ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
«ДОМ ЮНОШЕСКОГО ТЕХНИЧЕСКОГО ТВОРЧЕСТВА»
ЦЕНТР ЦИФРОВОГО ОБРАЗОВАНИЯ «IT-КУБ» Г. САТКА

ПРИНЯТО на заседании
педагогического совета
ГБУ ДО «ДЮТТ Челябинской области»
протокол № 135 от 15 июня 2023

УТВЕРЖДАЮ:
Директор ГБУ ДО «ДЮТТ
Челябинской области»
_____ В.Н. Халамов
Приказ №352 от «23» июня 2023 г.

СЕТЬ ЦЕНТРОВ ЦИФРОВОГО ОБРАЗОВАНИЯ ДЕТЕЙ «IT-КУБ»
IT-CUBE.САТКА

ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ОБЩЕРАЗВИВАЮЩАЯ ПРОГРАММА

«Разработка на языке Kotlin для платформы Android – Базовый уровень»

Направленность: техническая
Уровень программы: базовый
Срок освоения программы:1 год
Возрастная категория обучающихся: 14-18 лет

Автор-составитель:
Гайнанов Максим Вячеславович,
педагог дополнительного образования

г. Сатка
2023

Содержание

Раздел 1. Комплекс основных характеристик программы
1.1 Пояснительная записка 3
1.3. Цель и задачи программы 7
1.4. Содержание программы 7
1.5. Учебный план 25
1.6. Планируемые результаты 28
Раздел 2. Комплекс организационно-педагогических условий 29
2.1 календарный учебный график 29
2.2 Условия реализации программы 29
2.3. Формы аттестации 29
2.4. Оценочные материалы 30
2.5 Методические материалы 33
2.6 Воспитательные компоненты 33
2.7. Информационные ресурсы и литература 36
Приложение 37
Приложение 1 37
Приложение 2 38
Приложение 3 39

Раздел 1. Комплекс основных характеристик программы

1.1 Пояснительная записка

Настоящее Положение о порядке разработки и реализации дополнительных общеобразовательных программ ГБУ ДО «ДОТТ Челябинской области» разработано на основании:

Программа разработана на основании:

- Федерального закона от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации»;
- Федерального закона от 24.07.1998 № 124-ФЗ «Об основных гарантиях прав ребенка в Российской Федерации»;
- Распоряжения Правительства РФ от 12.11.2020 № 2945-р «Об утверждении плана мероприятий по реализации в 2021 — 2025 г. г. Стратегии развития воспитания в Российской Федерации на период до 2025 года»;
- Концепции развития дополнительного образования детей до 2030 года и плана мероприятий по ее реализации, утвержденной распоряжением Правительства РФ от 31.03.2022 № 678-р;
- Указа Президента Российской Федерации «Стратегия научно- технологического развития Российской Федерации» (редакция от 15.03.2021г. N*143);
- Постановления Главного государственного санитарного врача РФ от 28.09.2020 N. 28 «Об утверждении санитарных правил СП 2.4. 3648-20 «Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи»;
- Паспорта приоритетного проекта «Доступное дополнительное образование для детей», утвержденного президиумом Совета при Президенте РФ по стратегическому развитию и приоритетным проектам 30 ноября 2016 г;
- Приказ Министерства просвещения Российской Федерации от 27 июля 2022г. № 629 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»;
- Методических рекомендаций по проектированию дополнительных общеразвивающих программ, разработанных Министерством образования и науки России совместно с ГАОУ ВО «Московский государственный педагогический университет», ФГАУ «Федеральный институт развития образования», АНО дополнительного профессионального образования «Открытое образование»;
- Письмо Минобрнауки РФ от 18.11.2015 г. № 09-3242 «Методические рекомендации по проектированию дополнительных общеразвивающих программ (включая разноуровневые)»;
- Письмо Минобрнауки России от 29 марта 2016 г. № ВК-641/09 «О направлении методических рекомендаций» (вместе с «Методическими рекомендациями по реализации адаптированных дополнительных общеобразовательных программ, способствующих социально-психологической реабилитации, профессиональному самоопределению детей с ограниченными возможностями здоровья, включая детей-инвалидов, с учетом их особых образовательных потребностей»);
- Распоряжение Правительства ЧО № 901-рп от 20.09.2022 г. «Об утверждении регионального плана мероприятий на 2022 – 2024 годы по реализации Концепции развития дополнительного образования детей до 2030 года»;
- Приказ Министерства труда и социальной защиты РФ № 652-н от 21.09.2021 г «Об утверждении профессионального стандарта «Педагог дополнительного образования детей и взрослых»;
- Приказ Минпросвещения России от 03.09.2019 N 467 (ред. от 21.04.2023) «Об утверждении Целевой модели развития региональных систем дополнительного образования детей» (Зарегистрировано в Минюсте России 06.12.2019 N 56722);

- Закона Челябинской области от 29.08.2013 № 515-ЗО «Об образовании в Челябинской области»;
- Устава ГБУ ДО «Дом юношеского технического творчества Челябинской области»

Актуальность программы. Программа разработки на языке Kotlin для платформы Android является актуальной по нескольким причинам:

1. Kotlin - это язык программирования, разработанный JetBrains, который стал "официальным" языком разработки Android в 2017 году. Он предоставляет разработчикам множество преимуществ по сравнению с Java, таких как короткий и лаконичный синтаксис, поддержка функционального программирования, улучшенная безопасность типов и многое другое. Kotlin позволяет более эффективно и быстро писать код, что является основным преимуществом в разработке мобильных приложений.

2. Огромное сообщество разработчиков Kotlin на платформе Android. За несколько лет с момента принятия Kotlin в качестве официального языка для Android, множество разработчиков перешли на Kotlin и создали большое сообщество энтузиастов, которые активно делятся опытом, решениями и разработчиками библиотек на Kotlin. Это делает процесс разработки более удобным и эффективным.

3. Поддержка Kotlin во множестве инструментов и библиотек. Kotlin полностью совместим с Java, поэтому практически все существующие инструменты и библиотеки на Java могут быть использованы и в проектах на Kotlin. Кроме того, многие инструменты разработки, такие как Android Studio, поддерживают Kotlin "из коробки" и предоставляют средства для автоматического преобразования кода Java в Kotlin.

4. Kotlin продолжает развиваться и улучшаться. Kotlin активно разрабатывается и поддерживается компанией JetBrains. В последние годы он получил множество новых функций и дополнений, и его функциональность и производительность постоянно улучшаются. Это означает, что разработчики могут быть уверены в том, что язык Kotlin будет развиваться и соответствовать современным требованиям разработки мобильных приложений.

В целом, Разработка на языке Kotlin для платформы Android – Базовый уровень является актуальной и востребованной, и участие в программе позволит разработчикам освоить современные технологии и инструменты для создания качественных мобильных приложений.

Педагогическая целесообразность. Программа "Разработка на языке Kotlin для платформы Android – Базовый уровень" имеет ряд педагогических преимуществ и целесообразна по нескольким причинам:

1. Заинтересованность обучающихся: Мобильные приложения - это тема, которая актуальна и интересна для большинства современных обучающихся. Разработка мобильных приложений и игр является популярной и захватывающей деятельностью, которая может вызывать положительные эмоции и мотивацию для изучения и углубления в эту область.

2. Практическое применение знаний: Программа "Разработка на языке Kotlin для платформы Android – Базовый уровень" ориентирована на практическую разработку мобильных проектов. Обучающиеся имеют возможность применить полученные знания и навыки в создании собственных игр и приложений. Это помогает им понять, как применять теоретические знания на практике и улучшить свои навыки программирования.

3. Развитие творческого мышления: Разработка на языке Kotlin для платформы Android – Базовый уровень требует творческого подхода и воображения для создания уникальных и интересных проектов. В процессе обучения обучающиеся будут сталкиваться с различными задачами и проблемами, которые потребуют от них поиска нестандартных решений и развития творческого мышления.

4. Командная работа: Разработка мобильного приложения часто требует командной работы. Программа позволяет учиться работать в команде, обмениваться знаниями и навыками, а также эффективно совместно решать задачи и достигать общих целей.

5. Подготовка к профессиональной деятельности: Программа "Разработка на языке Kotlin для платформы Android – Базовый уровень" предоставляет базовые знания и навыки в области разработки мобильных проектов. Это может стать отличной отправной точкой для дальнейшей профессиональной деятельности в области мобильных технологий и программирования.

В целом, программа "Разработка на языке Kotlin для платформы Android – Базовый уровень" имеет педагогическую целесообразность, так как она сочетает в себе актуальность темы, практическое применение знаний и развитие ключевых навыков, необходимых для успешной работы в области мобильной разработки.

Отличительные особенности программы заключаются в том, что она учитывает новые технологические уклады, которые требуют новый способ мышления и тесного взаимодействия при постоянном повышении уровня междисциплинарности проектов, а также использует новые формы диагностики и подведения итогов реализации программы, выполняемые в формате защиты проектов и участия во Всероссийском конкурсе мобильных приложений.

Адресат программы – школьники 14-18 лет, проявляющие интерес к информационным технологиям, и имеющие фундаментальные знания по математике, информатике и английскому языку. На обучение принимаются все желающие, без предварительной подготовки, по заявлению родителей или лиц, их заменяющих. Набор в объединение производится по желанию обучающихся и их родителей.

Особенности развития детей среднего школьного возраста: формируется самосознание – представление о себе самом, самооценивание умственных, моральных, волевых качеств. Происходит соотношение себя с идеалом, появляется возможность самовоспитания. Возрастает волевая регуляция. Ведущая деятельность – учебно-профессиональная. Стремление приобрести профессию – основной мотив познавательной деятельности. Возрастает концентрация внимания, объем памяти, сформировалось абстрактно-логическое мышление. Появляется умение самостоятельно разбираться в сложных вопросах. Формируется собственное мировоззрение, как целостная система взглядов, знаний, убеждений, своей жизненной философии. Стремление к самоуправлению, стремление заново осмыслить все окружающее, происходит жизненное определение человека.

Срок реализации и объем программы определяется содержанием программы составляет 1 год (144 академических часа).

Направленность: техническая.

Язык реализации программы: русский.

Особенности реализации программы: модульный принцип.

Уровень освоения программы: базовый.

Форма обучения – очная, с возможностью применения дистанционных технологий.

Формы организации: в группе 12 человек.

Режим занятий: 4 академических часа в неделю. 1 раз - 2 часа (академический час – 45 мин.). Через каждые 45 минут занятия следует 15-минутный перерыв, согласно требованиям, СанПиН.

Форма организации занятий: групповое, индивидуально-групповое.

Методы обучения: наглядный, практический, объяснительно- иллюстративный.

1.2. Сведения о программе

Название программы	Разработка на языке Kotlin для платформы Android – Базовый уровень (14-18 лет)
Возраст обучающихся	14-18 лет
Длительность программы (в часах)	144 часа
Количество занятий в неделю	2 часа 2 раза в неделю (академический час – 45 мин)
Цель, задачи	Цель: развитие интереса обучающихся к информационным и телекоммуникационным технологиям; реализация их творческих идей в области мобильной разработки. Задачи направлены на достижение цели и включают в себя обучающие, развивающие, воспитательные.
Краткое описание программы	Программа "Разработка на языке Kotlin для платформы Android – Базовый уровень" предлагает изучение и практическое применение языка программирования Kotlin для создания мобильных приложений для платформы Android. В ходе программы научатся работать над практическими проектами, решать задачи и создавать свои собственные мобильные приложения на платформе Android с использованием языка Kotlin. Они также будут ознакомлены с современными инструментами разработки и методологиями разработки мобильных приложений. Цель программы - обеспечить участникам навыки и практический опыт разработки приложений на языке Kotlin для платформы Android, чтобы они могли успешно создавать и поддерживать высококачественные мобильные приложения.
Первичные знания, необходимые для освоения программы	Базовые знания, полученные при изучении школьной программы.
Результат освоения	приобретение навыков и знаний для самостоятельной разработки мобильных приложений с использованием языка Kotlin. Они смогут создавать пользовательские интерфейсы, работать с данными, интегрировать сторонние сервисы и библиотеки, оптимизировать приложения.
Перечень соревнований, в которых учащиеся смогут принять участие	Конференция «Юные техники – инженеры» Фестиваль по IT-технологиям Ярмарка проектов “Играойя” (г.Сатка) Всероссийский конкурс мобильных приложений
Перечень основного оборудования, необходимого для освоения программы	Планшет, ноутбук, WEB-камера, наушники, моноблочное интерактивное устройство, напольная мобильная стойка для интерактивных досок или универсальное настенное крепление, сетевой фильтр

Преимущества данной программы (отличия от других подобных курсов)	Данная программа формирует профессиональные компетенции, которые позволят обучающимся в будущем успешно конкурировать в области мобильной разработки. Программа плотно связана с массовыми мероприятиями в научно-технической сфере для обучающихся (турнирами, состязаниями, конференциями), что позволяет, не выходя за рамки учебного процесса, принимать активное участие в конкурсах различного уровня

1.3. Цель и задачи программы

Целью программы является развитие интереса обучающихся к информационным, телекоммуникационным технологиям и реализация их творческих идей в области мобильной разработки.

