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. В.ДВ.04.01 Коллективная разработка информационных систем

Направление подготовки 02.03.02 Фундаментальная информатика и информационные технологии

Профиль Современные методы машинного обучения и компьютерного зрения

Форма обучения очная

Квалификация бакалавр

Краснодар 2025

Рабочая программа дисциплины «Коллективная разработка информационных систем» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии.

Программу составил(и):
Добровольская Н.Ю. доцент, канд. пед. наук, доцент _____
подпись

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

Заведующий кафедрой
Подколзин В.В. _____
подпись

Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики
протокол № 1 «28» августа 2025г.

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

Формирование у студентов системных знаний, практических умений и навыков в области организации и управления процессами коллективной разработки сложных информационных систем с использованием современных методологий, инструментов и практик индустрии (DevOps, CI/CD, Agile).

1.2 Задачи дисциплины

Изучить принципы и методологии коллективной разработки (Agile, Scrum, Kanban) и их применение в создании ИС.

Освоить инструменты контроля версий (Git) и платформы для командной работы (GitLab, GitHub).

Сформировать навыки проектирования и управления требованиями к информационной системе.

Научиться проектировать, настраивать и использовать пайплайны непрерывной интеграции и доставки (CI/CD).

Освоить основы контейнеризации (Docker) и оркестрации для обеспечения переносимости и масштабируемости ИС.

Изучить принципы управления проектами, ведения технической документации и проведения код-ревью.

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Коллективная разработка информационных систем» относится к «Часть, формируемая участниками образовательных отношений» Блока 1 «Дисциплины по выбору Б1.В.ДВ.4» учебного плана.

1.4 Профессиональные роли в структуре образовательной программы

Роль 1: Data Engineer (Инженер по данным)
Задачи:

1. Проектирование и построение ETL-процессов
2. Создание и оптимизация хранилищ данных
3. Обеспечение качества и доступности данных
4. Настройка инфраструктуры для обработки больших данных
5. Интеграция разрозненных источников данных
6. Работа с данными в области природопользования, медицины, связи и телекоммуникаций

Роль 2: ML Engineer (Инженер МО)
Задачи:

1. Реализация ML-моделей в продуктивных системах
2. Оптимизация производительности и масштабирование моделей
3. Разработка ML-пайплайнов и автоматизация процессов
4. Мониторинг качества моделей в продуктиве
5. Интеграция ML-решений с бизнес-приложениями

Роль 3: MLOps (Специалист по эксплуатации ИИ)
Задачи:

6. Автоматизация процессов обучения и развертывания моделей
7. Мониторинг производительности ML-систем
8. Управление версиями моделей и данных
9. Обеспечение CI/CD для ML-проектов
10. Оптимизация вычислительных ресурсов

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

Изучение данной учебной дисциплины направлено на формирование у обучающихся следующих компетенций:

УК-1	***Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач***
УК-1.2	Выбирает оптимальный вариант решения задачи, аргументируя свой выбор
	Знать: Критерии сравнения альтернативных решений. Уметь: Проводить сравнительный анализ вариантов. Владеть: Навыками аргументации и принятия решений.
УК-3	***Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде***
УК-3.2	Применяет методы командного взаимодействия; планирует и организует командную работу
	Знать: Методы и инструменты организации командной работы. Уметь: Распределять задачи и контролировать их выполнение. Владеть: Техниками командной координации и планирования.
УК-5	***Способен воспринимать межкультурное разнообразие общества в социально-историческом, этическом и философском контекстах***
УК-5.2	Интерпретирует проблемы современности с позиции этики и философских знаний
	Знать: Современные социально-этические проблемы. Уметь: Применять этические frameworks для анализа. Владеть: Навыками этической оценки технологий.
ПК-3	***Способен применять основные алгоритмические и программные решения в области информационно-коммуникационных технологий, а также участвовать в их разработке***
ПК-3.2	Определяет элементы проблемной области и их взаимодействие, архитектуру программной системы, ее функциональные возможности и логику работы с использованием основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования
	Знать: Принципы проектирования архитектуры ПО. Уметь: Проектировать компоненты системы и их взаимодействие. Владеть: Навыками проектирования программной архитектуры.
ПК-4	***Способен планировать необходимые ресурсы и этапы выполнения работ в области информационно-коммуникационных технологий, составлять соответствующие технические описания и инструкции***
ПК-4.2	Применяет современные приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов на базе языков программирования, баз данных и пакетов прикладных программ
	Знать: Приемы работы с инструментальными средствами. Уметь: Использовать инструменты для создания программных комплексов. Владеть: Техниками интеграции различных инструментов.
ПК-4.3	Способен использовать методы эффективного управления командой при разработке, внедрении и сопровождении программных продуктов
	Знать: Методы управления командой в IT-проектах. Уметь: Организовывать работу команды на разных этапах. Владеть: Практиками управления командой разработки.

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
	Знать: Принципы эффективной коммуникации в проектах. Уметь: Ясно излагать идеи и задачи команде. Владеть: Навыками проведения эффективных встреч.
SS-2.2	Учитывает профессиональные и ролевые особенности коллег при совместной разработке технических решений и представлении результатов
	Знать: Особенности ролей в проектной команде по ИИ. Уметь: Адаптировать стиль коммуникации под аудиторию. Владеть: Методами междисциплинарного взаимодействия.
SS-3	***Способен осуществлять свою трудовую функцию с учетом неопределенности как сущностной черты функционирования искусственного интеллекта***
SS-3.2	Определяет релевантность применения ИИ для решения конкретных задач, анализирует поведение ИИ в техническом, социальном и правовом контекстах, переносит идеи и методы за пределы исходной предметной области
	Знать: Критерии применимости ИИ для решения задач. Уметь: Анализировать поведение ИИ в различных контекстах. Владеть: Методами анализа границ применимости моделей.
LC-1	***Способен проводить анализ бизнес-проблем с оценкой перспективности применения ИИ для их решения, осуществлять постановку задачи машинного обучения, формулировать требования к системе ИИ***
LC-1.1	Оценивает технические требования и разрабатывает техническое задание на системы искусственного интеллекта в конкретной предметной области
	Знать: Структуру и содержание ТЗ на систему ИИ. Уметь: Формализовывать бизнес-требования в технические. Определяет и формализует проблему предметной области, решение которой требует применения искусственного интеллекта Владеть: Методами разработки технического задания.
LC-4.1	***Способен управлять процессом жизненного цикла ИИ-продукта***
LC-4.1.1	Осуществляет запуск и ведение проекта в области ИИ, в том числе планирование и контроль задач, оценку ресурсов
	Знать: Уметь: Осуществляет ведение (запуск и управление) проектов в области ИИ, в том числе подбор команды, планирование и контроль задач, оценка ресурсов Владеть:
LC-4.2	***Способен руководить работой команды проекта в области ИИ***
LC-4.2.1	Координирует и контролирует работу команд проекта с целью достижения общих целей проекта
	Знать: Методы и инструменты координации командной работы в проектах. Уметь: Обеспечивает взаимодействие всех участников проекта, включая бизнес-лидеров, команду Data Science и разработчиков ПО Владеть: Техниками контроля сроков и качества выполнения работ для достижения целей проекта

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 2 зач. ед. (72 часов), их распределение по видам работ представлено в таблице

Вид учебной работы		Всего часов	Семестры (часы)					
			8					
Контактная работа, в том числе:		**30,2**	**30,2**					
Аудиторные занятия (всего):		**28**	**28**					
Занятия лекционного типа		**14**	14					
Лабораторные занятия		**14**	14					
Занятия семинарского типа (семинары, практические занятия)								
Иная контактная работа:		**2,2**	**2,2**					
Контроль самостоятельной работы (КСР)		**2**	2					
Промежуточная аттестация (ИКР)		**0,2**	0,2					
Самостоятельная работа, в том числе:		**41,8**	**41,8**					
Курсовая работа								
Проработка учебного (теоретического) материала		**20**	20					
Выполнение индивидуальных заданий (подготовка сообщений, презентаций)		**21,8**	21,8					
Реферат								
Подготовка к текущему контролю								
Контроль:								
Подготовка к экзамену								
Общая трудоемкость	**час.**	**72**	**72**					
	в том числе контактная работа	**30,2**	**30,2**					
	зач. ед	**2**	**2**					

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.

