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.18 Технологии создания и поддержки ПО

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 «Технологии создания и поддержки программного обеспечения» 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):
Р.Ю. Вишняков, доцент КАДИИ, к.т.н., доцент
И.О. Фамилия, должность, ученая степень, ученое звание — подпись

Рабочая программа дисциплины «Технологии создания и поддержки программного обеспечения» утверждена на заседании кафедры анализа данных и искусственного интеллекта
протокол № 01 «28» августа 2025 г.
Заведующий кафедрой Коваленко А.В.
подпись

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

Цели изучения дисциплины определены государственным образовательным стандартом высшего образования и соотнесены с общими целями ООП ВО по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем, в рамках которой преподается дисциплина. Цели дисциплины «Технологии создания и поддержки программного обеспечения»: формирование у студентов комплексных знаний и практических навыков в области современных технологий создания и поддержки программного обеспечения на протяжении всего жизненного цикла разработки (ЖЦР).

1.2 Задачи дисциплины

- Изучить основные модели ЖЦР ПО и их особенности.
- Овоить современные методологии разработки ПО, включая Agile-подходы.
- Научиться использовать системы контроля версий для совместной разработки.
- Приобрести навыки тестирования программного обеспечения на различных уровнях.
- Познакомиться с принципами автоматизации сборки, развертывания и мониторинга ПО (CI/CD).
- Освоить базовые концепции DevOps и их применение в практике разработки.

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Технологии создания и поддержки программного обеспечения» относится к профильным дисциплинам, обеспечивающим формирование профессиональных компетенций выпускника по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем.

Для успешного освоения дисциплины необходимы следующие пререквизиты: информатика, основы программирование, операционные системы и сети, технологии программирования.

Дисциплина «Технологии создания и поддержки программного обеспечения» является основой для изучения следующих дисциплин.

Постреквизиты: Анализ, проектирование и разработка БД, Облачные технологии и бэкэндразработка, Разработка мобильных приложений, а также других дисциплин связанных с: разработкой, тестированием, поддержкой программного обеспечения, разработкой архитектур информационных систем. Углубляет знания, полученные в рамках дисциплин Основы программирования, Объектно-ориентированное программирование», Разработка пользовательского WEB интерфейса, Основы программирования на Python.

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:

Код и наименование индикатора*	**Результаты обучения по дисциплине**
ПК-3 Способен участвовать в разработке, сопровождении и администрировании информационных систем и программного обеспечения	
ПК-3.1 Способен разрабатывать алгоритмы и программы, пригодные для практического применения	Знает основные модели и методологии жизненного цикла ПО (каскадная, V-модель, Agile). Умеет выбирать и обосновывать выбор модели ЖЦР для конкретного проекта.
ПК-3.2 Способен принимать участие в управлении проектами создания информационных систем на стадиях жизненного цикла	Владеет системой контроля версий Git на уровне организации репозитория, ветвления (GitFlow), слияния и разрешения конфликтов. Применяет практики командной работы и code review в рамках общего репозитория.
ПК-3.3 Способен управлять процессами жизненного цикла ИТ-продукта	Умеет настраивать процессы непрерывной интеграции и доставки (CI/CD) с использованием современных инструментов (напр., GitHub Actions). Применяет принципы DevOps для автоматизации сборки, тестирования и развертывания ПО.
LC-4 Способен управлять процессом жизненного цикла ИИ-продукта	
LC-4.1 Осуществляет запуск и ведение проекта в области ИИ, в том числе планирование и контроль задач, оценку ресурсов	Подбирает методологию управления проектами (Agile, Scrum, Kanban) под ограничения задачи и ресурсное обеспечение. Осуществляет декомпозицию проекта на задачи, оценивает трудозатраты и строит рабочий план (бэклог) в инструментах управления (напр., Jira).
LC-4.2 Координирует и контролирует работу команд проекта с целью достижения общих целей проекта	Демонстрирует эффективное владение инструментами коммуникаций и проектного управления. Координирует взаимодействие между техническими и бизнес-командами в рамках скрам-митингов и итерационных обзоров. Применяет базовые принципы управления стейкхолдерами и работы с ожиданиями.

Формулировка компетенции	**Индикаторы компетенции**	**Уровень освоения индикаторов компетенции**
ПК-3 Способен участвовать в разработке, сопровождении и администрировании информационных систем и программного обеспечения	**ПК-3.1** Способен разрабатывать алгоритмы и программы, пригодные для практического применения	Знает основные модели и методологии жизненного цикла ПО (каскадная, V-модель, Agile). Умеет выбирать и обосновывать выбор модели ЖЦР для конкретного проекта.

	ПК-3.2 Способен принимать участие в управлении проектами создания информационных систем на стадиях жизненного цикла	Владеет системой контроля версий Git на уровне организации репозитория, ветвления (GitFlow), слияния и разрешения конфликтов. Применяет практики командной работы и code review в рамках общего репозитория.
	ПК-3.3 Способен управлять процессами жизненного цикла ИТ-продукта	Умеет настраивать процессы непрерывной интеграции и доставки (CI/CD) с использованием современных инструментов (напр., GitHub Actions). Применяет принципы DevOps для автоматизации сборки, тестирования и развертывания ПО.
LC-4 Способен управлять процессом жизненного цикла ИИ-продукта	**LC-4.1** Осуществляет запуск и ведение проекта в области ИИ, в том числе планирование и контроль задач, оценку ресурсов	Подбирает методологию управления проектами (Agile, Scrum, Kanban) под ограничения задачи и ресурсное обеспечение. Осуществляет декомпозицию проекта на задачи, оценивает трудозатраты и строит рабочий план (бэклог) в инструментах управления (напр., Jira).
	LC-4.2 Координирует и контролирует работу команд проекта с целью достижения общих целей проекта	Демонстрирует эффективное владение инструментами коммуникаций и проектного управления. Координирует взаимодействие между техническими и бизнес-командами в рамках скрам-митингов и итерационных обзоров. Применяет базовые принципы управления стейкхолдерами и работы с ожиданиями.

Результаты обучения по дисциплине достигаются в рамках осуществления всех видов контактной и самостоятельной работы обучающихся в соответствии с утвержденным учебным планом.

Индикаторы достижения компетенций считаются сформированными при достижении соответствующих им результатов обучения.

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 4 зачетных единиц (144 часов), их распределение по видам работ представлено в таблице

Виды работ	Всего Часов	Форма обучения	
		Очная	
		4 семестр (часы)	5 семестр (часы)
Контактная работа, в том числе:	**102,4**	**50,2**	**52,2**
Аудиторные занятия (всего):	**118**	**48**	**50**
занятия лекционного типа	50	16	16

лабораторные занятия		68	32	34
практические занятия		-	-	-
семинарские занятия		-	-	-
Иная контактная работа:		**4,4**	**2,2**	**2,2**
Контроль самостоятельной работы (КСР)		4	2	2
Промежуточная аттестация (ИКР)		0,4	0,2	0,2
Самостоятельная работа, в том числе:		**41,6**	**21,8**	**19,8**
Проработка и повторение лекционного материала и материала учебников и учебных пособий		41,6	21,8	19,8
Подготовка к текущему контролю				
Контроль:		**-**	**-**	**-**
Подготовка к экзамену		-	-	-
Общая трудоемкость	**час.**	**144**	**72**	**72**
	в том числе контактная работа	**102,4**	**50,2**	**52,2**
	зач. Ед	**4**	**2**	**2**

2.2 Содержание дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины. Разделы (темы) дисциплины, изучаемые в 4 семестре (очная форма обучения)

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная Работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1.	Жизненный цикл разработки программного обеспечения (ЖЦР)	**17**	4		8	5
2.	Методологии разработки ПО	**17**	4		8	5
3.	Системы контроля версий (Git)	**17**	4		8	5
4.	Тестирование программного обеспечения	**18,8**	4		8	6,8
	ИТОГО по разделам дисциплины	**69,8**	**16**		**32**	**21,8**
	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

Разделы (темы) дисциплины, изучаемые в 5 семестре (очная форма обучения)

