Министерство науки и высшего образования Российской Федерации

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

Кафедра «Программное обеспечение автоматизированных систем»

УТВЕРЖДАЮ:
Первый проректор

_____Т. Р. Змызгова

«02» сентября 2021 г.

Рабочая программа учебной дисциплины

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

образовательной программы высшего образования –
программы бакалавриата

09.03.04 Программная инженерия
направленность
Программное обеспечение автоматизированных систем

формы обучения – очная, заочная

Курган 2021

Рабочая программа дисциплины «Разработка мобильных приложений» составлена в соответствии с учебными планами по программе бакалавриата «Программная инженерия» (Программное обеспечение автоматизированных систем), утвержденными для очной формы обучения «30» августа 2021 года, для заочной формы обучения «30» августа 2021 года.

Рабочая программа дисциплины одобрена на заседании кафедры «Программное обеспечение автоматизированных систем» «1» сентября 2021 года, протокол № 1.

Рабочую программу составил:

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

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

Согласовано:

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

1. ОБЪЕМ ДИСЦИПЛИНЫ

Всего: 4 зачетных единиц трудоемкости (144 академических часа)

Очная форма обучения

Вид учебной работы	На всю дисциплину	Семестр 7
Аудиторные занятия (контактная работа с преподавателем), всего часов в том числе:	**48**	**48**
Лекции	16	16
Лабораторные работы	32	32
Аудиторные занятия в интерактивной форме, часов	-	-
Самостоятельная работа, всего часов в том числе:	**96**	**96**
Контрольная работа	18	18
Подготовка к зачёту	18	18
Другие виды самостоятельной работы (самостоятельное изучение тем (разделов) дисциплны)	60	60
Вид промежуточной аттестации	**Зачёт**	**Зачёт**
Общая трудоемкость дисциплины и трудоемкость по семестрам, часов	**144**	**144**

Заочная форма обучения

Вид учебной работы	На всю дисциплину	Курс 5, 10 семестр
Аудиторные занятия (контактная работа с преподавателем), всего часов в том числе:	**10**	**10**
Лекции	4	4
Лабораторные работы	6	6
Аудиторные занятия в интерактивной форме, часов	-	-
Самостоятельная работа, всего часов в том числе:	**134**	**134**
Контрольная работа	18	18
Подготовка к экзамену	27	27
Другие виды самостоятельной работы (самостоятельное изучение тем (разделов) дисциплны)	89	89
Вид промежуточной аттестации	**Экзамен**	**Экзамен**
Общая трудоемкость дисциплины и трудоемкость по семестрам, часов	**144**	**144**

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина «Разработка мобильных приложений» относится к модулю «Промышленные технологии разработки и сопровождения программного обеспечения», к части, формируемой участниками образовательных отношений, блока 1.

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

- Информатика.
- Основы программирования.
- Алгоритмы и структуры данных.
- Технологии разработки web-приложений.

Результаты изучения дисциплины используются при освоении профильных дисциплин, включенных в модули «Программное и аппаратное обеспечение информационно-коммуникационных систем», «Технологии разработки программных систем» и «Системы интеллектуальной обработки данных».

3. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

Целью освоения дисциплины « Разработки мобильных приложений» является формирование знаний и практических навыков разработке мобильных приложений.

Задачи дисциплины:

1) изучение:

– архитектура iOS/Android и нативные API;

– архитектура кроссплатформенных фреймворков;

– архитектура мобильных приложений;

– процесс разработки мобильных приложений и документация;

– создание интерфейсов с помощью макетов и виджетов;

– базы данных и библиотека компонентов архитектуры Room;

– библиотека WorkManager;

– Веб-серфинг и WebView.

2) практическое освоение:

– платформа Android Studio;

– язык программирования Kotlin;

– Git;

– библиотеки Glide (используется для загрузки изображений) и Retrofit (применяется для получения данных из сети);

Компетенции, формируемые в результате освоения дисциплины:

- владение навыками использования операционных систем, сетевых технологий, систем управления базами данных (ПК-6);

- способность осуществлять разработку, отладку, проверку работоспособности, оценку сложности программного обеспечения и рефакторинг программного кода (ПК-7);

- способность проводить установку, настройку и оптимизацию функционирования прикладного программного обеспечения (ПК-11).

В результате изучения дисциплины обучающийся должен:

Знать:

- операционные системы, сетевые технологии, системы управления базами данных (ПК-6);

- разработку, отладку, проверку работоспособности, оценку сложности программного обеспечения и рефакторинг программного кода (ПК-7);

- установку, настройку и оптимизацию функционирования прикладного программного обеспечения (ПК-11).

Уметь:

- применять операционные системы, сетевые технологии, системы управления базами данных (ПК-6);

- применять методы и средства разработки, отладки, проверки работоспособности, оценки сложности программного обеспечения и рефакторинг программного кода (ПК-7);

- проводить установку, настройку и оптимизацию функционирования прикладного программного обеспечения (ПК-11).

Владеть:

- навыками работы с операционными системами, сетевыми технологиями, системами управления базами данных (ПК-6);

- знаниями основных методов и инструментов разработки, отладки, проверки работоспособности, оценки сложности программного обеспечения и рефакторинг программного кода (ПК-7);

- навыками установки, настройки и оптимизации функционирования прикладного программного обеспечения (ПК-11).

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Учебно-тематический план. Очная форма обучения. Семестр 7

Рубеж	Номер раздела, темы	Наименование раздела, темы	Количество часов контактной работы с преподавателем	
			Лекции	Лабораторные работы
Рубеж 1	1	Архитектура и базовые сведения о платформе Android	2	2
	2	Основы разработки мобильных приложений	2	2
	3	Android и модель MVC	2	4
	4	Неявные интенты, интенты при работе с камерой	2	4

		Рубежный контроль №1	-	2
Рубеж 2	5	Создание интерфейсов с использованием макетов и виджетов	2	4
	6	Базы данных и Room Library	2	4
	7	Модульное тестирование и воспроизведение звуков	2	4
	8	Классы Looper и HandlerThread	2	4
		Рубежный контроль №2	-	2
		Всего:	**16**	**32**

4.2. Учебно-тематический план. Заочная форма обучения. Семестр 10

Рубеж	Номер раздела, темы	Наименование раздела, темы	Количество часов контактной работы с преподавателем	
			Лекции	Лабораторные работы
Рубеж 1	1	Архитектура и базовые сведения о платформе Android	0,5	0,5
	2	Основы разработки мобильных приложений	0,5	0,5
	3	Android и модель MVC	0,5	0,5
	4	Неявные интенты, интенты при работе с камерой	0,5	0,5
Рубеж 2	5	Создание интерфейсов с использованием макетов и виджетов	0,5	1
	6	Базы данных и Room Library	0,5	1
	7	Модульное тестирование и воспроизведение звуков	0,5	1
	8	Классы Looper и HandlerThread	0,5	1
		Всего:	**4**	**6**

4.3 Содержание лекционных занятий
Семестр 7

Тема 1. Архитектура и базовые сведения о платформе Android

Нативные и кроссплатформенные инструменты разработки. Архитектура iOS/Android. Нативный iOS. Нативный Android. Нативный Windows UMP. Архитектура Android. Уровень ядра. Уровень библиотек. Уровень каркаса приложений. Компоненты Android-приложения. Activity. Service. Broadcast Receiver. Архитектуры кроссплатформенных фреймворков:PhoneGap, ReactNative, Qt, Flutter, Xamarin, Xamarin.Forms.