Разделы (темы) дисциплины, изучаемые в 8 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 коллективную разработку.	10	2		2	6
2.	Управление требованиями и проектирование ИС.	10	2		2	6
3.	Инструменты контроля версий и командной работы.	10	2		2	6
4.	Практики разработки и обеспечение качества.	10	2		2	6
5.	Непрерывная интеграция и доставка (CI/CD).	10	2		2	6
6.	Контейнеризация и оркестрация.	10	2		2	6
7.	Управление проектом и итоговая аттестация.	9,8	2		2	5,8
ИТОГО по разделам дисциплины		**69,8**	**14**		**14**	**41,8**

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
Контроль самостоятельной работы (КСР)		2				
Промежуточная аттестация (ИКР)		0,2				
Подготовка к текущему контролю						
Общая трудоемкость по дисциплине		**72**				

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 коллективную разработку.	Понятие информационной системы (ИС). Жизненный цикл ИС. Особенности и проблемы коллективной разработки. Обзор современных методологий: Waterfall, Agile (Scrum, Kanban).	Т, ЛР
2.	Управление требованиями и проектирование ИС.	Функциональные и нефункциональные требования. Техники сбора требований. Use Case диаграммы, пользовательские истории. Проектирование архитектуры ИС (микросервисы vs монолит).	Т, ЛР
3.	Инструменты контроля версий и командной работы.	Системы контроля версий (Git). Ветвление (GitFlow, GitHub Flow). Платформы GitLab/GitHub: Issues, Merge Requests, Project Boards.	Т, ЛР
4.	Практики разработки и обеспечение качества.	Code Review: цели, процесс и лучшие практики. Стандарты кодирования. Понятие технического долга. Автоматизация статического анализа кода (линтеры).	Т, ЛР
5.	Непрерывная интеграция и доставка (CI/CD).	Принципы CI/CD. Проектирование пайплайнов в GitLab CI/CD. Автоматизация сборки, тестирования и развертывания.	Т, ЛР
6.	Контейнеризация и оркестрация.	Введение в Docker: образы, контейнеры, Dockerfile. Docker Compose для управления мультиконтейнерными приложениями. Основы оркестрации (Kubernetes).	Т, ЛР
7.	Управление проектом и итоговая аттестация.	Роли в команде (Team Lead, DevOps, разработчик). Инструменты управления проектами (Jira, Notion). Подведение итогов, защита проектов.	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.

2.3.2 Занятия семинарского типа

Не предусмотрены

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.

2.3.3 Лабораторные занятия

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
1.	Введение в коллективную разработку.	ЛР1. Организация командного репозитория. Создание репозитория на GitLab/GitHub. Освоение базовых команд Git (commit, push, pull). Работа с ветками, создание Merge/Pull Request, разрешение конфликтов.	Т, ЛР
2.	Управление требованиями и проектирование ИС.	ЛР2. Настройка проекта и планирование спринта. Создание проекта на GitLab/GitHub/Jira. Формирование бэклога продукта и спринта на основе пользовательских историй.	Т, ЛР
3.	Инструменты контроля версий и командной работы.	ЛР3. Настройка и проведение код-ревью. Настройка линтера (например, ESLint, Pylint). Создание Merge Request и проведение взаимного код-ревью по заданному чек-листу.	Т, ЛР
4.	Практики разработки и обеспечение качества.	ЛР4. Создание базового CI-пайплайна. Написание .gitlab-ci.yml для автоматической сборки и запуска юнит-тестов проекта при каждом пуше в репозиторий.	Т, ЛР
5.	Непрерывная интеграция и доставка (CI/CD).	ЛР5. Создание Docker-образа для компонента ИС. Написание Dockerfile для backend/frontend-приложения. Сборка образа и запуск контейнера.	Т, ЛР
6.	Контейнеризация и оркестрация.	ЛР6. Развертывание в тестовом окружении. Модификация пайплайна для автоматического развертывания Docker-контейнера на тестовом сервере (например, используя GitLab CI и SSH).	ЛР
7.	Управление проектом и итоговая аттестация.	ЛР7. Сквозная задача: реализация пайплайна CI/CD для микросервиса. Интеграция всех пройденных этапов: от кода до развертывания, включая тестирование и контейнеризацию.	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 предусмотрены

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	2	3
1	Изучение теоретического материала	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019
2	Решение задач	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019

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

В соответствии с требованиями ФГОС в программа дисциплины предусматривает использование в учебном процессе следующих образовательные технологии: чтение лекций с использованием мультимедийных технологий; метод малых групп, разбор практических задач и кейсов.

При обучении используются следующие образовательные технологии:

– Технология коммуникативного обучения – направлена на формирование коммуникативной компетентности студентов, которая является базовой, необходимой для адаптации к современным условиям межкультурной коммуникации.

– Технология разноуровневого (дифференцированного) обучения – предполагает осуществление познавательной деятельности студентов с учётом их индивидуальных способностей, возможностей и интересов, поощряя их реализовывать свой творческий потенциал. Создание и использование диагностических тестов является неотъемлемой частью данной технологии.

– Технология модульного обучения – предусматривает деление содержания дисциплины на достаточно автономные разделы (модули), интегрированные в общий курс.

– 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.

– Технология индивидуализации обучения – помогает реализовывать личностно-ориентированный подход, учитывая индивидуальные особенности и потребности учащихся.

– Проектная технология – ориентирована на моделирование социального взаимодействия учащихся с целью решения задачи, которая определяется в рамках профессиональной подготовки, выделяя ту или иную предметную область.

– Технология обучения в сотрудничестве – реализует идею взаимного обучения, осуществляя как индивидуальную, так и коллективную ответственность за решение учебных задач.

– Игровая технология – позволяет развивать навыки рассмотрения ряда возможных способов решения проблем, активизируя мышление студентов и раскрывая личностный потенциал каждого учащегося.

– Технология развития критического мышления – способствует формированию разносторонней личности, способной критически относиться к информации, умению отбирать информацию для решения поставленной задачи.

Комплексное использование в учебном процессе всех вышеназванных технологий стимулируют личностную, интеллектуальную активность, развивают познавательные процессы, способствуют формированию компетенций, которыми должен обладать будущий специалист.

Основные виды интерактивных образовательных технологий включают в себя:

– работа в малых группах (команде) - совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путём творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности;

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

Семестр	**Вид занятия**	**Используемые интерактивные образовательные технологии**	**количество интерактивных часов**
	ЛР	Практические занятия в режимах взаимодействия «преподаватель – студент» и «студент – студент»	12

Семестр	Вид занятия	Используемые интерактивные образовательные технологии	количество интерактивных часов
Итого			12

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

Темы, задания и вопросы для самостоятельной работы призваны сформировать навыки поиска информации, умения самостоятельно расширять и углублять знания, полученные в ходе лекционных и практических занятий.

Подход разбора конкретных ситуаций широко используется как преподавателем, так и студентами при проведении анализа результатов самостоятельной работы.

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

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.

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

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

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 «Коллективная разработка информационных систем».

Оценочные средства включает контрольные материалы для проведения **текущего контроля** в форме тестовых заданий, кейсов и **промежуточной аттестации** в форме вопросов и заданий к **зачету**.

