационно-справочным системам.
Перечень учебно-методического обеспечения:

1. Основная образовательная программа высшего профессионального образования федерального государственного бюджетного образовательного учреждения высшего образования «Кубанский государственный университет» по направлению подготовки.
2. Положение о проведении текущего контроля успеваемости и промежуточной аттестации в федеральном государственном бюджетном образовательном учреждении высшего образования «Кубанский государственный университет».
3. Общие требования к построению, содержанию, оформлению и утверждению рабочей программы дисциплины Федерального государственного образовательного стандарта высшего профессионального образования.
4. Методические рекомендации по содержанию, оформлению и применению образовательных технологий и оценочных средств в учебном процессе, основанном на Федеральном государственном образовательном стандарте.
5. Учебный план основной образовательной программы по направлению подготовки.
6. Федеральный государственный образовательный стандарт высшего профессионального образования по направлению подготовки.

В освоении дисциплины инвалидами и лицами с ограниченными возможностями здоровья большое значение имеет индивидуальная учебная работа (консультации) – дополнительное разъяснение учебного материала.

Индивидуальные консультации по предмету являются важным фактором, способствующим индивидуализации обучения и установлению воспитательного контакта между преподавателем и обучающимся инвалидом или лицом с ограниченными возможностями здоровья.

Важнейшим компонентом курса является самостоятельная проектная работа, в ходе которой студент разрабатывает законченное решение для решения задач (кейсов) индустриальных партнеров. Допускается выполнение проектов в командах.

Подход, определяющий установление соответствия кейсов ИП и УГТ (5-7), позволяет четко соотносить этапы развития технологии с вовлеченностью партнера и снижать риски при переходе от лабораторных испытаний к промышленному внедрению.

А. Кейсы дисциплины Облачные технологии и бэкэндразработка для ПАО «Сбербанк»

Кейс 1: Разработка системы онлайн-мониторинга клиентских обращений

Задача: Разработать внутреннюю систему для агрегации и анализа в реальном времени всех обращений клиентов-юрлиц из разных каналов (почта, мессенджеры, телефон) для оперативного выявления инцидентов.

Проблема: Разрозненность информации приводит к медленному реагированию на проблемы клиентов. Необходимо в режиме, близком к реальному времени, выявлять типовые проблемы (например, "не работает функция в приложении") и оценивать настроение клиентов.

Техническое задание для студентов:

1. **Проектирование архитектуры:** Предложить микросервисную архитектуру, которая будет интегрироваться с телефонией (Avaya), системой транскрибации (SmartSpeech) и шиной данных (Kafka).
2. **Реализация бэкенда:** Разработать на Go сервис, который будет потреблять текстовые транскрипты из Kafka, сохранять их в PostgreSQL и рассчитывать метрики для "облака тегов".
3. **Работа с данными:** Реализовать алгоритм формирования "облака тегов", где размер слова зависит от частоты употребления. Добавить "светофор" — индикатор, который меняет цвет (зеленый/желтый/красный) при росте частоты упоминания проблемных слов.
4. **Обеспечение производительности:** Система должна обрабатывать тысячи диалогов одновременно с минимальной задержкой.

Кейс 2: Создание модуля транскрибации и анализа телефонных переговоров

Задача: Организовать процесс преобразования аудио звонков в текст в реальном времени и его последующей обработки для службы мониторинга.

Проблема: Аудиозаписи звонков не приносят пользы, пока их не проанализирует человек. Необходимо автоматизировать преобразование речи в текст и выделение ключевой информации.

Техническое задание для студентов:

1. **Проектирование потока данных:** Спроектировать и реализовать на Go конвейер обработки данных, который принимает RTP-аудиопоток от Avaya, передает его в транскрибатор (условный SmartSpeech), получает текстовые реплики и сохраняет их в БД.
2. **Обработка в реальном времени:** Обеспечить минимальную задержку между окончанием фразы оператором/клиентом и появлением текста в системе. Использовать многопоточность и буферизацию для эффективной работы.
3. **Интеграция с AI-моделями:** Разработать API для передачи транскрибированного текста в Python-модули с NLP-моделями для анализа тональности и извлечения сущностей (например, упомянутые суммы, названия продуктов).