Задачи:

Личностные:

• Формирование умения самостоятельной деятельности.

• Формирование умения работать в команде.

• Формирование коммуникативных навыков.

• Формирование навыков анализа и самоанализа.

• Формирование целеустремленности и усидчивости в процессе творческой, исследовательской работы и учебной деятельности.

Предметные:

• Формирование представления о программном обеспечении и сетевом оборудовании организаций.

• Формирование представления об устройстве персонального компьютера и принципе его работы.

• Формирование представления о принципах работы сетей.

• Формирование умений по работе с различным программным обеспечением.

Метапредметные:

• Формирование умения ориентироваться в системе знаний.

• Формирование умения выбирать наиболее эффективные способы решения задач на компьютере в зависимости от конкретных условий.

• Формирование приемов проектной деятельности, включая умения видеть проблему, формулировать тему и цель проекта, составлять план своей деятельности, осуществлять действия по реализации плана, результат своей деятельности соотносить с целью, классифицировать, наблюдать, проводить эксперименты, делать выводы и заключения, доказывать, защищать свои идеи, оценивать результаты своей работы.

• Формирование умения распределения времени.

• Формирование умений успешной самопрезентации.

1.4. Содержание программы

1.Введение. Знакомство.

Теория: Начальное знакомство. Описание задач на год.

Практика: Первичная диагностика. Тестирование.

Модуль 1. Введение в Kotlin и платформу Android

1.1. Что такое Kotlin и почему он используется для разработки на платформе Android

Теория: Kotlin — это статически типизированный язык программирования, который работает на платформе Java Virtual Machine (JVM). Он был создан командой разработчиков из компании JetBrains в 2011 году и официально представлен для разработки на платформе Android в 2017 году.

Практика:

1. Создайте новый проект в Android Studio и выберите язык программирования Kotlin.
2. Ознакомьтесь с базовыми синтаксическими конструкциями Kotlin, такими как переменные, функции, условные операторы и циклы.
3. Используйте Kotlin Android Extensions для удобного доступа к элементам пользовательского интерфейса (UI) в коде приложения.
4. Разработайте логику приложения, включая обработку событий, взаимодействие с базой данных и сетевые запросы.
5. Используйте библиотеки и фреймворки, доступные для Kotlin, чтобы упростить разработку и добавить функциональность вашему приложению.
6. Отлаживайте и тестируйте ваше приложение, исправляя ошибки и обеспечивая правильное

1.2. Основы платформы Android и структура приложения

Теория: Основы платформы Android:

- История и эволюция Android
- Архитектура Android: ядро Linux, слои системы, приложения
- Виртуальная машина Dalvik и ART
- Разновидности устройств Android: телефоны, планшеты, телевизоры, носимые устройства и т. д.

Практика: Создание проекта в Android Studio:

- Установка среды разработки Android Studio
- Создание нового проекта
- Настройка настройки проекта: минимальная версия Android, пакет приложения, иконки и т. д.

1.3. Создание первого проекта на Kotlin

Теория:

Создание проекта в Android Studio:

- Установка среды разработки Android Studio
- Создание нового проекта
- Настройка настройки проекта: минимальная версия Android, пакет приложения, иконки и т. д.

Практика: При создании первого проекта на Kotlin для мобильных игр и приложений важно ознакомиться с основными компонентами Android-разработки, которые используются в процессе разработки

Модуль 2. Основы синтаксиса Kotlin

2.1. Приведение примеров использования переменных, функций и классов в Kotlin

Теория: Kotlin - объектно-ориентированный язык программирования, который поддерживает все основные концепции ООП, такие как классы, объекты, наследование, полиморфизм и инкапсуляцию.

Практика: Использования переменных, функций и классов в Kotlin для разработки мобильных игр и приложений

2.2. Создание функций с различными параметрами и возвращаемыми значениями

Теория: В Kotlin вы можете создавать функции с различными параметрами и возвращаемыми значениями, что позволяет вам создавать более гибкий и мощный код.

Практика: Простая функция без параметров и возвращаемого значения, Функция с параметрами, Функция с параметрами по умолчанию, Перегрузка функций, Функция высшего порядка.

2.3. Изучение механизмов управления потоками

Теория: Механизмы управления потоками являются важной частью разработки мобильных игр и приложений. Потоки используются для выполнения параллельных операций, таких как загрузка изображений, обработка пользовательских вводов, анимация и обновление экрана.

Практика:

1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.
2. Создайте новый класс, расширяющий AsyncTask. Реализуйте методы doInBackground(), onPreExecute() и onPostExecute().
3. В методе doInBackground() выполните фоновую операцию, например, загрузку данных из сети или обработку изображений.
4. В методе onPreExecute() выполняйте настройку пользовательского интерфейса, например, отображение прогресс-бара или текстового сообщения.
5. В методе onPostExecute() обновляйте пользовательский интерфейс на основе результатов фоновой операции, например, отображение полученных данных или обработанных изображений.
6. Используйте созданный класс AsyncTask в вашем приложении, чтобы выполнить фоновую операцию и взаимодействовать с пользовательским интерфейсом.
7. Тестирование вашего приложения и отладка потенциальных проблем с потоками и синхронизацией доступа к ресурсам.

Модуль 3. Изучение пользовательского интерфейса Android

3.1. Основы построения пользовательского интерфейса в Android

Теория: при разработке приложений для платформы Android важную роль играет построение пользовательского интерфейса (UI). Пользовательский интерфейс - это то, через что пользователь взаимодействует с приложением.

Практика:

1. Создайте новый проект Android Studio и настройте его для разработки на языке Kotlin.
2. Откройте файл разметки (XML), который представляет пользовательский интерфейс вашей активности. Этот файл находится в папке res/layout.
3. Используйте компоновщики (layout managers) и представления (views) для создания пользовательского интерфейса. Например, добавьте кнопки, текстовые поля, изображения и другие элементы на экран.
4. Настройте свойства элементов пользовательского интерфейса с помощью атрибутов XML, таких как размеры, цвета, шрифты и другие.
5. В вашей активности Kotlin свяжите элементы пользовательского интерфейса (views) с соответствующими переменными Kotlin, используя findViewById() или синтаксис свойств (property syntax).
6. Обработайте события пользовательского взаимодействия, такие как нажатия кнопок или изменения текста в текстовых полях. Для этого можно использовать слушатели событий (event listeners) или привязку данных (data binding).
7. Запустите приложение на эмуляторе или физическом устройстве Android и проверьте, как работает ваш пользовательский интерфейс.

3.2. Работа с различными элементами пользовательского интерфейса: текст, изображения, кнопки и т.д.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Откройте макет вашей активности (activity_main.xml) и разместите на нем несколько элементов пользовательского интерфейса, например, TextView, ImageView и Button.
3. В файле MainActivity.kt свяжите созданные элементы с кодом Kotlin с помощью функции findViewById() и получите ссылки на них.
4. Используйте методы и свойства соответствующих элементов пользовательского интерфейса для изменения их содержимого и поведения. Например, используйте метод setText() для изменения текста TextView, setImageResource() для загрузки изображения в ImageView и установите слушатели OnClickListener для обработки нажатий на кнопку.

5. Запустите приложение на эмуляторе или физическом устройстве Android и проверьте работу элементов пользовательского интерфейса.

3.3. Изучение работы с языком разметки XML

Теория: Язык разметки XML (eXtensible Markup Language) играет важную роль в разработке приложений для платформы Android. XML используется для создания пользовательского интерфейса, описания ресурсов, настроек приложения и других структурированных данных.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Откройте файл разметки activity_main.xml и изучите его содержимое.

3. Измените элементы XML-разметки, добавляя новые элементы или изменяя существующие атрибуты.

4. Запустите приложение на эмуляторе или реальном устройстве и проверьте изменения в пользовательском интерфейсе.

5. Изучите документацию Android SDK и примеры кода для более сложных элементов и атрибутов XML-разметки.

6. Практикуйте создание различных макетов пользовательского интерфейса, используя разные элементы и атрибуты XML.

7. Тестирование и отладка вашего приложения, чтобы убедиться, что пользовательский интерфейс работает должным образом.

Модуль 4. Работа со списками и адаптерами

4.1. Изучение основ работы со списками на примере ListView

Теория: В разработке приложений для платформы Android широко используются списки для отображения данных. Одним из основных инструментов для работы со списками в Android является компонент ListView. ListView представляет собой прокручиваемый список элементов.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Откройте файл разметки activity_main.xml и добавьте компонент ListView

3. Создайте новый файл activity.kt и определите в нем класс активности. В методе onCreate() инициализируйте ListView и установите адаптер

4. Запустите приложение на эмуляторе или реальном устройстве и проверьте отображение списка с тремя элементами "Item 1", "Item 2" и "Item 3".

5. Добавьте обработку события клика на элементе списка. Для этого добавьте метод setOnItemClickListener() к ListView

6. Внутри метода setOnItemClickListener() добавьте необходимые действия при клике на элементе списка, например, открытие новой активности или вывод дополнительной информации о выбранном элементе.

4.2. Создание пользовательских адаптеров. Интеграция адаптеров с пользовательским интерфейсом

Теория: Пользовательские адаптеры являются важной частью разработки мобильных приложений для платформы Android. Они используются для связи данных с пользовательским интерфейсом, особенно для отображения списков данных.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Создайте новый класс, расширяющий класс ArrayAdapter или RecyclerView.Adapter, в зависимости от выбранной библиотеки для отображения списков.

3. Реализуйте необходимые методы адаптера, такие как getCount(), getItem(), getView() или onCreateViewHolder(), onBindViewHolder(), getItemCount().

4. Создайте layout-файл для элемента списка и определите необходимые элементы пользовательского интерфейса.

5. В методе getView() или onCreateViewHolder() создайте новый экземпляр ViewHolder и заполните его ссылками на элементы пользовательского интерфейса.

6. В методе onBindViewHolder() заполните элементы пользовательского интерфейса данными, связанными с текущим элементом списка.

7. В активности или фрагменте, где будет отображаться список, создайте экземпляр адаптера и свяжите его с ListView или RecyclerView. Установите адаптер для списка.

8. Заполните данные в адаптере, чтобы они отобразились в списке.

9. Добавьте обработчики событий для элементов списка, если необходимо.

10. Тестирование вашего приложения и отладка проблем, связанных с адаптером и интеграцией с пользовательским интерфейсом.

Модуль 5. Работа с базами данных

5.1. Изучение принципов работы и использования баз данных в Android

Теория: Принципы работы и использования баз данных в Android играют важную роль в разработке мобильных приложений. База данных используется для хранения и организации структурированных данных, таких как пользовательские данные, настройки приложения, кэшированная информация и многое другое.

Практика:

1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Определите схему базы данных, включая создание таблиц, указание полей, индексов и ограничений.

3. Создайте класс, расширяющий SQLiteOpenHelper, и определите методы onCreate() и onUpgrade().

4. В методе onCreate() создайте таблицы и необходимые индексы согласно определенной схеме базы данных.

5. В методе onUpgrade() выполните обновление схемы базы данных при необходимости.

6. Создайте класс (или классы) для взаимодействия с базой данных, реализовав необходимые методы для работы с данными.

7. Используйте созданный класс для выполнения операций с базой данных в других частях вашего приложения, таких как активности или фрагменты.

8. При необходимости создайте контент-провайдер для предоставления доступа

5.2. Работа с базой данных SQLite с помощью API Room

Теория: Работа с базой данных SQLite является важным аспектом разработки мобильных приложений для платформы Android. Для упрощения взаимодействия с базой данных существует API Room, которое предоставляет удобные средства для создания, доступа и управления базой данных SQLite.