Тема 2. Основы разработки мобильных приложений

Создание проекта Andoid. Навигация в Android Studio. Создание макета пользовательского интерфейса. Иерархия представления. Атрибуты виджетов. Создание строковых ресурсов. Предварительный просмотр макета. От

разметки XML к объектам View. Ресурсы и идентификаторы ресурсов. Разработка виджетов. Установка ссылок на виджеты. Назначение слушателей. Уведомления. Выполнение в эмуляторе. Процесс сборки Android-приложений. Инструменты сборки.

Тема 3. Android и модель MVC

Создание нового класса. Архитектура «Модель–Представление–Контроллер» и Android. Преимущества MVC. Обновление уровня представления. Обновление уровня контроллера. Добавление ресурсов в проект. Ссылка на ресурсы в XML. Запуск на устройстве. Многослойный MVVM. Декомпозиция по слоям. Связи внутри слоёв. Связи между слоями. Структуры данных на основе UI.

Тема 4. Неявные интенты, интенты при работе с камерой

Использование неявных интентов. Строение неявного интента. Отправка отчёта. Запрос контакта у Android. Проверка реагирующих activity. Хранилище файлов. Использование FileProvider. Использование интента камеры. Масштабирование и отображение растровых изображений.

Тема 5. Создание интерфейсов с использованием макетов и виджетов

ConstraintLayout. Использование графического инструмента макетов. Освобождение пространства. Добавление виджетов. Внутренние механизмы ConstrainLayout. Редактирование свойств. Динамическое поведение элементов списка. Стили, темы, атрибуты тем.

Тема 6. Базы данных и Room Library

Библиотека компонентов архитектуры Room. Создание базы данных. Отделение сущности. Создание класса базы данных. Определение объекта доступа к данным. Доступ к базе данных с помощью шаблона репозитория. Тестирование запросов. Загрузка тестовых данных. Потоки приложения. Фоновые потоки. Использование LiveData.

Тема 7. Модульное тестирование и воспроизведение звуков

Создание объекта SounPool. Доступ к активам. Воспроизведение звука. Зависимости от тестирования. Создание класса теста. Подготовка теста. Настройка тестируемых объектов. Написание тестов. Взаимодействия тестируемых объектов. Обратные вызовы привязки данных. Выгузка звуков.

Тема 8. Классы Looper и HandlerThread

Подготовка RecyclerView. Подготовка к загрузке через URL. Множественные загрузки. Создание фонового потока. Запуск и остановка HandlerThread. Сообщения и обработчики сообщений. Структура сообщений. Структура обработчика. Использование обработчиков. Прослушивание жизненного цикла представления. Сохранённые фрагменты.

4.3 Лабораторные занятия. Очная форма обучения. Семестр 7

Номер раздела, темы	Наименование раздела, Темы	Наименование лабораторной работы	Норматив времени, час.
1	Архитектура и базовые сведения о платформе Android	Основные принципы создания проектов Android	2
2	Основы разработки мобильных приложений	Разработка игры Tetris	2
3	Android и модель MVC	Разработка приложения для хранения данных	4
4	Неявные интенты, интенты при работе с камерой	Воспроизведение мультимедийного контента	4
		Рубежный контроль №1	2
5	Создание интерфейсов с использованием макетов и виджетов	Нестандартный лаунчер	4
6	Базы данных и Room Library	Загрузка и отображение фотографий	4
7	Модульное тестирование и воспроизведение звуков	Графическое приложение обработки событий касания и создание нестандартных представлений	4
8	Классы Looper и HandlerThread	Приложение с анимацией	4
		Рубежный контроль №2	2
		Всего:	**32**

4.4 Лабораторные занятия. Заочная форма обучения. Семестр 10

Номер раздела, темы	Наименование раздела, Темы	Наименование лабораторной работы	Норматив времени, час.
1	Архитектура и базовые сведения о платформе Android	Основные принципы создания проектов Android	0,5
2	Основы разработки мобильных приложений	Разработка игры Tetris	0,5
3	Android и модель MVC	Разработка приложения для хранения данных	0,5
4	Неявные интенты, интенты при работе с камерой	Воспроизведение мультимедийного контента	0,5

5	Создание интерфейсов с использованием макетов и виджетов	Нестандартный лаунчер	1
6	Базы данных и Room Library	Загрузка и отображение фотографий	1
7	Модульное тестирование и воспроизведение звуков	Графическое приложение обработки событий касания и создание нестандартных представлений	1
8	Классы Looper и HandlerThread	Приложение с анимацией	1
		Всего:	**6**

4.5 Контрольная работа (для очной, заочной формы обучения)

Контрольная работа посвящена разработке мобильного приложения, по вариантам задания, согласно методических рекомендаций.

4.5.1 Назначение, цели и задачи контрольной работы

Контрольная работа выполняется по вариантам заданий или по теме, предложенной студентом, и согласованной с преподавателем.

В ходе выполнения контрольной работы студент проектирует и реализует мобильное приложение.

Основная учебная цель выполнения контрольной работы – закрепление теоретических знаний, полученных в процессе изучения дисциплины «Разработка мобильных приложений», и приобретение практических навыков в разработке мобильных приложений.

Основные задачи, решаемые студентом в процессе выполнения контрольной работы:

- создание проекта в Android Studio;
- создание пользовательского интерфейса;
- добавление активности, навигации и действий;
- выполнение тест-драйва приложения в эмуляторе;
- оформление документации.

4.5.2 Требования к контрольной работе

4.5.2.1 Требования к функциональным характеристикам

Функциональные требования к приложению:

- роли пользователей: какие уровни доступа должны быть у разных пользователей, например у гостя и авторизованного пользователя;
- форматы данных: как будет реализован обмен данными в приложении;

- интеграция: должно ли приложение поддерживать совместную работу с другими сервисами, например с платежными системами и почтовыми серверами;

- интерфейсы доступа: как приложение будет обмениваться данными с внешними сервисами;

- дополнительные функции: должно ли приложение уметь что-то еще, например работать с файлами или библиотеками шифрования;

- конфигурация и администрирование: с помощью каких элементов администратор будет управлять приложением;

- состав системы: из чего состоит мобильное приложение, то есть экраны, пуш-уведомления, система аутентификации и т.д.

4.5.2.2 Требования к эксплуатационным характеристикам

- Модульность.
- Расширяемость.
- Кроссплатформенность.
- Отказоустойчивость.

4.5.2.3 Требования к программному обеспечению

- Язык программирования Kotlin, Java;
- Платформа Android.

4.5.2.4 Требования к содержанию контрольной работы

К защите контрольной работы должен быть представлены мобильное приложение и пояснительная записка:

- экран загрузки;
- регистрация и авторизация;
- основной экран;
- меню;
- поиск;
- уведомления.

4.5.3 Варианты заданий контрольной работы

1 Мобильное приложение «Поиск вакансий».
2 Мобильное приложение «Участники спортивных соревнований»
3 Мобильное приложение «Cash Organizer»
4 Мобильное приложение «Прогноз погоды».
5 Мобильное приложение «Тренажёр для развития памяти».
6 Мобильное приложение «Калькулятор ОСАГО».
7 Мобильное приложение «Записная книжка».
8 Мобильное приложение «Карманный навигатор».

9 Мобильное приложение «Домашняя библиотека».

10 Мобильное приложение «Информационная система салона магазина по продаже мобильных устройств»

11 Мобильное приложение, рисующее движение автомобиля через перекрёсток

12 Мобильное приложение «Результаты аттестации студенческой группы».