Кейс 3: Миграция высоконагруженной системы в облако и импортозамещение СУБД

Задача: Перенести систему, работающую с большими данными, с зарубежных стека (Spark, Hadoop) на российское ПО, используя один инстанс высокопроизводительной СУБД PostgreSQL (Sber Edition).

Проблема: Зависимость от иностранных технологических стеков и высокая стоимость лицензий. Необходимо доказать, что отечественное решение справляется с нагрузками уровня Сбера.

Техническое задание для студентов:

1. **Сравнительный анализ:** Изучить отличия в работе Hadoop/Spark и PostgreSQL при обработке больших данных. Выявить потенциальные "узкие места".

2. **Проектирование схемы БД:** Оптимизировать схему базы данных и написать эффективные SQL-запросы для замены типовых операций, которые раньше делались в Spark (агрегации, фильтрация, соединения).

3. **Реализация ETL-процесса:** Написать на Go сервис, который имитирует загрузку и обработку данных из нескольких источников (например, из Kafka и файловых логов) в PostgreSQL.

4. **Тестирование под нагрузкой:** Провести нагрузочное тестирование, чтобы определить максимальную пропускную способность решения и точки отказа.

Кейс 4: Внедрение low-code платформы для ускорения бэкенд-разработки

Задача: Освоить и применить low-code платформу Platform V DataSpace для быстрого прототипирования и создания микросервиса в рамках одного из внутренних проектов.

Проблема: Длительный цикл разработки бэкенд-сервисов замедляет вывод новых продуктов на рынок.

Техническое задание для студентов:

1. **Изучение платформы:** Разобраться с принципами предметно-ориентированного проектирования (Domain-Driven Design) в контексте Platform V.

2. **Создание визуальной модели:** Используя визуальный конструктор Platform V, спроектировать модель данных для нового сервиса (например, сервис управления заявками на кредитование).

3. **Генерация и доработка микросервиса:** Сгенерировать бэкенд-приложение и доработать его на Go, интегрировав специфическую бизнес-логику, которую невозможно описать в low-коде.

4. **Настройка API:** Настроить GraphQL API для созданного сервиса и реализовать политики разграничения прав доступа.

Кейс 5: Оптимизация конкурентности в высоконагруженном финансовом сервисе

Задача: Исправить проблемы с параллелизмом и избежать дедлоков в системе, обрабатывающей одновременные операции по группе счетов.

Проблема: В оригинальной задаче с Joker 2018 "Вилларибо и Виллабаджо" ненадежный банк блокировал счета в произвольном порядке, что приводило к дедлокам и увеличению времени обработки операций в 10 раз.

Техническое задание для студентов:

1. **Анализ алгоритма блокировок:** Проанализировать, при каких условиях возникает дедлок при обработке 12 операций над 10 счетами, где каждая операция затрагивает до 3 счетов.

2. **Разработка алгоритма без дедлоков:** Реализовать на Go алгоритм, аналогичный "надежному банку", который всегда блокирует ключи по возрастанию их ID, чтобы исключить возможность взаимной блокировки.

3. **Сравнение эффективности:** Сравнить производительность "ненадежного" (с возможностью дедлоков) и "надежного" (с сортировкой блокировок) подходов, оценив, в каких условиях каждый из них выигрывает.

4. **Реализация с использованием примитивов Go:** Использовать горутины, каналы и мьютексы из стандартной библиотеки Go для безопасного параллельного выполнения операций.

Б. Кейсы дисциплины Облачные технологии и бэкэндразработка для компании AVA Lab

Кейс 1: Микросервисная архитектура для управления продажами недвижимости

Бизнес-задача: Существующая монолитная система CRM не справляется с нагрузкой в периоды горячих продаж. Клиенты жалуются на зависания при бронировании квартир.