Практика:

1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Добавьте зависимость для API Room в файле build.gradle.

3. Определите Entity классы, которые представляют таблицы в базе данных, с помощью аннотации @Entity и аннотаций для определения столбцов и первичного ключа.

4. Создайте интерфейсы DAO, которые объявляют методы для доступа и управления данными в базе данных, с помощью аннотации @Dao и аннотаций для определения SQL-запросов или операций.

5. Создайте класс Database, который расширяет класс RoomDatabase, и отметьте его аннотацией @Database, указывая версию базы данных и список Entity классов.

6. В каждом DAO интерфейсе определите методы для выполнения операций с базой данных, используя аннотации, такие как @Insert, @Query или @Update.

7. В вашем приложении создайте экземпляр Database класса, используя метод Room.databaseBuilder(), и получите экземпляр DAO с помощью метода database.dao().

8. Используйте методы DAO для выполнения операций с базой данных, таких как вставка, выборка или обновление данных.

9. Тестирование вашего приложения и отладка проблем, связанных с базой данных и интеграцией с API Room.

5.3. Создание, обновление и удаление записей в базе данных

Практика:

1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Создайте класс, который расширяет класс SQLiteOpenHelper, и переопределите его методы onCreate(), onUpgrade() и другие, по мере необходимости.

3. В методе onCreate() создайте таблицу в базе данных с помощью SQL выражения CREATE TABLE.

4. В методе onUpgrade() обновите схему базы данных, если это необходимо.

5. Создайте класс, представляющий сущность базы данных, например, таблицу или модель данных. Определите поля и методы для добавления, обновления и удаления записей. Используйте методы SQLiteDatabase для выполнения операций с базой данных.

6. Получите ссылку на базу данных и используйте методы класса для добавления, обновления и удаления записей.

7. Проверьте результаты операций и убедитесь, что данные правильно вставляются, обновляются и удаляются.

8. Обработайте запросы на выборку данных с помощью методов класса и отобразите данные на пользовательском интерфейсе вашего приложения.

Модуль 6. Работа с сенсорным экраном, геолокацией и сетевыми запросами

6.1. Промежуточная Аттестация. Тестирование

Теория: Тестирование

6.2. Изучение работы с сенсорным экраном и извлечение информации из него

Теория: Разработка на языке Kotlin для платформы Android – Базовый уровень включает работу с сенсорным экраном устройства. Сенсорный экран является основной формой ввода информации в большинстве современных мобильных устройств, и поэтому очень важно уметь работать с ним.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Добавьте разрешение доступа к сенсорному экрану в манифест-файле вашего приложения.

3. Создайте пользовательский интерфейс, который будет взаимодействовать с сенсорным экраном. Например, добавьте кнопку или поле для отображения информации о касании.

4. В коде вашей активности или фрагмента добавьте код для обработки касания на сенсорном экране. Используйте методы класса MotionEvent для получения информации о касании, такой как координаты и тип действия.

5. Обработайте информацию о касании согласно вашим потребностям. Например, отобразите информацию о касании на пользовательском интерфейсе или выполните определенные действия на основе типа действия.

6. При необходимости добавьте обработку жестов с помощью классов GestureDetector или ScaleGestureDetector. Создайте экземпляры этих классов и переопределите методы обратного вызова для обработки жестов.

6.3. Изучение использования геолокации на Android

Теория: Использование геолокации на платформе Android - это важный аспект разработки приложений, который позволяет получить информацию о текущем местоположении устройства. Это может быть полезно для различных сценариев, таких как поиск ближайших объектов, навигация, отслеживание перемещений пользователя и другие.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Добавьте разрешение ACCESS_FINE_LOCATION или ACCESS_COARSE_LOCATION в файл манифеста приложения, чтобы запросить доступ к геолокации.

3. Создайте класс для работы с геолокацией, который будет управлять подключением к службе геолокации и запросом обновлений местоположения.

4. Используйте Fused Location Provider API для подключения к службе геолокации и запроса обновлений местоположения. Обработайте полученные данные и отобразите их на пользовательском интерфейсе или выполните другие действия с ними.

5. Используйте API геолокации для определения текущего местоположения пользователя и получения дополнительной информации о местоположении, такой как адрес или страна.

6. Используйте API геолокации для поиска ближайших мест или объектов, таких как рестораны, магазины или банкоматы. Отобразите результаты на карте или в списке.

7. Используйте API геолокации для навигации между двумя точками. Рассчитайте расстояние между двумя координатами, отобразите маршрут на карте и предоставьте инструкции по навигации.

8. Тестируйте приложение и убедитесь, что использование геолокации работает правильно и соответствует ожиданиям.

6.4. Работа с сетевыми запросами

Теория: Разработка на языке Kotlin для платформы Android – Базовый уровень включает работу с сетевыми запросами, чтобы получать и отправлять данные через сеть. Для выполнения сетевых запросов в разработке Android используются библиотеки, такие как Retrofit или Volley.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Добавьте зависимость для Retrofit в файле build.gradle.

3. Создайте интерфейс, который будет служить API для выполнения сетевых запросов. Определите методы с использованием аннотации @GET, @POST и других, указав URL-адрес и другие параметры запроса.

4. Создайте объект данных, который будет представлять тело запроса или данные, которые вы получите в ответе.

5. Создайте экземпляр Retrofit с использованием метода Retrofit.Builder(). Установите базовый URL-адрес и добавьте конвертеры данных (например, Gson) при необходимости.

6. Создайте экземпляр интерфейса с помощью метода create(), который возвращает объект, реализующий этот интерфейс.

7. Вызовите методы интерфейса, чтобы выполнить сетевые запросы и получить ответы.

8. Обработайте ответы в коллбеках или с помощью сопрограмм Kotlin, проверьте статус кода ответа и преобразуйте данные в необходимом формате.

9. Отобразите данные на пользовательском интерфейсе вашего приложения или выполняйте другие операции с полученными данными.

Модуль 7. Работа с мультимедиа и уведомлениями

7.1. Изучение работы с мультимедиа на Android

Теория: Разработка на языке Kotlin для платформы Android – Базовый уровень включает работу с мультимедиа, такую как аудио, видео и изображения. Для работы с мультимедиа на Android используются различные API и библиотеки, такие как MediaPlayer, ExoPlayer, Glide или Picasso.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на языке Kotlin.

2. Добавьте необходимые зависимости или библиотеки для работы с мультимедиа, такие как MediaPlayer, ExoPlayer, Glide или Picasso, в файл build.gradle.

3. Создайте элементы интерфейса пользователя, такие как кнопки для управления воспроизведением аудио или видео, или ImageView для отображения изображений.

4. Импортируйте необходимые классы или объекты API или библиотеки в код Kotlin.

5. Создайте экземпляры классов, которые представляют мультимедийные ресурсы, такие как MediaPlayer или ImageView.

6. Настройте параметры для мультимедийных ресурсов, такие как громкость или размер изображения, с помощью соответствующих методов.

7.2. Создание пользовательских уведомлений/Интеграция уведомлений с пользовательским интерфейсом

Теория: Создание пользовательских уведомлений в разработке на Kotlin для платформы Android включает работу с классами и методами, которые предоставляют Android API для управления уведомлениями. Уведомления являются важным средством связи с пользователем и могут быть использованы для различных целей, таких как информирование о новых сообщениях, напоминания о событиях, предупреждения о важных обновлениях и т.д.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Определите, где и когда должно отображаться уведомление в вашем пользовательском интерфейсе, например, при нажатии кнопки или при получении нового сообщения.

3. Создайте экземпляр класса NotificationManager для управления уведомлениями

Модуль 8. Создание приложений с использованием паттерна MVVM

8.1. Изучение паттерна MVVM и его реализации на Kotlin

Теория: Паттерн MVVM (Model-View-ViewModel) является одним из популярных паттернов архитектуры, который используется в разработке на Kotlin для платформы Android. Он предлагает разделение бизнес-логики приложения от пользовательского интерфейса и упрощает тестирование и поддержку кода.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Создайте классы для модели данных, которые представляют данные, необходимые вашему приложению.

3. Создайте классы модели представления, в которых будет содержаться логика и данные, связанные с вашими представлениями.

4. Создайте пользовательский интерфейс с помощью разметки XML или программного кода.

5. Свяжите данные из модели представления с элементами пользовательского интерфейса, используя механизмы двустороннего связывания данных (например, с помощью Data Binding).

8.2. Создание приложения с использованием MVVM

Теория: Разработка приложения на языке Kotlin для платформы Android с использованием архитектурного паттерна MVVM включает организацию кода приложения на три основных компонента: Модель (Model), Представление (View) и Модель представления (ViewModel). MVVM помогает разделить бизнес-логику приложения от его пользовательского интерфейса, упрощает тестирование и повышает его поддерживаемость.

Практика:

1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Определите компоненты модели, представления и модели представления вашего приложения.

3. Создайте классы для модели, представления и модели представления.

4. Определите данные и методы, необходимые в каждом из этих классов.

5. Реализуйте пользовательский интерфейс вашего приложения с использованием разметки XML и соответствующих классов Activity

8.3. Изучение практических примеров использования MVVM

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Определите модель, которая будет предоставлять данные и бизнес-логику вашего приложения. Создайте классы, которые будут представлять данные и методы для получения и обновления данных.

3. Определите представление, которое будет отображать данные пользователю и реагировать на пользовательский ввод. Создайте активити, фрагменты или другие пользовательские интерфейсы, которые будут отображать данные и иметь элементы пользовательского ввода.

4. Определите представление-модель, которая будет связывать модель и представление. Создайте классы ViewModel, которые будут содержать методы для обработки пользовательского ввода и обновления данных из модели.

Модуль 9. Изучение архитектурных компонентов

9.1. Изучение архитектурных компонентов Android Jetpack

Теория: Android Jetpack — это набор компонентов и библиотек, предоставляемых Google для упрощения разработки приложений для платформы Android. Он включает в себя различные архитектурные компоненты, которые обеспечивают более эффективную и масштабируемую разработку приложений. Ниже приведены основные компоненты Android Jetpack

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Изучите документацию и примеры использования каждого компонента Android Jetpack.

3. Определите, какие компоненты Android Jetpack будут наиболее полезны для вашего проекта.

4. Создайте классы и код для использования выбранных компонентов Android Jetpack в вашем приложении.

5. Протестируйте функциональность каждого компонента Android Jetpack в вашем приложении и убедитесь, что они работают как, ожидается.

6. Используйте документацию и руководства по использованию Android Jetpack, чтобы узнать больше о каждом компоненте

9.2. Создание приложений с использованием Room, ViewModel и LiveData

Теория: Room, ViewModel и LiveData — это популярные компоненты, которые используются при разработке приложений на языке Kotlin для платформы Android. Они предоставляют удобные средства для работы с базой данных и связи данных между моделью и представлением.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Определите структуру базы данных и создайте сущности, используя аннотации Room. Это могут быть таблицы базы данных и классы, представляющие данные.

3. Создайте интерфейсы DAO для доступа к данным в базе данных. Определите методы для выполнения операций чтения и записи данных в базу данных.

4. Создайте класс базы данных, который наследуется от RoomDatabase. Внутри этого класса определите абстрактные методы для доступа к DAO и настройте параметры базы данных.

5. Создайте класс ViewModel, который наследуется от ViewModel. Внутри ViewModel определите методы для получения данных из базы данных и обновления данных.

6. Свяжите ViewModel с представлением, например, с помощью LiveData. Наблюдайте за изменениями данных в ViewModel и обновляйте представление при изменении данных.

7. Реализуйте логику представления, которая отображает данные из ViewModel и реагирует на пользовательский ввод. Свяжите пользовательский ввод с методами ViewModel для обновления данных.

8. Проверьте работу приложения, проверьте, что данные регулярно обновляются из базы данных и отображаются в представлении.

9.3. Ввод в понятие Navigation Components

Теория: Navigation Components — это часть Android Jetpack, предоставляющая инструменты и рекомендации для навигации по различным экранам вашего приложения на платформе Android. Основная цель Navigation Components - упростить и улучшить навигацию внутри приложения, уменьшить количество кода и облегчить его поддержку.

Практика: 1. Обновите свой проект до последней версии Android Gradle Plugin и добавьте зависимость на Navigation Components в файл build.gradle.

2. Создайте навигационный граф в файле XML, определите различные экраны и связи между ними. Используйте визуальный редактор для упрощения этого процесса.