13 Мобильное приложение «Домашняя бухгалтерия».

14 Мобильное приложение «Чтение QR-кодов».

15 Мобильное приложение-приложение «Мир путешествий» (при запуске приложения открывается карта Google, на которую нанесены точки (маркеры) – переходы к вопросу о месте, в котором она расположена. Цель: ответить правильно на все вопросы и пройти все точки на карте)

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

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

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

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

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

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

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

Выполнение самостоятельной работы подразумевает самостоятельное изучение разделов дисциплины, подготовку к лабораторным работам, к ру-

бежным контролям (для очной формы обучения), выполнение контрольной работы, подготовку к зачёту (для очной формы обучения), подготовку к экзамену (для заочной формы обучения).

Рекомендуемая трудоемкость самостоятельной работы для очной и заочной формы обучения представлена в таблицах:

Рекомендуемый режим самостоятельной работы

Очная форма

Наименование вида самостоятельной работы	**Рекомендуемая трудоемкость, акад. час.**
	Семестр 7
Самостоятельное изучение тем дисциплины:	**28**
Жизненный цикл activity	6
Отладка Android-приложений	6
Версии Android SDK и совместимость	6
UI-фрагменты и FragmentManager	6
Вывод списков и RecyclerView	4
Подготовка к лабораторным занятиям (по 2 часа на каждое занятие)	**28**
Подготовка к рубежным контролям (по 2 часа на каждый рубеж)	**4**
Выполнение контрольной работы	**18**
Подготовка к зачёту	**18**
Всего:	**96**

Заочная форма

Наименование вида самостоятельной работы	**Рекомендуемая трудоемкость, акад. час.**
	Семестр 10
Самостоятельное изучение тем дисциплины:	**83**
Жизненный цикл activity	16
Отладка Android-приложений	16
Версии Android SDK и совместимость	17
UI-фрагменты и FragmentManager	17
Вывод списков и RecyclerView	17
Подготовка к лабораторным занятиям (по 2 часа на каждое занятие)	**6**
Выполнение контрольной работы	**18**
Подготовка к экзамену	**27**
Всего:	**134**

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

6.1. Перечень оценочных средств

1 Балльно-рейтинговая система контроля и оценки академической активности студентов в КГУ (для очной формы обучения).

2 Отчеты студентов по лабораторным работам.

3 Банк заданий к рубежным контролям № 1, № 2 (для очной формы обучения).

4 Банк заданий к зачёту (очная форма обучения).

5 Контрольная работа.

6 Банк заданий к экзамену (для заочной формы обучения).

6.2. Система балльно-рейтинговой оценки работы студентов по дисциплине

Очная форма обучения

<table>
<tr><td>№</td><td>Наименование</td><td colspan="7">Содержание</td></tr>
<tr><td rowspan="3">1</td><td rowspan="3">Распределение баллов за семестры по видам учебной работы, сроки сдачи учебной работы (доводятся до сведения студентов на первом учебном занятии)</td><td colspan="7">Распределение баллов,
7 семестр</td></tr>
<tr><td>Вид учебной работы:</td><td>Посещение лекций</td><td>Выполнение и защита отчетов по лабораторным работам</td><td>Рубежный контроль №1</td><td>Рубежный контроль №2</td><td>Контрольная работа</td><td>Зачёт</td></tr>
<tr><td>Балльная оценка:</td><td>2б*8=16б</td><td>5б*8=40б</td><td>5</td><td>5</td><td>4</td><td>30</td></tr>
<tr><td>2</td><td>Критерий пересчета баллов в традиционную оценку по итогам работы в семестре и зачёта</td><td colspan="7">60 и менее баллов – незачтено;
61…73 – зачтено;
74… 90 – зачтено;
91…100 – зачтено</td></tr>
<tr><td>3</td><td>Критерии допуска к промежуточной аттестации, возможности получения автоматического зачета (экзаменационной оценки) по дисциплине, возможность получения бонусных баллов</td><td colspan="7">Для допуска к промежуточной аттестации (зачету) студент должен набрать по итогам текущего и рубежного контроля не менее 50 баллов, выполнить все лабораторные работы и контрольную работу.
Для получения «автоматически» оценки «зачтено» студенту необходимо набрать 61 балл.
По согласованию с преподавателем студенту могут быть добавлены дополнительные (бонусные) баллы за активность на консультациях, активное участие в научной и методической работе, оригинальность принятых решений в ходе выполнения практических работ, за участие в значимых учебных и внеучебных мероприятиях кафедры.</td></tr>
</table>

4	Формы и виды учебной работы для неуспевающих (восстановившихся на курсе обучения) студентов для получения недостающих баллов в конце семестра	В случае если к промежуточной аттестации (зачету) набрана сумма менее 50 баллов, студенту необходимо набрать недостающее количество баллов за счет выполнения дополнительных заданий, до конца последней (зачетной) недели семестра. При этом необходимо проработать материал всех пропущенных лабораторных занятий. Формы дополнительных заданий (назначаются преподавателем): - выполнение и защита пропущенного лабораторного занятия (при невозможности проведения дополнительного занятия преподаватель самостоятельно устанавливает форму дополнительного задания по тематике пропущенного лабораторного занятия) – до 8 баллов. Ликвидация академических задолженностей, возникших из-за разности в учебных планах при переводе или восстановлении, проводится путем выполнения дополнительных заданий, форма и объем которых определяется преподавателем.

6.3 Процедура оценивания результатов освоения дисциплины

Рубежные контроли (для очной формы обучения) проводятся в виде ответов на вопросы в письменной форме. Зачёт (для очной формы обучения) и экзамен (для заочной формы обучения) проводится в виде ответов на вопросы билета в устной форме.

Перед проведением рубежного контроля (для очной формы обучения) преподаватель прорабатывает с обучающимися основной материал соответствующих разделов дисциплины в форме краткой лекции-дискуссии.

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

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

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

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

Билет зачёта (для очной формы обучения) содержит 1 вопрос. Вопросы к зачёту доводятся до обучающегося на последней лекции в семестре. На

подготовку ответа обучающему отводится 1 астрономический час. Вопрос оценивается в 30 баллов.

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

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

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

6.4 Примеры оценочных средств для рубежных контролей, зачёта, экзамена

6.4.1 Примеры заданий для рубежного контроля №1 Очная форма обучения, 7 семестр

Вариант 1_1

1 Кроссплатформенные приложения – приложения, разрабатываемые под конкретную аппаратно-программную платформу на языках, созданных для данной платформы?

1 Да.

2 Нет.

2 Какие преимущества нативных приложений?

1 Высокая производительность.

2 Простое и быстрое развёртывание.

3 Покрывают широкую аудиторию.

4 Один код доступен для повторного использования на других платформах.

3 Какие недостатки нативных приложений?

1 Значительные финансовые и временные затраты на разработку.

2 Отсутствие гибкости.

3 Несовместимость с другими мобильными операционными системами.

4 Несоответствие UI в различных платформах.

4 Какие преимущества кроссплатформенных приложений?

1 Высокая производительность.

2 Простое и быстрое развёртывание.

3 Покрывают широкую аудиторию.

4 Один код доступен для повторного использования на других платформах.

5 Какие недостатки кроссплатформенных приложений?

1 Значительные финансовые и временные затраты на разработку.

2 Отсутствие гибкости.

3 Несовместимость с другими мобильными операционными системами.

4 Несоответствие UI в различных платформах.

