МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКІЙ ГОСУДАРСТВЕННЫЙ УНІВЕРСІТЕТ»
Факультет компьютерных технологий и прикладной математики
Центр искусственного интеллекта

УТВЕРЖДАЮ:
Проректор по учебной работе,
качеству образования –
первый проректор
_____ Хагуров Т.А.
подпись
«29» мая 2026

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1.О.30 СОВРЕМЕННЫЕ СРЕДЫ РАЗРАБОТКІ ІНТЕРАКТІВНЫХ ПРІЛОЖЕНІЙ

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных

Направленность: Искусственный интеллект и аналитика данных

Форма обучения очная

Квалификация бакалавр

Краснодар 2026

Рабочая программа дисциплины «Современные среды разработки интерактивных приложений» 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 составили:
А.В. Коваленко, заведующий КАДИИ,
д-р.тех.н., профессор

подпись

А.В. Щеголев
Ведущий инженер-разработчик
ООО «Кар Икс Технолоджис»

подпись

Е.Н. Калайдин, доктор физ.-мат.наук,
профессор

подпись

Г.В. Калайдина, канд. физ.-мат.наук,
доцент

подпись

Рабочая программа дисциплины утверждена на заседании центра искусственного интеллекта
протокол № 4 «14» мая 2026 г.
Руководитель центра ИИ Коваленко А.В.

подпись

Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики
протокол № 3 «15» мая 2026 г.
Председатель УМК факультета Добровольская Н.Ю.

подпись

Экспертиза:
Экспертная группа индустриального партнера ООО «CarX Technologies»: Фокин Д.Н., Генеральный директор, ceo@carx-tech.com; Щеголев А.В., Ведущий инженер-разработчик, a.shchegolev@carx-tech.com; Болоцкий Д.Д., DevOps-инженер, d.bolotskiy@carx-tech.com.

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 ФГБОУ ВО «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

Освоение современных интегрированных сред и фреймворков для создания интерактивных приложений (десктопных, мобильных, игровых, симуляционных) с акцентом на архитектурные паттерны, управление состоянием и пользовательский интерфейс, включая применение профессионального игрового движка Unity на примере реальных проектов лидера гоночных симуляторов – компании CarX TECHNOLOGIES (МИП).

(МИП – материалы индустриального партнера интегрированные в РПД)

1.2 Задачи дисциплины

1. Изучить принципы RAD-методологии (прототипирование, итеративность, визуальное проектирование) на примере инструментов: WinForms, .NET MAUI, Avalonia.
2. Освоить событийную модель Windows Forms и паттерн MVVM в XAML-фреймворках, научиться сравнивать их эффективность для разных типов приложений.
3. Сформировать умение реализовывать асинхронное взаимодействие с пользовательским интерфейсом (async/await, CancellationToken, ProgressBar) для длительных операций, включая вызов ML-моделей.
4. Научиться управлять состоянием приложения через сериализацию (JSON/XML) и паттерны Repository, Unit of Work с БД SQLite.
5. Развить навыки интеграции классических алгоритмов машинного обучения (ML.NET, ONNX) в интерактивные приложения без блокировки UI-потока.
6. Освоить архитектуру и ключевые механизмы разработки интерактивных 3D-приложений в среде Unity на языке C# (МИП).
7. Сформировать практические навыки реализации масштабируемых, оптимизированных игровых решений с использованием компонентного подхода, событийной модели, сериализации и асинхронных операций на основе кейсов индустриального партнёра CarX TECHNOLOGIES (МИП).

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Современные среды разработки интерактивных приложений» относится к обязательной части Блока 1 "Дисциплины (модули)" учебного плана. В соответствии с рабочим учебным планом дисциплина изучается на 2. курсе по очной форме обучения. Вид промежуточной аттестации: зачет.

Пререквизиты: (Формируют инфраструктуру для освоения базовых уровней компетенций дисциплины) Основы программирования, Алгоритмы и структуры данных, Объектно-ориентированное программирование, Инструментальные средства разработки ИИ.

Постреквизиты (***не содержат развития компетенций на следующие уровни: средний, продвинутый):*** Разработка мобильных приложений (прямое развитие MAUI), Технологии тестирования ПО (контроль качества создаваемых приложений), Машинное обучение / Глубокое обучение (углубление ML-части), Разработка ИИ агентов / Методы обучения с подкреплением (использование Unity), Высоконагруженные приложения (развитие тем асинхронности и оптимизации).

1.4 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:

Код и наименование индикатора достижения компетенции	Результаты обучения по дисциплине
ОПК-2. Способен применять современный математический аппарат, связанный с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности	
	Знает: методы системного анализа предметной (проблемной) области для интерактивных приложений; способы

Код и наименование индикатора достижения компетенции	Результаты обучения по дисциплине
ОПК-2.1 Способен применять системный подход к анализу предметной (проблемной) области, выявлению требований к ИС.	выявления, классификации и документирования требований к ИС с использованием RAD-сред; архитектуру игрового движка Unity: жизненный цикл объектов, событийную модель, систему компонентов, персистентность данных (МИП).
	Умеет: применять системный подход для декомпозиции требований в архитектурные элементы (UI, данные, логика) с выбором целесообразной среды разработки (WinForms, MAUI, Avalonia, Unity).
	Владеет: навыками построения моделей требований (use case, прототипы интерфейсов) в средах визуального проектирования .NET и Unity; навыками работы со сценами, префабами, компонентами Transform и Camera в Unity (МИП).
ОПК-2.2 Применяет современный математический аппарат при построении моделей в различных областях человеческой деятельности.	Знает: современный математический аппарат: конечные автоматы, алгоритмы сериализации (JSON/XML), методы асинхронного программирования, основы линейной алгебры для 3D-преобразований; паттерны проектирования в Unity (Object Pool, ScriptableObject), методы сериализации состояния (PlayerPrefs, JSON), асинхронную загрузку сцен (МИП).
	Умеет: применять математические модели для управления состоянием, физикой, оптимизации; реализовывать систему сохранения с миграцией данных, управлять физикой и событиями коллизий в Unity (МИП).
	Владеет: методами количественной оценки качества (латентность, throughput); навыками оптимизации памяти (пул объектов), сборки кроссплатформенного приложения в Unity (МИП).
PL-3 Способен применять языки программирования платформы .NET для решения задач в области ИИ	
PL-3.2 – Разрабатывает и отлаживает решения для задач МО и ИИ на языке программирования C#	Знает: способы интеграции ML.NET в RAD-проекты, вызов ONNX-моделей из UI-событий.
	Умеет: встраивать предсказание (классификация/регрессия) в кроссплатформенное приложение, обрабатывать результаты модели асинхронно.
	Владеет: приемами привязки данных к выходам ML-модели в XAML.
ML-3 Способен применять классические алгоритмы машинного обучения с пониманием их математических основ и областей применения	
ML-3.1 – Обосновывает способы и варианты применения классических методов и моделей МО в задачах ИИ, включая их математическое преобразование и адаптацию	Знает: типовые задачи (классификация изображений, анализ текста), решаемые в RAD-приложениях, ограничения моделей в реальном времени.
	Умеет: подбирать предобученную модель для интеграции в десктоп/мобильное приложение, описывать математическое преобразование входных данных.
	Владеет: методами предобработки данных внутри UI-событийного цикла.
ML-3.3 – Оценивает результативность применения классических методов и моделей МО в задачах ИИ на основе сопоставления с аналогами	Знает: метрики качества (точность, latency) при инференсе модели в GUI.
	Умеет: сравнивать производительность модели в WinForms vs MAUI, измерять время отклика интерфейса.
	Владеет: построением простых A/B-тестов внутри лабораторных приложений.

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.

Примечание: достижение результатов обучения, отмеченных (МИП), обеспечивается за счёт освоения разделов дисциплины, реализованных с участием индустриального партнёра – компании CarX TECHNOLOGIES.

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 2 зачетных единиц (72 часа), их распределение по видам работ представлено в таблице

Виды работ		Всего часов	Форма обучения
			очная
			4 семестр (часы)
Контактная работа, в том числе:		**50,2**	**50,2**
Аудиторные занятия (всего):		**48**	**48**
занятия лекционного типа		16	16
лабораторные занятия		32	32
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:		**21.8**	**21.8**
Самостоятельное изучение разделов, самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к лабораторным и практическим занятиям, коллоквиумам и т.д.)		21,8	21,8
Контроль:			
Подготовка к экзамену			
Общая трудоемкость	**час.**	**72**	**72**
	в том числе контактная работа	**50,2**	**50,2**
	зач. ед	**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.	RAD-методология и событийное программирование (WinForms, SQLite, сериализация)	17,8	4		8	5,8
2.	Архитектуры MVVM/MVP и кроссплатформенная разработка (MAUI/Avalonia, XAML, привязки)	12	4		4	4
3.	Асинхронность, управление состоянием и интеграция ИИ (ML.NET, ONNX, фоновые задачи)	16	4		8	4
4.	Разработка интерактивных приложений в среде Unity (с использованием материалов индустриального партнёра CarX TECHNOLOGIES) (МИП)	22	4		12	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

2.3 Содержание разделов (тем) дисциплины

2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Соответствие индикаторам компетенций	Форма текущего контроля Пояснение
Раздел 1. RAD-методология и событийное программирование (WinForms, SQLite, сериализация)				
1.	Введение в RAD-методологию	Принципы быстрой разработки (прототипирование, итеративность, визуальное проектирование, повторное использование компонентов). Примеры RAD-сред: Delphi, Visual Studio (WinForms), Qt Designer, Access. Роль RAD в создании интерактивных приложений.	ОПК-2.1	*Дискуссия* Применение современных RAD-сред для определения структуры приложений.
2.	Windows Forms: событийная модель и визуальный конструктор	Событийная модель .NET, визуальный конструктор, элементы управления (Button, TextBox, DataGridView, ListBox, ComboBox). Структура проекта WinForms, жизненный цикл формы. Обработчики событий.	ОПК-2.1, ОПК-2.2	*Опрос* Использование визуального конструктора и событий для разработки GUI.
Раздел 2. Архитектуры MVVM/MVP и кроссплатформенная разработка (MAUI/Avalonia, XAML, привязки)				
3.	Архитектура GUI-приложений	Сравнение классической событийной модели (WinForms) с паттернами MVVM (WPF, MAUI) и MVP. Преимущества и недостатки. Применимость в задачах с интенсивным обновлением интерфейса.	ОПК-2.1, ОПК-2.2	*Опрос* Выбор архитектур (MVVM/MVP) под требования задачи.
4.	.NET MAUI / Avalonia	XAML-разметка, принцип привязки данных (Binding), навигация, жизненный цикл кроссплатформенного приложения. Сравнение с WinForms по сложности разработки, гибкости и производительности.	ОПК-2.1, ОПК-2.2	*Опрос* Освоение кроссплатформенных сред – расширение инструментария.