3. Создайте навигационный хост, который будет отображать различные экраны вашего приложения.

Модуль 10. Изучение многомодульной архитектуры

10.1. Изучение базовой концепции многомодульной архитектуры на примере приложения

Теория: Многомодульная архитектура — это подход к разработке приложений, когда приложение состоит из нескольких независимых модулей, каждый из которых отвечает за конкретную функциональность или слой приложения. Это позволяет лучше организовать код, повысить его переиспользуемость, облегчить сопровождение и более эффективно работать в команде.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Определите функциональность вашего приложения и разбейте ее на отдельные модули. Например, вы можете создать модуль для работы с базой данных, модуль для работы с сетью и модуль для пользовательского интерфейса.

3. Создайте модули Gradle для каждого модуля вашего приложения. В каждом модуле определите зависимости от других модулей, которые они используют.

4. Реализуйте функциональность каждого модуля. Каждый модуль должен быть независимым и иметь свою собственную структуру каталогов.

5. Определите интерфейсы и контракты, которые модули будут использовать для взаимодействия друг с другом. Это может быть набор интерфейсов для работы с базой данных, классы для передачи данных через сеть и т. д.

6. Реализуйте взаимодействие между модулями через определенные интерфейсы и контракты. Каждый модуль должен использовать только публичные методы и интерфейсы, определенные в других модулях.

10.2. Создание и интеграция различных модулей

Теория: В разработке на языке Kotlin для платформы Android интеграция различных модулей является важной практикой. Модули позволяют разделить функциональность приложения на отдельные части, что облегчает поддержку, тестирование и масштабирование приложения.

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Определите функциональность приложения и разделите ее на отдельные модули. Модули могут быть библиотечными, фич-модулями или динамическими модулями, в зависимости от требований.

3. Создайте отдельные модули для разных функциональностей приложения. Создайте отдельные проекты для библиотечных модулей или добавьте новые модули в основной проект, если они являются частью основного приложения.

4. Определите зависимости между модулями. Добавьте зависимости между модулями в файлы build.gradle модулей. Используйте механизмы зависимостей, предоставляемые Android Studio, такие как implementation и api, чтобы определить, какие модули должны быть доступны из других модулей.

5. Разработайте функциональность внутри каждого модуля, включая бизнес-логику, пользовательский интерфейс и другие необходимые компоненты.

6. Объедините модули вместе и проверьте работу приложения.

Модуль 11. Работа с FireBase

11.1. Изучение сервиса Firebase и его использование для разработки приложений

Теория: Firebase — это сервис разработки приложений, предоставляемый компанией Google. С его помощью можно создавать приложения для мобильных, веб- и серверных платформ. Firebase включает в себя множество инструментов и сервисов для разработки и управления

приложениями, включая аутентификацию пользователей, хранение и синхронизацию данных, аналитику и многое другое.

Практика: 1. Создайте новый проект Android Studio с поддержкой языка Kotlin.

2. Подключите сервис Firebase к вашему проекту. Включите необходимые модули Firebase, такие как Firebase аутентификация, Firebase Realtime Database и Firebase Cloud Messaging.

3. Настройте аутентификацию пользователей с использованием Firebase. Реализуйте функции регистрации, входа и выхода пользователей, используя методы Firebase аутентификации.

4. Используйте Firebase Realtime Database для хранения и синхронизации данных в реальном времени. Создайте модели данных и настройте их связь с Firebase Database. Работайте с данными: добавляйте, изменяйте и удаляйте их, и наблюдайте за их синхронизацией с другими подключенными клиентами.

5. Используйте Firebase Cloud Messaging для отправки уведомлений на мобильные устройства. Настройте прием уведомлений на вашем Android-устройстве и отправьте тестов

11.2. Работа с базой данных Firebase

Практика: 1. Создайте новый проект Android Studio и настройте его для разработки на Kotlin.

2. Создайте проект Firebase и настройте его для работы с базой данных Firebase Realtime Database или Firestore.

3. Добавьте зависимости Firebase в файле build.gradle вашего проекта.

4. Инициализируйте Firebase в вашем приложении.

5. Создайте класс для доступа к базе данных, включая получение, запись и удаление данных.

6. Используйте методы доступа к базе данных для получения данных из базы данных и отображения их в пользовательском интерфейсе вашего приложения. Используйте также методы для записи данных в базу данных, обновления или удаления данных по мере необходимости.

11.3. Работа с Push уведомлениями на Firebase

Теория: Firebase Cloud Messaging (FCM) — это сервис для отправки push-уведомлений на устройства Android и веб-браузеры. Он позволяет разработчикам отправлять сообщения одному или нескольким устройствам, используя один интерфейс API. FCM поддерживает различные типы сообщений, включая текстовые сообщения, уведомления с изображениями, данные пакеты и многое другое.

Практика: 1. Создайте проект в консоли Firebase и настройте проект для использования Firebase Cloud Messaging.

2. Добавьте Firebase Cloud Messaging библиотеку в проект Android Studio, следуя инструкциям документации Firebase.

3. Настройте манифест вашего приложения для подключения к Firebase Cloud Messaging. Убедитесь, что у вас есть разрешение на использование интернета в манифесте приложения.

4. Регистрация устройства в Firebase Cloud Messaging. Получите токен устройства, который будет использоваться для отправки уведомлений на это устройство.

5. Настройте серверную часть для отправки push-уведомлений. Это может быть отдельное приложение сервера, которое будет обрабатывать запросы на отправку уведомлений через Fireboaser SDK или API FCM.

6. Отправьте тестовое уведомление с сервера на зарегистрированное устройство. Удостоверьтесь, что уведомление корректно отображается на устройстве.

7. Разработайте логику вашего приложения для работы с push-уведомлениями. Обработайте получение уведомлений на устройстве и выполните необходимые действия, когда уведомление прибывает.

Модуль 12. Изучение инструментов и библиотек

12.1. Изучение различных вспомогательных инструментов и библиотек для Android на Kotlin

Теория: Разработка на языке Kotlin для платформы Android – Базовый уровень может включать использование различных вспомогательных инструментов и библиотек, которые

помогают ускорить разработку, повысить качество кода и добавить новые функциональные возможности в приложение

Практика: 1. Создайте новый проект Android Studio на языке Kotlin.

2. Изучите документацию и примеры использования каждого из инструментов и библиотек. Ознакомьтесь с основными концепциями, функциональностью и способами использования каждого инструмента.

3. Включите необходимые зависимости в файле build.gradle вашего проекта, чтобы использовать инструменты и библиотеки. Используйте механизм зависимостей, предоставляемый Android Studio, чтобы добавить библиотеки в проект.

4. Реализуйте примеры использования инструментов и библиотек

12.2. Изучение вспомогательных библиотек для работы с RecyclerView, картами, фотографиями, анимациями и т.д.

Практика: 1. Выберите одну или несколько вспомогательных библиотек, которые вы хотите изучить, например, RecyclerView, карты, фотографии или анимации.

2. Изучите документацию и руководства по использованию выбранных библиотек. Понимайте базовые концепции и способы использования библиотеки.

3. Создайте новый проект в Android Studio и настройте его для работы с выбранной библиотекой. Это может включать добавление соответствующих зависимостей в файл Gradle, настройку разрешений и другие необходимые шаги.

4. Разработайте простое приложение, используя выбранную библиотеку. Попробуйте реализовать основной функционал, предоставляемый библиотекой, например, отображение списка с использованием RecyclerView или загрузку и отображение изображений с использованием библиотеки для работы с фотографиями.

5. Протестируйте и проверьте работу вашего приложения с использованием выбранной библиотеки. Удостоверьтесь, что функционал работает корректно и приложение выглядит и работает ожидаемым образом.

12.3. Работа со сторонними API

Теория: Работа со сторонними API является важной частью разработки на языке Kotlin для платформы Android. Сторонние API позволяют взаимодействовать с различными сервисами и ресурсами, такими как базы данных, социальные сети, карты и т.д., чтобы получать данные или выполнять определенные задачи.

Практика: 1. Выберите стороннее API, с которым вы хотите работать. Например, это может быть API для работы с социальными сетями, картами, базами данных и т.д.

2. Изучите документацию API, чтобы понять его функциональность, требования и способы взаимодействия.

3. Зарегистрируйтесь и получите ключ API, если это необходимо.

4. Добавьте библиотеку API в зависимости вашего проекта и настройте ее, следуя инструкциям в документации. Это может потребовать добавления зависимостей в файл build.gradle модуля.

5. Создайте класс или функции для работы с API. Включите методы для создания и отправки запросов к API, аутентификации и авторизации, обработки ответов и ошибок.

6. Разработайте UI-компоненты

Модуль 13. Работа над проектом

13.1. Организация работы над проектом на Kotlin

Теория: 1. Определение цели проекта: прежде чем начать работу над проектом, необходимо ясно определить его цель и ожидаемый результат. Это может быть создание приложения, игры или другого программного решения.

2. Постановка задач: Разбейте проект на более мелкие задачи, чтобы их было проще управлять. Например, разделите задачи по функциональности или между разработчиками.

3. Определение требований: Разработайте список требований к проекту, включая функциональные и нефункциональные требования. Это поможет вам лучше понять, что должно быть реализовано в проекте.

4. Определение стека технологий: на этом этапе нужно определить, какие технологии и инструменты будут использоваться в проекте. Например, Kotlin, Android Studio, Git и т. д.

5. Создание архитектуры проекта: Придумайте, как будет устроена структура проекта, какие модули и компоненты будут использоваться. Например, это может быть применение паттерна MVVM или Clean Architecture.

6. Разработка пользовательского интерфейса: Создайте макеты и дизайн пользовательского интерфейса вашего проекта. Рассмотрите, какие элементы интерфейса вам понадобятся и как они будут взаимодействовать друг с другом.

7. Написание кода: Начните писать код с учетом требований, архитектуры и дизайна. Используйте язык программирования Kotlin и соответствующие библиотеки для создания функциональности вашего проекта.

8. Тестирование и отладка: Проверьте работоспособность и качество вашего кода, выполнив тестирование и отладку. Используйте инструменты и фреймворки для автоматизации тестирования, такие как JUnit или Espresso.

Практика: 1. Начните с простого проекта: для практического изучения организации работы над проектом на Kotlin для платформы Android, начните с простого проекта, такого как создание приложения для заметок или калькулятора.

2. Определите цель и требования проекта: определите, какая функциональность будет реализована в вашем проекте и какие требования нужно учесть при его разработке.

3. Составьте список задач: Разбейте проект на более мелкие задачи, чтобы их было проще управлять. Например, создание пользовательского интерфейса, написание кода для логики приложения и тестирование.

4. Напишите код: Используя язык программирования Kotlin и Android Studio, начните писать код для вашего проекта. Следуйте архитектуре и дизайну, определенным ранее.

5. Тестируйте и отлаживайте: Проверьте ваш код, выполнив тестирование и отладку.

13.2. Разработка технической спецификации проекта

Теория: Разработка технической спецификации (ТЗ) проекта является важным этапом перед началом работы над проектом на языке Kotlin для платформы Android. ТЗ определяет требования к проекту, его функциональность, интерфейс пользователя, базу данных и другие аспекты проекта. Он служит основой для разработки и взаимодействия команды разработчиков.

Практика: 1. Выберите проект: Выберите проект, над которым вы хотите работать. Это может быть приложение для социальных сетей, игра или другое программное решение.

2. Определите требования: определите требования к вашему проекту. Разделите их на функциональные и нефункциональные требования. Например, функциональные требования могут включать функции, такие как регистрация пользователя, создание и редактирование постов, а нефункциональные требования - производительность и безопасность.

13.3. Дизайн пользовательского интерфейса проекта

Практика: 1. Изучение материалов по дизайну UI: изучите основные принципы дизайна пользовательского интерфейса. Посмотрите примеры хорошо спроектированных приложений на Android, чтобы понять, как расположены элементы интерфейса, какое цветовое решение используется и какие анимации применяются.

2. Определение целевой аудитории: определите, для кого будет разрабатываться ваш проект. Например, если это игра, определите, какие возрастные группы будут самыми интересующимися вашим приложением. Это поможет определить стиль и содержание интерфейса.

3. Создание прототипа интерфейса: Создайте прототип интерфейса вашего проекта. Используйте соответствующие инструменты, такие как Adobe XD, Figma или Sketch, чтобы создать макеты экранов вашего приложения. Определите расположение и взаимодействие элементов интерфейса.