6 Kotlin – типизированный объектно-ориентированный язык программирования, функционирующий на основе виртуальной машины Java (JVM)?

1 Да

2 Нет

7 Какие преимущества имеет язык Kotlin по сравнению с языком Java?

1 кроссплатформенность

2 нулевая безопасность

3 произволительность

4 наличие функций расширений

8 Что такое выведение типов данных в языке Kotlin?

1 ручное определение типа переменной на основании данных, которыми переменная инициализируется

2 ручное определение типа переменной на основании вводимых данных

3 автоматическое определение типа переменной на основании вводимых данных

4 автоматическое определение значения переменной на основании анализа данных, которыми переменная инициализируется

9 Какие методы языка Kotlin используются для вызова функций?

1 invoke( )

2 invake( )

3 invike( )

4 invuke( )

10 Какую правильную форму имеет оператор Elvis?

1 (выражение) :? значение2

2 (выражение) !: значение2

3 (выражение) ?: значение2

4 (выражение) :! значение2

11 Какой принцип реализует система Android?

1 принцип наибольших привилегий

2 принцип наименьших привилегий

3 принцип изменяемых привилегий

4 принцип постоянных привилегий

12 Какие подкаталоги содержит каталог ресурсов Android-проекта app\src\main\res?

1 *res\able*

2 *res\json*

3 *res\draw*

4 *res\xml*

13 Какие компоненты применяются при создании Android-приложений?

1 Text providers
2 Graphic providers
3 Content providers
4 Double providers

14 Какие типы компонента *Services* сообщают системе, как управлять приложением?

1 Quick services
2 Started services
3 Layout services
4 Low services

15 Что такое Видимый процесс (Visible Process)?

1 процесс, в котором выполняется служба
2 процесс, не содержащий активных компонентов приложений
3 процесс, не имеющий приоритетных компонентов
4 процесс с которым работает в текущий момент пользователь

16 Какие действия выполняются в файле манифеста приложения AndroidManifest.xml?

1 Объявляет минимальный уровень API
2 Объявляет максимальный уровень API
3 Объявляет функции расширения
4 Объявляет аппаратные и программные функции

17 Какой правильный листинг файла манифеста приложения AndroidManifest.xml?

1
```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application android:icon="@drawable/app_icon.png" ... >
    <activity android:name="com.example.project.ExampleActivity"
          android:name="@string/example_label" ... >
    </activity>
    ...
  </application>
</manifest>
```

2
```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application android:icon="@drawable/app_icon.png" ... >
    <activity android:name="com.example.project.ExampleActivity"
          android:label="@string/example_label" ... >
    </activity>
    ...
  </application>
</manifest>
```

```
3 <?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application android:icon="@drawable/app_icon.png" ... >
    <activity android:name="com.example.project.Activity"
          android:name="@string/example_label" ... >
    </activity>
    ...
  </application>
</manifest>

4 <?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application android:icon="@drawable/app_icon.png" ... >
    <activity android:name="com.example.project.Activity"
          android:label="@string/example_label" ... >
    </activity>
    ...
  </application>
</manifest>
```

18 Какая правильная последовательность этапов запроса разрешенияAndroid-приложений?

1 *Этапы запроса разрешения:*

Этап 1. В файле манифеста вашего приложения укажите разрешения, которые может потребоваться запросить вашему приложению.

Этап 2. Разработайте пользовательский интерфейс вашего приложения таким образом, чтобы конкретные действия в вашем приложении были связаны с определенными разрешениями среды выполнения. Пользователи должны знать, какие действия могут потребовать от них предоставления вашему приложению разрешения на доступ к личным пользовательским данным.

Этап 3. Дождитесь, пока пользователь вызовет в вашем приложении задачу или действие , требующее доступа к определенным личным пользовательским данным. В это время ваше приложение может запросить разрешение среды выполнения, необходимое для доступа к этим данным.

Этап 4. Проверьте, должно ли ваше приложение показывать пользователю обоснование, объясняющее, почему вашему приложению требуется, чтобы пользователь предоставил определенное разрешение на выполнение. Если система определит, что ваше приложение не должно показывать обоснование, переходите непосредственно к следующему шагу, не показывая элемент пользовательского интерфейса.

Этап 5. Проверьте, предоставил ли пользователь уже разрешение на выполнение, которое требуется вашему приложению. Если это так, ваше приложение может получить доступ к личным данным пользователя. Если нет, переходите к следующему шагу.

Этап 6. Запросите разрешение среды выполнения, необходимое вашему приложению для доступа к личным пользовательским данным. Система отображает запрос разрешения среды выполнения, подобный тому, который показан на странице обзора разрешений.

Этап 7. Проверьте ответ пользователя, независимо от того, решил ли он предоставить или отклонить разрешение на выполнение.

Этап 8. Если пользователь предоставил разрешение вашему приложению, вы можете получить доступ к личным данным пользователя. Если вместо этого пользователь отказал в разрешении, корректно ухудшите работу вашего приложения, чтобы оно предоставляло пользователю функциональные возможности даже без информации, защищенной этим разрешением.

2 *Этапы запроса разрешения:*

Этап 1. В файле манифеста вашего приложения укажите разрешения, которые может потребоваться запросить вашему приложению.

Этап 2. Разработайте пользовательский интерфейс вашего приложения таким образом, чтобы конкретные действия в вашем приложении были связаны с определенными разрешениями среды выполнения. Пользователи должны знать, какие действия могут потребовать от них предоставления вашему приложению разрешения на доступ к личным пользовательским данным.

Этап 3. Проверьте, предоставил ли пользователь уже разрешение на выполнение, которое требуется вашему приложению. Если это так, ваше приложение может получить доступ к личным данным пользователя. Если нет, переходите к следующему шагу.

Этап 4. Дождитесь, пока пользователь вызовет в вашем приложении задачу или действие , требующее доступа к определенным личным пользовательским данным. В это время ваше приложение может запросить разрешение среды выполнения, необходимое для доступа к этим данным.

Этап 5. Проверьте, должно ли ваше приложение показывать пользователю обоснование, объясняющее, почему вашему приложению требуется, чтобы пользователь предоставил определенное разрешение на выполнение. Если система определит, что ваше приложение не должно показывать обоснование, переходите непосредственно к следующему шагу, не показывая элемент пользовательского интерфейса.

Этап 6. Запросите разрешение среды выполнения, необходимое вашему приложению для доступа к личным пользовательским данным. Система отображает запрос разрешения среды выполнения, подобный тому, который показан на странице обзора разрешений.

Этап 7. Проверьте ответ пользователя, независимо от того, решил ли он предоставить или отклонить разрешение на выполнение.

Этап 8. Если пользователь предоставил разрешение вашему приложению, вы можете получить доступ к личным данным пользователя. Если вместо этого пользователь отказал в разрешении, корректно ухудшите работу вашего приложения, чтобы оно предоставляло пользователю функциональные возможности даже без информации, защищенной этим разрешением.

3 *Этапы запроса разрешения:*

Этап 1. В файле манифеста вашего приложения укажите разрешения, которые может потребоваться запросить вашему приложению.

Этап 2. Разработайте пользовательский интерфейс вашего приложения таким образом, чтобы конкретные действия в вашем приложении были связаны с определенными разрешениями среды выполнения. Пользователи должны знать, какие действия могут потребовать от них предоставления вашему приложению разрешения на доступ к личным пользовательским данным.

Этап 3. Дождитесь, пока пользователь вызовет в вашем приложении задачу или действие , требующее доступа к определенным личным пользовательским данным. В это время ваше приложение может запросить разрешение среды выполнения, необходимое для доступа к этим данным.