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

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
1	Введение в коллективную разработку.	УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1	*Лабораторная работа №1*	*Вопросы к зачету 1-3*
2	Управление требованиями и проектирование ИС.	УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1	*Лабораторная работа №2*	*Вопросы к зачету 4-7, проектное задание*
3	Инструменты контроля версий и командной работы.	УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1	*Лабораторная работа №3*	*Вопросы к зачету 8-10, проектное задание*
4	Практики разработки и обеспечение качества.	УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1	*Лабораторная работа №4*	*Вопросы к зачету 11-13, проектное задание*
5	Непрерывная интеграция и доставка (CI/CD).	УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3;	*Лабораторная работа №5*	*Вопросы к зачету 14-17, проектное задание*

		SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1		
6	Контейнеризация и оркестрация.	УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1	*Лабораторная работа №6*	*Вопросы к зачету 18-20, проектное задание*
7	Управление проектом и итоговая аттестация.	УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1	*Лабораторная работа №7*	*Вопросы к зачету 21-25, проектное задание*

Показатели, критерии и шкала оценки сформированных компетенций

Соответствие **пороговому уровню** освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **удовлетворительно /зачтено**):

УК-1	***Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач***
УК-1.2	Выбирает оптимальный вариант решения задачи, аргументируя свой выбор
	Знать: Критерии сравнения альтернативных архитектурных и технологических решений. Уметь: Проводить сравнительный анализ вариантов реализации компонентов ИС. Владеть: Навыками аргументации технических решений в команде.
УК-3	***Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде***
УК-3.2	Применяет методы командного взаимодействия; планирует и организует командную работу
	Знать: Методы и инструменты организации командной работы в процессе разработки ИС. Уметь: Распределять разработческие задачи и контролировать их выполнение. Владеть: Техниками командной координации и планирования в гибкой методологии.
УК-5	***Способен воспринимать межкультурное разнообразие общества в социально-историческом, этическом и философском контекстах***
УК-5.2	Интерпретирует проблемы современности с позиции этики и философских знаний
	Знать: Современные социально-этические проблемы в сфере информационных технологий.

	Уметь: Применять этические frameworks для анализа процесса разработки и результатов. Владеть: Навыками этической оценки создаваемых информационных систем.
ПК-3	***Способен применять основные алгоритмические и программные решения в области информационно-коммуникационных технологий, а также участвовать в их разработке***
ПК-3.2	Определяет элементы проблемной области и их взаимодействие, архитектуру программной системы, ее функциональные возможности и логику работы с использованием основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования
	Знать: Принципы проектирования архитектуры ПО для масштабируемых ИС. Уметь: Проектировать компоненты системы и их взаимодействие в командном проекте. Владеть: Навыками проектирования программной архитектуры с учетом разделения ответственности.
ПК-4	***Способен планировать необходимые ресурсы и этапы выполнения работ в области информационно-коммуникационных технологий, составлять соответствующие технические описания и инструкции***
ПК-4.2	Применяет современные приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов на базе языков программирования, баз данных и пакетов прикладных программ
	Знать: Приемы работы с инструментальными средствами в процессе командной разработки. Уметь: Использовать инструменты для создания и интеграции программных комплексов. Владеть: Техниками интеграции различных инструментов в единый контур разработки.
ПК-4.3	Способен использовать методы эффективного управления командой при разработке, внедрении и сопровождении программных продуктов
	Знать: Методы управления командой разработчиков в IT-проектах. Уметь: Организовывать работу команды на разных этапах жизненного цикла ИС. Владеть: Практиками управления распределенной командой разработки.
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
	Знать: Принципы эффективной коммуникации в проектах по разработке ИС. Уметь: Ясно излагать технические идеи и задачи команде. Владеть: Навыками проведения эффективных проектных встреч.
SS-2.2	Учитывает профессиональные и ролевые особенности коллег при совместной разработке технических решений и представлении результатов
	Знать: Особенности ролей в проектной команде по разработке ИС. Уметь: Адаптировать стиль коммуникации под техническую и нетехническую аудиторию. Владеть: Методами междисциплинарного взаимодействия в проекте ИС.

SS-3	***Способен осуществлять свою трудовую функцию с учетом неопределенности как сущностной черты функционирования искусственного интеллекта***
SS-3.2	Определяет релевантность применения ИИ для решения конкретных задач, анализирует поведение ИИ в техническом, социальном и правовом контекстах, переносит идеи и методы за пределы исходной предметной области
	Знать: Критерии применимости ИИ для решения задач в контексте разрабатываемой ИС. Уметь: Анализировать поведение ИИ как компонента ИС в различных контекстах. Владеть: Методами анализа границ применимости моделей в проектируемой системе.
LC-1	***Способен проводить анализ бизнес-проблем с оценкой перспективности применения ИИ для их решения, осуществлять постановку задачи машинного обучения, формулировать требования к системе ИИ***
LC-1.1	Оценивает технические требования и разрабатывает техническое задание на системы искусственного интеллекта в конкретной предметной области
	Знать: Структуру и содержание ТЗ на систему ИИ, включая разделы по взаимодействию с другими компонентами и командой. Уметь: Формализовывать бизнес-требования в технические спецификации, понятные для всех участников команды разработки. Владеть: Методами совместной разработки и согласования технического задания.
LC-4.1	***Способен управлять процессом жизненного цикла ИИ-продукта***
LC-4.1.1	Осуществляет запуск и ведение проекта в области ИИ, в том числе планирование и контроль задач, оценку ресурсов
	Знать: Основы проектного управления в области ИИ и инструменты для коллективного планирования. Уметь: Осуществлять ведение (запуск и управление) проектов в области ИИ, в том числе подбор команды, планирование и контроль задач, оценка ресурсов. Владеть: Инструментами планирования, контроля и координации командной работы в проекте.
LC-4.2	***Способен руководить работой команды проекта в области ИИ***
LC-4.2.1	Координирует и контролирует работу команд проекта с целью достижения общих целей проекта
	Знать: Методы и инструменты координации командной работы в проектах, включая практики Agile. Уметь: Обеспечивать взаимодействие всех участников проекта, включая бизнес-лидеров, команду Data Science и разработчиков ПО. Владеть: Техниками контроля сроков, качества выполнения работ и синхронизации усилий команды.

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

1. Тесты

Тема: "Инструменты контроля версий" (Git)

Вопрос 1: Какой командой в Git можно просмотреть историю коммитов с визуализацией ветвления?
a) git log --oneline
b) git status
c) git log --oneline --graph --all
d) git history
Вопрос 2: Что делает команда git revert <commit-hash>?
a) Удаляет коммит из истории, перезаписывая все последующие коммиты.
b) Создает новый коммит, который отменяет изменения, внесенные указанным коммитом.
c) Возвращает рабочий каталог к состоянию указанного коммита.
d) Отменяет все незакоммиченные изменения в рабочем каталоге.
Вопрос 3: В чем основное различие между стратегиями ветвления GitFlow и GitHub Flow?
a) GitFlow использует одну основную ветку, а GitHub Flow — несколько.
b) GitFlow имеет отдельные ветки для разработки, релизов и хотфиксов, в то время как GitHub Flow ориентирован на непрерывную доставку с одной основной веткой и короткоживущими feature-ветками.
c) GitHub Flow запрещает использование Merge Requests.
d) GitFlow не поддерживает теги.
Вопрос 4: При выполнении git merge feature-branch в main возник конфликт в файле app.js. Каковы ваши дальнейшие шаги?
a) Удалить файл app.js и выполнить git merge --continue.
b) Вручную отредактировать файл app.js**, разрешив конфликтные участки (отмечены** <<<<<<<, =======, >>>>>>>**), затем выполнить** git add app.js **и** git commit**.**
c) Выполнить git abort для отмены слияния.
d) Выполнить git push --force.