Техническое задание:

Разработать микросервисы на Go:

- booking-service — бронирование квартир
- payment-service — интеграция с банками
- notification-service — SMS/email уведомления

Реализовать межсервисную коммуникацию через gRPC

Настроить кеширование в Redis для каталога квартир

Обеспечить идемпотентность операций бронирования

go

```go
// Пример структуры сервиса бронирования
type BookingService struct {
    redis *redis.Client
    db    *sql.DB
}

func (s *BookingService) ReserveApartment(ctx context.Context, req *ReserveRequest) (*ReserveResponse, error) {
    // Проверка доступности через Redis
    // Блокировка квартиры на 15 минут
    // Создание заявки в PostgreSQL
    // Отправка события в Kafka для нотификаций
}
```

Кейс 2: Real-time дашборд аналитики для отдела продаж

Бизнес-задача: Менеджеры не видят актуальную информацию о продажах. Отчеты формируются раз в сутки, что мешает оперативному принятию решений.

Техническое задание:

Реализовать сбор метрик в Prometheus

Настроить потоковую обработку данных через Kafka

Создать WebSocket сервер на Go для real-time обновлений

Разработать API для агрегации данных

go

```go
// Metrics collector
type MetricsCollector struct {
    prometheus.Counter
    kafkaProducer *kafka.Writer
}

func (m *MetricsCollector) RecordSale(sale *Sale) {
    m.Increment()
    m.kafkaProducer.WriteMessages(ctx, kafka.Message{
        Key:   []byte("sale"),
        Value: serializeSale(sale),
    })
}
```

Кейс 3: Система управления документацией и согласований

Бизнес-задача: Процесс согласования ДДУ и других документов занимает недели из-за бумажного документооборота.

Техническое задание:
Разработать workflow engine на Go
Интегрировать с электронной подписью
Реализовать WebHook для уведомлений
Настроить версионность документов

```go
type DocumentWorkflow struct {
    steps []WorkflowStep
    currentStep int
}

func (w *DocumentWorkflow) ProcessStep(approver string, decision bool) error {
    // Логика перехода между этапами
    // Валидация прав подписанта
    // Обновление статуса документа
}
```

Кейс 4: Единый customer portal для клиентов

Бизнес-задача: Клиенты не имеют единой точки входа для отслеживания статуса строительства, платежей и документов.

Техническое задание:
Создать API Gateway на Go
Реализовать OAuth2 аутентификацию
Интегрировать с 1С для данных о платежах
Разработать систему ролевого доступа

```go
// API Gateway routing
func SetupRoutes(r *mux.Router) {
    r.HandleFunc("/api/payments", authMiddleware(paymentHandler))
    r.HandleFunc("/api/construction-progress", authMiddleware(progressHandler))
    r.HandleFunc("/api/documents", authMiddleware(documentsHandler))
}
```

Кейс 5: IoT платформа для мониторинга хода строительства

Бизнес-задача: Отсутствует автоматизированный контроль сроков строительства. Данные с объектов собираются вручную.

Техническое задание:
Разработать MQTT брокер для IoT устройств
Реализовать обработку потоковых данных с камер и датчиков
Создать систему алертинга при отклонениях от графика
Настроить long-term хранение телеметрии

```go
// IoT Data Processor
type IoTProcessor struct {
    mqttClient *mqtt.Client
    timescaleDB *sql.DB
}

func (p *IoTProcessor) HandleSensorData(topic string, payload []byte) {
    // Парсинг данных с датчиков
    // Агрегация метрик за последний час
    // Проверка на аномалии
    // Сохранение в TimescaleDB для аналитики
}
```

7. Материально-техническое обеспечение по дисциплине (модулю))