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная Работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1.	Автоматизация сборки и развертывания ПО (CI/CD)	**17**	4		8	5
2.	Основы DevOps	**18**	4		9	5
3.	Контейнеризация (Docker)	**18**	4		9	5
4.	Оркестрация контейнеров (Kubernetes)	**16,8**	4		8	4,8
	ИТОГО по разделам дисциплины	**69,8**	**16**		**34**	**19,8**
	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.	**Жизненный цикл разработки программного обеспечения (ЖЦР)**	– Классические модели их преимущества и недостатки, области применения: Waterfall, V-model, Spiral model. – Итеративные и инкрементные модели: Rational Unified Process (RUP). – Agile методологии их принципы, роли, артефакты, церемонии: Scrum, Kanban, Extreme Programming (XP). – Сравнение Agile с традиционными подходами.	Устные ответы;
2.	**Методологии разработки ПО**	– Принципы SOLID и DRY. – Паттерны проектирования: основные типы паттернов (создающие, структурные, поведенческие). – Code Review – принципы проведения и инструменты.	Устные ответы;
3.	**Системы контроля версий (Git)**	– Основные понятия: репозиторий, коммит, ветка, и т.д. – Команды Git: init, clone, add, commit, push, pull, branch, merge, rebase. – Работа с удаленными репозиториями (GitHub, GitLab), создание репозиториев, клонирование, отправка изменений, работа с issue tracker. – Стратегии ветвления: Gitflow, GitHub Flow.	Устные ответы; Лабораторная работа.
4.	**Тестирование программного обеспечения**	– Уровни тестирования: модульное тестирование (unit testing), интеграционное тестирование, системное тестирование, приемочное тестирование. – Виды тестирования: функциональное тестирование, нефункциональное тестирование (нагрузочное, стрессовое, security). – Методики тестирования: black-box testing, white-box testing. – Инструменты автоматизированного тестирования.	Устные ответы; Тесты; Лабораторная работа.
5.	**Автоматизация сборки и развертывания ПО (CI/CD)**	– Принципы CI/CD. – Инструменты CI/CD: Jenkins, Travis CI, CircleCI, GitLab CI. – Настройка пайплайнов CI/CD для автоматической сборки, тестирования и развертывания приложений.	Устные ответы; Лабораторная работа.
6.	**Основы DevOps**	– Культура DevOps: Collaboration, Automation, Measurement, Sharing (CAMS). – Инфраструктура как код (IaC). – Мониторинг и логирование.	Устные ответы; Лабораторная работа.
7.	**Контейнеризация (Docker)**	– Основные понятия: образы, контейнеры, Dockerfile. – Создание и запуск контейнеров. – Docker Compose для управления многоконтейнерными приложениями. – Разработка Dockerfile для простого приложения.	Устные ответы; Лабораторная работа.
8.	**Оркестрация контейнеров (Kubernetes)**	– Основные понятия: поды, сервисы, деплойменты, namespace. – Развертывание приложений в Kubernetes. – Масштабирование и обновление приложений в Kubernetes. – Практическое развертывание простого приложения в Kubernetes.	Устные ответы; Лабораторная работа.

2.3.2 Занятия семинарского типа (практические / семинарские занятия/ лабораторные работы)

№	Наименование раздела (темы)	Тематика занятий/рабор	Форма текущего контроля
1.	**Системы контроля версий (Git)**	Практическое применение Git для совместной разработки небольшого проекта, имитирующего реальный рабочий процесс команды разработчиков. Акцент на понимании принципов ветвления и разрешения конфликтов. Освоить базовые команды Git для управления репозиторием, включая инициализацию, добавление файлов, коммиты, ветвление и слияние. Научиться работать с удаленными репозиториями на GitHub/GitLab, создавать pull requests и проводить code review.	ЛР
2.	**Тестирование программного обеспечения**	Разработка тестового набора для обеспечения качества кода, выявление ошибок на ранних этапах разработки и повышение надежности программного продукта. Написать модульные тесты (unit tests) для существующего кода с использованием выбранного фреймворка (JUnit, pytest). Реализовать интеграционные тесты для проверки взаимодействия между различными компонентами приложения. Изучить и применить различные техники тестирования (black-box, white-box).	ЛР
3.	**Автоматизация сборки и развертывания ПО (CI/CD)**	Автоматизация рутинных задач разработки, ускорение процесса поставки программного обеспечения и снижение риска ошибок при развертывании. Настроить CI/CD-пайплайн с использованием Jenkins/Travis CI/GitLab CI для автоматической сборки, тестирования и развертывания небольшого приложения при каждом изменении кода в репозитории.	ЛР
4.	**DevOps**	Применение принципов DevOps для автоматизации управления инфраструктурой, повышения эффективности работы команды разработки и обеспечения стабильности и масштабируемости приложений. Использовать инструменты инфраструктуры как код для автоматического создания и настройки виртуальных машин или облачных ресурсов. Настроить базовый мониторинг приложения.	ЛР
5.	**Контейнеризация (Docker)**	Упаковка приложений в изолированные контейнеры для обеспечения переносимости, воспроизводимости и упрощения развертывания. Создать Dockerfile для контейнеризации простого приложения (например, веб-сервера). Научиться собирать образы Docker, запускать контейнеры и управлять ими. Использовать Docker Compose для управления многоконтейнерными приложениями.	ЛР
6.	**Оркестрация контейнеров (Kubernetes)**	Автоматизация развертывания, управления и масштабирования контейнеризированных приложений в кластерной среде, обеспечение высокой доступности и отказоустойчивости. Развернуть контейнеризированное приложение в Kubernetes-кластере с использованием YAML-манифестов. Настроить масштабирование приложения и обновление версий. Изучить основные компоненты Kubernetes: поды, сервисы, деплойменты.	ЛР

Защита лабораторной работы (ЛР), выполнение курсового проекта (КП), курсовой работы (КР), расчетно-графического задания (РГЗ), написание реферата (Р), эссе (Э), коллоквиум (К), тестирование (Т) и т.д.

2.3.3 Курсовые работы

Курсовые работы 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	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 кафедры анализа данных и искусственного интеллекта факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 22.03.2023 г. Методические указания по выполнению самостоятельной работы, утвержденные на заседании кафедры анализа данных и искусственного интеллекта факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 22.03.2023 г.
2	Подготовка к лабораторным занятиям	Методические указания по выполнению лабораторных работ, утвержденные на заседании кафедры анализа данных и искусственного интеллекта факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 22.03.2023 г.
3	Подготовка к решению задач и тестов	Методические указания по выполнению самостоятельной работы, утвержденные на заседании кафедры анализа данных и искусственного интеллекта факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 22.03.2023 г.
4	Подготовка к текущему контролю	Методические указания по выполнению самостоятельной работы, утвержденные на заседании кафедры анализа данных и искусственного интеллекта факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 22.03.2023 г.
5	Подготовка докладов	Методические указания для подготовки эссе, рефератов, курсовых работ, утвержденные на заседании кафедры анализа данных и искусственного интеллекта факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 22.03.2023 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

В соответствии с требованиями ФГОС в 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 средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Технологии создания и поддержки программного обеспечения».

Оценочные средства включают контрольные материалы для проведения **текущего контроля** в форме тестов и лабораторных работ, и **промежуточной аттестации** в форме вопросов к зачету.

Текущий контроль успеваемости осуществляется в течение семестра, в ходе повседневной учебной работы и предполагает овладение материалами лекций,

литературы, программы, работу студентов в ходе проведения лабораторных занятий, а также систематическое выполнение лабораторных работ, решение практических задач и иных заданий для самостоятельной работы студентов. Данный вид контроля стимулирует у студентов стремление к систематической самостоятельной работе по изучению дисциплины. Он предназначен для оценки самостоятельной работы слушателей по решению задач, выполнению лабораторных работ, подведения итогов тестирования. Оценивается также активность и качество результатов практической работы на занятиях, участие в дискуссиях, обсуждениях и т.п. Индивидуальные и групповые самостоятельные, аудиторные работы по всем темам дисциплины организованы единообразным образом. Для контроля освоения содержания дисциплины используются оценочные средства. Они направлены на определение степени сформированности компетенций.