Тема: "Практики разработки" (Code Review, Качество кода)
Вопрос 1: Какая из перечисленных практик является примером *плохого* код-ревью?
a) Проверка не только функциональности, но и читаемости кода.
b) Комментарий "Этот код ужасен, переделай всё" без конкретных указаний на проблемы.
c) Предложение альтернативных решений с обоснованием.
d) Проверка на соответствие стандартам кодирования проекта.
Вопрос 2: Что такое "Технический долг" (Technical Debt)?
a) Накопленные в кодовой базе проблемы (например, неоптимальные решения, "костыли"), решение которых отложено на будущее, что усложняет дальнейшую разработку.
b) Деньги, которые компания должна заплатить за использование открытого ПО.
c) Время, которое разработчик должен отработать сверхурочно.
d) Стоимость облачной инфраструктуры для проекта.
Вопрос 3: Для чего используются инструменты статического анализа кода (линтеры), такие как Pylint или ESLint?
a) Для автоматического развертывания приложения.
b) Для автоматического выявления потенциальных ошибок, нарушения стиля кодирования и "запахов кода" (code smells) до запуска программы.
c) Для проведения нагрузочного тестирования.
d) Для управления виртуальными окружениями.
Вопрос 4: Какое из утверждений лучше всего описывает цель пользовательской истории (User Story) в Agile?
a) Это техническое задание для разработчика с полным перечнем функций.
b) Это краткое описание функции, рассказанное с точки зрения пользователя,

которая отвечает на вопросы "Кто?", "Что?" и "Зачем?".
c) Это диаграмма последовательности действий в системе.
d) Это протокол совещания с заказчиком.

2. Перечень индивидуальных проектов (примеры)

Проект 1: "Система управления задачами (Trello-клон)"

- **Описание:** Разработка веб-приложения для управления проектами по методологии Kanban. Основные сущности: доски, колонки (например, "Сделать", "В работе", "Готово"), карточки задач.
- **Технологический стек:** Backend - Python (Django/FastAPI) / Node.js, Frontend - React/Vue.js, База данных - PostgreSQL.
- **Задачи для CI/CD:**
    1. Настроить пайплайн, который запускает юнит-тесты для backend и линтеры для frontend/backend.
    2. Автоматически собирать Docker-образы для frontend и backend.
    3. Развертывать приложение на тестовом сервере с помощью `docker-compose` при мерже кода в ветку `develop`.

Проект 2: "REST API для блога или новостного агрегатора"

- **Описание:** Создание бэкенд-сервиса с API для статей, пользователей, комментариев и тегов. Реализация аутентификации по JWT-токенам.
- **Технологический стек:** Python (FastAPI/Django REST Framework), PostgreSQL, Redis (для кеширования).
- **Задачи для CI/CD:**
    1. Настроить автоматический запуск миграций базы данных при развертывании.
    2. Интегрировать в пайплайн этап сборки и тестирования.
    3. Реализовать деплой на тестовый сервер (например, VPS) с помощью Ansible или через SSH в GitLab CI.

Проект 3: "Микросервис для обработки и валидации данных"

- **Описание:** Разработка отдельного микросервиса, который принимает файл (например, CSV или JSON) через API, проверяет его структуру и данные по заданным правилам и возвращает отчет о ошибках или преобразованные данные.
- **Технологический стек:** Python (Flask/FastAPI), Pydantic (для валидации), Celery (для асинхронной обработки больших файлов).
- **Задачи для CI/CD:**
    1. Создать многоконтейнерное приложение (Docker Compose), включающее сам сервис, брокер сообщений (Redis/RabbitMQ) и воркер Celery.
    2. Настроить пайплайн, который собирает все образы и запускает интеграционные тесты всего стека.
    3. Реализовать деплой на тестовый кластер Kubernetes (например, используя Minikube) или через Docker Compose на сервер.

Проверяемые компетенции комплексом практических заданий: УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1

Зачетно-экзаменационные материалы для промежуточной аттестации (зачет)

Вопросы для подготовки к зачету

Блок A: Методологии и управление проектами

1. Опишите роли и артефакты в методологии Scrum.
2. В чем разница между бэклогом продукта (Product Backlog) и бэклогом спринта (Sprint Backlog)?
3. Какие инструменты для управления проектами вы знаете? Сравните их по функциональности (например, Jira vs Trello).
4. Что такое "минимизация циклов обратной связи" (Feedback Loops) в Agile и почему это важно?
5. Как проводится ретроспектива спринта и какова ее цель?

Блок B: Системы контроля версий (Git)

6. Объясните жизненный цикл файла в Git (untracked, staged, committed).

7. Для чего используются команды git stash и git cherry-pick? Приведите примеры использования.

8. Что такое "submodule" в Git и в каких сценариях его использование оправдано?

9. Чем отличается git fetch от git pull?

10. Как организовать эффективную командную работу с помощью Merge/Pull Requests? Опишите идеальный процесс.

Блок C: Практики разработки и обеспечение качества

11. Каковы ключевые принципы написания "чистого кода" (Clean Code)?

12. Что такое "интеграция по функциям" (Feature Toggles) и как она помогает в непрерывной поставке?

13. Опишите различные типы автоматических тестов (юнит, интеграционные, e2e) и их место в CI/CD-пайплайне.

14. Какую пользу приносит автоматизация сборки и развертывания помимо экономии времени?

Блок D: Непрерывная интеграция и доставка (CI/CD)

15. Из каких основных этапов состоит CI/CD-пайплайн? Дайте краткую характеристику каждому.

16. Что такое "инфраструктура как код" (Infrastructure as Code, IaC)? Какие инструменты вы знаете (Terraform, Ansible)?

17. Как обеспечить безопасность пайплайна (например, хранение секретов, сканирование зависимостей)?

18. Опишите стратегии развертывания (Deployment Strategies): сине-зеленое развертывание, канареечный релиз.

Блок E: Контейнеризация и оркестрация

19. В чем преимущества контейнеризации приложений перед традиционным развертыванием?

20. Что такое Dockerfile и какие лучшие практики написания Dockerfile вы знаете (например, использование многоступенчатой сборки)?

21. Какова основная задача оркестратора, такого как Kubernetes?

22. Какие проблемы решает использование Docker Compose в процессе разработки и тестирования?

Блок F: Командная работа и коммуникация

23. Почему важна хорошая техническая документация и как ее можно поддерживать в актуальном состоянии?

24. Как эффективно проводить онбординг нового разработчика в проект?

25. Каковы ваши обязанности как члена команды при проведении код-ревью (и как автора, и как ревьювера)?

Перечень компетенций (части компетенции), проверяемых оценочным средством
УК-1.2; УК-3.2; УК-5.2; ПК-3.2; ПК-4.2; ПК-4.3; SS-2.1; SS-2.2; SS-3.2; LC-1.1; LC-4.1.1; LC-4.2.1

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

Методические рекомендации, определяющие процедуры оценивания на экзамене:

Процедура промежуточной аттестации проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

Итоговой формой контроля сформированности компетенций у обучающихся по дисциплине является зачет. Студенты обязаны сдать экзамен в соответствии с расписанием и учебным планом.

ФОС промежуточной аттестации состоит из вопросов к зачету и результатов текущего контроля.

Экзамен по дисциплине преследует цель оценить работу студента за курс, получение теоретических знаний, их прочность, развитие творческого мышления, приобретение навыков самостоятельной работы, умение применять полученные знания для решения практических задач.

Форма проведения зачета: устно.

Экзаменатору предоставляется право задавать студентам дополнительные вопросы по всей учебной программе дисциплины.

Результат сдачи зачета заноситься преподавателем в экзаменационную ведомость и зачетную книжку.

Оценивание уровня освоения дисциплины основывается на качестве выполнения студентом заданий текущего контроля и ответов на вопросы зачета.

Критерии оценки:

Для получения оценки «Зачтено» студент должен выполнить все следующие условия:

1. Проектная работа (Минимальный порог — «Удовлетворительно»)

Критерии оценки проекта:

«Не зачтено» выставляется, если:

Проект не сдан или его функциональность не соответствует заявленным минимальным требованиям.