Раздел 3. Асинхронность, управление состоянием и интеграция ИИ (ML.NET, ONNX, фоновые задачи)				
5.	Асинхронное программирование в RAD	async/await, BackgroundWorker, Task.Run. Избегание блокировки UI-потока. Реализация длительных операций (в т.ч. вызов ML-моделей) с сохранением отклика интерфейса.	ОПК-2.2	*Опрос* Обоснование асинхронности для неблокирующего вызова ML-моделей.
6.	Управление состоянием и сериализация	Сохранение настроек и данных: JSON, XML, реестр Windows, файловая система. Паттерны Unit of Work, Repository. Применение для кэширования моделей ИИ и истории взаимодействий.	ОПК-2.2	*Опрос* Паттерны Repository, Unit of Work – методы проектирования.
7.	Интеграция искусственного интеллекта в RAD-приложения	Вызов ML.NET из UI-событий, загрузка ONNX-моделей, асинхронный инференс. Отображение результатов (уверенность, класс). Обработка ошибок и падение производительности.	PL-3.2, ML-3.1	*Опрос* Разработка на C# вызовов ML.NET/ONNX, обоснование применения.
8.	Инструментальные средства разработки ИИ для RAD	Обзор ML.NET Model Builder, ONNX Runtime, Hugging Face API. Подготовка входных данных (преобразование изображений, текста) в C#. Примеры: классификация изображений, анализ тональности.	ML-3.1, ML-3.3	*Опрос* Обзор средств, выбор модели, оценка применимости в GUI.
Раздел 4. Разработка интерактивных приложений в среде Unity				
9.	Архитектурные основы разработки приложений в среде Unity: управление жизненным циклом объектов, событийная модель и персистентность данных (МИП)	Введение в экосистему Unity: компонентно-ориентированная архитектура, иерархия GameObject-Component. Жизненный цикл объектов: методы Awake, Start, Update, FixedUpdate, LateUpdate, OnDestroy. Событийная модель: обработка ввода (Input), триггеры и коллизии (OnTriggerEnter, OnCollisionEnter), пользовательские события и делегаты. Персистентность данных: PlayerPrefs (простые типы), сериализация через JSON (JsonUtility), сохранение и загрузка состояния приложения. Особенности разработки высокопроизводительных игр на примере проектов CarX TECHNOLOGIES (дрифт-симуляторы).	ОПК-2.1, ОПК-2.2	Дискуссия, опрос
10	Методы проектирования масштабируемых и оптимизированных приложений: шаблонное проектирование, оптимизация, сериализация состояния и финализация продукта (МИП)	Шаблоны проектирования в Unity: Object Pool (пул объектов), ScriptableObject (отделение данных от поведения), Singleton, Event Bus. Оптимизация производительности: профилирование (Profiler), снижение Draw Calls, LOD, управление памятью, избегание аллокаций в Update. Сериализация состояния: кастомные решения для миграции данных (версионирование), работа с JSON и бинарными файлами, применение Application.persistentDataPath.	ОПК-2.1, ОПК-2.2	Опрос, анализ кейсов

		Финализация продукта: настройка сборки (Build Settings) под Windows, Android; тестирование, логирование, создание инсталлятора. Реальные практики компании CarX TECHNOLOGIES (CarX Drift Racing, CarX Street) по оптимизации графики и физики на мобильных платформах.		

2.3.2 Лабораторные занятия

№	Наименование раздела (темы)	Содержание раздела (темы)	**Соответствие индикаторам компетенций**
Раздел 1. RAD-методология и событийное программирование (WinForms, SQLite, сериализация)			
1.	Простое приложение на Windows Forms	Создание проекта WinForms. Размещение Button, TextBox, ListBox, ComboBox. Написание обработчиков событий. Валидация ввода. Реализация простого калькулятора или записной книжки.	ОПК-2.1, ОПК-2.2
2.	DataGridView + SQLite + сериализация (часть 1)	Подключение SQLite к проекту, создание таблицы. Отображение данных в DataGridView. Реализация добавления/удаления записей.	ОПК-2.2
3.	DataGridView + SQLite + сериализация (часть 2)	Сериализация таблицы в XML и JSON. Восстановление данных из файлов. Паттерн Repository для доступа к данным.	ОПК-2.2
4.	Работа с элементами управления и событиями (закрепление)	Самостоятельное мини-задание: «Каталог книг» с поиском и фильтрацией, без БД (List<T>, BindingSource).	ОПК-2.1
Раздел 2. Архитектуры MVVM/MVP и кроссплатформенная разработка (MAUI/Avalonia, XAML, привязки)			
5.	Сравнительная лабораторная: приложение «Заметки» на WinForms (часть 1)	Разработка минимального приложения заметок на WinForms (событийная модель). Сохранение/загрузка через JSON.	ОПК-2.2
6.	Сравнительная лабораторная: приложение «Заметки» на .NET MAUI (часть 2)	Реализация того же функционала на MAUI с использованием MVVM, привязок данных, команд. Настройка под Android/Windows. Оценка различий в подходах.	ОПК-2.1, ОПК-2.2
Раздел 3. Асинхронность, управление состоянием и интеграция ИИ (ML.NET, ONNX, фоновые задачи)			
7.	Интеграция ONNX модели (классификация изображений) в WinForms – асинхронный инференс.	Выбор задачи (классификация изображений или анализ тональности текста). Загрузка предобученной модели ONNX или использование ML.NET. Создание интерфейса для ввода данных (Image, TextBox). Асинхронный вызов модели.	PL-3.2, ML-3.1
8.	Интеграция ML.NET (анализ тональности) в MAUI.	Отображение результата (класс, вероятность) с анимацией загрузки. Сохранение истории предсказаний в JSON. Сравнение производительности модели в WinForms vs MAUI. Оформление отчета.	PL-3.2, ML-3.1
9.	Предобработка данных для ML в UI, сохранение истории предсказаний в JSON.	Преобразование входных данных (изображение → тензор, текст → токены) непосредственно в обработчиках событий. Отображение предобработанных данных для отладки.	ML-3.1
10.	Асинхронная загрузка данных в WinForms	Реализация «долгой» задачи (например, вычисление или загрузка из сети). ProgressBar, BackgroundWorker или Task.Run + IProgress<T>. Использование CancellationToken для отмены. Обеспечение отклика UI.	ОПК-2.2
Раздел 4. Разработка интерактивных приложений в среде Unity (МИП)			
11.	Основы проектирования, разработки интерактивных сцен в Unity: настройка рабочего пространства, управление сценой,	Изучение архитектуры игровой сцены. Освоение принципов работы с примитивами, компонентами Transform и Camera. Реализация скриптового управления виртуальной камерой в реальном времени. Освое-	ОПК-2.1, ОПК-2.2

	камерой и динамическая генерация объектов (МИП)	ние механизмов инстанцирования и работы с префабами. Формирование навыков работы с системой контроля версий Git в контексте Unity-проекта. *Пример из практики CarX: создание тестовой трассы с динамической генерации препятствий.*	
12.	Разработка прототипа игрового приложения: реализация физически-корректного управления объектами, построение событийно-ориентированной системы взаимодействий и архитектуры многоуровневого приложения (МИП)	Реализация базовых игровых механик: перемещение физического тела, обработка пользовательского ввода. Проектирование системы событий (триггеры, коллизии) для сбора игровых предметов. Разработка многоуровневой архитектуры сцен с возможностью переключения. Изучение механизмов сохранения игрового прогресса с использованием PlayerPrefs и асинхронной загрузки сцен.	ОПК-2.2
13.	Архитектурные решения и управление данными в Unity-приложениях: проектирование адаптивного пользовательского интерфейса, применение паттерна ScriptableObject, методы сериализации состояния, внедрение асинхронных операций (МИП)	Разработка многоэкранного пользовательского интерфейса (UI) с использованием системы Canvas. Изучение паттерна проектирования «ScriptableObject» для отделения данных от поведения. Освоение методов сериализации и десериализации данных в формат JSON (JsonUtility) для реализации системы сохранения настроек и прогресса пользователя. Реализация меню паузы и управления состоянием игрового процесса.	ОПК-2.1, ОПК-2.2
14.	Оптимизация и завершающий этап разработки приложения в Unity: интеграция визуальных эффектов, методы повышения эффективности приложений и генерация исполняемого файла (МИП)	Освоение паттерна программирования «Пул объектов» (Object Pool) для оптимизации управления памятью при частом создании/уничтожении игровых сущностей. Интеграция систем частиц (Particle System) для обеспечения визуальной обратной связи. Реализация дополнительных элементов интерфейса (миникарта, индикаторы состояния). Изучение процесса сборки (Build) кроссплатформенного приложения под целевую операционную систему Windows.	ОПК-2.2

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 Примерная тематика курсовых работ (проектов)

Не предусмотрены учебным планом

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, применяемые при освоении дисциплины (модуля)

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.

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 средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Современные среды разработки интерактивных приложений».