Этап 4. Проверьте, предоставил ли пользователь уже разрешение на выполнение, которое требуется вашему приложению. Если это так, ваше приложение может получить доступ к личным данным пользователя. Если нет, переходите к следующему шагу.

Этап 5. Проверьте, должно ли ваше приложение показывать пользователю обоснование, объясняющее, почему вашему приложению требуется, чтобы пользователь предоставил определенное разрешение на выполнение. Если система определит, что ваше приложение не должно показывать обоснование, переходите непосредственно к следующему шагу, не показывая элемент пользовательского интерфейса.

Этап 6. Проверьте ответ пользователя, независимо от того, решил ли он предоставить или отклонить разрешение на выполнение.

Этап 7. Запросите разрешение среды выполнения, необходимое вашему приложению для доступа к личным пользовательским данным. Система отображает запрос разрешения среды выполнения, подобный тому, который показан на странице обзора разрешений.

Этап 8. Если пользователь предоставил разрешение вашему приложению, вы можете получить доступ к личным данным пользователя. Если вместо этого пользователь отказал в разрешении, корректно ухудшите работу вашего приложения, чтобы оно предоставляло пользователю функциональные возможности даже без информации, защищенной этим разрешением.

4 *Этапы запроса разрешения:*

Этап 1. В файле манифеста вашего приложения укажите разрешения, которые может потребоваться запросить вашему приложению.

Этап 2. Разработайте пользовательский интерфейс вашего приложения таким образом, чтобы конкретные действия в вашем приложении были связаны с определенными разрешениями среды выполнения. Пользователи должны знать, какие действия могут потребовать от них предоставления вашему приложению разрешения на доступ к личным пользовательским данным.

Этап 3. Дождитесь, пока пользователь вызовет в вашем приложении задачу или действие , требующее доступа к определенным личным пользова-

тельским данным. В это время ваше приложение может запросить разрешение среды выполнения, необходимое для доступа к этим данным.

Этап 4. Проверьте, предоставил ли пользователь уже разрешение на выполнение, которое требуется вашему приложению. Если это так, ваше приложение может получить доступ к личным данным пользователя. Если нет, переходите к следующему шагу.

Этап 5. Проверьте, должно ли ваше приложение показывать пользователю обоснование, объясняющее, почему вашему приложению требуется, чтобы пользователь предоставил определенное разрешение на выполнение. Если система определит, что ваше приложение не должно показывать обоснование, переходите непосредственно к следующему шагу, не показывая элемент пользовательского интерфейса.

Этап 6. Запросите разрешение среды выполнения, необходимое вашему приложению для доступа к личным пользовательским данным. Система отображает запрос разрешения среды выполнения, подобный тому, который показан на странице обзора разрешений.

Этап 7. Проверьте ответ пользователя, независимо от того, решил ли он предоставить или отклонить разрешение на выполнение.

Этап 8. Если пользователь предоставил разрешение вашему приложению, вы можете получить доступ к личным данным пользователя. Если вместо этого пользователь отказал в разрешении, корректно ухудшите работу вашего приложения, чтобы оно предоставляло пользователю функциональные возможности даже без информации, защищенной этим разрешением.

19 Какой правильный листинг запроса разрешения с помощью кода запроса?

1
```
when {
  ContextCompat.checkSelfPermission(
      CONTEXT,
      Manifest.permission.REQUESTED_PERMISSION
      ) == PackageManager.PERMISSION_GRANTED -> {
    // You can use the API that requires the permission.
    Action(...)
  }
  shouldShowRequestPermissionRationale(...) -> {
    // In an educational UI, explain to the user why your app requires this
    // permission for a specific feature to behave as expected. In this UI,
    // include a "cancel" or "no thanks" button that allows the user to
    // continue using your app without granting the permission.
    showInContextUI(...)
  }
  else -> {
    // You can directly ask for the permission.
    requestPermissions(CONTEXT,
        arrayOf(Manifest.permission.REQUESTED_PERMISSION),
```

```
            REQUEST_CODE)
    }
}

2 when {
    ContextCompat.checkSelfPermission(
            CONTEXT,
            Manifest.permission.REQUESTED_PERMISSION
            ) == PackageManager.PERMISSION_GRANTED -> {
        // You can use the API that requires the permission.
        performAction(...)
    }
    shouldShowRequestPermissionRationale(...) -> {
        // In an educational UI, explain to the user why your app requires this
        // permission for a specific feature to behave as expected. In this UI,
        // include a "cancel" or "no thanks" button that allows the user to
        // continue using your app without granting the permission.
        showInContextUI(...)
    }
    else -> {
        // You can directly ask for the permission.
        requestPermissions(CONTEXT,
                arrayOf(Manifest.permission.REQUESTED_PERMISSION),
                REQUEST_CODE)
    }
}
3 when {
    ContextCompat.checkSelfPermission(
            CONTEXT,
            Permission.REQUESTED_PERMISSION
            ) == PackageManager.PERMISSION_GRANTED -> {
        // You can use the API that requires the permission.
        performAction(...)
    }
    shouldShowRequestPermissionRationale(...) -> {
        // In an educational UI, explain to the user why your app requires this
        // permission for a specific feature to behave as expected. In this UI,
        // include a "cancel" or "no thanks" button that allows the user to
        // continue using your app without granting the permission.
        showInContextUI(...)
    }
    else -> {
        // You can directly ask for the permission.
        requestPermissions(CONTEXT,
                arrayOf(Manifest.permission.REQUESTED_PERMISSION),
```

```
            REQUEST_CODE)
        }
    }

4 when {
    ContextCompat.checkSelfPermission(
            CONTEXT,
            Manifest.permission.REQUESTED_PERMISSION
            ) == PackageManager.PERMISSION_GRANTED -> {
        // You can use the API that requires the permission.
        performAction(...)
    }
    shouldShowRequestPermissionRationale(...) -> {
        // In an educational UI, explain to the user why your app requires this
        // permission for a specific feature to behave as expected. In this UI,
        // include a "cancel" or "no thanks" button that allows the user to
        // continue using your app without granting the permission.
        showInContextUI(...)
    }
    else if -> {
        // You can directly ask for the permission.
        requestPermissions(CONTEXT,
            arrayOf(Manifest.permission.REQUESTED_PERMISSION),
            REQUEST_CODE)
    }
}
```