4. Определение стилей и цветовой палитры: определите стиль вашего проекта - будет ли это минималистический, футуристический или другой стиль. Выберите цветовую палитру, которая будет соответствовать вашему стилю и помогать пользователям ориентироваться в приложении.

5. Расположение элементов интерфейса: разместите элементы интерфейса на экранах вашего приложения с учетом их функции и взаимосвязи. Обеспечьте удобную навигацию и логическую организацию экранов.

6. Разработка элементов интерфейса: Разработайте элементы интерфейса, такие как кнопки, текстовые поля, изображения и иконки, в соответствии с определенными стилем и цветовой палитрой. Обратите внимание на пропорции, четкость и удобство использования.

7. Тестирование и улучшение интерфейса: Протестируйте ваш интерфейс на реальных пользователях или с помощью протоколирования и записи экрана. Исправьте все замечания и улучшите интерфейс с учетом отзывов и рекомендаций.

8. Адаптация интерфейса для разных устройств: Учтите различия в размерах экранов и разрешениях разных устройств Android. Адаптируйте интерфейс так, чтобы он выглядел хорошо и был удобен для использования на разных устройствах.

9. Участие в сообществе разработчиков: присоединитесь к сообществу разработчиков Kotlin и Android, чтобы получить обратную связь и советы от опытных разработчиков. Обменяйтесь опытом и участвуйте в обсуждениях о дизайне интерфейса.

10. Постоянное развитие и обновление: Учтите, что дизайн интерфейса является процессом, который требует постоянного развития и обновления. Следите за последними трендами дизайна пользовательского интерфейса Android

Модуль 14. Тестирование и отладка приложений

14.1. Изучение методов тестирования и отладки приложений на платформе Android

Теория: Изучение методов тестирования и отладки приложений на платформе Android является важной частью разработки на языке Kotlin. Тестирование и отладка позволяют разработчикам проверить функциональность, исправить ошибки и обеспечить качество приложения перед его выпуском.

Практика: 1. Напишите unit-тесты для отдельных компонентов вашего приложения, используя фреймворк JUnit. Проверьте корректность работы классов и методов ваших компонентов.

2. Напишите интеграционные тесты для проверки взаимодействия компонентов приложения, используя фреймворк Espresso. Проверьте корректность работы активностей, фрагментов и служб в вашем приложении.

3. Напишите UI-тесты для проверки функциональности и отображения пользовательского интерфейса приложения, используя фреймворк UI Automator. Проверьте правильность отображения и работу интерфейсных элементов вашего приложения.

5. Используйте отладчик в Android Studio, чтобы идентифицировать и исправить ошибки в коде вашего приложения. Установите точки останова и анализируйте состояние приложения на каждом этапе выполнения. Проверьте значения переменных и выполнение кода.

6. Используйте инструменты анализа кода, такие как Lint, чтобы проверить соответствие кода стандартам и рекомендациям разработки на языке Kotlin. Исправьте обнаруженные потенциальные проблемы и улучшите качество вашего кода.

7. Постепенно увеличивайте покрытие тестами вашего приложения, добавляя новые unit-тесты, интеграционные тесты и UI-тесты для проверки новых функций или исправления ошибок.

8. Повторяйте процесс тестирования и отладки после каждого обновления приложения, чтобы убедиться в его работоспособности и качестве перед выпуском новой версии.

14.2. Изучение основных ошибок, с которыми сталкиваются приложения на Android

Теория: при разработке приложений на платформе Android с использованием языка Kotlin, разработчики могут столкнуться с различными ошибками, которые могут повлиять на функциональность и качество приложения. Ниже перечислены основные ошибки, с которыми может столкнуться разработчик приложений на Android

Практика: 1. Создайте простое приложение на языке Kotlin. Напишите код, связанный с пользовательским интерфейсом, взаимодействием с базой данных или сетью, асинхронными операциями и многопоточностью.

2. Откройте приложение на устройстве или эмуляторе и проверьте его работу на предмет возможных ошибок. Обратите внимание на работу пользовательского интерфейса, взаимодействие с внешними компонентами и функциональность приложения.

3. Проведите анализ кода приложения с использованием инструментов статического анализа, таких как Android Lint или SonarLint. Обратите внимание на предупреждения и ошибки, указанные инструментами статического анализа. Исправьте обнаруженные проблемы, такие как неправильное использование ресурсов, потенциальные утечки памяти или несоответствие стандартам безопасности.

4. Создайте набор тестов, который проверяет различные аспекты функциональности и корректности вашего приложения. Напишите unit-тесты для тестирования отдельных компонентов или модулей вашего приложения, интеграционные тесты для проверки взаимодействия между компонентами и UI-тесты для проверки функциональности пользовательского интерфейса. Запустите тесты и убедитесь, что они проходят успешно.

5. Запустите ваше приложение в режиме отладки и используйте отладчик Android Studio для идентификации и исправления ошибок в коде. Установите точки останова, проанализируйте значения переменных и выполнение кода для выявления возможных проблем.

6. Проанализируйте код вашего приложения с помощью средств профилирования, таких как Profiler в Android Studio. Определите узкие места, утечки памяти или производительственные проблемы и поправьте их.

7. Читайте документацию и учебные материалы, посвященные разработке на платформе Android и языке Kotlin. Изучите многочисленные советы и рекомендации, которые помогут вам избежать распространенных ошибок и улучшить качество своего кода.

8. Присоединяйтесь к онлайн-сообществам разработчиков Android и Kotlin, где вы сможете задавать вопросы, делиться опытом и получать советы от опытных разработчиков.

14.3. Работа с отладочными инструментами

Практика: 1. Создайте проект на языке Kotlin для платформы Android.

2. Изучите документацию по использованию отладочных инструментов в Android Studio, включая логгирование, отладчик, инспектор представлений и другие инструменты.

3. Используйте отладчик для установки точек останова в коде и анализа значений переменных и состояния приложения.

4. Используйте инспектор представлений для анализа и проверки

Модуль 15. Оптимизация приложений на Android

15.1. Изучение техник оптимизации приложений на платформе Android

Теория: Изучение алгоритмов и структур данных, Оптимизация работы с памятью, Оптимизация работы с сетью,Улучшение производительности интерфейса, Профилирование и оптимизация кода, Тестирование производительности, Анализ данных о производительности, Оптимизация ресурсов

Практика: 1. Изучение алгоритмов и структур данных:

- Изучите основные алгоритмы и структуры данных, такие как сортировка, поиск, хеш-таблицы, списки и деревья.

- Разберитесь, какие алгоритмы и структуры данных наиболее эффективны для различных операций в вашем приложении.

2. Оптимизация работы с памятью:

- Изучите различные способы уменьшить потребление памяти в вашем приложении.

- Определите, какие данные вы можете лениво загружать или кэшировать, чтобы избежать необходимости повторной загрузки или создания объектов.

3. Оптимизация работы с сетью:

- Изучите различные способы оптимизации сетевых запросов и передачи данных.

- Определите, какие данные вы можете комбинировать в один запрос или сжимать, чтобы уменьшить количество сетевых операций и объем передаваемых данных.

4. Улучшение производительности интерфейса:

- Изучите различные подходы к оптимизации интерфейса вашего приложения, такие как использование асинхронных операций, плоского дизайна и отложенной загрузки данных.

- Определите, какие операции занимают больше всего времени и установите приоритеты для их оптимизации.

5. Профилирование и оптимизация кода:

- Используйте инструменты профилирования для идентификации узких мест в вашем коде.

- Определите, какие операции занимают больше всего времени или ресурсов и оптимизируйте их, например, путем использования более эффективных алгоритмов или оптимизации памяти.

6. Тестирование производительности:

- Создайте набор тестов, который позволяет измерить производительность вашего приложения.

- Запустите тесты на разных устройствах и оцените, как ваше приложение справляется с различными нагрузками.

7. Анализ данных о производительности:

- Изучите результаты профилирования и тестирования производительности вашего приложения.

- Определите, какие аспекты требуют дополнительной оптимизации и принимайте меры для улучшения производительности вашего приложения.

8. Оптимизация ресурсов:

- Изучите, какие ресурсы (изображения, аудио, видео) используются в вашем приложении и оптимизируйте их.

- Сжимайте изображения, используя эффективные алгоритмы сжатия, и оптимизируйте использование звука и видео, чтобы уменьшить размеры файлов и улучшить производительность приложения.

9. Тестирование и отладка:

- Протестируйте ваши оптимизации, чтобы убедиться, что они не вводят новые ошибки или проблемы.

- Используйте отладчик Android Studio, чтобы идентифицировать и исправить ошибки в своем коде.

10. Итеративный подход:

- Повторяйте процесс оптимизации и тестирования, чтобы продолжать улучшать производительность вашего приложения.

15.2. Изучение методов улучшения скорости работы и уменьшению использования ресурсов памяти

Теория:

1. Обзор понятия профилирование и его значение в разработке на платформе Android.
2. Изучение основных типов профилировщиков, доступных для работы с Android приложениями.
3. Ознакомление с основными функциями и возможностями профилировщиков, такими как Android Profiler и Systrace.
4. Изучение различных метрик и показателей, которые можно измерять с помощью профилировщиков, например, использование CPU, памяти, сети, энергии и др.
5. Понимание того, как эффективно использовать профилировщики для оптимизации производительности и исправления проблем в приложении.

Практика:

1. Знакомство с UI и основными функциями Android Profiler в Android Studio.
2. Запуск профилирования для вашего приложения и изучение полученных результатов.
3. Анализ работы приложения через профилировщик, например, измерение и анализ использования CPU, памяти и сети.
4. Идентификация узких мест и проблем в производительности приложения на основе полученных данных.
5. Применение оптимизаций для улучшения производительности и исправление проблем, выявленных в профилировщике.

6. Запуск профилировщика Systrace и изучение получаемых данных о работе системы и приложений.
7. Использование Systrace для анализа задержек, потерь сигнала, использования процессора и энергопотребления в вашем приложении.
8. Итеративный процесс оптимизации, включающий запуск профилировщиков, анализ результатов, внесение изменений, повторный анализ и т. д.

15.3. Работа с профилировщиком приложений

Теория:

1. Введение в профилирование приложений для платформы Android.
2. Обзор основных понятий и терминологии, используемых при работе с профилировщиками.
3. Изучение возможностей и функций профилировщиков для разработки на языке Kotlin и платформы Android.
4. Понимание принципов работы профилировщиков, таких как сбор информации о производительности, обнаружение узких мест и проблем производительности, анализ и оптимизация кода.
5. Изучение различных инструментов и техник профилирования, таких как Android Profiler, Systrace, анализаторы памяти и процессора и другие.

Практика:

1. Запуск профилировщика Android Profiler для вашего приложения на языке Kotlin.
2. Изучение доступных функций и настроек Android Profiler.
3. Использование Android Profiler для анализа производительности вашего приложения, например, измерение времени выполнения, использование CPU и памяти.
4. Идентификация узких мест и проблем в производительности вашего приложения на основе полученных данных.
5. Применение оптимизаций для улучшения производительности вашего приложения, например, оптимизация использования памяти, оптимизация работы с сетью и другие.
6. Запуск профилировщика Systrace и изучение получаемых данных о работе системы и приложений на языке Kotlin.
7. Использование Systrace для анализа энергопотребления, задержек, использования ресурсов и других параметров вашего приложения.
8. Итеративный процесс оптимизации, включающий запуск профилировщиков, анализ результатов, внесение изменений, повторный анализ и т. д.

Модуль 16. Работа с различными SDK и аддонами

16.1. Изучение SDK и аддонов для разработки приложений на платформе Android

Теория:

1. Обзор основных SDK (Software Development Kit) для разработки приложений на платформе Android, таких как Android SDK, Android Jetpack, Google Play Services и другие.
2. Изучение основных инструментов и ресурсов, предоставляемых SDK, таких как Android Studio, эмуляторы, документация, примеры кода и т.д.
3. Понимание важных концепций и паттернов разработки на платформе Android, таких как жизненный цикл компонентов, макеты, обработка событий, работа с ресурсами и другие.
4. Ознакомление с аддонами и библиотеками для разработки на платформе Android, такими как Retrofit, Glide, Firebase, Dagger и другие.