Структура оценочных средств для текущей и промежуточной аттестации

№ п/п	Код и наименование индикатора (в соответствии с п. 1.4)	Результаты обучения (в соответствии с п. 1.4)	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
1	**ОПК-2.1** Способен применять системный подход к анализу предметной (проблемной) области, выявлению требований к ИС.	Знает: методы системного анализа предметной (проблемной) области для интерактивных приложений; способы выявления, классификации и документирования требований к ИС с использованием RAD-сред; архитектуру игрового движка Unity: жизненный цикл объектов, событийную модель, систему компонентов, персистентность данных (МИП). Умеет: применять системный подход для декомпозиции требований в архитектурные элементы (UI, данные, логика) с выбором целесообразной среды разработки (WinForms, MAUI, Avalonia, Unity). Владеет: навыками построения моделей требований (use case, прототипы интерфейсов) в средах визуального проектирования .NET и Unity; навыками работы со сценами, префабами, компонентами Transform и Camera в Unity (МИП)..	Защита ЛР №1,2,5,6, (МИП) №11,12. Рубежный тест №1. Кейс №1 или №5 (МИП).	Практический кейс №1 (выбор RAD-среды и проектирование интерфейса) или кейс №5 (МИП) (Unity – система сохранений с миграцией). Билет содержит 2 вопроса из диапазона №1–13 (из разных блоков) и практическую задачу.
2	**ОПК-2.2** Применяет современный математический аппарат при построении моделей в различных областях человеческой деятельности.	Знает: современный математический аппарат: конечные автоматы, алгоритмы сериализации (JSON/XML), методы асинхронного программирования, основы линейной алгебры для 3D-преобразований; паттерны проектирования в Unity (Object Pool, ScriptableObject), методы сериализации состояния (PlayerPrefs, JSON), асинхронную загрузку сцен (МИП). Умеет: применять математические модели для	Защита ЛР №2,3,5,6, (МИП) №13,14. Рубежный тест №2. Кейс №1 или №5 (МИП).	Практический кейс №1 (асинхронный экспорт) или кейс №5 (МИП) (система сохранений). Билет содержит 2 вопроса из диапазона №14–23 и практическую задачу (например, асинхронный экспорт, сериализация или миграция данных).

		управления состоянием, физикой, оптимизации; реализовывать систему сохранения с миграцией данных, управлять физикой и событиями коллизий в Unity (МИП). Владеет: методами количественной оценки качества (латентность, throughput); навыками оптимизации памяти (пул объектов), сборки кроссплатформенного приложения в Unity (МИП).		
1	PL-3.2 Разрабатывает и отлаживает решения для задач МО и ИИ на языке C#	*Знает:* способы интеграции ML.NET в RAD-проекты, вызов ONNX-моделей из UI-событий. *Умеет:* встраивать предсказание (классификация/регрессия) в кроссплатформенное приложение, обрабатывать результаты модели асинхронно. *Владеет:* приемами привязки данных к выходам ML-модели в XAML.	Защита ЛР №7, №8. Вопросы №24–30 (блок 5, кроме специфики сравнения).	Практический кейс №2 (интеграция ONNX/ML.NET) или билет, который включает 1 вопрос из №24–28 и практическую задачу на интеграцию ONNX/ML.NET в заготовку проекта.
2	ML-3.1 Обосновывает способы и варианты применения классических методов и моделей МО в задачах ИИ	*Знает:* типовые задачи (классификация изображений, анализ текста), решаемые в RAD-приложениях, ограничения моделей в реальном времени. *Умеет:* подбирать предобученную модель для интеграции, описывать математическое преобразование входных данных. Владеет: методами предобработки данных внутри UI-событийного цикла.	Защита лабораторных работ 7, 9. Рубежный тест №2 (вопросы по ML).	Билет содержит 2 вопроса из №24–28 или устное обоснование выбора модели в рамках практического кейса.
3	ML-3.3 Оценивает результативность применения классических методов и моделей МО на основе сопоставления с аналогами	*Знает:* метрики качества (точность, latency) при инференсе модели в GUI. *Умеет:* сравнивать производительность модели в WinForms vs MAUI, измерять время отклика интерфейса. *Владеет:* построением простых A/B-тестов внутри лабораторных приложений.	Защита лабораторных работ 8 (сравнение платформ). Отчёт по сравнительной части.	Билет включает вопрос №29 или №30 и практическое измерение латентности при защите кейса.

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

Примерный перечень вопросов и заданий

Рубежный тест №1 (разделы 1–2)

Темы: RAD-методология, WinForms, архитектуры GUI (событийная, MVVM, MVP), .NET MAUI / Avalonia, кроссплатформенность, привязки данных.

Максимум: 30 баллов (далее порог – 18 баллов, 60%).

Время: 40 минут.

Часть А. Закрытые вопросы (выбор одного правильного ответа) – 1 балл каждый

1. **Что из перечисленного НЕ является принципом RAD-методологии?**

a) Прототипирование
b) Итеративность
c) Длительное предварительное проектирование
d) Визуальное проектирование интерфейса

2. **В событийной модели Windows Forms связь между элементом и обработчиком осуществляется через:**

a) Делегат и ключевое слово event
b) XAML-привязку
c) Атрибут [Bind]
d) Магический метод __onEvent

3. **Какой элемент управления в WinForms лучше всего подходит для отображения табличных данных?**

a) ListBox
b) DataGridView
c) ComboBox
d) Panel

4. **Как в WinForms избежать блокировки UI при выполнении длительной задачи?**

a) Использовать Application.DoEvents() в цикле
b) Запустить фоновую задачу через Task.Run и обновить UI через Invoke
c) Вызвать Thread.Sleep(0)
d) Установить для формы свойство DoubleBuffered = true

5. **Что означает аббревиатура MVVM?**

a) Model-View-ViewManager
b) Model-View-ViewModel
c) Model-View-ViewModule
d) Main-View-ViewModel

6. **В паттерне MVVM для уведомления View об изменении свойств ViewModel используется:**

a) INotifyPropertyChanged
b) INotifyDataErrorInfo
c) IBindingList
d) ObservableCollection

7. **Как в XAML выполнить двустороннюю привязку свойства** Text **элемента** Entry **к свойству** UserName **в ViewModel?**

a) {Binding UserName}

b) {Binding UserName, Mode=TwoWay}

c) {x:Bind UserName, Mode=TwoWay} (в UWP/MAUI допустимо, но в вопросе ожидается стандартный вариант – Binding с режимом)

d) {StaticResource UserName}

8. **Что из перечисленного является преимуществом .NET MAUI перед WinForms?**

a) Поддержка Windows только

b) Отсутствие XAML

c) Кроссплатформенность (Android, iOS, macOS, Windows)

d) Более простая событийная модель

9. **Какой паттерн из перечисленных наиболее близок к классической событийной модели WinForms?**

a) MVVM

b) MVP (Model-View-Presenter) с пассивным View

c) MVC

d) VIPER

10. **Для чего** служит ObservableCollection<T> в **контексте MVVM?**

a) Для автоматического уведомления об изменении коллекции (добавление/удаление)

b) Для сериализации данных в JSON

c) Для асинхронной загрузки данных

d) Для хранения команд

Часть Б. Открытые вопросы (краткий ответ) – 2 балла каждый

11. Назовите два основных отличия RAD-разработки от каскадной («водопадной») модели.

12. Опишите порядок событий при закрытии формы WinForms (укажите минимум два события, которые возникают).

13. Приведите пример XAML-разметки для Button, привязанной к команде AddCommand в ViewModel (используя Binding).

14. В чём разница между DataGridView и ListBox в WinForms?

15. Что такое CommandParameter в MAUI / WPF и для чего он используется?

Часть В. Практическая задача на написание кода – 10 баллов

Задача:

Дан фрагмент кода ViewModel на C# для MAUI:

```
C#
public class MainViewModel : INotifyPropertyChanged
{
    private string _searchText;
    public string SearchText
    {
        get => _searchText;
        set { _searchText = value; OnPropertyChanged(); }
    }

    public ObservableCollection<string> Items { get; set; } = new();
    public ICommand FilterCommand { get; }
```

```
    public MainViewModel()
    {
        // Инициализация Items (100 элементов)
        FilterCommand = new Command(OnFilter);
    }

    private void OnFilter()
    {
        // Необходимо реализовать фильтрацию Items по SearchText (без учёта регистра)
    }

    public event PropertyChangedEventHandler PropertyChanged;
    protected void OnPropertyChanged([CallerMemberName] string name = null) =>
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
}
```

Требуется:

Написать тело метода OnFilter(), который выполняет фильтрацию исходного списка allItems (предположим, он хранится в отдельной переменной) и присваивает отфильтрованный результат в Items. Учесть, что ObservableCollection должна обновляться в UI-потоке.

(Студент должен привести код с использованием временного списка, очисткой и добавлением или созданием нового экземпляра ObservableCollection с маршаллингом в UI-поток).

Пример правильного ответа (кратко):

```
C#
private void OnFilter()
{
    var filtered = allItems.Where(x => x.Contains(SearchText, StringComparison.Ordina
lIgnoreCase)).ToList();
    Items.Clear();
    foreach (var item in filtered)
        Items.Add(item);
```

Ключи (правильные ответы) к тесту №1

№	Ответ
1	c
2	a
3	b
4	b
5	b
6	a
7	b
8	c
9	b (MVP с пассивным View – ближе всего)
10	a
11	Итеративность, прототипирование, короткий цикл разработки, активное участие пользователя; отсутствие жёсткой фиксации требований в начале. (Любые два)
12	FormClosing, FormClosed (также Load, Shown – не при закрытии)

13	<Button Command="{Binding AddCommand}" Text="Добавить" />
14	DataGridView – таблица, много колонок, редактирование; ListBox – простой список строк.
15	Параметр, передаваемый в команду при вызове (например, идентификатор элемента).
16 (задача)	Код фильтрации с использованием allItems и обновлением Items.

Рубежный тест №2 (раздел 3)

Темы: Асинхронное программирование в RAD (async/await, CancellationToken, BackgroundWorker), управление состоянием (сериализация, паттерны Repository/Unit of Work), интеграция ИИ (ML.NET, ONNX), оценка результативности.

Максимум: 30 баллов.

Время: 45 минут.