Полностью отсутствует работа в команде (не участвовал в общих обсуждениях, не выполнял свои задачи).

Код проекта не запускается или его основная функция не работает.

Отсутствует обязательная техническая документация (например, README файл с инструкцией по запуску).

«Зачтено» выставляется, если проект:

Реализован и работает: Основная заявленная функциональность реализована, система запускается и выполняет свою основную задачу.

Демонстрирует работу в команде: Использована система контроля версий (Git), и виден вклад студента (коммиты, пул-реквесты). Вклад может быть небольшим, но осмысленным и завершенным.

Имеет минимальную документацию: Есть инструкция по запуску и краткое описание проекта.

Сдан в установленные сроки.

2. Тест (Письменный или компьютерный) (Минимальный порог — 60% правильных ответов)

Критерии:

«Не зачтено»: Менее 60% правильных ответов.

«Зачтено»: 60% и более правильных ответов.

3. Ответы на вопросы к зачету (Минимальный порог — 2/3 полноценных ответа)

Студент отвечает на несколько случайно выбранных вопросов из списка (3 из 25).

Критерии оценки ответа на каждый вопрос:

Полноценный ответ («Зачет» по вопросу): Ответ точный, раскрывает суть, возможно, приведен пример. Студент демонстрирует понимание.

Неполный/Неверный ответ («Незачет» по вопросу): Ответ поверхностный, содержит фактические ошибки или отсутствует.

Итоговый критерий для блока:

«Не зачтено»: Студент не дал полноценных ответов на более чем 1 вопрос (например, если отвечал на 3 вопроса, и 2 из них — слабые).

«Зачтено»: Студент дал полноценные ответы как минимум на 2 из 3 вопросов (т.е. продемонстрировал знание большей части материала).

Итоговая таблица для оценки

Компонент курса	Критерий для «Зачтено»	Критерий для «Не зачтено»
Проект	Сдан, работает, виден командный вклад.	Не сдан, не работает, нет командной работы.
Тест	≥ 60% правильных ответов.	< 60% правильных ответов.
Вопросы на зачете	2 из 3 (или более) ответов — полноценные.	2 из 3 (или более) ответов — неполноценные.

«Зачтено» выставляется только если по всем трем компонентам студент получил «Зачтено».

«Не зачтено» выставляется, если хотя бы по одному из компонентов студент не преодолел минимальный порог.

Методические рекомендации, определяющие процедуры оценивания лабораторных работ:

Процедура оценивания лабораторных работ проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

По каждой лабораторной работе оформляется отчет. Отчеты сдаются на проверку руководителю в течение курса по мере их выполнения, и защищаются студентами в установленном порядке.

При защите отчета студенту могут быть заданы вопросы и дополнительные задания по сути лабораторной работы, в том числе из списка контрольных вопросов к данной лабораторной работе. При неудовлетворительной оценке знаний студента по теме данного отчета, студент возвращается к повторному изучению соответствующих материалов, после чего допускается к повторной защите. Неудовлетворительно выполненный отчет также возвращается на доработку.

Отчет должен содержать заголовок, тему лабораторной работы, цель, задание, индивидуальную тему, описание хода выполнения работы, необходимые прикладные материалы (схемы, макеты документов и т.п.), в соответствии с требованиями к содержанию, и выводы по работе.

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 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.

4.3. Методические указания по организации вычислительной инфраструктуры

1. Цель и задачи

Цель: Сформировать у студентов практические навыки развертывания и управления современной инфраструктурой для поддержки полного жизненного цикла коллективной разработки программного обеспечения.

Задачи:

Освоить принципы инфраструктуры как кода (IaC).

Получить опыт работы с системами контейнеризации и оркестрации.

Настроить pipelines непрерывной интеграции и поставки (CI/CD).

Организовать удаленную совместную работу над кодом с использованием Git.

Обеспечить базовый мониторинг и управление конфигурациями.

2. Рекомендуемая архитектура инфраструктуры

Для каждого проектного коллектива (3-5 человек) рекомендуется развернуть следующую схему:

text

[Локальные машины разработчиков]

|

| (Git push/pull)

v

[Удаленный Git-хостинг (GitLab / GitHub)]

|

| (Webhook trigger)

v

[CI/CD Сервер (GitLab Runner / GitHub Actions)]

|

| (Docker build/push)

v

[Контейнерный реестр (Docker Hub / GitLab Registry)]

|

| (Deploy command)

v

[Продакшен-сервер / Хостинг-площадка]

|

v

[Контейнеризованное приложение (Docker Compose)]

3. Технологический стек и инструменты (рекомендуемый)

Обязательный минимум:

Система контроля версий: Git

Удаленный хостинг Git-репозиториев: GitHub, GitLab или Bitbucket. Предпочтение — GitLab, так как он предоставляет встроенный CI/CD, Issue Tracker и Container Registry в одном продукте.

Контейнеризация: Docker

Оркестрация контейнеров (для проектов повышенной сложности): Docker Compose

Инфраструктура как код (IaC):

Базовый уровень: docker-compose.yml

Продвинутый уровень: Ansible playbooks для деплоя или Terraform для управления облачной инфраструктурой (если есть доступ к облачным кредитам).

CI/CD:

GitLab CI/CD (.gitlab-ci.yml) или GitHub Actions (.github/workflows/).

Pipeline должен включать стадии: build, test, deploy.

Опционально (для углубленного изучения):

Веб-сервер / Reverse Proxy: Nginx (для раздачи статики и проксирования запросов к приложению).

Мониторинг: Prometheus + Grafana (для сбора и визуализации метрик), или готовые облачные решения (Datadog, New Relic — по бесплатным тарифам).

Базы данных: Управляемая облачная БД (AWS RDS, DigitalOcean Managed Database) либо контейнеризованная версия (PostgreSQL/MySQL в Docker).

4. Практические задания и этапы внедрения

Этап 1: Базовая настройка репозитория и CI/CD

Создать группу/организацию на выбранном Git-хостинге.

Создать проект, настроить доступ для всех членов команды.

Настроить .gitignore, README.md, LICENSE.

Создать базовый Dockerfile для приложения.

Написать простейший .gitlab-ci.yml (или аналог для GitHub Actions), который выполняет сборку Docker-образа и запуск unit-тестов.

Этап 2: Контейнеризация и оркестрация

Определить все сервисы приложения (frontend, backend, БД, кэш).

Создать docker-compose.yml для локальной разработки и для продакшена.

Настроить переменные окружения для управления конфигурацией.

Этап 3: Настройка автоматического деплоя

Зарегистрировать/настроить сервер для деплоя (рекомендуется VPS от DigitalOcean, VK Cloud, Selectel и др. с помесячной оплатой).

Настроить в CI/CD pipeline автоматический деплой на сервер при пуше в ветку main/master.

Способ: Использование Docker Compose на удаленном сервере через ssh.

Альтернативный способ: Развертывание через Ansible-плейбук, запускаемый из CI/CD.

Этап 4: Мониторинг и документация

Настроить логирование приложения.

(Опционально) Развернуть стек мониторинга (Prometheus+Grafana) для сбора базовых метрик (CPU, RAM, нагрузка сети).

Задокументировать всю инфраструктуру: схему, инструкции по развертыванию, описание pipeline.

5. Критерии оценки инфраструктурной составляющей проекта

Инфраструктура является неотъемлемой частью проектной работы и оценивается в рамках общего критерия по проекту.

На «Зачтено» необходимо:

Работоспособность: Приложение должно быть доступно по URL (или на проверочном сервере) после автоматического деплоя.

Контейнеризация: Приложение и все его зависимости должны быть упакованы в Docker-контейнеры.

CI/CD: Настроен автоматический pipeline, который проходит как минимум стадии build и test. В Git-репозитории видна история запусков pipeline.