Промежуточная аттестация студентов осуществляется в рамках завершения изучения дисциплины и позволяет определить качество усвоения изученного материала, предполагает контроль и управление процессом приобретения студентами необходимых знаний, умения и навыков, определяемых по ФГОС ВО по соответствующему направлению подготовки в качестве результатов освоения учебной дисциплины.

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 зачете;

– 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	Жизненный цикл разработки программного обеспечения (ЖЦР)	ПК-3.2, ПК-3.3, LC-4.1	-	Вопросы к зачету 1-2

2	Методологии разработки ПО	ПК-3.2, ПК-3.3, LC-4.1, LC-4.2	-	Вопросы к зачету 3-4
3	Системы контроля версий (Git)	ПК-3.1, ПК-3.2, LC-4.2	ЛР№1	Вопросы к зачету 5-9
4	Тестирование программного обеспечения	ПК-3.1, ПК-3.3	ЛР№2	Вопросы к зачету 10-14
5	Автоматизация сборки и развертывания ПО (CI/CD)	ПК-3.1, ПК-3.3, LC-4.1	ЛР№3	Вопросы к зачету 15-19
6	Основы DevOps	ПК-3.2, ПК-3.3, LC-4.1, LC-4.2	ЛР№4	Вопросы к зачету 20-21
7	Контейнеризация (Docker)	ПК-3.1, ПК-3.3	ЛР№5	Вопросы к зачету 22-23
8	Оркестрация контейнеров (Kubernetes)	ПК-3.2, ПК-3.3, LC-4.2	ЛР№6	Вопросы к зачету 24-25

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**)				
на пороговом уровне:				
1.	**ПК-3.1** Способен разрабатывать алгоритмы и программы, пригодные для практического применения	Знать: базовые синтаксис и структуры языков программирования, основные команды Git, принципы модульного тестирования. Уметь: писать простой код, использовать Git для базовых операций (коммит, ветвление), запускать готовые модульные тесты. Владеть: навыками отладки простых ошибок, работой с системой контроля версий на уровне пользователя.	ЛР	Вопросы к зачету: 5-15, 22, 23
2.	**ПК-3.2** Способен принимать участие в управлении проектами создания информационных систем на стадиях жизненного цикла	Знать: основные этапы ЖЦ ПО, названия и общие принципы Agile/Scrum, назначение Pull Request. Уметь: определить этап ЖЦ проекта, участвовать в обсуждении задач в рамках SCRUM-митинга, создать PR по шаблону. Владеть: базовой терминологией в области управления проектами.	ЛР	Вопросы к зачету: 1-4, 8, 20, 24, 25
3.	**ПК-3.3** Способен управлять процессами	Знать: определения CI/CD, DevOps, контейнеризации;	ЛР	Вопросы к зачету: 11-

	жизненного цикла ИТ-продукта	назначение основных инструментов (Jenkins, Docker, Kubernetes). Уметь: запустить готовый CI/CD-пайплайн, собрать Docker-образ по готовому Dockerfile, развернуть приложение в Kubernetes по готовому манифесту. Владеть: навыками выполнения стандартных операций в рамках заданного процесса.		15, 17-20, 22-25
4.	**LC-4.1** Осуществляет запуск и ведение проекта в области ИИ, в том числе планирование и контроль задач, оценку ресурсов	Знать: основы планирования задач (бэклог, спринт), принципы декомпозиции требований. Уметь: составить простой план задач для небольшого проекта, оценить необходимые вычислительные ресурсы для запуска контейнеризованного приложения. Владеть: навыками работы с инструментами для автоматизации сборки и развертывания (CI/CD) для запуска проекта.	ЛР	Вопросы к зачету: 1-4, 11, 12, 15-17, 20
5.	**LC-4.2** Координирует и контролирует работу команд проекта с целью достижения общих целей проекта	Знать: роль код-ревью и инструментов коллаборации (Git) в координации команды. Уметь: проводить базовое код-ревью, формулировать комментарии к PR, работать в общей ветке репозитория. Владеть: навыками взаимодействия в команде с использованием Git и процессов CI/CD.	ЛР	Вопросы к зачету: 4, 8, 9, 18, 24, 25
Соответствие освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **хорошо /зачтено**)				
<u>на базовом уровне</u>:				
1.	**ПК-3.1** Способен разрабатывать алгоритмы и программы, пригодные для практического применения	**Знать:** принципы написания чистого и поддерживаемого кода, различные стратегии слияния в Git, виды и цели тестирования (black-box/white-box). **Уметь:** применять различные команды Git (merge, rebase) для управления историей, писать модульные и интеграционные тесты, конфигурировать этапы в CI/CD-пайплайне. **Владеть:** навыками рефакторинга кода, разрешения	ЛР	Вопросы к зачету: 5-15, 22, 23

		конфликтов слияния, настройки инструментов тестирования и сборки.		
2.	**ПК-3.2** Способен принимать участие в управлении проектами создания информационных систем на стадиях жизненного цикла	**Знать:** отличия методологий (Waterfall, Agile), роли и артефакты в Scrum, принципы управления инфраструктурой как код (IaC). **Уметь:** выбирать методологию в зависимости от контекста проекта, участвовать в планировании спринта, описывать простую инфраструктуру в виде кода. **Владеть:** навыками работы с инструментами IaC (например, Terraform), практиками проведения код-ревью.	ЛР	Вопросы к зачету: 1-4, 8, 20, 21, 24, 25
3.	**ПК-3.3** Способен управлять процессами жизненного цикла ИТ-продукта	**Знать:** преимущества и недостатки разных подходов к контейнеризации и оркестрации, этапы CI/CD-пайплайна. **Уметь:** настраивать CI/CD-пайплайн для автоматического развертывания, оптимизировать Dockerfile, конфигурировать базовые объекты Kubernetes (Deployment, Service). **Владеть:** навыками отладки пайплайнов, управления жизненным циклом контейнеров.	ЛР	Вопросы к зачету: 11-15, 17-20, 22-25
4.	**LC-4.1** Осуществляет запуск и ведение проекта в области ИИ, в том числе планирование и контроль задач, оценку ресурсов	**Знать:** методы оценки трудозатрат и ресурсов для DevOps-инфраструктуры. **Уметь:** планировать этапы внедрения CI/CD и контейнеризации в проект, контролировать выполнение задач по настройке инфраструктуры. **Владеть:** навыками мониторинга выполнения пайплайнов и использования ресурсов кластера.	ЛР	Вопросы к зачету: 1-4, 11, 12, 15-17, 20, 21
5.	**LC-4.2** Координирует и контролирует работу команд проекта с целью достижения общих целей проекта	**Знать:** лучшие практики проведения код-ревью и совместной работы в Git. **Уметь:** координировать процесс слияния веток в команде, обеспечивать соблюдение стандартов кодирования через CI/CD. **Владеть:** навыками организации workflow в Git, настройки проверок кода в пайплайне.	ЛР	Вопросы к зачету: 4, 6, 8, 9, 18, 24, 25

Соответствие освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **отлично /зачтено**)				
<u>на продвинутом уровне:</u>				
1.	**ПК-3.1** Способен разрабатывать алгоритмы и программы, пригодные для практического применения	**Знать:** передовые практики DevOps (GitOps, Infrastructure as Code), принципы построения отказоустойчивых и масштабируемых систем. **Уметь:** проектировать и реализовывать сложные CI/CD-пайплайны с несколькими стадиями и окружениями, оптимизировать процессы сборки и развертывания. **Владеть:** навыками глубокой отладки и оптимизации всего цикла разработки и поставки ПО.	ЛР	Вопросы к зачету: 5-15, 22, 23
2.	**ПК-3.2** Способен принимать участие в управлении проектами создания информационных систем на стадиях жизненного цикла	**Знать:** современные тенденции в управлении IT-проектами (DevOps-культура, SRE). **Уметь:** выбирать и адаптировать инструменты и практики (CI/CD, оркестрация) под конкретные бизнес-процессы проекта. **Владеть:** навыками комплексного внедрения и управления процессами ЖЦ ПО в команде.	ЛР	Вопросы к зачету: 1-4, 8, 20, 21, 24, 25
3.	**ПК-3.3** Способен управлять процессами жизненного цикла ИТ-продукта	**Знать:** архитектурные паттерны для микросервисов и их оркестрации, продвинутые техники мониторинга и обеспечения безопасности в CI/CD и Kubernetes. **Уметь:** проектировать и внедрять сквозные процессы ЖЦ для сложных распределенных систем. **Владеть:** навыками кристического анализа и выбора инструментов и технологий для управления ЖЦ продукта.	ЛР	Вопросы к зачету: 11-15, 17-20, 22-25
4.	**LC-4.1** Осуществляет запуск и ведение проекта в области ИИ, в том числе планирование и контроль задач, оценку ресурсов	**Знать:** подходы к управлению MLOps-проектами, включая версионирование данных и моделей в пайплайнах. **Уметь:** интегрировать процессы машинного обучения в сквозные CI/CD/ML-пайплайны, планировать ресурсы для сложных ML-развертываний в Kubernetes. **Владеть:** навыками полного контроля над жизненным циклом AI-продукта от кода до	ЛР	Вопросы к зачету: 1-4, 11, 12, 15-17, 20, 21