Практика:

1. Изучение основных компонентов приложения на основе созданного проекта, например, создание активности, фрагментов, сервисов и других.
2. Использование инструментов и ресурсов SDK, таких как Android Studio, эмуляторы и документация, для разработки и отладки приложения.
3. Изучение и использование различных аддонов и библиотек для улучшения функциональности и производительности вашего приложения, например, работа с сетью, хранение данных, обработка изображений и другие.

4. Тестирование и отладка созданного приложения на платформе Android с использованием инструментов и функций SDK.
5. Оптимизация и доработка созданного приложения на платформе Android с использованием аддонов и библиотек для улучшения его функциональности и производительности.

16.2. Создание и интеграция различных SDK и аддонов в приложения

Теория:

1. Ознакомление с основами языка Kotlin и его отличиями от Java.
2. Изучение основных принципов разработки приложений на платформе Android с использованием языка Kotlin.
3. Определение основных понятий и компонентов в разработке на платформе Android, таких как активности, фрагменты, макеты, обработчики событий и другие.
4. Изучение основных SDK (Software Development Kit) и аддонов для разработки на платформе Android, таких как Android SDK, Android Jetpack, Google Play Services и другие.
5. Ознакомление с основными функциями и возможностями SDK и аддонов, такими как работа с сетью, хранение и обработка данных, аналитика, реклама и другие.
6. Понимание важных паттернов и принципов разработки на платформе Android, таких как MVP (Model-View-Presenter), MVVM (Model-View-ViewModel) и другие.
7. Изучение и использование различных инструментов и сред разработки на платформе Android, таких как Android Studio, эмуляторы, системы версионирования и другие.

Практика:

1. Создание нового проекта на платформе Android с использованием языка Kotlin.
2. Интеграция различных SDK и аддонов в созданный проект, например, добавление библиотеки Retrofit для работы с сетью или Google Play Services для аутентификации пользователей.
3. Использование функций и возможностей SDK и аддонов для разработки различных функций приложения, таких как загрузка данных с сервера, работа с базой данных, реализация аутентификации и другие.
4. Тестирование и отладка приложения с интегрированными SDK и аддонами на платформе Android.
5. Оптимизация и улучшение функциональности и производительности приложения с помощью SDK и аддонов, например, добавление аналитики или рекламы.

16.3. Изучение примеров использования сторонних сервисов и инструментов в приложениях

Теория:

1. Ознакомление с основами языка Kotlin и его применением для разработки на платформе Android.
2. Изучение основных компонентов и архитектуры приложений на платформе Android.
3. Обзор и изучение различных сторонних сервисов и инструментов, доступных для разработки на платформе Android. Примеры таких сервисов могут включать Firebase, Crashlytics, и другие.
4. Понимание принципов работы и примеров использования каждого из сторонних сервисов и инструментов.
5. Изучение возможностей и функций сервисов и инструментов для улучшения функциональности и производительности приложения.

Практика:

1. Создание нового проекта на платформе Android с использованием языка Kotlin.
2. Интеграция выбранного стороннего сервиса или инструмента в созданный проект, например, добавление Firebase для аналитики или Facebook SDK для аутентификации.
3. Разработка функциональности приложения, используя возможности выбранного сервиса или инструмента, например, отправка уведомлений с помощью Firebase Cloud Messaging или получение данных из Facebook Graph API.
4. Тестирование и отладка приложения с использованием интегрированного сервиса или инструмента.

5. Оптимизация и улучшение функциональности и производительности приложения с помощью стороннего сервиса или инструмента, например, оптимизация запросов к базе данных или добавление аналитики для отслеживания использования приложения.

Модуль 17. Создание индивидуального проекта

17.1. Разработка индивидуального проекта на Kotlin для платформы Android

Теория: Разработка индивидуального проекта мобильного приложения или игры на платформе по выбору учащегося является заключительным этапом обучения по программе "Разработка на языке Kotlin для платформы Android – Базовый уровень".

Практика: Определение целей и требований проекта. Обучающиеся должны определить цель проекта и требования к нему. Например, если проект заключается в разработке приложения, то обучающиеся должны определить, какую проблему приложение будет решать, как он будет использоваться, кто будет использовать приложение и т.д.

17.2. Работа с преподавателем по разработке и созданию проекта

Теория: В процессе обучения в программе "Разработка на языке Kotlin для платформы Android – Базовый уровень" обучающимся доступно множество ресурсов и инструментов для создания своих проектов. Однако, помощь и поддержка со стороны преподавателя могут оказаться важными для решения сложных задач и получения советов по оптимизации проекта.

Практика: использовать все доступные ресурсы, такие как книги и документация, чтобы получить информацию о технологиях и методах, используемых в разработке мобильных приложений.

17.3. Аттестация по итогам прохождения программы

Практика: Защита проектов

1.5. Учебный план

№ п/п	Название модуля, темы	Кол-во часов			Формы аттестации/ контроля
		Всего часов	Теоретические занятия	Практические занятия	
1	**Введение. Знакомство**	**2**	**2**	**0**	
Модуль 1. Введение в Kotlin и платформу Android		**8**	**3**	**5**	
1.1	Что такое Kotlin и почему он используется для разработки на платформе Android	2	1	1	
1.2	Основы платформы Android и структура приложения	2	1	1	
1.3	Создание первого проекта на Kotlin	4	1	3	
Модуль 2. Основы синтаксиса Kotlin		**8**	**3**	**5**	
2.1	Приведение примеров использования переменных, функций и классов в Kotlin	2	1	1	
2.2	Создание функций с различными параметрами и возвращаемыми значениями	4	1	3	
2.3	Изучение механизмов управления потоками	2	1	1	
Модуль 3. Изучение пользовательского интерфейса Android		**8**	**2**	**6**	
3.1	Основы построения пользовательского интерфейса в Android	2	1	1	

3.2	Работа с различными элементами пользовательского интерфейса: текст, изображения, кнопки и т.д.	2	0	2	
3.3	Изучение работы с языком разметки XML	4	1	3	
Модуль 4. Работа со списками и адаптерами		**8**	**2**	**6**	
4.1	Изучение основ работы со списками на примере ListView	4	1	3	
4.2	Создание пользовательских адаптеров. Интеграция адаптеров с пользовательским интерфейсом	4	1	3	
Модуль 5. Работа с базами данных		**8**	**2**	**6**	
5.1	Изучение принципов работы и использования баз данных в Android	2	1	1	
5.2	Работа с базой данных SQLite с помощью API Room	4	1	3	
5.3	Создание, обновление и удаление записей в базе данных	2	0	2	
Модуль 6. Работа с сенсорным экраном, геолокацией и сетевыми запросами		**10**	**3**	**7**	
6.1	Промежуточная Аттестация. Тестирование	2	0	2	
6.2	Изучение работы с сенсорным экраном и извлечение информации из него	2	1	1	
6.3	Изучение использования геолокации на Android	2	1	1	
6.4	Работа с сетевыми запросами	4	1	3	
Модуль 7. Работа с мультимедиа и уведомлениями		**8**	**2**	**6**	
7.1	Изучение работы с мультимедиа на Android	4	1	3	
7.2	Создание пользовательских уведомлений/Интеграция уведомлений с пользовательским интерфейсом	4	1	3	
Модуль 8. Создание приложений с использованием паттерна MVVM		**8**	**2**	**6**	
8.1	Изучение паттерна MVVM и его реализации на Kotlin	2	1	1	
8.2	Создание приложения с использованием MVVM	4	1	3	
8.3	Изучение практических примеров использования MVVM	2	0	2	
Модуль 9. Изучение архитектурных компонентов		**8**	**3**	**5**	
9.1	Изучение архитектурных компонентов Android Jetpack	2	1	1	
9.2	Создание приложений с использованием Room, ViewModel и LiveData	4	1	3	
9.3	Ввод в понятие Navigation Components	2	1	1	
Модуль 10. Изучение многомодульной		**8**	**2**	**6**	

архитектуры					
10.1	Изучение базовой концепции многомодульной архитектуры на примере приложения	4	1	3	
10.2	Создание и интеграция различных модулей	4	1	3	
Модуль 11. Работа с FireBase		**8**	**2**	**6**	
11.1	Изучение сервиса Firebase и его использование для разработки приложений	2	1	1	
11.2	Работа с базой данных Firebase	2	0	2	
11.3	Работа с Push уведомлениями на Firebase	4	1	3	
Модуль 12. Изучение инструментов и библиотек		**8**	**2**	**6**	
12.1	Изучение различных вспомогательных инструментов и библиотек для Android на Kotlin	2	1	1	
12.2	Изучение вспомогательных библиотек для работы с RecyclerView, картами, фотографиями, анимациями и т.д.	2	0	2	
12.3	Работа со сторонними API	4	1	3	
Модуль 13. Работа над проектом		**8**	**2**	**6**	
13.1	Организация работы над проектом на Kotlin	2	1	1	
13.2	Разработка технической спецификации проекта	4	1	3	
13.3	Дизайн пользовательского интерфейса проекта	2	0	2	
Модуль 14. Тестирование и отладка приложений		**8**	**2**	**6**	
14.1	Изучение методов тестирования и отладки приложений на платформе Android	2	1	1	
14.2	Изучение основных ошибок, с которыми сталкиваются приложения на Android	4	1	3	
14.3	Работа с отладочными инструментами	2	0	2	
Модуль 15. Оптимизация приложений на Android		**8**	**3**	**5**	
15.1	Изучение техник оптимизации приложений на платформе Android	2	1	1	
15.2	Изучение методов улучшения скорости работы и уменьшению использования ресурсов памяти	4	1	3	
15.3	Работа с профилировщиком приложений	2	1	1	
Модуль 16. Работа с различными SDK и аддонами		**10**	**3**	**7**	
16.1	Изучение SDK и аддонов для разработки приложений на платформе Android	2	1	1	
16.2	Создание и интеграция различных SDK и	4	1	3	

	аддонов в приложения				
16.3	Изучение примеров использования сторонних сервисов и инструментов в приложениях	4	1	3	
Модуль 17. Создание индивидуального проекта		**10**	**2**	**8**	
17.1	Разработка индивидуального проекта на Kotlin для платформы Android	4	1	3	
17.2	Работа с преподавателем по разработке и созданию проекта	4	1	3	
17.3	Аттестация по итогам прохождения программы	2	0	2	
Итого		**144**	**42**	**102**	

1.6. Планируемые результаты

Планируемые результаты программы "Разработка на языке Kotlin для платформы Android – Базовый уровень" включают:

1. Приобретение фундаментальных знаний о языке программирования Kotlin и его особенностях для разработки мобильных приложений на платформе Android.
2. Умение создавать пользовательские интерфейсы для мобильных приложений с использованием элементов и компонентов Android.
3. Навыки работы с данными, включая методы получения, обработки и хранения данных в приложении на платформе Android.
4. Умение интегрировать сторонние библиотеки и сервисы в мобильные приложения с помощью языка Kotlin.
5. Знание методов оптимизации производительности мобильных приложений и умение отлаживать и исправлять ошибки.
6. Навыки тестирования мобильных приложений и развертывания их на устройствах и в магазинах приложений.
7. Умение создавать качественные мобильные приложения на платформе Android, отвечающие требованиям и ожиданиям пользователей.

В результате программы участники будут готовы самостоятельно разрабатывать и поддерживать мобильные приложения на платформе Android с использованием языка Kotlin, а также продолжать углублять свои знания и навыки в этой области.

Метапредметные результаты:

- формирование навыков самоорганизации;
- формирование навыков сотрудничества: работа в коллективе, в команде, микро-группе;
- воспитание бережного отношение к технике;
- воспитание самостоятельности, инициативности;
- развитие навыков анализа и оценки получаемой информации.

Личностные:

- развитие личностных качеств (активность, инициативность, воля, любознательность и т. п.);
- развитие внимания, памяти, восприятия, образного мышления;
- развитие логического и пространственного воображения;
- развитие творческих способностей и фантазии;
- развитие мотивации к познанию и творчеству;
- формирование положительных черт характера: трудолюбия, аккуратности, собранности, усидчивости, отзывчивости;
- развитие мотивации к профессиональному самоопределению.