Документация: В README.md присутствуют четкие инструкции по запуску проекта локально и описание процесса деплоя.

Работа в команде: Вклад в инфраструктурную часть распределен между членами команды (видно по истории коммитов).

4.4. Методические указания по организации лабораторных работ

1. Цель и задачи

Цель: Сформировать у студентов устойчивые практические навыки работы в распределенной команде разработчиков с использованием современного инструментария и методологий.

Задачи:

Освоить жизненный цикл коллективной разработки от планирования до поставки ПО.

Сформировать компетенции работы с системами контроля версий.

Приобрести опыт настройки процессов непрерывной интеграции и поставки.

Отработать процедуры код-ревью и совместной разработки.
Закрепить навыки контейнеризации приложений.

2. Формат проведения работ
Периодичность: 6 лабораторных работ в течение семестра
Формат: Выполнение в командах по 3-4 человека
Сдача: Защита каждой работы с демонстрацией результата
Ведение: Общий репозиторий команды с историей выполнения всех ЛР

3. Структура и содержание лабораторных работ
Лабораторная работа 1: «Организация рабочего процесса команды»
Цель: Настроить инфраструктуру для коллективной работы и освоить базовые workflow Git.
Задачи:
Создать организацию/группу на GitHub/GitLab
Настроить SSH-ключи для всех участников
Создать проект с README, .gitignore, LICENSE

Отработать Git Flow: fork → clone → feature branch → commit → push → Pull Request
Настроить Issues и Projects для трекинга задач
Результат: Репозиторий с настроенным доступом для всех членов команды, минимум 3 merged PR от разных участников.

Лабораторная работа 2: «Проектирование и начало разработки»
Цель: Спроектировать архитектуру приложения и начать разработку по методологии Scrum.
Задачи:
Провести планирование спринта (2 недели)
Разработать техническое задание и архитектурную схему
Создать milestone и распределить задачи между участниками
Реализовать базовую структуру приложения
Провести код-ревью всех изменений
Результат: ТЗ в wiki, заполненный backlog, работающая заготовка приложения.

Лабораторная работа 3: «Настройка CI/CD пайплайна»
Цель: Настроить автоматическую сборку и тестирование проекта.
Задачи:
Написать Dockerfile для приложения
Настроить CI-пайплайн для автоматического запуска тестов
Добавить статический анализ кода (linters)
Настроить сборку Docker-образов
Добавить бейджи статуса сборки в README
Результат: Рабочий CI-пайплайн, проходящий все стадии, Docker-образ в registry.

Лабораторная работа 4: «Внедрение практик код-ревью»
Цель: Отработать процессы проверки кода и управления качеством.
Задачи:
Настроить protected branches
Создать checklist для код-ревью
Реализовать feature с обязательным ревью 2 участников
Настроить template для Pull Request
Внедрить требование успешной сборки для мержа PR

Результат: Процесс код-ревью документально оформлен и отработан на практике.

Лабораторная работа 5: «Настройка окружений и деплоя»
Цель: Настроить многоступенчатый деплой приложения.

Задачи:
Создать docker-compose.yml для разработки и продакшена
Настроить CD-пайплайн для деплоя на staging-сервер
Реализовать деплой по тегам на production-сервер
Настроить миграции БД при деплое
Добавить health-check для контейнеров
Результат: Приложение автоматически деплоится на два окружения.

Лабораторная работа 6: «Мониторинг и документация»
Цель: Настроить мониторинг работы приложения и завершить документацию.
Задачи:
Настроить логирование в контейнерах
Добавить сбор метрик приложения
Настроить дашборд для мониторинга
Задокументировать API (Swagger/OpenAPI)
Составить инструкцию по развертыванию и troubleshooting
Результат: Полная документация проекта, работающая система мониторинга.

4. Критерии оценки лабораторных работ
Общие требования для защиты:
Соблюдение сроков сдачи
Участие всех членов команды
Соответствие заданиям ЛР
Качество выполнения и оформления
Критерии оценки каждой ЛР:

Критерий	Вес	Описание
Техническая реализация	40%	Корректность выполнения, работоспособность решения
Работа в команде	30%	Вклад каждого участника, история коммитов, код-ревью
Документация	20%	Описание решения, скриншоты, инструкции
Качество кода	10%	Соблюдение code style, чистота репозитория

Шкала оценки:
«Зачтено»: 70-100% от максимального балла
«Не зачтено»: менее 70% от максимального балла

5. Методические рекомендации
Для студентов:
Распределяйте роли в команде (тимлид, разработчик, тестировщик, DevOps)
Проводите ежедневные стендапы (10-15 минут)
Ведите подробную документацию всех процессов
Своевременно обращайтесь за консультациями

6. Требования к отчетности
Для каждой ЛР команда предоставляет:
Ссылку на репозиторий с выполненной работой
Пулл-реквесты с выполненными задачами
Скриншоты работающего функционала
Краткий отчет в wiki репозитория:
Цель работы
Выполненные задачи
Распределение работы между участниками
Проблемы и пути их решения

Выводы

7. Рекомендуемый инструментарий
Хостинг кода: GitHub Education, GitLab
CI/CD: GitHub Actions, GitLab CI
Контейнеризация: Docker, Docker Compose
Коммуникация: Telegram, Discord, Mattermost
Документация: Wiki GitHub/GitLab, Markdown

4.5. Методические указания по организации проектной деятельности студентов

1. Цель и задачи

Цель: Интегрировать полученные в ходе лабораторных работ знания и навыки в рамках реализации сквозного проекта, максимально приближенного к реальным условиям промышленной разработки.

Задачи:

Сформировать опыт полного жизненного цикла разработки ПО в команде

Закрепить навыки проектного планирования, управления задачами и командной коммуникации

Развить умение принимать архитектурные и технологические решения

Отработать процедуры поставки и сопровождения программного продукта

2. Организация проектной деятельности
Продолжительность: 1 семестр (параллельно с лабораторными работами)
Формат: Сквозной проект с поэтапной сдачей результатов
Команды: 3-4 человека с распределением ролей
Процесс: Гибкая методология разработки (Scrum/Kanban)
Результат: Работающее приложение с полной инфраструктурой

3. Этапы выполнения проекта
Этап 1: Инициация и планирование (Недели 1-3)

Результат: Утвержденное техническое задание и план проекта
Содержание:
Выбор темы проекта из утвержденного перечня или предложение собственной
Формирование команды, распределение ролей (Team Lead, Developer, DevOps, QA)
Разработка технического задания с описанием функциональных требований
Создание дорожной карты (Roadmap) проекта
Планирование первого спринта, создание бэклога продукта
Настройка проектной инфраструктуры (репозиторий, Issue Tracker, CI/CD)

Этап 2: Проектирование архитектуры (Недели 4-5)
Результат: Архитектурная документация и прототип
Содержание:
Выбор технологического стека
Проектирование архитектуры системы (микросервисы/монолит)
Проектирование схемы базы данных
Разработка API-спецификации (OpenAPI/Swagger)
Создание прототипа основного функционала
Презентация архитектурного решения

Этап 3: Активная разработка (Недели 6-12)
Результат: Работающая система с основным функционалом
Содержание:

Проведение регулярных спринтов (2-3 недели каждый)
Ежедневные стендапы, планирование спринтов, ретроспективы
Реализация функциональности согласно бэклогу
Настройка и поддержка CI/CD пайплайна
Написание unit- и integration-тестов
Регулярное код-ревью и мерж Pull Requests
Деплой на тестовые окружения

Этап 4: Тестирование и стабилизация (Недели 13-14)
Результат: Стабильная версия продукта
Содержание:
Интеграционное и системное тестирование
Исправление критических ошибок
Нагрузочное тестирование (опционально)
Подготовка production-окружения
Финальный деплой и проверка безопасности