		продакшена.		
5.	**LC-4.2** Координирует и контролирует работу команд проекта с целью достижения общих целей проекта	**Знать:** методы повышения эффективности командной работы через автоматизацию (например, автотесты в CI как условие мержа). **Уметь:** выстраивать и оптимизировать процессы взаимодействия между командами разработки, тестирования и эксплуатации (DevOps). **Владеть:** навыками организации и координации распределенных команд в рамках единого технологического процесса.	ЛР	Вопросы к зачету: 4, 6, 8, 9, 18, 24, 25

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

Пример опроса по темам с ответами

Тема 1: Жизненный цикл разработки программного обеспечения (ЖЦР)
Вопрос 1: Назовите и охарактеризуйте основные этапы жизненного цикла ПО.
Ответ: Стандартный жизненный цикл включает: 1) Сбор и анализ требований: определение целей и функциональности проекта. 2) Проектирование: создание архитектуры системы. 3) Реализация (кодирование): написание исходного кода. 4) Тестирование: выявление и устранение ошибок. 5) Внедрение: развертывание системы у пользователей. 6) Сопровождение и поддержка: исправление ошибок, обновление и улучшение.

Вопрос 2: В чем различия между каскадной (Waterfall) и итеративной моделью ЖЦ?
Ответ: Каскадная модель — строго последовательный процесс, где переход на следующий этап возможен только после полного завершения предыдущего. Негибка, сложно вносить изменения. Итеративная модель — разработка ведется циклами (итерациями), на каждой из которых создается работающая версия продукта. Это позволяет рано получать обратную связь и гибко реагировать на изменения требований.

Тема 2: Методологии разработки ПО
Вопрос 3: Опишите основные принципы Agile-манифеста.
Ответ: 1) Люди и взаимодействие важнее процессов и инструментов. 2) Работающий продукт важнее исчерпывающей документации. 3) Сотрудничество с заказчиком важнее согласования условий контракта. 4) Готовность к изменениям важнее следования первоначальному плану.

Вопрос 4: Что такое Scrum и какие основные роли и артефакты в нем определены?
Ответ: Scrum — это фреймворк для реализации Agile. Основные роли: Владелец Продукта (формирует требования), Scrum-мастер (устраняет препятствия), Команда разработки (создает продукт). Ключевые артефакты: Бэклог продукта (список всех задач), Бэклог спринта (задачи на текущую итерацию), Инкремент продукта (результат спринта).

Тема 3: Системы контроля версий (Git)
Вопрос 5: Что такое система контроля версий и зачем она нужна в разработке программного обеспечения?
Ответ: Система контроля версий (СКВ) – это инструмент, позволяющий отслеживать изменения в файлах проекта. Она необходима для совместной работы над проектом, сохранения истории изменений, возможности отката к предыдущим версиям и ветвления разработки. Без СКВ сложно поддерживать стабильность кода и эффективно работать в команде.

Вопрос 6: Объясните разницу между командами git merge и git rebase. В каких ситуациях целесообразно использовать каждую из них?
Ответ: git merge создает новый коммит, объединяющий изменения из двух веток. История изменений сохраняется в виде ветвей и слияний. Команда git rebase перемещает последовательность коммитов одной ветки на вершину другой, создавая линейную историю. Rebase полезен для очистки истории перед отправкой изменений, но может привести к проблемам при совместной работе, если ветка уже опубликована.

Вопрос 7: Что такое .gitignore и как он используется? Приведите примеры файлов или папок, которые обычно добавляют в этот файл.
Ответ: .gitignore – это текстовый файл, содержащий список файлов и папок, которые Git должен игнорировать при отслеживании изменений. Он позволяет исключить из репозитория временные файлы, скомпилированный код, логи и другие артефакты, не относящиеся к исходному коду проекта. Примеры: /bin/, /obj/, .idea/, *.log.

Вопрос 8: Что такое Pull Request (PR) и зачем он нужен? Опишите процесс создания и рецензирования PR на GitHub/GitLab.
Ответ: Pull Request – это запрос на слияние изменений из одной ветки в другую. Он позволяет другим разработчикам просмотреть код, оставить комментарии и предложить улучшения перед интеграцией изменений в основную ветку. Процесс включает создание PR, рецензирование кода, внесение правок (при необходимости) и утверждение PR для слияния.

Вопрос 9: Как разрешить конфликт при слиянии веток в Git? Опишите шаги, которые необходимо предпринять.
Ответ: Конфликты возникают, когда изменения в разных ветках затрагивают одни и те же строки кода. Для разрешения конфликта нужно открыть файл с конфликтом, найти маркеры конфликта (<<<<<<<, =======, >>>>>>>), вручную выбрать нужные изменения или объединить их, удалить маркеры, добавить исправленный файл в индекс (git add) и закоммитить результат (git commit).

Тема 4: Тестирование программного обеспечения
Вопрос 10: Какие основные уровни тестирования вы знаете? Опишите каждый из них.
Ответ: Модульное тестирование (Unit testing) – проверка отдельных модулей или функций кода. Интеграционное тестирование – проверка взаимодействия между различными компонентами системы. Системное тестирование – проверка всей системы в целом на соответствие требованиям. Приемочное тестирование – проверка системы конечным пользователем для подтверждения ее готовности к эксплуатации.

Вопрос 11: Что такое TDD (Test-Driven Development)? Опишите процесс разработки по методологии TDD.
Ответ: TDD – это методология разработки, в которой сначала пишутся тесты, а затем код,

который эти тесты проходит. Процесс: 1) Red: написать тест, который не проходит. 2) Green: написать минимальный объем кода для прохождения теста. 3) Refactor: улучшить структуру и читаемость кода, сохраняя "зеленый" статус тестов.

Вопрос 12: В чем разница между black-box и white-box тестированием? Приведите примеры.
Ответ: Black-box – тестирование без знания внутренней структуры кода, основанное на входных данных и ожидаемых результатах (например, функциональное тестирование интерфейса). White-box – тестирование с доступом к исходному коду, позволяющее проверить логику работы программы (например, модульное тестирование).

Вопрос 13: Что такое покрытие кода (code coverage) и зачем оно нужно?
Ответ: Покрытие кода – это метрика, показывающая процент строк, ветвей или условий кода, которые были выполнены тестами. Оно позволяет оценить эффективность тестового набора и выявить непротестированные участки кода. Важно помнить, что высокое покрытие не гарантирует отсутствие ошибок, но повышает уверенность в качестве кода.

Вопрос 14: Какие инструменты для автоматизации тестирования вы знаете? Приведите примеры.
Ответ: Модульные тесты: JUnit (Java), pytest (Python). Интеграционные и системные тесты: Selenium (веб-интерфейсы), Postman (API). Фреймворки: Jest (JavaScript). Эти инструменты позволяют создавать и запускать тесты автоматически.

Тема 5: Автоматизация сборки и развертывания ПО (CI/CD)
Вопрос 15: Что такое CI/CD? Объясните значение каждого термина.
Ответ: CI (Continuous Integration) – это практика автоматической сборки и тестирования каждого изменения в коде. CD (Continuous Delivery/Deployment) – это практика автоматизации процесса доставки (Delivery) или развертывания (Deployment) программного обеспечения в среду (тестовую или продакшен).