Часть А. Закрытые вопросы (1 балл)

1. **Какой из способов является предпочтительным для выполнения асинхронной операции без блокировки UI в WinForms?**

a) Task.Run с последующим Control.Invoke

b) Thread.Sleep(0)

c) Application.DoEvents() в цикле

d) new Thread(() => { }).Start() без синхронизации

2. **Для отмены асинхронной задачи в .NET используется:**

a) CancellationTokenSource и CancellationToken

b) Thread.Abort

c) Task.Stop

d) ManualResetEvent

3. **Что произойдёт, если в обработчике события кнопки написать** Task.Run(() => { Thread.Sleep(5000); }).Wait();**?**

a) UI не будет блокироваться, задача выполнится в фоне

b) UI заблокируется на 5 секунд

c) Будет выброшено исключение

d) Приложение завершится

4. **Какой формат сериализации легче всего читается человеком и часто используется для конфигурационных файлов?**

a) BinaryFormatter

b) JSON

c) Protobuf

d) SOAP

5. **Что такое паттерн Repository?**

6. a) Способ организации кода, изолирующий логику доступа к данным

7. b) Шаблон для создания объектно-реляционного отображения (ORM)

8. c) Синоним Unit of Work

9. d) Паттерн для построения GUI

10. **Какой пакет NuGet необходимо установить для работы с ONNX-моделями в .NET?**

a) Microsoft.ML

b) Microsoft.ML.OnnxRuntime

c) TensorFlow.NET

d) TorchSharp

11. **Как в** ML.NET **загрузить предобученную модель из файла** model.zip**?**

a) MLContext.Model.Load("model.zip", out _)

b) Model.Load("model.zip")

c) OnnxRuntime.Load("model.zip")

d) PredictionEngine.Load("model.zip")

12. **Как измерить latency (время инференса) ML-модели в приложении?**

a) Использовать Stopwatch до и после вызова Predict()

b) Замерить время загрузки модели

c) Никак, это невозможно

d) Посмотреть в документации модели

13. **Что означает высокое значение уверенности модели (0.95) при низкой точности на тестовой выборке?**

a) Модель переобучена (overfitting) или калибровка вероятностей нарушена

b) Модель работает идеально

c) Данные для инференса неверны

d) Такого не бывает

14. **Какой метод сериализации в .NET является предпочтительным для сохранения состояния приложения в JSON с поддержкой асинхронности?**

a) JsonSerializer.Serialize + File.WriteAllText

b) JsonSerializer.SerializeAsync + FileStream

c) BinaryFormatter.Serialize

d) XmlSerializer.Serialize

Часть Б. Открытые вопросы (2–3 балла)

11. (2 балла) **Напишите фрагмент кода, который асинхронно сохраняет объект** data **в JSON-файл** state.json **без блокировки вызывающего потока.**

12. (2 балла) **В чём разница между** async/await и BackgroundWorker**? Назовите минимум два отличия.**

13. (2 балла) **Зачем нужен** ConfigureAwait(false) **в библиотечном коде?**

14. (3 балла) **Опишите последовательность шагов для интеграции ONNX-модели классификации изображений в приложение WinForms (от добавления пакета до получения предсказания).**

15. (2 балла) **Приведите пример метрики, которую можно использовать для сравнения производительности одной и той же ML-модели на WinForms и на MAUI.**

Часть В. Практическая задача (10 баллов)

Задача:

Дан код обработчика кнопки в WinForms, который выполняет синхронную загрузку и предсказание:

C#

```
private void btnPredict_Click(object sender, EventArgs e)
{
   var text = txtInput.Text;
   var prediction = _model.Predict(text);  // синхронный вызов, может занимать 2-3 секунды
   lblResult.Text = prediction.Sentiment ? "Позитив" : "Негатив";
}
```

Требуется:

Переписать этот метод в асинхронный вариант, не блокируя UI. Добавить:

- Кнопку отмены btnCancel (изначально неактивна).
- ProgressBar для индикации выполнения.
- Возможность отмены операции через CancellationToken.

Напишите код метода (или нескольких методов) с использованием async/await, CancellationTokenSource, IProgress<int>.

Пример правильного ответа (структура):

C#

```
private CancellationTokenSource _cts;

private async void btnPredict_Click(object sender, EventArgs e)
{
```

```
    _cts = new CancellationTokenSource();
    btnPredict.Enabled = false;
    btnCancel.Enabled = true;
    progressBar1.Visible = true;

    var progress = new Progress<int>(p => progressBar1.Value = p);
    try
    {
        var result = await Task.Run(() => PredictWithProgress(txtInput.Text, _cts.Token, progress),
_cts.Token);
        lblResult.Text = result ? "Позитив" : "Негатив";
    }
    catch (OperationCanceledException)
    {
        lblResult.Text = "Отменено";
    }
    finally
    {
        btnPredict.Enabled = true;
        btnCancel.Enabled = false;
        progressBar1.Visible = false;
        _cts.Dispose();
    }
}

private bool PredictWithProgress(string text, CancellationToken token, IProgress<int> progress)
{
    for (int i = 0; i <= 100; i += 10)
    {
        token.ThrowIfCancellationRequested();
        Thread.Sleep(200); // имитация
        progress.Report(i);
    }
    return _model.Predict(text); // синхронный вызов в фоновом потоке
}

private void btnCancel_Click(object sender, EventArgs e)
{
    _cts?.Cancel();
}
```

Ключи (ответы) к тесту №2

№	Ответ
1	a
2	a
3	b

№	Ответ
4	b
5	a
6	b
7	a
8	a
9	a
10	b
11	await using var fs = File.Create("state.json"); await JsonSerializer.SerializeAsync(fs, data);
12	async/await – языковая конструкция, не создаёт отдельный поток; BackgroundWorker – устаревший компонент, создаёт поток, имеет события отчёта о прогрессе.
13	Указывает, что продолжение не должно захватывать исходный контекст синхронизации (SynchronizationContext), повышает производительность и предотвращает deadlock в библиотеках.
14	1) Установить Microsoft.ML.OnnxRuntime; 2) добавить .onnx файл в проект; 3) создать InferenceSession; 4) подготовить входной тензор; 5) выполнить Run; 6) интерпретировать выходные данные.
15	Время отклика интерфейса (latency) при инференсе, использование памяти, FPS, время от запуска до получения результата.
16 (задача)	Код должен содержать: async void, CancellationTokenSource, прогресс-репорт, блокировку/разблокировку UI, отмену.

Оба теста позволяют оценить как знание теории, так и практические навыки. Порог сдачи: **≥18 баллов из 30** (60%). При необходимости баллы можно масштабировать.

Описание четырёх кейсов для промежуточной аттестации (зачёт) по дисциплине. Каждый кейс включает: исходные данные, пошаговую задачу для студента, требования к реализации, критерии оценки и примерный объём кода.

Кейс 1. «Асинхронная загрузка + сериализация» (WinForms)

Исходные данные (дано студенту)

- Проект **WinForms (.NET 6/8)** на C# с уже реализованным функционалом:
  - Форма MainForm содержит DataGridView с колонками: Id, Название, Цена, Количество.
  - База данных **SQLite** с таблицей Products, заполненной 100–200 записями.
  - В проекте подключен System.Data.SQLite или Microsoft.Data.Sqlite.

- На форме есть кнопка «Загрузить данные», которая заполняет DataGridView синхронно (без асинхронности).
- Проект компилируется и работает.

Задача студента (что нужно сделать)

1. **Добавить на форму:**
   - Кнопку **«Экспорт в JSON»**.
   - ProgressBar (от 0 до 100).
   - Кнопку **«Отмена»** (изначально неактивна).
2. **Реализовать асинхронный экспорт** данных из DataGridView (или напрямую из БД) в JSON-файл:
   - При нажатии «Экспорт»:
     - Кнопка экспорта блокируется, кнопка «Отмена» активируется.
     - Запускается асинхронная операция (Task.Run или async/await).
     - В фоновом потоке данные читаются (имитация «долгой» операции: можно добавить Task.Delay или реальную генерацию JSON для большого объёма).
     - ProgressBar обновляется через IProgress<int> (например, после каждых 10%).
   - **Возможность отмены** через CancellationToken:
     - При нажатии «Отмена» операция прерывается, пользователь получает сообщение «Экспорт отменён».
   - По завершении (успех или отмена) кнопка экспорта разблокируется, кнопка «Отмена» деактивируется.
   - При успехе: выбор папки через SaveFileDialog, сохранение JSON-файла (сериализация списка объектов Product).
3. **Обработка ошибок:** Если экспорт не удался (нет прав на запись, ошибка БД) – показывать MessageBox с описанием.
4. **Код должен быть потокобезопасным** – обновление UI (сообщения, прогресс, состояние кнопок) через Invoke или с использованием Progress<T>.

Ожидаемый объём изменений

- Добавить примерно 100–150 строк кода (обработчики событий, метод асинхронного экспорта).
- Использовать System.Text.Json для сериализации.

Критерии оценки «зачёт» (выполнено / не выполнено)

Критерий	Требование
Работоспособность	Приложение экспортирует данные в корректный JSON-файл.
Асинхронность	UI не блокируется во время экспорта; прогресс-бар обновляется.
Отмена	Кнопка «Отмена» прерывает операцию, состояние кнопок корректно.
Качество кода	Отсутствие Task.Result, Wait(), Thread.Sleep в UI-потоке; обработка исключений.

Кейс 2. «Интеграция простой ML-модели» (WinForms или MAUI)

Исходные данные (дано студенту)

- Проект **WinForms** или **.NET MAUI** (на выбор студента или по указанию преподавателя).
- Интерфейс содержит:
  - TextBox для ввода текста (на русском или английском).
  - Кнопку **«Анализ тональности»**.
  - Label для вывода результата (позитив / негатив).
  - Label для вывода уверенности модели (0.00–1.00).

- В папку проекта добавлен файл модели: **наивная бинарная ONNX-модель** (например, модель анализа тональности на 1000 предложениях) или ML.NET **модель** (zip-файл). Преподаватель предоставляет модель (или ссылку на скачивание).
- Установлены NuGet-пакеты: Microsoft.ML.OnnxRuntime или Microsoft.ML.

Задача студента (что нужно сделать)

1. **Загрузить модель** при старте приложения (или лениво при первом вызове).
2. **Реализовать предобработку текста** (токенизация, приведение к нижнему регистру, удаление знаков препинания) – в соответствии с требованиями модели (документация прилагается).
3. **При нажатии кнопки:**
   - Взять текст из TextBox, проверить, что он не пустой.
   - Выполнить инференс **асинхронно** (чтобы не блокировать UI).
   - Показать индикатор загрузки (например, ProgressBar или заменить текст кнопки на «Анализирую...»).
   - Получить результат (например, два числа: score для негатива и позитива).
   - Вывести в Label:
     - «Позитив» / «Негатив» (или «Нейтрально» при необходимости).
     - Уверенность (максимальная вероятность) в процентах.
4. **Обработать ошибки:** если модель не загружена, текст пуст, или инференс упал – показать сообщение.

Особенности для MAUI (если выбран)

- Использовать CommunityToolkit.MVVM или стандартный INotifyPropertyChanged.
- Команда для кнопки (RelayCommand) с асинхронным выполнением.
- Обновление UI через OnPropertyChanged.