Этап 5: Защита проекта (Недели 15-16)
Результат: Презентация и демонстрация проекта
Содержание:
Подготовка финальной документации
Создание презентации проекта
Демонстрация работающего продукта
Ответы на вопросы комиссии

4. Роли в команде
Team Lead:
Координация работы команды
Ведение бэклога продукта
Проведение совещаний
Контроль сроков и качества
Developer (2 человека):
Реализация функциональности
Написание тестов
Участие в код-ревью
Рефакторинг кода
DevOps Engineer:
Настройка инфраструктуры
Поддержка CI/CD
Деплой и мониторинг
Контейнеризация приложения

5. Критерии оценки проекта
Общий вес проекта в итоговой оценке: 50%

Критерий	Вес	Показатели оценки
Техническая реализация	30%	Работоспособность, соответствие ТЗ, качество кода, архитектурные решения
Процесс разработки	25%	Следование методологии, история коммитов, код-ревью, управление задачами
Инфраструктура и DevOps	20%	CI/CD пайплайн, контейнеризация, автоматизация деплоя, мониторинг

Документация	15%	Техническое задание, архитектурная документация, руководство пользователя
Презентация и защита	10%	Качество демонстрации, ответы на вопросы, командная работа

Минимальные требования для защиты:
Реализован основной заявленный функционал
Проект запускается по инструкции и работает
Настроен CI/CD пайплайн
Все члены команды внесли вклад в разработку
Предоставлена полная документация

6. Требования к проекту
Обязательные компоненты:
Система контроля версий (Git) с историей разработки
CI/CD пайплайн с автоматическими тестами
Контейнеризация приложения (Docker)
Автоматический деплой на сервер
Документация в README.md и wiki
Issue tracking с историей задач

5. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)

5.1 Основная литература:

1. Методы программирования : учебно-методическое пособие / авторы В. В. Подколзин, А. Н. Полетайкин, Е. П. Лукащик [и др.] ; Министерство науки и высшего образования Российской Федерации, Кубанский государственный университет. - Краснодар : Кубанский государственный университет, 2020. - 174 с.
2. Носова, Л. С. Case-технологии и язык UML [Электронный ресурс] : учебно методическое пособие / Л. С. Носова. — 2-е изд. — Электрон. текстовые данные. — Челябинск, Саратов : Южно-Уральский институт управления и экономики, Ай Пи Эр Медиа, 2019. — 67 с. — 978-5-4486-0670-0. — Режим доступа: http://www.iprbookshop.ru/81479.html.
3. Доррер, Г. А. Методология программной инженерии : учебное пособие / Г. А. Доррер. — Красноярск : Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева, 2021. — 190 с.
4. Иванов, П. К. Глубокое обучение для NLP: от RNN до Transformer / П. К. Иванов, Е. А. Смирнова. – Санкт-Петербург : БХВ-Петербург, 2022. – 256 с. – ISBN 978-5-9775-0987-6.
5. Кузнецов, М. А. Языковые модели и их применение : учебник / М. А. Кузнецов. – Москва : Лань, 2023. – 415 с. – ISBN 978-5-8114-4567-8.
6. Петров, Д. С. NLP на Python: от классики до нейросетей / Д. С. Петров. – Москва : Питер, 2022. – 320 с. – ISBN 978-5-4461-2345-6.
7. 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
8. 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

9. Lu Y., Xu X., Wang L. Smart manufacturing process and system automation–a critical review of the standards and envisioned scenarios //Journal of Manufacturing Systems. – 2020. – Т. 56. – С. 312-325.

10. Ajiga D. et al. The role of software automation in improving industrial operations and efficiency //International Journal of Engineering Research Updates. – 2024. – Т. 7. – №. 1. – С. 22-35.

11. Laplante P. A., Kassab M. What every engineer should know about software engineering. – CRC Press, 2022.

12. Документация по библиотекам NLP:
– spaCy: https://spacy.io/usage
– Hugging Face Transformers: https://huggingface.co/docs/transformers/index
– NLTK: https://www.nltk.org/

5.2 Дополнительная литература:

1. Полетайкин, А. Н. Социальные и экономические информационные системы: законы функционирования и принципы построения : учеб. пособие / А. Н. Полетайкин ; Сиб. гос. ун-т телекоммуникаций и информатики. - Новосибирск : СибГУТИ, 2016. - 240 с. : ил.

2. Влацкая, И.В. Проектирование и реализация прикладного программного обеспечения : учебное пособие / И.В. Влацкая, Н.А. Заельская, Н.С. Надточий ; Кафедра компьютерной безопасности и математического обеспечения информационных систем, Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Оренбургский государственный университет». - Оренбург : ОГУ, 2015. - 119 с. – http://biblioclub.ru/index.php?page=book_red&id=439107&sr=1

3. Леоненков, А.В. Язык UML в анализе и проектировании программных систем и бизнес-процессов. Лекция 1. Базовые принципы и понятия технологии разработки объектно-ориентированных информационных систем на основе UML 2. Презентация / А.В. Леоненков. - М. : Национальный Открытый Университет «ИНТУИТ», 2014. - 34 с.– http://biblioclub.ru/index.php?page=book_red&id=238441&sr=1

4. Мостовой Я.А. Управление программными проектами [Электронный ресурс]: учебное пособие/ Мостовой Я.А.— Электрон. текстовые данные.— Самара: Поволжский государственный университет телекоммуникаций и информатики, 2016.— 103 с.— Режим доступа: http://www.iprbookshop.ru/71894.html.— ЭБС «IPRbooks».

5. Громов, К. Л. Генеративные модели в NLP: GPT и не только / К. Л. Громов. – Москва : Интернет-Университет, 2022. – 275 с. – ISBN 978-5-9556-0123-4. Лебедев, С. А. Этика искусственного интеллекта: NLP и большие языковые модели / С. А. Лебедев. – Санкт-Петербург : Альфа-книга, 2023. – 150 с. – ISBN 978-5-98281-456-9.

6. Тихонов, Р. В. Оптимизация NLP-моделей: квантование, дистилляция / Р. В. Тихонов. – Москва : Техносфера, 2022. – 195 с. – ISBN 978-5-94836-678-1.

7. Polykovskiy, Daniil, et al. "Molecular sets (MOSES): a benchmarking platform for molecular generation models." Frontiers in pharmacology 11 (2020): 565644.

8. 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.

5.3. Периодические издания:

1. Базы данных компании «Ист Вью» http://dlib.eastview.com
2. Электронная библиотека GREBENNIKON.RU https://grebennikon.ru/

Конференции A*:

1. https://openreview.net/forum?id=FMMF1a9ifL
2. https://openreview.net/forum?id=ElUrNM9U8c#discussion
3. https://openreview.net/forum?id=JoO6mtCLHD
4. https://aclanthology.org/2024.findings-emnlp.760/
5. https://aclanthology.org/2020.coling-main.588/
6. https://link.springer.com/chapter/10.1007/978-3-030-72113-8_30
7. https://link.springer.com/chapter/10.1007/978-3-031-42448-9_10
8. https://aclanthology.org/2024.findings-naacl.288/

5.4. 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. Президентская библиотека им. Б.Н. Ельцина https://www.prlib.ru/
8. База данных CSD Кембриджского центра кристаллографических данных (CCDC) https://www.ccdc.cam.ac.uk/structures/
9. Springer Journals: https://link.springer.com/
10. Springer Journals Archive: https://link.springer.com/
11. Nature Journals: https://www.nature.com/
12. Springer Nature Protocols and Methods: https://experiments.springernature.com/sources/springer-protocols
13. Springer Materials: http://materials.springer.com/
14. Nano Database: https://nano.nature.com/
15. Springer eBooks (i.e. 2020 eBook collections): https://link.springer.com/
16. "Лекториум ТВ" http://www.lektorium.tv/
17. Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Бесплатные образовательные ресурсы

1. Jupyter Notebook – интерактивные вычисления
2. Visual Studio Code – редактор кода с поддержкой Python
3. Google Scholar/arXiv – доступ к научным публикациям

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/

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

По курсу предусмотрено проведение лекционных занятий, на которых дается основной систематизированный материал по методологиям и инструментам коллективной разработки. В ходе лекций разбираются ключевые концепции: от классических (Waterfall) и гибких (Agile, Scrum, Kanban) методологий до современных практик DevOps и CI/CD. Особое внимание уделяется инструментальным средствам - системам контроля версий (Git), трекерам задач (Jira), платформам код-ревью (GitLab, GitHub), контейнеризации (Docker) и автоматизации развертывания. После каждой лекции рекомендуется выполнить практические задания по освоению рассмотренных инструментов и методик.