Вопрос 16: Какие преимущества дает использование CI/CD?
Ответ: Ускорение цикла разработки и выпуска обновлений, снижение риска ошибок при развертывании, повышение качества кода за счет автоматического тестирования, автоматизация рутинных задач, более частые и предсказуемые релизы.

Вопрос 17: Опишите основные этапы типичного CI/CD-пайплайна.
Ответ: 1) Получение изменений из репозитория. 2) Сборка проекта (build). 3) Запуск автоматических тестов (юнитов, интеграционных). 4) Статический анализ кода (линтеры, анализаторы). 5) Создание артефакта (Docker-образ). 6) Развертывание в тестовую среду. 7) Запуск тестов в тестовой среде. 8) Развертывание в прод (для Continuous Deployment).

Вопрос 18: Что такое Jenkinsfile? Для чего он используется?
Ответ: Jenkinsfile – это текстовый файл, содержащий определение CI/CD-пайплайна в виде кода (Pipeline-as-Code). Он позволяет описать все этапы сборки, тестирования и развертывания в декларативном или скриптовом формате, что делает пайплайн версионируемым и воспроизводимым.

Вопрос 19: Какие инструменты можно использовать для мониторинга CI/CD-пайплайна?
Ответ: Встроенные интерфейсы Jenkins, GitLab CI/CD, CircleCI; инструменты для визуализации метрик, такие как Grafana; системы логирования, например, ELK-стек (Elasticsearch, Logstash, Kibana).

Тема 6: Основы DevOps
Вопрос 20: Что такое DevOps и какие проблемы он решает?
Ответ: DevOps — это культурная философия, практики и инструменты, которые повышают способность организации быстро доставлять приложения и услуги. Он решает проблему разрыва между командами разработки (Dev) и эксплуатации (Ops), автоматизируя процессы сборки, тестирования и развертывания, что приводит к более коротким циклам разработки и повышенной надежности систем.

Вопрос 21: Что такое "Инфраструктура как код" (IaC)? Назовите принципы и инструменты.
Ответ: IaC — это подход к управлению инфраструктурой с помощью конфигурационных файлов, а не вручную. Принципы: идемпотентность, версионирование, повторное использование. Инструменты: Terraform (для provisioning), Ansible, Chef, Puppet (для конфигурационного менеджмента).

Тема 7: Контейнеризация (Docker)
Вопрос 22: В чем преимущества контейнеризации перед классической виртуализацией?
Ответ: Контейнеризация использует ядро хостовой ОС, что делает контейнеры более легковесными и быстрыми при запуске по сравнению с виртуальными машинами, которым требуется полноценная гостовая ОС. Это обеспечивает лучшую плотность размещения и более эффективное использование ресурсов.

Вопрос 23: Что такое Docker-образ и из каких слоев он состоит?
Ответ: Docker-образ — это шаблон только для чтения, используемый для создания контейнеров. Он состоит из нескольких слоев, каждый из которых представляет собой набор изменений файловой системы. Слои кэшируются и используются повторно, что ускоряет сборку образов и уменьшает их объем.

Тема 8: Оркестрация контейнеров (Kubernetes)
Вопрос 24: Каковы основные задачи оркестратора Kubernetes?
Ответ: Основные задачи: 1) Развертывание и управление: запуск приложений в контейнерах, репликация, обновление без простоя. 2) Сервис-дискавери и балансировка нагрузки. 3) Распределение ресурсов между приложениями. 4) Самовосстановление: перезапуск контейнеров при сбоях. 5) Масштабирование: автоматическое или ручное изменение количества реплик.

Вопрос 25: Опишите назначение основных сущностей (resources) в Kubernetes: Pod, Deployment, Service.
Ответ: Pod — наименьшая и простейшая единица в Kubernetes, представляющая один или несколько контейнеров, разделяющих ресурсы. Deployment — декларативный ресурс для управления жизненным циклом Pod'ов (обеспечивает обновления, откаты, репликацию). Service — абстракция, которая определяет логический набор Pod'ов и политику доступа к ним (например, балансировщик нагрузки).

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

Контрольные вопросы	**Перечень компетенций (части компетенции)**
Назовите и охарактеризуйте основные этапы жизненного цикла ПО.	ПК-3.1, LC-4.1
В чем различия между каскадной (Waterfall) и итеративной моделью ЖЦ?	ПК-3.1, LC-4.1

Опишите основные принципы Agile-манифеста.	ПК-3.1, LC-4.1
Что такое Scrum и какие основные роли и артефакты в нем определены?	ПК-3.1, LC-4.1, LC-4.2
Что такое система контроля версий и зачем она нужна в разработке программного обеспечения?	ПК-3.2
Объясните разницу между командами git merge и git rebase.	ПК-3.2
Что такое .gitignore и как он используется?	ПК-3.2
Что такое Pull Request (PR) и зачем он нужен?	ПК-3.2, LC-4.2
Как разрешить конфликт при слиянии веток в Git?	ПК-3.2, LC-4.2
Какие основные уровни тестирования вы знаете?	ПК-3.3
Что такое TDD (Test-Driven Development)?	ПК-3.3
В чем разница между black-box и white-box тестированием?	ПК-3.3
Что такое покрытие кода (code coverage) и зачем оно нужно?	ПК-3.3
Какие инструменты для автоматизации тестирования вы знаете?	ПК-3.3
Что такое CI/CD? Объясните значение каждого термина.	ПК-3.3
Какие преимущества дает использование CI/CD?	ПК-3.3, LC-4.1
Опишите основные этапы типичного CI/CD-пайплайна.	ПК-3.3
Что такое Jenkinsfile? Для чего он используется?	ПК-3.3
Какие инструменты можно использовать для мониторинга CI/CD-пайплайна?	ПК-3.3
Что такое DevOps и какие проблемы он решает?	ПК-3.3, LC-4.2
Что такое "Инфраструктура как код" (IaC)? Назовите принципы и инструменты.	ПК-3.3
В чем преимущества контейнеризации перед классической виртуализацией?	ПК-3.3
Что такое Docker-образ и из каких слоев он состоит?	ПК-3.3
Каковы основные задачи оркестратора Kubernetes?	ПК-3.3
Опишите назначение основных сущностей (resources) в Kubernetes: Pod, Deployment, Service.	ПК-3.3

Пример технического задания к лабораторной работе

Техническое задание на выполнение лабораторной работы № [Номер]

По дисциплине: «Технологии создания и поддержки программного обеспечения» На тему «Автоматизация сборки и развертывания ПО (CI/CD)»

1. Цель работы

Ознакомление с принципами и инструментами CI/CD, приобретение практических навыков автоматизации процессов сборки, тестирования и развертывания программного обеспечения.

2. Задачи работы

- Изучить основные концепции Continuous Integration (CI) и Continuous Delivery/Deployment (CD).
- Освоить работу с системой управления версиями Git.
- Настроить автоматическую сборку проекта при внесении изменений в репозиторий.
- Автоматизировать запуск тестов после сборки проекта.
- Развернуть собранное приложение на тестовый сервер (или локальную виртуальную машину).

3. Исходные данные

- Репозиторий с исходным кодом простого приложения [выдается ссылка на

репозиторий].

- Доступ к платформе CI/CD (например, GitLab или GitHub) [указывается конкретная платформа и необходимые учетные данные, если требуются].
- Тестовый сервер (или локальная виртуальная машина) с установленным необходимым окружением для запуска приложения [указываются характеристики сервера/ВМ: ОС, версия языка программирования, веб-сервер и т.д.].

4. Требования к выполняемой работе

- Создать pipeline CI/CD в выбранной платформе.
- Pipeline должен выполнять следующие этапы:
  - Сборка. Клонирование репозитория, установка зависимостей.
  - Тестирование. Запуск unit-тестов и/или интеграционных тестов (если они есть в проекте).
  - Развертывание. Копирование собранного приложения на тестовый сервер или виртуальную машину.
- Настроить автоматический запуск pipeline при каждом коммите в основную ветку репозитория (например, *main* или *master*).
- Предоставить скриншоты настроек pipeline и логи успешного выполнения всех этапов.