Критерии оценки

Критерий	Требование
Загрузка модели	Модель загружается корректно, без ошибок.
Предобработка	Текст преобразуется в нужный формат (например, List<long> для токенов или строка с нормализацией).
Асинхронность	UI не зависает во время вычислений.
Результат	Выводится вердикт и уверенность, интерфейс понятен.

Кейс 3. «MVVM и привязки» (MAUI)

Исходные данные (дано студенту)

- Проект **.NET MAUI** (или .NET MAUI Blazor Hybrid) с одной страницей NotesPage.xaml.
- В NotesPage.xaml уже есть:
  - Entry для ввода новой заметки.
  - Button «Добавить» (без реализации команды).
  - CollectionView или ListView для отображения списка заметок.
  - Label для сообщений об ошибках.
- В проекте есть **ViewModel** NotesViewModel с заглушками:
  - ObservableCollection<string> Notes (пустая).
  - Свойство NewNoteText.
  - Команда AddNoteCommand не реализована (бросает NotImplementedException).
  - Метод SaveNotes() – пустой.
  - Метод LoadNotes() – пустой.

Задача студента (что нужно сделать)

1. **Реализовать привязку данных (Binding):**
   - Entry.Text привязать к NewNoteText (двусторонняя).

- CollectionView.ItemsSource привязать к Notes.
- Кнопку «Добавить» привязать к AddNoteCommand.

2. **Реализовать команду** AddNoteCommand (асинхронно):
   - Проверить, что NewNoteText не пустой и не состоит из пробелов.
   - Добавить текст в ObservableCollection<Note> (лучше использовать класс Note с Id, Text, CreatedAt).
   - Очистить NewNoteText.
   - Автоматически сохранять в JSON после каждого изменения.
3. **Сохранение и загрузка (без блокировки UI):**
   - Метод SaveNotes() – асинхронно сериализует Notes в JSON-файл в локальной папке приложения.
   - Метод LoadNotes() – асинхронно загружает при запуске страницы (в событии OnAppearing или в конструкторе ViewModel).
   - Использовать File.WriteAllTextAsync и File.ReadAllTextAsync.
4. **Дополнительно (по желанию):**
   - Добавить удаление заметки (свайпом или кнопкой) с подтверждением.
   - Сортировка по дате.

Критерии оценки

Критерий	Требование
Привязки	UI корректно отображает список и обновляется при изменении коллекции.
Команда	Кнопка работает, добавление происходит без зависаний.
Сохранение/загрузка	После перезапуска приложения заметки восстанавливаются.
Отсутствие блокировок	Операции ввода-вывода асинхронны.

Кейс 4. «Сравнение архитектур» (WinForms vs MAUI)

Исходные данные (дано студенту)

- **Два готовых приложения**, функционально идентичных:
  - **WinForms-версия** (NotesWinForms):
    - Форма с TextBox (ввод), Button (добавить), ListBox (список заметок).
    - Событийная модель: обработчик нажатия кнопки добавляет строку в ListBox и сохраняет список в JSON синхронно (сохранение в файл – потенциально долгое).
    - Загрузка при старте – также синхронная.
  - **MAUI-версия** (NotesMAUI):
    - Та же функциональность, но с MVVM: ObservableCollection, команды, асинхронное сохранение/загрузка.
    - XAML-разметка с привязками.

Оба приложения работают, но имеют **проблему** – в WinForms при большом списке и синхронной загрузке интерфейс подвисает.

Задача студента (что нужно сделать)

1. **Внести одинаковое изменение** в обе версии:
   - Добавить **поле поиска** (TextBox и кнопка «Поиск» или TextBox с автоматической фильтрацией).
   - Реализовать фильтрацию списка заметок по вхождению подстроки (без учёта регистра).
2. **В WinForms**:
   - Добавить TextBox и Button.
   - По нажатию кнопки фильтровать элементы в ListBox (исходный список хранить отдельно). Обновлять ListBox.
   - Синхронно – возможна задержка при частой фильтрации.
3. **В MAUI**:

- Добавить SearchBar (или Entry).
- Привязать текст поиска к свойству во ViewModel.
- Использовать ICollectionView или свойство FilteredNotes, которое обновляется через OnPropertyChanged.
- Фильтрация автоматическая (по мере ввода) или по кнопке – без задержек.

4. **Написать отчёт (1–2 страницы)**:
   - Описать, какие изменения были внесены в каждой версии.
   - Сравнить **сложность реализации** (количество строк кода, необходимость ручного обновления списка).
   - Сравнить **производительность** (как ведёт себя UI при фильтрации 1000+ элементов).
   - Сделать вывод: в каких случаях событийная модель предпочтительнее, а когда лучше MVVM.
   - Приложить скриншоты и ссылку на репозиторий с обеими версиями.

Критерии оценки

Критерий	Требование
Функциональность	В обеих версиях поиск работает корректно.
Отчёт	Чёткое описание различий, понимание архитектурных плюсов и минусов.
Демонстрация	Студент может показать работу приложений и объяснить код.

Общие требования ко всем кейсам для промежуточной аттестации

- **Время выполнения:** 90 минут (компьютерный зачёт) или в формате домашнего задания (по решению преподавателя).
- **Формат сдачи:** репозиторий + демонстрация. Для кейса №4 – дополнительно текстовый отчёт.
- **Оценка:** бинарная («зачёт»/«не зачёт») по критериям выше. Если студент не справляется, ему даётся ещё одна попытка (другой кейс) в течение недели.

Кейсы покрывают все индикаторы компетенций (ОПК-2.1, ОПК-2.2, PL-3.2, ML-3.1, ML-3.3) и требуют от студента практического применения знаний, полученных в лекциях и лабораторных работах.

Кейс 5. «Разработка версионно-устойчивой системы сохранений с поддержкой миграции данных и многослотовым управлением» (Unity, МИП)

Исходные данные (дано студенту):

- Проект **Unity** (версия 2021 LTS или новее) с реализованным игровым прототипом: простой гоночный трек или сцена сбора предметов, где есть такие параметры, как:
  - рекордное время заезда (`float bestTime`),
  - количество собранных монет (`int coins`),
  - выбранный автомобиль (`int carIndex`),
  - настройки громкости (`float volume`).
- В проекте отсутствует система сохранений. Требуется разработать **многослотовую систему сохранения прогресса** (минимум 3 слота), поддерживающую версионирование и миграцию данных.

Задача студента (что нужно сделать):

1. **Спроектировать класс данных сохранения** (`SaveData`), включающий:
   - поле `int version` (текущая версия 1);
   - все игровые параметры;
   - предусмотреть возможность расширения (например, `Dictionary<string, object>` для нестандартных полей).
2. **Реализовать базовую сериализацию/десериализацию** одного слота:

- использовать JsonUtility для преобразования в JSON;
- сохранять файлы в Application.persistentDataPath (например, save_slot_0.json, save_slot_1.json, save_slot_2.json);
- создать простой SaveManager с методами Save(int slotIndex), Load(int slotIndex), HasSave(int slotIndex).

3. **Добавить версионирование и миграцию:**
   - при загрузке проверять version из файла;
   - если версия файла меньше текущей (например, 0 → 1 или 1 → 2), применить миграционные функции для преобразования старых данных в новый формат;
   - **продемонстрировать миграцию**: создать тестовый файл старой версии (без некоторых полей) и показать, что после загрузки недостающие поля заполняются значениями по умолчанию, а старые данные сохраняются.
4. **Реализовать многослотовое управление:**
   - расширить SaveManager для работы с массивом из 3 слотов;
   - создать UI (Canvas) с тремя кнопками выбора слота, отображением метаданных (скриншот, дата последнего сохранения, время игры);
   - при сохранении автоматически создавать скриншот текущего вида игры (через ScreenCapture.CaptureScreenshot) и сохранять его рядом с JSON (или в отдельную папку).
5. **Обеспечить сброс прогресса:**
   - добавить кнопку «Сбросить слот» с подтверждением;
   - по нажатию удалить файл сохранения и установить значения по умолчанию.
6. **Обработать ошибки:**
   - повреждённый или невалидный JSON – вывод сообщения пользователю;
   - отсутствие файла – инициализация новым сохранением.

Ожидаемый объём изменений (кода): 200–250 строк C# (классы данных, менеджер сохранений, UI-скрипты, мигратор).

Технологии и инструментарий:

- Unity Editor, C# (MonoBehaviour, ScriptableObject – опционально), System.IO, JsonUtility, ScreenCapture, Application.persistentDataPath.
- Система контроля версий Git.

Критерии оценки «зачёт»:

Критерий	Требование
Работоспособность	Приложение сохраняет и загружает прогресс в выбранный слот, метаданные (скриншот, дата) отображаются корректно.
Версионирование и миграция	Сохранение старой версии успешно загружается, новые поля получают значения по умолчанию, миграция выполняется без потери данных.
Многослотовость	Независимо работают 3 слота, выбор слота сохраняется или переключается через UI.
Сброс прогресса	Слот корректно очищается (файл удаляется или перезаписывается начальными данными).
Качество кода	Код разделён на логические модули (менеджер, мигратор, UI), отсутствуют «магические числа», есть обработка исключений.
Понимание архитектуры	Студент объясняет, почему выбрана версия в сохранении, как работает миграция, какие альтернативы существуют.

Результаты защиты кейса:

- **Демонстрация работы** в Unity Editor (или собранном приложении) – 5 минут.
- **Ответы на вопросы** по коду, выбору архитектуры, принципам обратной совместимости.

- **Предоставление ссылки на репозиторий** с исходным кодом и файлами старых версий для тестирования миграции.

Пример мини-проекта (повышающий бонус)
Название проекта: «Классификатор изображений с историей предсказаний»

Цель проекта

Разработать интерактивное кроссплатформенное приложение (WinForms или .NET MAUI), которое:

- загружает предобученную ONNX-модель для классификации изображений (например, модель ResNet-50 на 1000 классов ImageNet),
- позволяет пользователю выбрать изображение с диска,
- выполняет асинхронный инференс с отображением прогресса,
- показывает топ-3 предсказанных класса с процентами уверенности,
- сохраняет историю всех предсказаний в JSON-файл с указанием времени,
- предоставляет возможность просмотра и очистки истории.