20 Какой правильный листинг передачи кода запроса?

```
1 override fun onRequestPermissionsResult(requestCode: Int,
        permissions: Array<String>, grantResults: IntArray) {
    when (requestCode) {
        PERMISSION_REQUEST_CODE -> {
            // If request is cancelled, the result arrays are empty.
            if ((grantResults.isNotEmpty() &&
                grantResults[0] == PackageManag-
er.PERMISSION_GRANTED)) {
                // Permission is granted. Continue the action or workflow
                // in your app.
            } else {
                // Explain to the user that the feature is unavailable because
                // the features requires a permission that the user has denied.
                // At the same time, respect the user's decision. Don't link to
                // system settings in an effort to convince the user to change
                // their decision.
            }
            return
```

```
            }
            // Add other 'when' lines to check for other
            // permissions this app might request.
            else -> {
               // Ignore all other requests.
            }
         }
      }
      2 override fun onRequestPermissionsResult(requestCode: Int,
           permissions: Array<String>, grantResults: IntArray) {
         when (requestCode) {
            PERMISSION_REQUEST_CODE -> {
               // If request is cancelled, the result arrays are empty.
               if ((grantResults.isNotEmpty() &&
                     grantResults[0]          ==          PackageManag-
er.PERMISSION_GRANTED)) {
                  // Permission is granted. Continue the action or workflow
                  // in your app.
               } else {
                  // Explain to the user that the feature is unavailable because
                  // the features requires a permission that the user has denied.
                  // At the same time, respect the user's decision. Don't link to
                  // system settings in an effort to convince the user to change
                  // their decision.
               }
               return 0
            }
            // Add other 'when' lines to check for other
            // permissions this app might request.
            else -> {
               // Ignore all other requests.
            }
         }
      }
      3 override fun onRequestPermissionsResult(requestCode: Int,
           permissions: Array<String>, grantResults: IntArray) {
         when (requestCode) {
            PERMISSION_REQUEST_CODE -> {
               // If request is cancelled, the result arrays are empty.
               if ((grantResults.isNotEmpty() &&
                     grantResults[0]          ==          PackageManag-
er.PERMISSION_GRANTED)) {
                  // Permission is granted. Continue the action or workflow
                  // in your app.
               } else if {
```

```
                // Explain to the user that the feature is unavailable because
                // the features requires a permission that the user has denied.
                // At the same time, respect the user's decision. Don't link to
                // system settings in an effort to convince the user to change
                // their decision.
            }
            return
        }
        // Add other 'when' lines to check for other
        // permissions this app might request.
        else -> {
            // Ignore all other requests.
        }
    }
}
4 override fun onRequestPermissionsResult(requestCode: Int,
    permissions: Array<String>, grantResults: IntArray) {
    switch (requestCode) {
        PERMISSION_REQUEST_CODE -> {
            // If request is cancelled, the result arrays are empty.
            if ((grantResults.isNotEmpty() &&
                grantResults[0]          ==          PackageManag-
er.PERMISSION_GRANTED)) {
                // Permission is granted. Continue the action or workflow
                // in your app.
            } else {
                // Explain to the user that the feature is unavailable because
                // the features requires a permission that the user has denied.
                // At the same time, respect the user's decision. Don't link to
                // system settings in an effort to convince the user to change
                // their decision.
            }
            return
        }
        // Add other 'when' lines to check for other
        // permissions this app might request.
        else -> {
            // Ignore all other requests.
        }
    }
}
```

21 Какие существуют способы выполнения разметки?

1 Редактирование *XML-файла* окна приложения

2 Редактирование *JSON-файла* окна приложения

3 Интерактивный

4 Графический

22 Какие существуют типы разметки?

1 *VerticalLayout*

2 *LinearLayout*

3 *FrameLayout*

4 *TextLayout*

23 Какие существуют способы использования макетов в файле ресурсов?

1 Назначение макета на роль корневого представления в *Services*

2 Назначение макета на роль корневого представления в *Activity*

3 Добавление содержимого макета в существующее дерево решения

4 Добавление содержимого макета в существующее дерево представлений

24 Какой правильный атрибут пространства имен, ссылающийся на схему Android?

1 *http://schemas.android.com/apk/res/android*

2 *http://schemas.android.com/atk/res/android*

3 *http://schemas.android.json/apk/res/android*

4 *http://schemas.android.com/apk/xml/android*

25 Какой правильный листинг файла макета?

1 *<?xml versi.on="l.0" encoding="utf-8"?>*
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:rayout_width = "match_parent"
android:1ayout_height="match_parent">
android:orientation="vertical">
<android.support.design.widget.AppBarLayout
android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
<android.support.V7.widget.Too1bar
android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
app:navigationIcon="?attr/homleAUpIndicator" />
</android.support.design.widget.AppBarLayout>
<FrameLayout
android:1ayout_width="match_parent"
android:1ayout_height="0dp"
androi.d:1ayout_weight="1">
</FrameLayout>
</LinearLayout>

2 *<?xml versi.on="l.0" encoding="utf-8"?>*
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:1ayout_width = "match_parent"
android:1ayout_height="match_parent">

android:orientation=”vertical”>
<android.support.design.widget.AppBarLayout
android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
<android.support.V7.widget.Too1bar
android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
app:navigationIcon="?attr/homleAUpIndicator" />
</android.support.design.widget.AppBarLayout>
<TableLayout
android:1ayout_width="match_parent"
android:1ayout_height="0dp"
androi.d:1ayout_weight="1">
</FrameLayout>
</LinearLayout>
3 *<?xml versi.on="l.0" encoding="utf-8"?>*
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:1ayout_width = “match_parent”
android:1ayout_height="match_parent">
android:orientation=”vertical”>
<android.support.design.widget.AppBarLayout
android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
<android.support.V7.widget.Too1bar
android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
app:navigationJcon="?attr/homleAUpIndicator" />
</android.support.design.widget.AppBarLayout>
<FrameLayout
android:1ayout_width="match_parent"
android:1ayout_height="0dp"
androi.d:1ayout_weight="1">
</FrameLayout>
</LinearLayout>
4 *<?xml versi.on="l.0" encoding="utf-8"?>*
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:1ayout_width = “match_parent”
android:1ayout_height="match_parent">
android:orientation=”vertical”>
<android.support.design.widget.AppBarLayout
android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
<android.support.V7.widget.Too1bar

android:1ayout_width="match_parent"
android:1ayout_height="wrap_content"
app:navigationIcon="?attr/homleAUpIndicator" />
</android.support.design.widget.AppBarLayout>
<FrameLayout
android:1ayout_width="match_parent"
android:1ayout_height="0dp"
androi.d:1ayout_weight="1">
</FrameLayout>
</LinearLayout>

6.4.2 Примеры заданий для рубежного контроля №2
Очная форма обучения, 7 семестр

Вариант 2_1

1 Какие типы уведомлений отображают Android приложения?

1 Local Notification
2 Toast Notification
3 View Notification
4 Status Bar Notification

2 Какой класс используется для отображения всплывающего сообщения?

1 Dialog
2 Snackbar
3 Toast
4 View

3 Какое предназначение имеет метод setGravity?

1 Отображение уведомления на экране
2 Загрузка представления
3 Включение представления в иерархическое дерево представлений
4 Удаление представления из иерархии представлений, конда оно уже не отображается

4 Какую строковую константу необходимо передать в качестве параметра методу getSystemService( ), чтобы получить ссылку на NotificationManager?

1 NOTIFICATION_VIEW
2 NOTIFICATION_TOAST
3 NOTIFICATION_SERVICE
4 NOTIFICATION_SHOW

5 Какой механизм используют для реализации поддержки нестандартных жестов?

1 Переопределение метода onEventTouch в экземпляре ViewGroup (или Activity) и управление каждым событием в соответствии с потребностями

2 Переопределение метода onTouchEvent в экземпляре ViewGroup (или MyActivity) и управление каждым событием в соответствии с потребностями

3 Переопределение метода onEventTouchEvent в экземпляре GroupView (или Activity) и управление каждым событием в соответствии с потребностями

4 Переопределение метода onTouchEvent в экземпляре ViewGroup (или Activity) и управление каждым событием в соответствии с потребностями

6 Какие обработчики сенсорных событий реализуют для полного контроля при использовании GestureDetectorCompat?

1 onUpPress

2 onDownPress

3 onShowPress

4 onShortPress

7 Какие системные значения использует экземпляр GestureDetector?

1 Сила тяготения

2 Сила притяжения

3 Сила давления

4 Сила инерции

8 Какие существуют модели систем передачи сообщений?