Раздел 2. Комплекс организационно-педагогических условий

2.1 календарный учебный график

Год обучения	Всего учебных недель	Количество учебных часов	Режим занятий
1 год	**36**	**144**	**1 раз в неделю – 2 часа**

2.2 Условия реализации программы

Материально-техническое обеспечение:

Для реализации учебных занятий используется следующее оборудование и материалы:

- Ноутбуки, оснащенные выходом в Интернет;
- Интерактивная доска;
- Планшеты;

Информационное обеспечение:

- операционная система Monjaro;
- Интернет-источники;
- поддерживаемые браузеры (для работы LMS): Yandex Browser, Chrome, Chrome Mobile, Firefox, Opera;
- варианты демонстрационных программ, материалы по терминологии ПО;
- инструкции по настройке оборудования;
- учебная и техническая литература
- методические пособия, разрабатываемые преподавателем с учётом конкретных условий;
- техническая библиотека объединения, содержащая справочный материал, учебную и техническую литературу.
- обязательным является инструктаж по технике безопасности и беседы о здоровье сберегающем поведении в процессе работы на компьютере, интенсивной интеллектуальной деятельности.

Кадровое обеспечение: Программа реализуется Гайнановым М.В., Среднее профессиональное ГБПОУ «Саткинский горно-керамический колледж имени А.К. Савина», 30.06.2021 Техник-программист, Свидетельство Оператор электронно-вычислительных и вычислительных машин (4 разряд); Профессиональная переподготовка ГБУ ДПО «ЧИППКРО», 03..06.2022 Право на ведение профессиональной деятельности в сфере дополнительного образования.

2.3. Формы аттестации

Система контроля знаний и умений обучающихся представляется в виде:

Текущий контроль осуществляется после изучения отдельных тем, раздела программы. В практической деятельности результативность оценивается качеством выполнения практических работ, поиску и отбору необходимого материала, умению работать с различными источниками информации. Анализируются положительные и отрицательные стороны работы, корректируются недостатки. Контроль знаний осуществляется с помощью заданий педагога (решение практических задач средствами языка программирования); взаимоконтроля, самоконтроля и др. Они активизируют, стимулируют работу обучающихся, позволяют более полно проявлять полученные знания, умения, навыки.

Промежуточная аттестация осуществляется в конце I полугодия учебного года.

Форма контроля: тестирование, решение практических задач средствами языка программирования.

Аттестация по итогам освоения программы осуществляется в конце учебного года.

Форма контроля: защита проекта.

Проект является одним из видов самостоятельной работы, предусмотренной в ходе обучения по программе. Педагог-наставник оказывает консультационную помощь в выполнении проекта.

Индивидуальный (групповой) проект оценивается формируемой комиссией. Состав комиссии (не менее 3-х человек): педагог-наставник, администрация учебной организации, приветствуется привлечение IT-профессионалов, представителей высших и других учебных заведений.

Компонентами оценки индивидуального (группового) проекта являются (по мере убывания значимости): качество индивидуального проекта, отзыв руководителя проекта, уровень презентации и защиты проекта. Если проект выполнен группой обучающихся, то при оценивании учитывается не только уровень исполнения проекта в целом, но и личный вклад каждого из авторов. Решение принимается коллегиально.

2.4. Оценочные материалы

Промежуточная аттестация

Тестирование проводится с обучающимися индивидуально, во время проведения занятия.

Перечень вопросов для тестирования:

1. Что такое Kotlin? Какие основные отличия Kotlin от Java для разработки на платформе Android?
2. Расскажите о базовых синтаксических конструкциях Kotlin, таких как переменные, функции, условные операторы и циклы.
3. Каким образом можно создать пользовательский интерфейс в приложении на платформе Android с использованием Kotlin?
4. Каким образом можно работать с данными в приложении на платформе Android с использованием Kotlin? Упомяните методы получения данных, работу с базами данных и сетевыми запросами.
5. Как можно интегрировать сторонние библиотеки и сервисы в приложение, разработанное на платформе Android с использованием Kotlin?
6. Каким образом можно оптимизировать производительность мобильного приложения на платформе Android с использованием Kotlin? Укажите некоторые методы и техники.
7. Как можно отлаживать и исправлять ошибки в приложении, разработанном на платформе Android с использованием Kotlin? Упомяните инструменты и методы.
8. Как проводить тестирование мобильного приложения, созданного на платформе Android с использованием Kotlin? Укажите основные этапы и методы тестирования.
9. Каким образом можно развернуть мобильное приложение на устройствах или в магазинах приложений после его разработки на платформе Android с использованием Kotlin?
10. Какие основные преимущества и недостатки Kotlin для разработки на платформе Android вы можете выделить по сравнению с другими языками программирования, такими как Java?

Практическая задача:

Приложение будет отображать статистику активности в виде графиков и диаграмм.

Вот некоторые шаги, которые можно предпринять для разработки такого приложения на платформе Android с использованием Kotlin для детей и подростков:

1. Создайте новый проект в среде разработки Android Studio.

2. Определите модель данных для активности. Создайте класс Activity с необходимыми свойствами, такими как название активности, пройденное расстояние, длительность активности и количество сожженных калорий.

3. Создайте пользовательский интерфейс для приложения. Добавьте элементы управления, такие как текстовые поля и кнопки, с помощью XML-ресурсов.

4. Создайте активити (Activity) для ввода информации о физической активности. Свяжите нужные элементы интерфейса с кодом активити с помощью функции findViewById().

5. Реализуйте методы для сохранения информации о физической активности. Например, создайте метод для добавления активности в список и сохранения списка в файл или базу данных.

6. Используйте RecyclerView для отображения списка активностей в пользовательском интерфейсе. Создайте адаптер (Adapter) для связывания данных из списка активностей с элементами списка в RecyclerView.

7. Добавьте возможность просмотра статистики активности в приложении. Реализуйте графики или диаграммы, которые отражают суммарное пройденное расстояние или количество сожженных калорий в зависимости от даты активности.

8. Убедитесь, что приложение обрабатывает события пользователя, такие как нажатие на кнопку "Добавить активность" или выбор активности из списка.

9. Проверьте работоспособность приложения на эмуляторе Android или реальном устройстве. Используйте отладчик для исправления ошибок и отслеживания работы приложения.

10. Разработайте интерфейс приложения с учетом дизайна и удобства использования, адаптированного к детям и подросткам.

Критерии оценивания обучающихся

Описание правил проведения аттестации:

1.Знание теории

По результатам ответов на вопросы определяется уровень теоретической подготовки.

Уровень подготовки определяется по количеству набранных баллов.

Без ответа – 0 баллов

Ответ неверный – 1 балл

Ответ частично неверный - 2 балла

Ответ развёрнутый и верный 3 балла

Максимальное количество баллов за ответ 3 балла.

Максимальное количество -30 баллов

Критерии оценивания:

Высокий уровень: 30-26 баллов;

Средний уровень: 25-11 баллов;

Низкий уровень: 10-1 баллов.

1.Знание практики

По результатам выполнения практического задания уровень практической подготовки.

Уровень подготовки определяется по количеству набранных баллов. Правильное выполнение задачи 5 баллов, при частичном выполнении от 1-4 баллов. Максимальное количество -5 баллов

Критерии оценивания:

Высокий уровень: 5-4 баллов;

Средний уровень: 3- 2 баллов;

Низкий уровень: 2-1 баллов.

Аттестация по итогам освоения программы.

Разработка и защита проекта

Исследовательский проект

Тема проекта:

Творческое название (при наличии):

Основополагающий вопрос:

Авторы:

1.

2.
3.
...
Предметная область:
Краткая аннотация:
Проблемные вопросы учебной темы:
Темы исследования учащихся:
Этапы выполнения проекта:
На этапе презентации участники представляют проект на обсуждение.
Этап рефлексии отводится под обсуждение итогов проекта, оценки своих действий, формулирование выводов. Для оценивания проекта могут быть разработаны специальные оценочные листы. Ниже представлен пример оценочного листа:
Таблица
Лист оценивания проекта

Критерий оценивания	**1-я группа**
Актуальность темы	
Соответствие содержания проекта заявленной теме	
Техническая сложность	
Оригинальность	
Дизайн	
Уровень проработанности проекта	
Итоговое количество баллов	

Система оценки результатов освоения программы
Предметом диагностики и контроля являются внешние образовательные продукты учащихся, а также их внутренние личностные качества (освоенные способы деятельности, знания, умения), которые относятся к целям и задачам программы. Основой для оценивания деятельности учащихся являются результаты анализа его продукции, деятельности по ее созданию, уровень защиты проекта на конференции.

Оценке подлежит в первую очередь уровень достижения учеником минимально необходимых результатов, обозначенных в целях и задачах программы.

Оцениванию подлежат также те направления и результаты деятельности учащихся, которые определены в рабочей программе педагога и в индивидуальных образовательных маршрутах учащихся (при наличии таковых).

Критерии оценки защиты проекта и уровня выполнения работы учащимся	**Оценка**
Проект полный, оригинальный, обладает степенью новизны и практической пользы, не содержит ошибок. Учащийся способен обеспечить подачу проекта целевой аудитории, обобщить материал, сделать собственные выводы, выразить свое мнение, привести примеры, ответить на вопросы по теме проекта.	отлично

Проект полный, обладает оригинальностью, и практической пользой, не содержит значительных ошибок. Учащийся способен обеспечить подачу проекта целевой аудитории, сделать собственные выводы, ответить на вопросы по теме проекта. Собственное мнение по теме проекта недостаточно четко выражено.	хорошо
Проект типовой, не содержит значительных ошибок. Не обладает лаконичностью. Есть ошибки в деталях и/или они просто отсутствуют. Подача проекта сумбурная. Мнение по теме проекта сформировано частично. Затрудняется с ответами по теме проекта.	удовлетворительно

2.5 Методические материалы

Организация образовательного процесса в данной программе происходит в очной форме обучения, с возможностью применения дистанционных технологий, и групповой форме.

При реализации программы используются различные методы обучения:

- Объяснительно-иллюстративный (предъявление информации различными способами (объяснение, рассказ, беседа, инструктаж, демонстрация, работа с технологическими картами и др.);
- Проблемный (постановка проблемы и самостоятельный поиск её решения обучающимися);
- Репродуктивный (воспроизводство знаний и способов деятельности по аналогу);
- Метод проектов (технология организации образовательных ситуаций, в которых учащийся ставит и решает собственные задачи).

Методические материалы

- методы обучения (словесный, наглядный, практический, объяснительно-иллюстративный, интегрированный, метод сравнения, репродуктивный, частично-поисковый, аналитический, дедуктивный, исследовательский, проблемный, игровой, дискуссионный, проектный и др.) и воспитания (убеждение, поощрение, упражнение, стимулирование, мотивация, метод положительного примера и др.);

- формы организации образовательного процесса: индивидуальная, индивидуально-групповая и групповая; выбор той или иной формы обосновывается с позиции профиля деятельности (технического).;

-формы организации учебного занятия – беседа, выставка, защита проектов, конкурс, мастер-класс, «мозговой штурм», наблюдение, олимпиада, открытое занятие, практическое занятие, презентация, семинар, соревнование, объяснение материала, моделирование и др.;

-образовательные (педагогические) технологии – технология индивидуализации обучения, технология группового обучения, технология коллективного взаимообучения, технология программированного обучения, технология модульного обучения, технология блочно-модульного обучения, технология дифференцированного обучения, технология разноуровневого обучения, технология развивающего обучения, технология проблемного обучения, технология дистанционного обучения, технология проектной деятельности, технология коллективной творческой деятельности, технология решения изобретательских задач, здоровьесберегающая технология.

2.6 Воспитательные компоненты

Общей целью воспитания является формирование у обучающихся духовно-нравственных ценностей, способности к осуществлению ответственного выбора собственной индивидуальной образовательной траектории, способности к успешной социализации в обществе.

Задачи воспитания:

- поддерживать и развивать традиции учреждения, коллективные творческие формы деятельности, реализовать воспитательные возможности ключевых дел, формировать у обучающихся чувство солидарности и принадлежности к образовательному учреждению;
- реализовывать воспитательный потенциал общеобразовательных общеразвивающих программ и возможности учебного занятия и других форм образовательных событий;
- развивать социальное партнерство как один из способов достижения эффективности воспитательной деятельности.
- организовывать работу с семьями обучающихся, их родителями или законными представителями, активно их включать в образовательный процесс, содействовать формированию позиции союзников в решении воспитательных задач;
- использовать в воспитании детей возможности занятий по дополнительным общеобразовательным общеразвивающим программам как источник поддержки и развития интереса к познанию и творчеству;
- содействовать приобретению опыта личностного и профессионального самоопределения на основе личностных проб в совместной деятельности и социальных практиках;
- формировать сознательное отношение обучающихся к своей жизни, здоровью, здоровому образу жизни, а также к жизни и здоровью окружающих людей.
- создавать инновационную среду, формирующую у детей и подростков изобретательское, креативное, критическое мышление через освоение дополнительных общеобразовательных общеразвивающих программ нового поколения в области инженерных и цифровых технологий;
- повышать разнообразие образовательных возможностей при построении индивидуальных образовательных траекторий (маршрутов) обучающихся;
- оптимизировать систему выявления, поддержки и развития способностей и талантов у детей и подростков, направленной на самоопределение и профессиональную ориентацию обучающихся.