Задачи проекта (что должен реализовать студент)

1. **Интеграция ML-модели** – подключить ONNX-модель через Microsoft.ML.OnnxRuntime или ML.NET.
2. **Разработка GUI** – создать окно с элементами:
   - кнопка «Выбрать изображение»,
   - PictureBox/Image для предпросмотра,
   - ListBox или DataGridView для вывода топ-3 результатов,
   - ProgressBar и кнопка «Отмена» (для демонстрации асинхронности),
   - кнопка «Сохранить историю» (возможность ручного экспорта),
   - кнопка «Показать историю» (загрузить и отобразить ранее сохранённые предсказания).
3. **Асинхронность** – вызов модели выполнять через Task.Run с CancellationToken, обновлять прогресс через IProgress<int>, не блокировать UI.
4. **Предобработка изображения** – преобразование выбранного изображения в тензор нужного размера (224x224, нормализация) средствами .NET (например, System.Drawing.Common или SixLabors.ImageSharp).
5. **Сериализация** – сохранять историю в формате JSON (список объектов: путь к файлу, время, массив предсказанных классов и вероятностей). Загрузка истории – десериализация и отображение в отдельной форме.
6. **Обработка ошибок** – если модель не загрузилась или изображение некорректно – сообщение пользователю.

Стек технологий (на выбор студента)

Компонент	Вариант 1 (проще)	Вариант 2 (сложнее)
Платформа	WinForms (.NET 6/8)	.NET MAUI (кроссплатформа)
ML-библиотека	ML.NET с ONNX	Microsoft.ML.OnnxRuntime
Работа с изображениями	System.Drawing.Common	SixLabors.ImageSharp
Сериализация	System.Text.Json	System.Text.Json + File.ReadAllTextAsync

Рекомендуемые ресурсы для студента

- **Готовая ONNX-модель:** SqueezeNet (легковесная, классификация ImageNet) – скачать из ONNX Model Zoo.
- **Пример предобработки:**

C#

// изменение размера до 224x224, преобразование в float[1,3,224,224]

- **Код для загрузки модели:**

C#

// изменение using var session = new InferenceSession("model.onnx");

МП 1. Пример выполнения мини-проекта «Классификатор изображений с историей предсказаний»

МП 1.1. Архитектура приложения (пример)

Проект реализован на **WinForms (.NET 8)** с использованием библиотеки **Microsoft.ML.OnnxRuntime**. Приложение состоит из следующих ключевых классов:

- Form1 – главное окно (UI).
- ImageClassifier – обёртка над ONNX-сессией (загрузка модели, предобработка, инференс).
- PredictionHistory – управление историей (загрузка/сохранение JSON).
- PredictionRecord – модель данных для истории (путь к файлу, время, топ-3 класса и вероятности).

Структура проекта:

text

```
ImageClassifierApp/
├── Forms/
│   └── Form1.cs (дизайн и код)
├── Models/
│   ├── PredictionRecord.cs
│   └── ImageClassifier.cs
├── Services/
│   └── PredictionHistory.cs
├── Resources/
│   └── squeezenet.onnx (модель)
└── Program.cs
```

МП 1.2. Ключевые фрагменты кода (для отчёта)

Загрузка модели (класс ImageClassifier)

C#

```
using Microsoft.ML.OnnxRuntime;
using Microsoft.ML.OnnxRuntime.Tensors;

public class ImageClassifier : IDisposable
{
   private readonly InferenceSession _session;
   private readonly string[] _imagenetLabels = File.ReadAllLines("imagenet_classes.txt");

   public ImageClassifier(string modelPath)
   {
      _session = new InferenceSession(modelPath);
   }
```

```
    public async Task<List<(string Label, float Probability)>> PredictAsync(byte[] imageBytes, I
Progress<int> progress = null)
    {
        return await Task.Run(() =>
        {
            // Предобработка: загрузка, изменение размера до 224x224, нормализация
            using var image = SixLabors.ImageSharp.Image.Load<Rgb24>(imageBytes);
            image.Mutate(x => x.Resize(224, 224));
            var input = new DenseTensor<float>(new[] { 1, 3, 224, 224 });
            // ... заполнение тензора (пиксели / 255.0, нормализация по каналам)

            progress?.Report(50); // 50% – предобработка завершена

            var inputs = new List<NamedOnnxValue>
            {
                NamedOnnxValue.CreateFromTensor("data", input)
            };
            using var results = _session.Run(inputs);
            var output = results.First().AsTensor<float>();

            // Постобработка: softmax, выбор top-3
            var probabilities = Softmax(output.ToArray());
            var top3 = GetTop3(probabilities, _imagenetLabels);

            progress?.Report(100);
            return top3;
        });
    }
}
```

Асинхронный вызов в UI

C#

```
private CancellationTokenSource _cts;

private async void btnClassify_Click(object sender, EventArgs e)
{
    if (openFileDialog.ShowDialog() != DialogResult.OK) return;

    _cts = new CancellationTokenSource();
    btnClassify.Enabled = false;
    btnCancel.Enabled = true;
    progressBar.Visible = true;

    var imageBytes = File.ReadAllBytes(openFileDialog.FileName);
    pictureBox.Image = Image.FromFile(openFileDialog.FileName);

    var progress = new Progress<int>(p => progressBar.Value = p);

    try
    {
        var results = await _classifier.PredictAsync(imageBytes, progress, _cts.Token);
        DisplayResults(results);
```

```
        // Сохранить в историю
        var record = new PredictionRecord
        {
            ImagePath = openFileDialog.FileName,
            Timestamp = DateTime.Now,
            TopPredictions = results
        };
        _history.Add(record);
        await _history.SaveAsync();
    }
    catch (OperationCanceledException)
    {
        lblStatus.Text = "Отменено";
    }
    finally
    {
        btnClassify.Enabled = true;
        btnCancel.Enabled = false;
        progressBar.Visible = false;
    }
}
```

Сериализация истории (JSON)

```
C#
public class PredictionHistory
{
    private readonly string _filePath = Path.Combine(Application.UserAppDataPath, "history.json
");
    public List<PredictionRecord> Records { get; set; } = new();

    public async Task LoadAsync()
    {
        if (!File.Exists(_filePath)) return;
        var json = await File.ReadAllTextAsync(_filePath);
        Records = JsonSerializer.Deserialize<List<PredictionRecord>>(json);
    }

    public async Task SaveAsync()
    {
        var json = JsonSerializer.Serialize(Records, new JsonSerializerOptions { WriteIndented = tr
ue });
        await File.WriteAllTextAsync(_filePath, json);
    }
}
```

МП 1.3. Скриншот готового приложения (пример)

[В отчёте студент вставляет скриншот своего работающего приложения с загруженным изображением, результатом классификации и списком истории.]

МП 2. Требования к защите мини-проекта

Защита проводится индивидуально в формате устной презентации (5–7 минут) с демонстрацией работающего приложения. Студент должен:

1. **Предоставить ссылку на репозиторий** (GitHub/GitLab) с исходным кодом, инструкцией по сборке и запуску.
2. **Продемонстрировать работу приложения**:
   - Загрузка изображения.
   - Асинхронный инференс (показать, что UI не блокируется, прогресс-бар работает).
   - Отмена операции во время выполнения.
   - Сохранение истории и её загрузка после перезапуска.
3. **Ответить на контрольные вопросы** (не менее 3 из приведённого списка):
   - Какая модель используется и почему выбран именно ONNX-формат?
   - Как выполняется предобработка изображения (масштабирование, нормализация)?
   - Почему инференс выполняется в отдельной задаче (`Task.Run`)? Что произойдёт, если сделать синхронный вызов?
   - Как реализована отмена операции? Объясните роль `CancellationTokenSource`.
   - Какой формат сериализации выбран для истории и почему?
   - Какие метрики производительности вы измеряли (время инференса, задержка UI)?
4. **Показать понимание кода** – объяснить любой фрагмент по просьбе преподавателя.

Критерии успешной защиты (дополнение к табличным):

- Проект компилируется и запускается без ошибок.
- Все обязательные функции (классификация, отмена, сохранение истории) работают.
- Студент дал верные и развёрнутые ответы на 2 из 3 заданных вопросов.
- Код соответствует принципам асинхронного программирования и потокобезопасности.

Штрафные санкции:

- Ошибки при работе с файлами (отсутствие обработки исключений) – минус 1 балл.
- Блокировка UI при инференсе – проект не принимается до исправления.

МП 3. Примерный шаблон отчёта по мини-проекту

Отчёт оформляется в электронном виде (формат .docx или .pdf). Рекомендуемый объём – 3–5 страниц.

Титульный лист

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет прикладной математики и компьютерных технологий

ОТЧЁТ ПО МИНИ-ПРОЕКТУ
«Классификатор изображений с историей предсказаний»

Выполнил: студент группы ____

ФИО
Проверил: преподаватель
ФИО

Краснодар, 2026

Раздел 1. Цель и задачи проекта

(Краткое описание: что разработано, какие технологии использованы, какие индикаторы компетенций проверяются.)

Раздел 2. Архитектура приложения

- Схема взаимодействия классов (можно в виде текста или диаграммы).
- Описание каждой основной части (загрузка модели, предобработка, UI, история).

Раздел 3. Реализация ключевых функций

3.1. Интеграция ONNX-модели

(Привести код класса ImageClassifier*, пояснить, как создаётся тензор, как вызывается* Run*.)*

3.2. Асинхронный инференс и отмена

(Показать фрагмент обработчика кнопки, объяснить CancellationTokenSource*,* IProgress*.)*

3.3. Сохранение и загрузка истории

(Привести код сериализации в JSON, объяснить выбор пути Application.UserAppDataPath*.)*

Раздел 4. Скриншоты работы приложения

- Скриншот главного окна до загрузки.
- Скриншот во время инференса (прогресс-бар, кнопка «Отмена» активна).
- Скриншот с результатами (топ-3 класса и вероятности).
- Скриншот окна истории (или списка в главном окне).
- Скриншот после перезапуска приложения – история сохранилась.

Раздел 5. Тестирование и измерение производительности

Изображение	Время инференса (мс)	Точность (субъективно)
cat.jpg	245	Высокая (кот)
car.png	238	Средняя (спорткар)

Вывод: модель работает быстро, UI не блокируется.

Раздел 6. Выводы

(Что освоено в ходе выполнения проекта, какие трудности возникли, как они преодолены, какие компетенции сформированы.)

Список литературы / источники

- Документация ONNX Runtime.