5. Ожидаемые результаты

- Рабочий pipeline CI/CD, автоматически собирающий, тестирующий и развертывающий приложение при внесении изменений в репозиторий.
- Отчет о выполненной работе, содержащий:
  - Краткое описание процесса настройки pipeline.
  - Скриншоты настроек pipeline.
  - Логи успешного выполнения всех этапов pipeline.
  - Выводы о полученном опыте и сложностях, возникших в процессе работы.

6. Дополнительные материалы

- При необходимости.

Документация по выбранной платформе CI/CD: [Ссылка на документацию]
Примеры конфигурационных файлов для pipeline: [Ссылка на примеры]

Преподаватель: [ФИО преподавателя]
Подпись студента: _____(подтверждает получение задания)

Методические рекомендации, определяющие процедуры оценивания выполнения лабораторных работ:

Оценивание выполнения лабораторных работ осуществляется с целью контроля освоения студентами практических навыков и теоретических знаний по дисциплине «Технологии создания и поддержки программного обеспечения». Задание считается выполненным при выполнении следующих условий:

1. Реализованный функционал соответствует требованиям, указанным в задании к лабораторной работе. Проверяется корректность работы программы / скрипта / конфигурации в соответствии с поставленной целью.

2. Код должен быть читаемым, хорошо структурированным и соответствовать общепринятым стандартам кодирования для используемого языка программирования.

Конфигурационные файлы должны быть корректно оформлены и содержать необходимые параметры.

3. Наличие краткого отчета о выполненной работе, включающего:
   - описание цели лабораторной работы;
   - краткое описание реализованного решения;
   - инструкцию по запуску и использованию разработанного продукта;
   - скриншоты, демонстрирующие работу программы/скрипта/конфигурации.

4. Лабораторная работа должна быть выполнена и представлена к проверке в установленные сроки. Задержка с представлением работы влечет снижение оценки.

5. Работа должна быть выполнена студентом самостоятельно. Плагиат или использование готовых решений без понимания принципов их работы недопустимы и приводят к нулевой оценке. При использовании сторонних библиотек/инструментов необходимо указать источники.

6. Наличие тестов, покрывающих основные сценарии использования разработанного функционала (для работ по тестированию). Проверка работоспособности тестов и их соответствие требованиям задания.

Критерии оценивания лабораторных работ:

- Отлично (5 баллов): все условия выполнены в полном объеме. Работа выполнена качественно, с соблюдением всех требований и стандартов. Демонстрируется глубокое понимание материала.
- Хорошо (4 балла): большинство условий выполнены. Незначительные недостатки не влияют на общую работоспособность решения. Проявляется хорошее понимание материала.
- Удовлетворительно (3 балла): выполнены основные условия, но имеются существенные недостатки в функциональности или качестве кода/конфигурации. Понимание материала поверхностное.
- Неудовлетворительно (2 балла и ниже): не выполнено большинство условий. Работа не соответствует требованиям задания. Продемонстрировано недостаточное понимание материала.

Процедура оценивания лабораторных работ:

1. Преподаватель проверяет представленную студентом работу на соответствие вышеперечисленным условиям.

2. В случае необходимости, преподаватель может запросить у студента демонстрацию работы программы/скрипта/конфигурации и ответы на вопросы по реализованному решению.

3. Оценка выставляется на основании комплексной оценки выполненной работы с учетом всех критериев.

Методические рекомендации, определяющие процедуры оценивания на зачете: Зачет по дисциплине «Технологии создания и поддержки программного обеспечения» проводится сцелью комплексной оценки освоения студентами теоретических знаний и практических навыков, полученных в процессе изучения курса. Оценка на зачете формируется из нескольких компонентов: выполнения лабораторных работ, решения теоретических вопросов и демонстрации проектной работы.

Компоненты оценки зачета:

1. Оценка по лабораторным работам (35% от общей оценки): учитывается средняя оценка, полученная студентом за все выполненные лабораторные работы в течение семестра. Оцениваются соответствие выполненных работ требованиям

заданий, качество кода/конфигураций, документирование и соблюдение сроков выполнения.

2. Теоретический опрос (35% от общей оценки): проводится в форме устного опроса или письменного теста (по решению преподавателя). Охватывает ключевые понятия, принципы и технологии, изученные в рамках курса. Вопросы направлены на проверку понимания теоретических основ разработки программного обеспечения, процессов CI/CD, DevOps, контейнеризации и оркестрации.

3. Демонстрация проектной работы (30% от общей оценки - *опционально*): в рамках курса предусмотрена разработка мини-проекта, студент должен продемонстрировать его работоспособность и объяснить принятые архитектурные решения. Оценивается соответствие проекта требованиям задания, качество кода, документирование и способность студента ответить на вопросы по проекту.

Критерии оценивания – зачет:

«Зачтено» с оценкой «отлично» (5 баллов): студент демонстрирует глубокое понимание теоретического материала, уверенно отвечает на вопросы, успешно выполнил все лабораторные работы с высоким качеством и (при наличии) представил работоспособный проект, соответствующий требованиям.

«Зачтено» с оценкой «хорошо» (4 балла): студент хорошо знает теоретический материал, может ответить на большинство вопросов, успешно выполнил все лабораторные работы с незначительными замечаниями и (при наличии) представил работоспособный проект, в котором есть небольшие недостатки.

«Зачтено» с оценкой «удовлетворительно» (3 балла): студент владеет основными понятиями и принципами дисциплины, может ответить на базовые вопросы, выполнил все лабораторные работы с существенными замечаниями или пропустил несколько работ и (при наличии) представил проект, требующий значительной доработки.

«Не зачтено» с оценкой «неудовлетворительно» (2 балла и ниже): студент демонстрирует недостаточное понимание теоретического материала, не может ответить на большинство вопросов, имеет большое количество непроверенных или неудовлетворительно выполненных или не выполненных лабораторных работ, не представил проект или представил неработоспособный проект.

Процедура оценивания – зачет

1. Студент допускается к зачету при условии успешного выполнения всех лабораторных работ (получение оценки не ниже удовлетворительной по каждой работе).

2. Теоретический опрос проводится в индивидуальном порядке или в форме группового опроса (по решению преподавателя).

3. При демонстрации проектной работы студент должен предоставить доступ к коду проекта и продемонстрировать его работоспособность.

4. Окончательная оценка на зачете формируется путем суммирования оценок по всем компонентам с учетом их весовых коэффициентов.

5. Решение о допуске студента к зачету принимается преподавателем на основании комплексной оценки всех компонентов.

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 зачете;

– 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.

5. Перечень учебной литературы, информационных ресурсов и технологий

5.1. Основные учебники и руководства

1. Гаврилов, М. В. Информатика и информационные технологии : учебник для вузов / М. В. Гаврилов, В. А. Климов. — 6-е изд., перераб. и доп. — Москва : Юрайт, 2025. — 318 с. — URL: https://urait.ru/bcode/581419 (дата обращения: 20.06.2025). — Режим доступа: для авториз. пользователей. — ISBN 978-5-534-20354-7. — Текст : электронный. URL: http://212.192.134.46/MegaPro/UserEntry?Action=Link_FindDoc&id=157708&idb=0

2. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Юрайт, 2025. — 432 с. — URL: https://urait.ru/bcode/561885 (дата обращения: 23.07.2025). — Режим доступа: для авториз. пользователей. — ISBN 978-5-534-07604-2. — Текст : электронный. URL: http://212.192.134.46/MegaPro/UserEntry?Action=Link_FindDoc&id=144867&idb=0

3. Гниденко, И. Г. Технологии и методы программирования : учебник для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Юрайт, 2025. — 241 с. — URL: https://urait.ru/bcode/581329 (дата обращения: 24.06.2025). — Режим доступа: для авториз. пользователей. — ISBN 978-5-534-18130-2. — Текст : электронный. URL: http://212.192.134.46/MegaPro/UserEntry?Action=Link_FindDoc&id=144857&idb=0