1 *sub/pub*

2 *pub/sub*

3 *rub/pub*

4 *pub/rub*

9 Какой правильный листинг программного кода обратного вызова?

1 *object Callbacks {*
fun requestData(callback: (data: Any?) –> Unit) {
// выполнить некоторый ввод/вывод с файлом
// или сетевым соединением.
val data = Any( )
callback(data)
} }

2
object Callbacks {
fun requestData(callback: (data: Any) –> Unit) {
// выполнить некоторый ввод/вывод с файлом
// или сетевым соединением.
val data = Any( )
callback(data)
} }

3
object Callbacks {
fun requestData(callback: (data: Any) –> Unit) {
// выполнить некоторый ввод/вывод с файлом

// или сетевым соединением.
val data = Any?( )
callback(data)
} }
4
object Callbacks {
fun requestData(callback: (data: Any?) –> Unit) {
// выполнить некоторый ввод/вывод с файлом
// или сетевым соединением.
val data = Any( )
callback(data)
}

10 Какой правильный синглтон, единственный экземпляр, существующий и используемый в приложении, применяемый для передачи сообщений между объектами, ничего не знающими друг о друге?

1 BroadcastReceiverManager
2 BroadcastSyncManager
3 ToastBroadcastManager
4 LocalBroadcastManager

11 Что хранит экземпляр IntentFilter?

1 Список строк с названиями экземпляров классов
2 Список строк с названиями действий
3 Список строк с названиями представлений
4 Список строк с названиями переменных приложения

12 Какие мультимедийные форматы поддерживает операционная система Android?

1 IMY
2 RTX
3 SBB
4 SMP

13 Какие существуют способы для записи и воспроизведения аудироликов?

1 AudioPlayer/AudioRecorder
2 MediaTrack/MediaRecorder
3 MediaPlayer/MediaRecorder
4 AudioTrack/AudioRecorder

14 Какие операции выполняются вне основного потока?

1 Доступ к файловой системе
2 Транзакции к базе данных
3 Сетевые запросы
4 Получение фокуса ввода

15 Какая правильная последовательность шагов выполнения работы вне основного потока?

1
Шаг 1. Создать новый экземпляр ThreadUI.

Шаг 2. Выполнить работу в методе run( ) экземпляра RunnJobable, переданного конструктору;

Шаг 3. Запустить поток вызовом метода start().

2

Шаг 1. Создать новый экземпляр Thread.

Шаг 2. Выполнить работу в методе run( ) экземпляра RunnWorkable, переданного конструктору;

Шаг 3. Запустить поток вызовом метода start().

3

Шаг 1. Создать новый экземпляр Thread.

Шаг 2. Выполнить работу в методе start( ) экземпляра Runnable, переданного конструктору;

Шаг 3. Запустить поток вызовом метода send( ).

4

Шаг 1. Создать новый экземпляр Thread.

Шаг 2. Выполнить работу в методе run( ) экземпляра Runnable, переданного конструктору;

Шаг 3. Запустить поток вызовом метода start( ).

16 Какие существуют способы отправки сообщений из фонового потока в основной?

1

1) View.send( );
2) Activity.runOnUiThread( );
3) Handler.

2

1) View.post( );
2) Activity.runOnUiThread( );
3) Handler.

3

1) View.post( );
2) Activity.runOnThread( );
3) Handler.

4

1) View.send( );
2) Activity.runOnThread( );
3) Handler.

17 Для чего предназначен метод interrupt( ) класса Thread?

1 Устанавливает логический флаг в потоке и не останавливает выполнение потока

2 Возвращает целое число, представляющее количество прочитанных байтов, значение (-1) обозначает конец потока.

3 Добавляет данные при внесении изменений в схему базы данных

4 Уничтожает экземпляр Activity

18 Касание – форма пользовательского ввода в большинстве современных мобильных приложений?

1 Да
2 Нет

19 Какие существуют классы обратной связи?

1
1) ViewController
2) Snackbar
3) Dialog

2
1) Toast
2) Snackbar
3) BottomBorderTextView

3
1) Toast
2) Snackbar
3) Dialog

4
1) Toast
2) SomeView
3) Dialog

20 Какие методы доступа к данным применяются в Android-приложениях?

1
1) непосредственный
2) предпочтения
3) база данных

2
1) приближённый
2) предпочтения
3) база данных

3
1) непосредственный
2) адаптивный
3) база данных

4
1) непосредственный
2) предпочтения
3) множественный

21 Предпочтения – это специальный механизм чтения и записи пар «ключ-значение» для примитивных типов данных?

1 Да
2 Нет

22 Какие существуют виды предпочтений?

1 Общие для всех активностей в приложении
2 Адаптивные для отдельных активностей
3 Универсальные для всех активностей в приложении

4 Настройки для отдельных активностей

23 Какие особенности работы с предпочтениями?

1

1) Хранение важных данных
2) Удаление данных вместе с приложением

2

1) Не хранить важные данные
2) Удаление данных вместе с приложением

3

1) Не хранить важные данные.
2) Отсутствие удаления данных вместе с приложением.

4

1) Хранение важных данных
2) Отсутствие удаления данных вместе с приложением

24 Какие существуют типы доступа к предпочтениям?

1 MODE_PUBLIC
2 MODE_WORLD_PUBLIC
3 MODE_PRIVATE
4 MODE_WORLD_PRIVATE

25 Какая система управления базами данных используется в операционной системе Android?

1 DB2
2 PostgreSQL
3 MySQL
4 SQLite

6.4.3 Таблица ответов

№ вопроса	Правильные ответы	
	Рубежный контроль №1	Рубежный контроль №2
	Вариант 1_1	Вариант 2_1
1	2	2, 4
2	1	3
3	1,3	1
4	2,3,4	3
5	2,4	4
6	1	3
7	2,4	1
8	3	2
9	1	1
10	3	4
11	2	2
12	4	1
13	3	3, 4

14	2	1, 2, 3
15	3	4
16	1,4	2
17	2	1
18	4	1
19	2	3
20	1	1
21	1,4	1
22	2,3	1, 4
23	2,4	2
24	1	3
25	4	4

6.4.4 Примерный перечень вопросов для зачёта

1 Особенности разработки мобильных приложений. Нативные и кроссплатформенные инструменты разработки.

2 Основы языка программирования Kotlin: преимущества перед языком Java, базовые понятия, операнды и операторы, типы данных, функции, управление ходом выполнения программы, пакеты, объектно-ориентированное программирование.

3 Контроллеры пользовательского интерфейса: создание начального контроллера пользовательского интерфейса приложения, изменение активного контроллера пользовательского интерфейса, этапы жизненного цикла контроллера пользовательского интерфейса.

4 Представления: создание нового представления, вложение представлений друг в друга, изменение состояния представлений.

5 Пользовательские компоненты: создание своего представления, использование своего представления.

6 Пользовательский ввод: получение события касания и реакция на него, получение события ввода с клавиатуры и реакция на него, обработка уже-стов.

7 Передача сообщений: использование обратных вызовов для реакции на действия, передача сообщений подписчикам, получение и обработка сообщений.

8 Файлы. Определение характеристик файла (размер или дата последнего изменения), чтение и запись данных в файлы, копирование данных из одного файла в другой.

9 Хранение данных: соединение с базой данных, создание таблицы или хранимого объекта, запись данных в таблицу или хранимый объект, чтение данных из таблицы или хранимого объекта.

10 Конкурентное (многопоточное) выполнение: запуск задачи в фоновом потоке, передача результатов из фонового потока в главный, завершение потока выполнения.