- Статья по предобработке изображений для ResNet.
- ...

Приложение. Листинг полного кода (или ссылка на репозиторий)

МП 4. Рекомендации по оцениванию отчёта

Преподаватель оценивает:

- **Полноту** – все разделы заполнены, код соответствует описанию.
- **Качество кода** – читаемость, комментарии, обработка исключений.
- **Демонстрацию** – студент способен запустить и объяснить работу.

Максимальный бонус к среднему баллу за семестр – **10%** (при условии защиты на «отлично»).

Зачетно-экзаменационные материалы для промежуточной аттестации (экзамен/зачет)

Перечень вопросов для подготовки к текущему контролю и промежуточной аттестации. Вопросы охватывают все лекционные темы и лабораторные работы дисциплины «Современные среды разработки интерактивных приложений».

Блок 1. RAD-методология и Windows Forms (вопросы 1–7)

1. **Перечислите основные принципы RAD-методологии.** В чём преимущество быстрой разработки приложений по сравнению с «водопадной» моделью?
2. **Назовите примеры RAD-сред.** Какие из них поддерживают визуальное проектирование интерфейса?
3. **Опишите событийную модель Windows Forms.** Как происходит связь элемента управления (Button) с его обработчиком события?
4. **Что такое визуальный конструктор?** Какие элементы управления (Button, TextBox, DataGridView, ListBox, ComboBox) вы знаете и для каких задач они используются?
5. **Жизненный цикл формы в WinForms.** Какие события возникают при создании, открытии, закрытии формы?
6. **Как в WinForms организовать подключение к SQLite и отобразить данные в DataGridView?** Приведите основные шаги.
7. **Что такое сериализация?** В чём отличие XML-сериализации от JSON-сериализации в контексте сохранения настроек приложения?

Блок 2. Архитектуры GUI (MVVM, MVP) и кроссплатформенная разработка (вопросы 8–13)

8. **Сравните классическую событийную модель (WinForms) и паттерн MVVM (WPF, MAUI).** Какие преимущества даёт MVVM при разработке сложных интерфейсов?
9. **Что такое паттерн Model-View-Presenter (MVP)?** В каких случаях он предпочтительнее MVVM?
10. **Расскажите о XAML.** Какие возможности он предоставляет для описания интерфейса (привязка данных, стили, ресурсы)?
11. **Как работает привязка данных (Data Binding) в .NET MAUI?** Что такое INotifyPropertyChanged и зачем он нужен?
12. **Особенности жизненного цикла кроссплатформенного приложения .NET MAUI.** Чем отличается запуск на Windows от запуска на Android?
13. **Сравнительная лабораторная: «Заметки» на WinForms и на MAUI.** Какие ключевые различия в подходах к реализации одного и того же функционала вы увидели?

Блок 3. Асинхронное программирование в RAD (вопросы 14–19)

14. **Что такое асинхронное программирование?** Зачем использовать async/await в GUI-приложениях?

15. **В чём разница между** Task.Run **и** BackgroundWorker**?** Когда что предпочтительнее?

16. **Как избежать блокировки UI-потока при выполнении длительной операции?** Приведите пример плохого и хорошего кода.

17. **Что такое** CancellationToken**?** Как организовать отмену асинхронной задачи (например, загрузки данных) по требованию пользователя?

18. **Как обновить ProgressBar из фонового потока?** Объясните роль IProgress<T> и захвата контекста синхронизации.

19. **В лабораторной работе №10 «Асинхронная загрузка данных» вы реализовывали долгую задачу с откликом UI.** Какие ошибки блокировки интерфейса вы обнаружили и как их исправили?

Блок 4. Управление состоянием и сериализация (вопросы 20–23)

20. **Какие способы сохранения состояния приложения вы знаете?** (реестр, файлы, JSON, XML). В чём плюсы и минусы каждого?

21. **Что такое паттерн Repository?** Как он применяется при работе с SQLite в лабораторной работе №2?

Вопросы (МИП)

22. **Что такое версионирование данных в системе сохранений?** Почему необходимо хранить версию в файле сохранения?

23. **Опишите алгоритм миграции данных при загрузке старого сохранения.** Какие шаги необходимо выполнить для обновления структуры?

24. **Как организовать многослотовое сохранение (3+ слота) в Unity?** Какие метаданные (скриншот, дата, время) полезно сохранять для каждого слота?

25. **Что такое паттерн Unit of Work?** В каких случаях он полезен в RAD-приложениях?

26. **В лабораторной работе по сериализации вы сохраняли таблицу в XML/JSON.** Как бы вы реализовали загрузку обратно с проверкой целостности данных?

Блок 5. Интеграция искусственного интеллекта (ML.NET, ONNX) – вопросы 24–30

27. **Какие задачи машинного обучения можно интегрировать в десктопное RAD-приложение?** Приведите примеры (классификация изображений, анализ тональности и т.д.).

28. **Что такое** ML.NET? Как добавить ML.NET в проект WinForms или MAUI?

29. **Как загрузить и использовать предобученную ONNX-модель в C# приложении?** Объясните роль OnnxRuntime и PredictionEngine.

30. **Почему вызов ML-модели должен быть асинхронным с точки зрения UI?** Как это реализовать?

31. **Как подготовить входные данные (изображение, текст) для модели внутри обработчика события?** Какие преобразования необходимы?

32. **В лабораторной работе вы сравнивали производительность одной и той же ML-модели в WinForms и MAUI.** Какие метрики вы измеряли (латентность, FPS интерфейса)? Кто показал лучший результат и почему?

33. **Как оценить результативность применения ML-модели в интерактивном приложении?** Какие метрики важны для конечного пользователя (помимо точности)?

Блок 6. Разработка интерактивных приложений в Unity, системы сохранений (МИП)

34. **Как в Unity выполнить сериализацию игровых данных в JSON с использованием** JsonUtility? В чём ограничения этого подхода?

35. **Что такое** Application.persistentDataPath? Почему рекомендуется использовать именно этот путь для сохранения файлов?

36. **Как реализовать восстановление значений по умолчанию (сброс прогресса) в многослотовой системе сохранений?**

Дополнительные практические задачи для устного билета (не включены в список вопросов, представленных выше, но могут использоваться)

- Напишите метод на C#, который асинхронно загружает строку по URL и отображает её в TextBox (с обработкой исключений).
- Реализуйте фрагмент кода для отмены задачи через CancellationToken при нажатии кнопки «Отмена».
- Создайте простейшую привязку в XAML: текстовое поле привязать к свойству в ViewModel.
- Напишите код для сохранения объекта в JSON с помощью System.Text.Json.

Текущий контроль включает:

- Защиту лабораторных работ – студент демонстрирует работающий код и отвечает на 2–3 контрольных вопроса.
- Рубежные тесты (два).
- Мини-проект (дополнительно, для повышения оценки) – интеграция ML-модели в приложение с сохранением истории.

Виды и сроки текущего контроля

№	Вид контроля	Срок (неделя семестра)	Кол-во	Форма фиксации	Вес в допуске
1	Защита лабораторных работ	2–15	8	Оценка «зачтено» / «не зачтено» в ведомости	Обязательное условие (все 8 – «зачтено»)
2	Рубежный тест №1 (по разделам 1–2)	6–7	1	Баллы (max 20)	20%
3	Рубежный тест №2 (по разделу 3)	12–13	1	Баллы (max 20)	20%
4	Выполнение мини-проекта (по желанию, для повышения)	14–15	1	Оценка «зачтено» / «не зачтено» (бонус +10% к допуску)	Доп. бонус
5	Посещаемость и активность	непрерывно	–	«зачтено» при ≥75% посещений	Не более 5% штрафа

Критерии оценивания результатов обучения

1. **Перечень вопросов** является единым как для текущего, так и для итогового контроля.

- При защите лабораторных работ преподаватель может задать 2–3 вопроса из соответствующих блоков (например, после ЛР 1.1 – вопросы 3–5).
- Рубежные тесты строятся на основе вопросов №1–13 (тест №1) и №14–30 (тест №2).

2. **Промежуточная аттестация (зачёт)** проводится в форме устного билета или практического кейса.

• **Билет** содержит 2 теоретических вопроса (из разных диапазонов в соответствии с таблицей) и 1 практическую задачу (например, написать фрагмент асинхронного кода или выполнить привязку в XAML).

• Вопросы для билета выбираются случайным образом из указанных диапазонов так, чтобы покрыть минимум два индикатора.

3. **Допуск к зачёт**у – все лабораторные работы защищены, рубежные тесты сданы не менее чем на 60%.

Пример формирования билета на зачёт

Билет №	Вопрос 1 (из блока 1–2, ОПК-2.1)	Вопрос 2 (из блока 3–5, ОПК-2.2 / PL-3.2)	Практическая задача
5	№6: Жизненный цикл формы в WinForms	№26: Как загрузить ONNX-модель в C#?	Напишите async-метод, загружающий данные и отображающий ProgressBar.

КРИТЕРИИ ОЦЕНИВАНИЯ ПО ЗАЧЕТУ:

- Правильный ответ на 2 теоретических вопроса из разных блоков (каждый оценивается до 2 баллов, всего 4 балла).
- Решение практической задачи (до 3 баллов).
- Итого: ≥ 4 баллов из 7 – «зачтено».

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.

5. Перечень учебной литературы, информационных ресурсов и технологий

5.1. Учебная литература

1. Казанский, А. А. Программирование на C# : учебное пособие для вузов / А. А. Казанский. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 181 с. — (Высшее образование). — ISBN 978-5-534-21381-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/584121 (дата обращения: 23.04.2026).

2. Демин, А. Ю. Информатика. Программирование на C# в Visual Studio : учебник для вузов / А. Ю. Демин, В. А. Дорофеев. — 2-е изд. — Москва : Издательство Юрайт, 2025. — 138 с. — (Высшее образование). — ISBN 978-5-534-20596-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/561363 (дата обращения: 23.04.2026).

3. Зыков, С. В. Объектно-ориентированное программирование : учебник и практикум для вузов / С. В. Зыков. — 2-е изд. — Москва : Издательство Юрайт, 2026. — 151 с. — (Высшее образование). — ISBN 978-5-534-16941-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/584131 (дата обращения: 23.04.2026).

4. Тузовский, А. Ф. Объектно-ориентированное программирование : учебник для вузов / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2025. — 213 с. — (Высшее образование). — ISBN 978-5-534-16316-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/561394 (дата обращения: 23.04.2026).