5.2 Методические пособия и практикумы

4. Постолит, Анатолий Владимирович. Основы искусственного интеллекта в примерах на Python : самоучитель / Анатолий Постолит. — 2-е изд., перераб. и доп. — Санкт-Петербург : БХВ-Петербург, 2024. — 446 с. : ил. — (Самоучитель). — Библиогр.: с. 440-443. — ISBN 978-5-9775-1818-5. — Текст : непосредственный. URL: http://212.192.134.46/MegaPro/UserEntry?Action=Link_FindDoc&id=276189&idb=0

5. Ростовцев, В. С. Искусственные нейронные сети : учебник для вузов / В. С. Ростовцев. — 5-е изд., стер. — Санкт-Петербург : Лань, 2025. — 216 с. — URL: https://e.lanbook.com/book/447392 (дата обращения: 07.11.2025). — Режим доступа: для авториз. пользователей. — ISBN 978-5-507-50568-5. — Текст : электронный. URL: http://212.192.134.46/MegaPro/UserEntry?Action=Link_FindDoc&id=282243&idb=0

6. Шапиро, Л. Компьютерное зрение : учебное пособие / Л. Шапиро, Д. Стокман. — 5-е изд. — Москва : Лаборатория знаний, 2024. — 763 с. — Режим

доступа: для авториз. пользователей. — URL: https://e.lanbook.com/book/417998 (дата обращения: 17.11.2025). — ISBN 978-5-93208-725-1. — Текст : электронный. URL: http://212.192.134.46/MegaPro/UserEntry?Action=Link_FindDoc&id=282317&idb=0

5.3 Дополнительные материалы и стандарты

7. ISO/IEC/IEEE 12207:2017 Системная и программная инженерия. Процессы жизненного цикла программных средств.
8. Рудаков М.В. DevOps: принципы, методики, инструменты. — М.: ДМК Пресс,2023.

5.4 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/

Электронный архив документов КубГУ http://docspace.kubsu.ru/

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

Настоящие методические указания предназначены для студентов, изучающих дисциплину «Технологии создания и поддержки программного обеспечения». Они содержат рекомендации по организации учебной деятельности, освоению теоретического материала, выполнению практических работ и подготовке к зачету.

6.1. Цели и задачи дисциплины

Целью изучения дисциплины является формирование у студентов системы знаний о современных технологиях разработки, тестирования, сборки, развертывания и поддержки программного обеспечения.

Задачи дисциплины:

- Изучение основных принципов и практик DevOps.
- Освоение инструментов автоматизации сборки и развертывания (CI/CD).
- Приобретение навыков работы с системами контроля версий (Git).
- Изучение технологий контейнеризации (Docker) и оркестрации (Kubernetes).
- Формирование понимания важности тестирования программного обеспечения и освоение инструментов автоматизированного тестирования.
- Ознакомление с методами мониторинга и логирования приложений.

6.2. Организация учебной деятельности

Дисциплина включает в себя лекционные занятия, практические работы (лабораторные), самостоятельную работу студентов и выполнение мини-проектной работы. **Лекции.** На лекциях рассматриваются теоретические основы дисциплины, ключевые понятия и технологии. Студентам рекомендуется вести конспекты, активно участвовать в обсуждениях и задавать вопросы преподавателю.

Лекция представляет собой систематическое изложение учебного материала, обеспечивающее целостное представление предмета на основе новейших научных данных. Ее цель – организация познавательной деятельности студентов по освоению программного материала дисциплины.

Основные задачи лекции: формирование системы знаний, развитие умения аргументированно излагать научный материал, расширение профессионального кругозора и освещение актуальных достижений науки.

Для успешной подготовки к лекциям необходимо изучить основную литературу

по теме и обратить внимание на вопросы для рассмотрения, предложенные в конце каждой темы. Усвоение материала будет наиболее эффективным при следующих условиях:

1. Систематическая работа на занятиях и самостоятельное закрепление знаний.
2. Добросовестное выполнение заданий на практических занятиях.
3. Анализ предпосылок, умозаключений и выводов учебного курса, а также взаимосвязей между его разделами.
4. Сопоставление различных точек зрения по рассматриваемым проблемам и критическая оценка научной литературы.
5. Разработка предложений по совершенствованию учебного процесса."

Лабораторные работы. Лабораторные работы направлены на закрепление теоретических знаний, приобретение практических навыков работы с инструментами разработки и эксплуатации программного обеспечения, а также формирование компетенций, полученных в ходе лекционных занятий и самостоятельной работы. К каждому лабораторному занятию преподаватель разрабатывает практические задания, определяет требования к их выполнению и предоставляет методические рекомендации, содержащиеся в фонде оценочных средств учебной дисциплины. В процессе подготовки к занятиям студент осуществляет сбор и анализ информации по тематике работы, используя открытые источники (научные публикации, аналитические материалы, ресурсы сети Интернет и др.), а также практический опыт и доступные данные об объекте исследования. Контроль за выполнением самостоятельной работы осуществляется в ходе изучения каждой темы дисциплины на лабораторных занятиях.

Самостоятельная работа. Самостоятельная работа студентов по дисциплине направлена на закрепление и систематизацию теоретических знаний, формирование практических навыков их применения при решении задач, а также углубленное изучение отдельных тем курса. Она включает в себя изучение основной и дополнительной литературы, подготовку к лекциям и лабораторным работам, выполнение домашних заданий и самостоятельный поиск информации с использованием ресурсов библиотеки, онлайн-курсов и документации по используемым инструментам.

Подготовка к каждому разделу дисциплины осуществляется поэтапно:

Этап 1. Изучение теоретического материала на основе лекций преподавателя, рекомендуемой основной литературы и научных публикаций для овладения ключевыми понятиями и формирования понимания аналитического инструментария в данной области.

Этап 2. Выполнение лабораторных работ, направленных на формирование практических умений и навыков в рамках заявленных компетенций. Этот этап предполагает самостоятельный поиск эмпирических данных, их обобщение и анализ по схеме,
рекомендованной преподавателем, а также формулирование выводов.

Контролируемая самостоятельная работа (КСР) представляет собой совокупность заданий, выполняемых студентом под руководством и контролем преподавателя с целью закрепления теоретических знаний, формирования умений и навыков, а также накопления практического опыта. Студенты руководствуются методическими указаниями преподавателя и инструкциями по выполнению типовых заданий.

Текущий контроль КСР осуществляется еженедельно в соответствии с программой дисциплины. Задания для самостоятельной работы и требования к их выполнению предоставляются преподавателем на основе фонда оценочных средств по дисциплине.

Для студентов-инвалидов и лиц с ограниченными возможностями здоровья важным элементом освоения дисциплины являются индивидуальные консультации, обеспечивающие дополнительное разъяснение учебного материала и способствующие индивидуализации обучения. Консультации также поддерживают воспитательный контакт между преподавателем и обучающимся.

6.3. Рекомендации по изучению теоретического материала

- Начните с изучения основных понятий и терминов.
- Используйте различные источники информации: учебники, статьи, онлайн- документацию.
- Старайтесь понять взаимосвязь между различными концепциями и технологиями.
- Приводите примеры из реальной практики для лучшего усвоения материала.
- Регулярно повторяйте изученный материал.

6.4. Рекомендации по выполнению лабораторных работ

- Внимательно читайте условия задания перед началом выполнения работы.
- Разбейте задачу на более мелкие подзадачи и решайте их последовательно.
- Используйте систему контроля версий (Git) для отслеживания изменений в коде.
- Пишите чистый, хорошо документированный код.
- Тестируйте свою работу после каждого этапа разработки.
- Соблюдайте сроки выполнения лабораторных работ.

6.5. Рекомендации для самостоятельного изучения

- Используйте учебники из раздела 5.
- Дополнительно ознакомьтесь с содержанием онлайн-курсов на таких платформах как: Coursera, Udemy, Stepik.
- Используйте официальную документацию к используемым инструментам и технологиям.
- Читайте популярные источники такие как профильные блоги и статьи: Medium, Habr и т.д.