11 Сетевые взаимодействия: загрузка текстового файла с удалённого сервера и его вывод, создание запроса HTTP POST, загрузка двоичного файла.

12 Обратная связь с пользователем: отражение обратной связи с пользователем системных инструментов, snackbar, изменение строки состояния пользователя

13 Предпочтения пользователя: сохранения предпочтений пользователя, чтение предпочтений пользователя, работа с предпочтениями в многопользовательских приложениях.

14 Сериализация и транспорты: сериализация и десериализация экземпляров объектов.

15 Расширения: добавления новых возможностей в существующие API.

16 Тестирование: создание и запуск модульных тестов, создание и запуск интеграционных тестов.

6.4.5 Примерный перечень вопросов для экзамена

1 Особенности разработки мобильных приложений. Нативные и кроссплатформенные инструменты разработки.

2 Основы языка программирования Kotlin: преимущества перед языком Java, базовые понятия, операнды и операторы, типы данных, функции, управление ходом выполнения программы, пакеты, объектно-ориентированное программирование.

3 Контроллеры пользовательского интерфейса: создание начального контроллера пользовательского интерфейса приложения, изменение активного контроллера пользовательского интерфейса, этапы жизненного цикла контроллера пользовательского интерфейса.

4 Представления: создание нового представления, вложение представлений друг в друга, изменение состояния представлений.

5 Пользовательские компоненты: создание своего представления, использование своего представления.

6 Пользовательский ввод: получение события касания и реакция на него, получение события ввода с клавиатуры и реакция на него, обработка ужестов.

7 Передача сообщений: использование обратных вызовов для реакции на действия, передача сообщений подписчикам, получение и обработка сообщений.

8 Файлы. Определение характеристик файла (размер или дата последнего изменения), чтение и запись данных в файлы, копирование данных из одного файла в другой.

9 Хранение данных: соединение с базой данных, создание таблицы или хранимого объекта, запись данных в таблицу или хранимый объект, чтение данных из таблицы или хранимого объекта.

10 Конкурентное (многопоточное) выполнение: запуск задачи в фоновом потоке, передача результатов из фонового потока в главный, завершение потока выполнения.

11 Сетевые взаимодействия: загрузка текстового файла с удалённого сервера и его вывод, создание запроса HTTP POST, загрузка двоичного файла.

12 Обратная связь с пользователем: отражение обратной связи с пользователем системных инструментов, snackbar, изменение строки состояния пользователя

13 Предпочтения пользователя: сохранения предпочтений пользователя, чтение предпочтений пользователя, работа с предпочтениями в многопользовательских приложениях.

14 Сериализация и транспорты: сериализация и десериализация экземпляров объектов.

15 Расширения: добавления новых возможностей в существующие API.

16 Тестирование: создание и запуск модульных тестов, создание и запуск интеграционных тестов.

6.5. Фонд оценочных средств

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

7. ОСНОВНАЯ И ДОПОЛНИТЕЛЬНАЯ УЧЕБНАЯ ЛИТЕРАТУРА

7.1. Основная учебная литература

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

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

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

4 Сомон, П. Волшебство Kotlin : практическое руководство / П. Сомон ; пер. с анг. А. Н. Киселева. - Москва : ДМК Пресс, 2020. - 536 с. - ISBN 978-5-

97060-801-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/1094968.

7.2. Дополнительная учебная литература

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

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

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

8. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

1. Семахин А.М. Разработка мобильных приложений. Методические указания к выполнению лабораторных и контрольных работ для студентов направлений подготовки09.03.03 «Прикладная информатика», 09.03.04 «Программная инженерия». Курган, КГУ, 2021. – 48 с. (электронный).

4. Семахин А.М. Разработка мобильных приложений: учебное пособие. – Курган : Изд-во КГУ, 2021 – 76 с. (электронный).

9. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И ИНФОРМАЦИОННЫЕ СПРАВОЧНЫЕ СИСТЕМЫ

ЭБС «Лань»
ЭБС «Консультант студента»
ЭБС «Znanium.com»
«Гарант» – справочно-правовая система

10. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Материально-техническое обеспечение по реализации дисциплины осуществляется в соответствии с требованиями ФГОС ВО по данной образовательной программе.

ПРИЛОЖЕНИЕ 1

Аннотация к рабочей программе дисциплины

«Разработка мобильных приложений»

образовательной программы высшего образования –
программы бакалавриата

09.03.04 – Программная инженерия
Направленность:
Программное обеспечение автоматизированных систем

Трудоемкость дисциплины: 4 ЗЕ (144 академических часа)
Семестр: 7 (очная форма обучения), 10 (заочная форма обучения)
Форма промежуточной аттестации: зачёт (для очной формы обучения), экзамен (для заочной формы обучения)

Содержание дисциплины

Нативные и кроссплатформенные инструменты разработки. Архитектура iOS/Android. Нативный iOS. Нативный Android. Нативный Windows UMP. Архитектура Android. Уровень ядра. Уровень библиотек. Уровень каркаса приложений. Компоненты Android-приложения. Activity. Service. Broadcast Receiver. Архитектуры кроссплатформенных фреймворков:PhoneGap, ReactNative, Qt, Flutter, {amarin, {amarin.Forms.

Создание проекта Andoid. Навигация в Android Studio. Создание макета пользовательского интерфейса. Иерархия представления. Атрибуты виджетов. Создание строковых ресурсов. Предварительный просмотр макета. От разметки XML к объектам View. Ресурсы и идентификаторы ресурсов. Разработка виджетов. Установка ссылок на виджеты. Назначение слушателей. Уведомления. Выполнение в эмуляторе. Процесс сборки Android-приложений. Инструменты сборки.

Создание нового класса. Архитектура «Модель–Представление–Контроллер» и Android. Преимущества MVC. Обновление уровня представления. Обновление уровня контроллера. Добавление ресурсов в проект. Ссылка на ресурсы в XML. Запуск на устройстве. Многослойный MVVM. Декомпозиция по слоям. Связи внутри слоёв. Связи между слоями. Структуры данных на основе UI.

Использование неявных интентов. Строение неявного интента. Отправка отчёта. Запрос контакта у Android. Проверка реагирующих activity. Хранилище файлов. Использование FileProvider. Использование интента камеры. Масштабирование и отображение растровых изображений.

ConstraintLayout. Использование графического инструмента макетов. Освобождение пространства. Добавление виджетов. Внутренние механизмы

ConstrainLayout. Редактирование свойств. Динамическое поведение элементов списка. Стили, темы, атрибуты тем.

Библиотека компонентов архитектуры Room. Создание базы данных. Отделение сущности. Создание класса базы данных. Определение объекта доступа к данным. Доступ к базе данных с помощью шаблона репозитория. Тестирование запросов. Загрузка тестовых данных. Потоки приложения. Фоновые потоки. Использование LiveData.

Создание объекта SounPool. Доступ к активам. Воспроизведение звука. Зависимости от тестирования. Создание класса теста. Подготовка теста. Настройка тестируемых объектов. Написание тестов. Взаимодействия тестируемых объектов. Обратные вызовы привязки данных. Выгузка звуков.

Подготовка RecyclerView. Подготовка к загрузке через URL. Множественные загрузки. Создание фонового потока. Запуск и остановка HandlerThread. Сообщения и обработчики сообщений. Структура сообщений. Структура обработчика. Использование обработчиков. Прослушивание жизненного цикла представления. Сохранённые фрагменты.