Наименование специальных помещений	Оснащенность специальных помещений	Перечень лицензионного программного обеспечения
Учебные аудитории для проведения занятий лекционного типа	Мебель: учебная мебель Технические средства обучения: экран, проектор, компьютер ауд. 129, 131, А-305, А-307	MS Office Word 2016 и выше Ms Power Point 2016 и выше
Учебные аудитории для проведения текущего контроля (Ауд. 101, 102, 105/1, 106 и 106а)	Мебель: учебная мебель Технические средства обучения: Экран, компьютер Оборудование: компьютерная техника с подключением к информационно-коммуникационной сети «Интернет» и доступом в электронную информационно-образовательную среду образовательной организации	Браузер Google Chrome, Jupyter Notebook 6.3.0 и выше (язык Python с библиотеками Numpy, Pandas, gensim, NLTK, PyMorphy, фреймворком PyTorch)
Учебные аудитории для проведения промежуточной аттестации (Ауд. 129, 131, А-305, А-307)	Мебель: учебная мебель	-
Учебные аудитории для проведения лабораторных работ (Ауд. 101, 102, 105/1, 106 и 106а)	Мебель: учебная мебель Технические средства обучения: экран, компьютер Оборудование: компьютерная техника с подключением к информационно-коммуникационной сети «Интернет» и доступом в электронную информационно-образовательную среду образовательной организации	Браузер Google Chrome, Jupyter Notebook 6.3.0 и выше (язык Python с библиотеками Numpy, Pandas, gensim, NLTK, PyMorphy, фреймворком PyTorch)

Для самостоятельной работы обучающихся предусмотрены помещения, укомплектованные специализированной мебелью, оснащенные компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду университета.

Наименование помещений для самостоятельной работы обучающихся	Оснащенность помещений для самостоятельной работы обучающихся	Перечень лицензионного программного обеспечения
Помещение для самостоятельной работы обучающихся (читальный зал Научной библиотеки)	Мебель: учебная мебель Комплект специализированной мебели: компьютерные столы Оборудование: компьютерная техника с подключением к информационно-коммуникационной сети «Интернет» и доступом в электронную информационно-образовательную среду образовательной организации, веб-камеры, коммуникационное оборудование, обеспечивающее доступ к сети интернет (проводное соединение и беспроводное соединение по технологии Wi-Fi)	MS Office Word 2016 и выше Ms Power Point 2016 и выше
Помещение для самостоятельной работы обучающихся (Ауд. 101, 102, 103, 105/1, 106 и 106а)	Мебель: учебная мебель Комплект специализированной мебели: компьютерные столы Оборудование: компьютерная техника с подключением к информационно-	Браузер Google Chrome, Matlab, Jupyter Notebook 6.3.0 и выше (язык Python с библиотеками Numpy, Pandas, gensim,

	коммуникационной сети «Интернет» и доступом в электронную информационно-образовательную среду образовательной организации, веб-камеры, коммуникационное оборудование, обеспечивающее доступ к сети интернет (проводное соединение и беспроводное соединение по технологии Wi-Fi)	NLTK, PyMorphy, фреймворком PyTorch)

№	Продукт	Параметры продукта	Кол-во	Кол-во конфигураций	Ед. изм.
1	Виртуальная машина	Виртуальная машина 10% vCPU 2 vCPU 4 RAM	1	60	Шт
		ОС Ubuntu 22.04	1		Шт
		Системный диск SSD	1		Шт
			10		Гб
		Аренда публичного IP	1		Шт
2	Виртуальная машина с GPU	Виртуальная машина с GPU NVIDIA® Tesla® V100 2 GPU 8 vCPU 128 ГБ RAM	1	1	Шт
		ОС Ubuntu_24.04	1		Шт
		Системный диск SSD	1		Шт
			2000		Гб
		Диск SSD	1		Шт
			4096		Гб
		Диск SSD	1		Шт
			4096		Гб
		Аренда публичного IP	1		Шт
3	K8S	Master node 8 vCPU 16 RAM	1	1	Шт
		Worker node 10% доля 4 vCPU 32 RAM	5		Шт
		Worker node SSD-NVME	64		Гб
		Аренда публичного IP	1		Шт
4	ML Inference Instance Type GPU	Время работы в месяц	40	1	Ч
		Инстанс 8 x NVIDIA® H100 NVLink PCIe 160 vCPU 1520 GB RAM	1		Шт
		Количество запросов к ML-моделям	1		Млн. Шт
		Кэш ML-моделей	160		Гб
5	LLM	Токены GigaChat 2 Max	50		Млн. Шт
		Токены Embeddings	400		Млн. Шт