5.2. Периодическая литература

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

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

Электронно-библиотечные системы (ЭБС):

1. ЭБС «ЮРАЙТ» https://urait.ru/
2. ЭБС «УНИВЕРСИТЕТСКАЯ БИБЛИОТЕКА ОНЛАЙН» 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. Web of Science (WoS) http://webofscience.com/
2. Scopus http://www.scopus.com/
3. ScienceDirect www.sciencedirect.com
4. Журналы издательства Wiley https://onlinelibrary.wiley.com/
5. Научная электронная библиотека (НЭБ) http://www.elibrary.ru/
6. Полнотекстовые архивы ведущих западных научных журналов на Российской платформе научных журналов НЭИКОН http://archive.neicon.ru
7. Национальная электронная библиотека (доступ к Электронной библиотеке диссертаций Российской государственной библиотеки (РГБ) https://rusneb.ru/
8. Президентская библиотека им. Б.Н. Ельцина https://www.prlib.ru/

9. Электронная коллекция Оксфордского Российского Фонда https://ebookcentral.proquest.com/lib/kubanstate/home.action
10. Springer Journals https://link.springer.com/
11. Nature Journals https://www.nature.com/siteindex/index.html
12. Springer Nature Protocols and Methods https://experiments.springernature.com/sources/springer-protocols
13. Springer Materials http://materials.springer.com/
14. zbMath https://zbmath.org/
15. Nano Database https://nano.nature.com/
16. Springer eBooks: https://link.springer.com/
17. "Лекториум ТВ" http://www.lektorium.tv/
18. Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Информационные справочные системы:

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

Ресурсы свободного доступа:

1. Американская патентная база данных http://www.uspto.gov/patft/
2. Полные тексты канадских диссертаций http://www.nlc-bnc.ca/thesescanada/
3. КиберЛенинка (http://cyberleninka.ru/);
4. Министерство науки и высшего образования Российской Федерации https://www.minobrnauki.gov.ru/;
5. Федеральный портал "Российское образование" http://www.edu.ru/;
6. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru/;
7. Единая коллекция цифровых образовательных ресурсов http://school-collection.edu.ru/ .
8. Федеральный центр информационно-образовательных ресурсов (http://fcior.edu.ru/);
9. Проект Государственного института русского языка имени А.С. Пушкина "Образование на русском" https://pushkininstitute.ru/;
10. Справочно-информационный портал "Русский язык" http://gramota.ru/;
11. Служба тематических толковых словарей http://www.glossary.ru/;
12. Словари и энциклопедии http://dic.academic.ru/;
13. Образовательный портал "Учеба" http://www.ucheba.com/;
14. 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. Среда модульного динамического обучения http://moodle.kubsu.ru
2. База учебных планов, учебно-методических комплексов, публикаций и конференций http://mschool.kubsu.ru/
3. Библиотека информационных ресурсов кафедры информационных образовательных технологий http://mschool.kubsu.ru;
4. Электронный архив документов КубГУ http://docspace.kubsu.ru/
5. 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)

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.

По курсу предусмотрено проведение лекционных занятий, на которых дается систематизированный материал. В ходе лекций рассматриваются ключевые концепции.

Лабораторные занятия курса посвящены практическому освоению предложенных технологий и подходов.

При самостоятельной работе студентам необходимо изучать рекомендованную литературу в виде официальной документации к используемым открытым программным продуктам, облачным платформам.

Важнейшим компонентом курса является самостоятельная проектная работа, в ходе которой студент разрабатывает законченное решение для решения задач (кейсов.

Название лабораторной работы работы

Асинхронная загрузка данных в Windows Forms

(Лабораторная работа №10, 4 часа)

Цель работы

Освоить принципы асинхронного программирования в RAD-среде на примере Windows Forms.

Научиться:

- выполнять длительные операции в фоновом потоке без блокировки пользовательского интерфейса;
- отображать ход выполнения задачи с помощью ProgressBar;
- реализовывать отмену операции через CancellationToken;
- корректно обновлять элементы управления из фонового потока.

Технологии и инструменты

- **Язык программирования:** C# (.NET 6/8)
- **Среда разработки:** Visual Studio 2022 (Community или выше)
- **Фреймворк:** Windows Forms
- **Ключевые библиотеки/пространства имён:**
  - System.Threading.Tasks (async/await, Task)
  - System.Threading (CancellationToken, CancellationTokenSource)
  - System.Windows.Forms (ProgressBar, Button, Label, DataGridView)
- **Дополнительно:** локальная база данных SQLite (для имитации загрузки данных) или генерация случайных данных.

Реализация (ход выполнения)

1. Подготовка проекта

1. Создайте новый проект **Windows Forms App (.NET)** с именем AsyncLoadLab.
2. Добавьте на главную форму Form1 следующие элементы управления (см. таблицу):

Элемент	Имя (Name)	Назначение
DataGridView	dataGridView1	Отображение загруженных данных
Button	btnLoad	Запуск асинхронной загрузки
Button	btnCancel	Отмена загрузки (изначально неактивна)
ProgressBar	progressBar1	Индикатор выполнения (минимум 0, максимум 100)

Label	lblStatus	Отображение текущего состояния («Загрузка...», «Отменено», «Готово»)

Расположите элементы удобно, например, DataGridView на всю ширину, под ним панель с кнопками и прогресс-баром.

2. Имитация источника данных

Создайте класс DataProvider, который будет асинхронно возвращать список строк (или объектов). Для имитации длительной операции используйте Task.Delay.

C#

```
public static class DataProvider
{
    public static async Task<List<string>> LoadDataAsync(int count, IProgress<int> progress, CancellationToken token)
    {
        var result = new List<string>();
        for (int i = 1; i <= count; i++)
        {
            token.ThrowIfCancellationRequested();
            await Task.Delay(50, token); // имитация задержки
            result.Add($"Элемент {i}");
            progress?.Report(i * 100 / count); // процент выполнения
        }
        return result;
    }
}
```

3. Реализация асинхронного обработчика кнопки «Загрузить»

В классе формы объявите поле CancellationTokenSource _cts;.

C#

```
private CancellationTokenSource _cts;

private async void btnLoad_Click(object sender, EventArgs e)
{
    // Подготовка UI к загрузке
    btnLoad.Enabled = false;
    btnCancel.Enabled = true;
    progressBar1.Value = 0;
    lblStatus.Text = "Загрузка...";
    dataGridView1.Rows.Clear();

    _cts = new CancellationTokenSource();
    var progress = new Progress<int>(p => progressBar1.Value = p);

    try
    {
        // Асинхронный вызов с передачей токена и прогресса
        var data = await DataProvider.LoadDataAsync(100, progress, _cts.Token);

        // Обновление DataGridView (должно быть в UI-потоке, мы уже в нём)
        foreach (var item in data)
            dataGridView1.Rows.Add(item);

        lblStatus.Text = $"Загружено {data.Count} записей";
    }
```

```
    catch (OperationCanceledException)
    {
        lblStatus.Text = "Загрузка отменена";
        progressBar1.Value = 0;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка");
        lblStatus.Text = "Ошибка";
    }
    finally
    {
        btnLoad.Enabled = true;
        btnCancel.Enabled = false;
        _cts?.Dispose();
        _cts = null;
    }
}
```

4. Реализация отмены

Добавьте обработчик для кнопки btnCancel:

csharp

```
private void btnCancel_Click(object sender, EventArgs e)
{
    _cts?.Cancel();
    btnCancel.Enabled = false; // предотвращаем повторную отмену
}
```

5. Дополнительное задание (для закрепления)

Модифицируйте приложение так, чтобы данные загружались из реальной SQLite-таблицы (замените DataProvider на вызовы async методов библиотеки Microsoft.Data.Sqlite). Добавьте отображение времени выполнения (используйте Stopwatch).

Результат работы

По итогу выполнения лабораторной работы студент должен представить:

1. **Исходный код проекта** (архив или ссылку на репозиторий).
2. **Демонстрацию работоспособности** (скриншоты или короткое видео):
   - Запуск приложения;
   - Нажатие кнопки «Загрузить» — прогресс-бар заполняется, интерфейс не блокируется (можно двигать окно, нажимать другие кнопки);
   - Нажатие кнопки «Отмена» во время загрузки — операция прерывается, появляется сообщение «Отменено».
3. **Ответы на контрольные вопросы** (письменно или устно при защите):
   - Чем отличается асинхронный метод от синхронного?
   - Для чего используется CancellationToken?
   - Как обновлять UI из фонового потока? Почему нельзя просто присвоить значение свойству?
   - Что будет, если в обработчике async void возникнет необработанное исключение?

Критерии оценки («зачтено»)

- Приложение компилируется и запускается без ошибок.
- Кнопка «Загрузить» инициирует асинхронную загрузку, интерфейс остаётся отзывчивым.
- ProgressBar отображает ход выполнения.

- Отмена операции корректно прерывает загрузку.
- Студент понимает и объясняет свой код.

Данные методические указания могут быть адаптированы под любую другую лабораторную работу (например, для интеграции ИИ или для сравнения архитектур). Структура (название, цель, технологии, реализация, результат) остаётся неизменной.

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	MAUI): Visual Studio 2022 (Community или Professional), .NET SDK 6.0/8.0, SQLite, Git Лабораторные работы (Unity): Unity Hub, Unity Editor (2021 LTS или новее), Visual Studio Tools for Unity Общее: Microsoft Windows, браузер (Chrome/Edge), средство для работы с репозиториями (GitHub Desktop или консольный Git)
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	Лабораторные работы (WinForms, MAUI): Visual Studio 2022 (Community или Professional), .NET SDK 6.0/8.0, SQLite, Git Лабораторные работы (Unity): Unity Hub, Unity Editor (2021 LTS или новее), Visual Studio Tools for Unity Общее: Microsoft Windows, браузер (Chrome/Edge), средство для работы с репозиториями (GitHub Desktop или консольный Git)

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	MS Office Word 2016 и выше Ms Power Point 2016 и выше

	интернет (проводное соединение и беспроводное соединение по технологии Wi-Fi)	
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)	Visual Studio 2022 (Community или Professional), .NET SDK 6.0/8.0, Unity Hub, Unity Editor, Git, браузер (Chrome/Edge).

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.

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.