Лабораторные занятия курса посвящены практическому освоению инструментов коллективной разработки. На занятиях детально разбираются рабочие процессы: от организации репозитория и ветвления до настройки пайплайнов непрерывной интеграции и развертывания. Студенты работают в командах над общим проектом, применяя такие инструменты как Git, Docker, GitLab CI/CD, GitHub Actions. После каждого лабораторного занятия предлагаются задания для самостоятельного закрепления материала - отработка команд Git, модификация CI/CD конфигураций, настройка процессов код-ревью.

При самостоятельной работе студентам необходимо изучать рекомендованную литературу (учебники, документацию инструментов, лучшие практики) для глубокого понимания принципов коллективной разработки. Выполняя проектные задания, студент должен уметь: формулировать требования к проекту; организовывать рабочее пространство команды; настраивать процессы совместной работы; проектировать и реализовывать архитектуру приложения; внедрять автоматизацию сборки, тестирования и развертывания. Особое внимание уделяется навыкам разрешения конфликтов, проведения код-ревью и оптимизации процессов разработки.

Важнейшим компонентом курса является самостоятельная проектная работа, в ходе которой студенты в командах разрабатывают законченное информационное системы с полным жизненным циклом разработки. Такой проект позволяет закрепить навыки проектирования и реализации комплексных решений в условиях, максимально приближенных к промышленной разработке, включая планирование спринтов, распределение задач, код-ревью, непрерывную интеграцию и поставку.

Для студентов с ограниченными возможностями здоровья предусмотрены дополнительные индивидуальные консультации, на которых преподаватель детально разъясняет сложные аспекты дисциплины, помогает адаптировать задания и обеспечивает специальные условия для освоения практических навыков работы с текстовыми данными. Индивидуальный подход позволяет таким студентам полноценно участвовать в учебном процессе и достигать требуемых результатов обучения.

Кейсы ПАО «Сбербанк»

1. Генеративный ИИ для автоматического составления инвестиционных обзоров

Описание:

Аналитики Сбера ежедневно составляют десятки аналитических и инвестиционных обзоров по рынкам, компаниям, макроэкономике. Задача — исследовать применение LLM для генерации кратких сводок и аналитических отчетов на основе входных данных: биржевые котировки, макроэкономические показатели, рыночные события.

Цель:

Разработать инструмент, способный по структурированным данным и краткому описанию формировать инвестиционный обзор в деловом стиле.

Ожидаемый результат:

Модель, генерирующая аналитические тексты длиной 500–1000 слов с разделами «обзор событий», «рекомендации», «прогнозы», оформленные в формате банка.

2. Генерация сценариев фишинговых писем для обучения сотрудников

Описание:

Банк проводит киберучения, включая рассылку тестовых фишинговых писем сотрудникам для повышения их устойчивости к социальным атакам. Проект предполагает использование генеративной модели для создания реалистичных фишинговых писем различных типов (поддельные счета, HR-запросы, ИТ-поддержка).

Цель:

Создать генератор, способный на основе заданных параметров (тема, стиль, уровень угрозы) создавать тексты фишинга для тренировок.

Ожидаемый результат:

Набор разнообразных примеров фишинга и оценка их эффективности по реакции сотрудников, а также классификация моделей угроз.

Кейсы от «АВАЛАБ»

3. Генерация рекламного контента для жилых комплексов

Описание:

«АВА ГРУПП» регулярно запускает маркетинговые кампании для жилых комплексов. Необходимо исследовать использование диффузионных моделей для генерации изображений (визуализации интерьеров, окрестностей, видов из окон) и LLM — для описаний квартир, преимуществ района, инфраструктуры.

Цель:

Создать инструменты для быстрой генерации продающих материалов без привлечения дизайнеров и копирайтеров на первых этапах.

Ожидаемый результат:

Набор сгенерированных карточек объектов с текстом, изображением и логикой «живого» рекламного сообщения.

4. Генерация документации и шаблонов договоров

Описание:

Юридический департамент регулярно работает с договорами долевого участия, актами приёма-передачи и другими документами. Использование LLM может значительно сократить время на подготовку черновиков — достаточно ввести параметры сделки.

Цель:

Создать систему, которая генерирует адаптированные тексты документов по вводным данным (тип объекта, этаж, площадь, ФИО, сроки и пр.).

Ожидаемый результат:

Генератор документов в формате Word или PDF с автоматической подстановкой параметров и соблюдением юридического стиля.

5. Обратная генерация — ИИ-помощник для покупателей квартир

Описание:

Будущие покупатели часто задают типовые вопросы о квартирах, планировках, ипотеке, акциях, сроках. Вместо call-центра предлагается реализовать LLM-бота, который обрабатывает текстовые и голосовые запросы, показывает планировки, ссылается на PDF-документы и может «объяснять» информацию простым языком.

Цель:

Упростить коммуникацию с клиентами на этапе выбора квартиры и повысить качество первичного контакта.

Ожидаемый результат:

Демо-бот, способный отвечать на вопросы о жилом комплексе, ориентируясь в его характеристиках и маркетинговых документах.

7. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю)

7.1 Перечень информационно-коммуникационных технологий

1. Облачные платформы и сервисы

Google Colab – облачная среда для выполнения кода на Python с GPU/TPU

Kaggle – платформа для работы с датасетами и соревнований по ML

Hugging Face Spaces – развертывание NLP-моделей в виде демо

AWS/GCP/Azure/YandexCloud – облачные вычисления для обучения и инференса моделей

2. Системы управления версиями и коллаборации

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

Notion/Trello – организация проектной деятельности

3. Инструменты для работы с данными

Label Studio – разметка датасетов

DVC (Data Version Control) – управление версиями данных

4. Коммуникационные технологии

Telegram – координация работы в команде

МТС Линк– проведение онлайн-консультаций и защиты проектов

7.2 Перечень лицензионного и свободно распространяемого программного обеспечения

1. Лицензионное ПО
VSCode – IDE для Python (свободнораспространяемое)
LibreOffice– оформление отчетов (свободнораспространяемое)

2. Свободное ПО (Open Source)
Библиотеки для NLP:
NLTK – обработка текста
spaCy – промышленные NLP-пайплайны
Hugging Face Transformers – предобученные модели (BERT, GPT)
Gensim – тематическое моделирование и word2vec

Фреймворки для ML:

PyTorch/TensorFlow – разработка нейросетей
scikit-learn – классические алгоритмы ML

Инструменты для визуализации:

Streamlit/Gradio – создание веб-интерфейсов для моделей
Matplotlib/Seaborn – графики и анализ данных

СУБД:

SQLite/PostgreSQL – хранение структурированных данных
FAISS/Annoy – векторный поиск

8. Материально-техническое обеспечение по дисциплине (модулю)

№	Вид работ	Наименование учебной аудитории, ее оснащенность оборудованием и техническими средствами обучения
1.	Лекционные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
2.	Лабораторные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, проектором, программным обеспечением
3.	Практические занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
4.	Групповые (индивидуальные) консультации	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
5.	Текущий контроль, промежуточная аттестация	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
6.	Самостоятельная работа	Кабинет для самостоятельной работы, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», программой экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета.

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.