Направления воспитательной работы

Основными направлениями воспитательной работы являются:

- воспитывать аккуратность и дисциплинированность при выполнении работы;
- способствовать формированию положительной мотивации к трудовой деятельности;
- развивать основы коммуникативных отношений внутри проектных групп и в коллективе в целом;
- воспитывать трудолюбие, уважение к труду;
- развивать навыки отношений делового сотрудничества, взаимоуважения.

Работа с родителями

Работа с родителями обучающихся в себя:

- организацию системы индивидуальной и коллективной работы (тематические беседы, собрания, индивидуальные консультации);
- содействие сплочению родительского коллектива и вовлечение родителей в жизнедеятельность детского объединения (организация и проведение открытых занятий, мероприятий в течение учебного года);

Примерный перечень мероприятий

Сроки	Уровень проведения соревнований	Название соревнований, конкурсов, мероприятий
1. Модуль « Воспитывающая среда»		

01.09.2023	на уровне учреждения	«День знаний»
декабрь	на уровне учреждения	«КвантоЕлка»
февраль	муниципальный	Конкурс 3D моделей приуроченных к 23 февраля
март	муниципальный	конкурс по созданию видео открыток в среде "Подарок Маме"
апрель	муниципальный	конкурс рисунков ко дню Космонавтики
май	на уровне учреждения	Организация выставки с достижениями детей
2. Модуль « Учебное занятие»		
в течение года	муниципальный	«Урок цифры»
май	муниципальный	«Урок Победы»
декабрь-январь	региональный	«Технологический диктант»
февраль	на уровне учреждения	«День науки»
3.Модуль «Руководство детским объединением (направлением) и взаимодействие с родителями»		
сентябрь, май	на уровне учреждения	Родительские собрание, мастер-классы
4.Модуль «Проектная деятельность»		
декабрь, май	на уровне учреждения	«Ярмарка проектов»
5.Модуль «Профориентационная работа и наставничество»		
апрель	на уровне учреждения	Дни открытых дверей в СУЗе
6.Модуль «Социальное партнерство и сетевое взаимодействие»		
ноябрь-май	Региональный	Участие в конкурсе инженерных команд «Инженерные кадры России» и «Икаренок»
сроки, указанные в проекте	муниципальный	Проекты, совместно разрабатываемые и реализуемые обучающимися, педагогами с организациями-партнерами различной направленности
7.Модуль «Каникулы»		

ноябрь, январь, март, июнь	муниципальный	Онлайн-лагерь в дни школьных каникул
8.Модуль «Профилактика и безопасность»		
сентябрь	на уровне учреждения	Проведение «Урока безопасности и навыков безопасного поведения в Интернете, информационной безопасности, повышение правовой грамотности»
сентябрь	на уровне учреждения	Проведение инструктажа по безопасности и охране жизни и здоровья

2.7. Информационные ресурсы и литература

Литература для педагогов:

1. "Разработка мобильных приложений на платформе Android с использованием Kotlin: методические рекомендации для педагогов" - автор Баранова О.В. (доступно в электронном виде).

Литература для обучающихся:

1. "Kotlin для начинающих: подробное руководство по изучению языка программирования Kotlin" - автор Миртл Ж.
2. "Программирование для Android на языке Kotlin" - автор Гасков А.
3. "Разработка мобильных приложений на платформе Android с использованием Kotlin" - автор Самохвалов А.
4. "Android разработка на языке Kotlin: руководство для начинающих" - автор Смит Д.

Электронные ресурсы:

1. Официальный сайт Kotlin - https://kotlinlang.org/
2. Официальный сайт Android Developers - https://developer.android.com/
3. Kotlin на платформе Android: документация - https://developer.android.com/kotlin/
4. Kotlin Android Extensions: документация - https://developer.android.com/topic/libraries/view-binding/
5. Видеокурсы по Kotlin для Android-разработки на YouTube - пример: "Kotlin for Android Developers" от Филиппа Ляцко.
6. Kotlin Android Development Masterclass на Udemy - https://www.udemy.com/course/kotlin-android/

Приложение

Приложение 1

ОЦЕНОЧНЫЙ ЛИСТ
ВХОДНОЙ ДИАГНОСТИКИ УЧАЩИХСЯ
_____ учебный год

Входная диагностика учащихся объединения: Мобильная разработка

Наименование образовательной программы: Разработка на языке Kotlin для платформы Android – Базовый уровень(14-18 лет)

Фамилия, имя, отчество педагога: Гайнанов Максим Вячеславович

Дата проведения: _____

Форма проведения: Наблюдение, беседа

Форма оценки результатов: уровень (высокий, средний, низкий)

3 балла (высокий уровень) – высокий уровень развития компетенции. Обучающийся (его знания, умения) выделяются на общем фоне своей успешностью (оригинальностью, качеством).

2 балла (средний уровень) – промежуточный уровень.

1 балл (низкий уровень) – трудности в понимании заданий и учебного материала; низкий уровень развития компетенции, недостаточная активность

Результаты итоговой аттестации

№	Фамилия имя ребенка	1	2	3	Итого	Результат
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Всего аттестовано 12 воспитанников. Из них по результатам аттестации:	
высокий уровень -__ чел. средний уровень -__ чел. низкий уровень - __ чел.	
Подпись педагога: Гайнанов Максим Вячеславович	
Подписи членов аттестационной комиссии	
А.В. Михайлов - руководитель ЦЦОД «IT-куб» г.Сатка;	
Н.В. Кириченко – заведующий учебной частью ЦЦОД «IT-куб» г.Сатка;	
Э. И. Макагон – методист ЦЦОД «IT-куб» г.Сатка	

Приложение 2

ОЦЕНОЧНЫЙ ЛИСТ
ПРОМЕЖУТОЧНОЙ ДИАГНОСТИКИ УЧАЩИХСЯ
_____ учебный год

Входная диагностика учащихся объединения: Мобильная разработка

Наименование образовательной программы: Разработка на языке Kotlin для платформы Android – Базовый уровень(14-18 лет)

Фамилия, имя, отчество педагога: Гайнанов Максим Вячеславович

Дата проведения: _____

Форма проведения: Решение практических задач

Форма оценки результатов: уровень (высокий, средний, низкий)

3 балла (высокий уровень) – высокий уровень развития компетенции. Обучающийся (его знания, умения) выделяются на общем фоне своей успешностью (оригинальностью, качеством).

2 балла (средний уровень) – промежуточный уровень.

1 балл (низкий уровень) – трудности в понимании заданий и учебного материала; низкий уровень развития компетенции, недостаточная активность

Результаты итоговой аттестации

№	Фамилия имя ребенка	1	2	3	Итого	Результат
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Всего аттестовано 12 воспитанников. Из них по результатам аттестации:	
высокий уровень -__ чел. средний уровень -__ чел. низкий уровень - __ чел.	
Подпись педагога: Гайнанов Максим Вячеславович	
Подписи членов аттестационной комиссии	
А.В. Михайлов - руководитель ЦЦОД «IT-куб» г.Сатка;	
Н.В. Кириченко – заведующий учебной частью ЦЦОД «IT-куб» г.Сатка;	
Э. И. Макагон – методист ЦЦОД «IT-куб» г.Сатка	

Приложение 3

План
педагога Гайнанова Максима Вячеславовича
по проведению аттестации по итогам освоения программы
по дополнительной общеобразовательной общеразвивающей программе «Разработка на языке Kotlin для платформы Android – Базовый уровень**»**

Сроки проведения:
Вид аттестации: аттестация по итогам освоения программы
Цель итоговой аттестации: оценка качества усвоения обучающимися содержания образовательной программы в конце учебного года.
Форма проведения: защита проектов
Форма оценки, уровень усвоения программы: высокий, средний, низкий.
Правила проведения аттестации: критерии оценки результата.

Описание правил проведения аттестации:
Модель реализации исследовательских проектов обучающихся

№	Этапы реализации проекта	Примерные виды деятельности
1	Организационный (подготовка). Текущая рефлексия	Определение темы проекта. Разработка плана реализации. Обсуждение
2	Планирование	Корректировка маршрута. Совместные исследования
3	Поиск	Поиск информации в мультимедийной энциклопедии, справочнике, сети Интернет, электронном каталоге
4	Промежуточные результаты и выводы. Текущая рефлексия	Обработка информации и полученных данных с использованием электронных шаблонов; создание отчета о проделанной работе, презентации, альбома и др. Обсуждение
5	Защита проекта. Рефлексия результатов	Демонстрация отчета о проделанной работе, вручение грамот, дипломов. Обсуждение итогов

Правила выбора проекта:

1. Тема должна быть интересна обучающемуся, должна увлекать его. Исследовательская работа эффективна только на добровольной основе. Тема, навязанная, какой бы важной она ни казалась педагогу не даст должного эффекта. Вместо живого увлекательного поиска обучающийся будет чувствовать себя вовлеченным в очередное скучное мероприятие.
2. Тема должна быть выполнима, решение ее должно быть полезно участникам исследования. Натолкнуть обучающегося на ту идею, в которой он максимально реализуется как

исследователь, раскроет лучшие стороны своего интеллекта, получит новые полезные знания, умения и навыки, – сложная, но необходимая задача для работы педагога.

3. Учитывая интересы обучающихся, необходимо держаться ближе к той сфере, в которой лучше всего разбираетесь, в которой чувствуете себя сильным. Увлечь другого может лишь тот, кто увлечен сам.
4. Тема должна быть оригинальной с элементами неожиданности, необычности. Оригинальность следует понимать как способность нестандартно смотреть на традиционные предметы и явления.
5. Тема должна быть такой, чтобы работа могла быть выполнена относительно быстро.
6. Тема должна быть доступной. Она должна соответствовать возрастным особенностям обучающихся. Это касается не только выбора темы исследования, но и формулировки и отбора материала для ее решения.
7. Сочетание желаний и возможностей. Выбирая тему, педагог должен учесть наличие требуемых средств и материалов – исследовательской базы. Ее отсутствие, невозможность собрать необходимые данные обычно приводят к поверхностному решению.
8. С выбором темы не стоит затягивать. Большинство обучающихся не имеют постоянных пристрастий, их интересы ситуативны. Поэтому, выбирая тему, действовать следует быстро, пока интерес не угас.

План рассказа о проекте

1. Поприветствовать аудиторию. Представиться. Озвучить тему проекта.
2. Озвучить тему, актуальность, цели и задачи проекта.
3. Рассказать о выбранном наборе данных: источник, структура, размер.
4. Рассказать об использованных подходах, моделях и методах: причины выбора, структура, принцип работы.
5. Дать оценку качества работы модели по выбранным критериям.
6. Привести примеры работы модели.
7. В выводах озвучить, насколько достигнута поставленная цель и как усовершенствовать модель.
8. Поблагодарить за внимание.
9. Ответить на вопросы аудитории.

Общие критерии оценки проекта можно представить так:

Высокий уровень – (16-20 баллов)

1. Продукт отличается сложностью
2. Правильно поняты и сформулированы цель, задачи выполнения проекта;
3. Проект оформлен в соответствии с требованиями. Обучающийся владеет

специальными терминами и понятиями.

1. Проявлены творчество, инициатива;
2. Предъявленный продукт деятельности отличается высоким качеством

исполнения;

Средний уровень – (9-15 баллов):

1. Правильно поняты цель, задачи выполнения проекта;
2. Проект оформлен в соответствии с требованиями. Обучающийся владеет

специальными терминами и понятиями, но имеются 1-2 ошибки в этапах,
в представлении продукта;

1. Самостоятельность проявлена на недостаточном уровне.

Низкий уровень – (8-1 балл)

Не набрано минимальное количество баллов, установленное комиссией, принимающей защиту; проект не выполнен или не завершен