6.6. Подготовка к зачету

- Повторите все темы, изученные в течение семестра.
- Просмотрите конспекты лекций и отчеты по лабораторным работам.
- Решите тестовые задания и задачи для самоконтроля.
- Подготовьте ответы на возможные вопросы преподавателя.
- При необходимости обратитесь к преподавателю за консультацией.

6.7. Дополнительные рекомендации

- Активно участвуйте в обсуждениях на лекциях и семинарах.
- Задавайте вопросы преподавателю, если что-то непонятно.
- Работайте в команде с другими студентами для обмена опытом и знаниями.
- Используйте возможности онлайн-ресурсов для самостоятельного изучения материала.
- Экспериментируйте и пробуйте новые технологии.

Кейс 1: Оптимизация кредитного скоринга для физических лиц в Сбербанке

Контекст

Сбербанк – крупнейший банк России, который активно использует данные для улучшения качества кредитных продуктов. Одной из ключевых задач является оптимизация кредитного скоринга – процесса оценки кредитоспособности клиентов. Некачественный скоринг может привести к увеличению рисков невозврата кредитов или, наоборот, к отказу надежным клиентам.

Цель кейса

Разработать теоретическую модель оптимизации кредитного скоринга для физических лиц, используя данные о клиентах и их кредитной истории.

1. Анализ данных

Задача: Определить, какие данные необходимы для построения модели скоринга. Ход выполнения:

- Перечислить возможные источники данных (например, кредитная история, доходы, возраст, семейное положение, регион проживания).
- Описать, какие переменные могут быть значимыми для оценки кредитоспособности.
- Обсудить, какие данные могут быть избыточными или нерелевантными.

Ожидаемый результат: Список ключевых переменных для модели скоринга с обоснованием их важности.

2. Сегментация клиентов

Задача: Разделить клиентов на группы по уровню риска. Ход выполнения:

- Предложить критерии сегментации (например, возраст, уровень дохода, наличие просрочек).
- Описать, как можно классифицировать клиентов на группы: низкий, средний и высокий риск.
- Обсудить, какие факторы могут влиять на переход клиента из одной группы в другую.

Ожидаемый результат: Описание сегментов клиентов и критериев их распределения.

3. Разработка модели скоринга

Задача: Теоретически обосновать подход к построению модели кредитного скоринга. Ход выполнения:

- Описать, какие методы анализа данных могут быть использованы (например, логистическая регрессия, деревья решений, кластерный анализ).
- Объяснить, как можно оценить качество модели (например, точность, полнота, AUC-ROC).
- Обсудить, какие метрики важны для банка (например, минимизация потерь от невозвратов, увеличение одобрений надежным клиентам).

Ожидаемый результат: Описание подхода к построению модели скоринга с обоснованием выбора методов и метрик.

4. Интерпретация результатов

Задача: Объяснить, как результаты модели могут быть использованы для принятия бизнес-решений. Ход выполнения:

- Описать, как банк может использовать модель для оптимизации кредитных продуктов.
- Обсудить, какие изменения в бизнес-процессах могут быть внедрены на основе результатов анализа.
- Предложить, как можно улучшить модель в будущем (например, добавление новых данных, регулярное обновление).

Ожидаемый результат: Рекомендации по использованию модели для улучшения кредитных процессов в банке.

Вопросы для проверки понимания

1. Какие данные, по вашему мнению, являются наиболее важными для оценки кредитоспособности клиента? Почему?
2. Как сегментация клиентов помогает банку снизить риски?
3. Какие метрики качества модели скоринга вы считаете наиболее важными для банка?

Почему?

4. Какие бизнес-решения может принять банк на основе результатов модели скоринга?
5. Какие риски могут возникнуть при использовании модели скоринга, и как их можно минимизировать?

Кейс 2: Оптимизация управления запасами строительных материалов

Контекст

Холдинг AVA Group – один из крупнейших застройщиков Краснодарского края, ежегодно реализующий десятки жилых и коммерческих проектов. Одной из ключевых проблем компании является неэффективное управление запасами строительных материалов, что приводит к:

- Перерасходу бюджета на хранение избыточных материалов.
- Задержкам в строительстве из-за нехватки критически важных ресурсов.
- Сложностям в прогнозировании спроса на материалы в зависимости от сезона, региона и типа проекта.

Компания AVALAB (IT-подразделение AVA Group) разрабатывает BI-платформу FastBoard, которая уже используется для визуализации данных. Однако для решения задачи оптимизации запасов требуется применение методов Data Science: анализ исторических данных, построение прогнозных моделей и интеграция результатов в дашборды FastBoard.

Теоретическая часть: Постановка задачи

1. Цели проекта

- Снизить издержки на хранение материалов на 15–20% за счет точного прогнозирования потребности.
- Уменьшить количество задержек в строительстве из-за нехватки материалов на 25%.
- Автоматизировать процесс формирования заказов на пополнение запасов.

2. Источники данных

Для решения задачи доступны следующие данные:

- Исторические данные о закупках и расходе материалов (за 3 года) по каждому объекту строительства.
- Данные о сезонах строительства (климатические условия, пиковые периоды).
- Информация о поставщиках (время доставки, надежность, цены).
- Планы строительства (графики работ, типы объектов).

3. Методы Data Science для решения задачи

Методы Data Science и их применение:

- ***Exploratory Data Analysis (EDA)*** - анализ распределения расхода материалов, выявление аномалий и сезонных трендов.
- ***Временные ряды (ARIMA, Prophet)*** - прогнозирование спроса на материалы с учетом сезонности.
- ***Кластеризация (K-Means, DBSCAN)*** - группировка объектов строительства по типу потребления материалов.
- ***Оптимизация запасов (EOQ, Safety Stock)*** - расчет оптимального уровня запасов для минимизации издержек.
- ***Машинное обучение (XGBoost, Random Forest)*** - построение модели для прогнозирования дефицита или избытка материалов.

4. Интеграция с FastBoard

Результаты анализа должны быть визуализированы в дашбордах FastBoard для:

- Мониторинга текущих запасов в реальном времени.

• Отображения прогнозов спроса и рекомендаций по закупкам.
• Аналитики эффективности поставщиков.

Практическая часть: Решение кейса

Шаг 1: Подготовка данных

Задача студента: Очистить и объединить данные из разных источников (Excel, SQL, API поставщиков) в единый датасет. Провести EDA: построить графики расхода материалов по времени, выявить выбросы и сезонные тренды.

Инструменты:

• Python (Pandas, Matplotlib, Seaborn).
• SQL для извлечения данных из базы FastBoard.

Шаг 2: Прогнозирование спроса

Задача студента: Построить модель временных рядов (например, Prophet) для прогнозирования ежемесячного расхода материалов по каждому объекту. Оценить точность модели с помощью метрик MAE, RMSE.

Шаг 3: Оптимизация запасов

Задача студента: Рассчитать оптимальный уровень запасов (EOQ) и страховой запас (Safety Stock) для каждого материала. Использовать формулы:

• EOQ = sqrt((2 * D * S) / H), где D – годовой спрос, S – стоимость заказа, H – стоимость хранения.
• Safety Stock = Z * σ * sqrt(L), где Z – коэффициент сервиса, σ – стандартное отклонение спроса, L – время доставки.

Шаг 4: Визуализация в FastBoard

Задача студента: Создать дашборд в FastBoard с следующими виджетами:

1. График фактического и прогнозного расхода материалов.
2. Таблица с рекомендациями по закупкам (что, когда и сколько заказывать).
3. Карта объектов строительства с цветовой индикацией уровня запасов.

Инструменты:

• FastBoard (drag-and-drop конструктор дашбордов).
• Интеграция с Python через REST API для автоматического обновления данных.

Вопросы для обсуждения

1. Какие еще методы Data Science можно применить для оптимизации логистики в строительстве?
2. Как интеграция с FastBoard упрощает принятие решений менеджерами?
3. Какие риски могут возникнуть при внедрении подобных систем, и как их минимизировать?

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-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)	Браузер Google Chrome, Jupyter Notebook 6.3.0 и выше (язык Python с библиотеками Numpy, Pandas, gensim, NLTK, PyMorphy, фреймворком PyTorch)

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.

№	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

Дополнительные облачные ресурсы предоставляются технологическим партнером Yandex Cloud.

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.