МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский национальный технический университет

Международный институт дистанционного обРАЗОВАНИЯ

#### Кафедра Информационные технологии в управлении

**КОНТРОЛЬНАЯ РАБОТА**

по учебной дисциплине **"Разработка программного обеспечения для мобильных платформ"**

вариант № 14

Студент(ка) *(подпись)* Прутенская Е.А

курса №5 группы 41702119 *05.02.2024*

специальность 1-40 01 01

Программное обеспечение информационных технологий

номер зачетной книжки 4170211929

Принял *(подпись)* Бумай Андрей Юрьевич

*(дата)*

МИНСК 2024

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc158025279)

[1. Способы диагностики качества кода 4](#_Toc158025280)

[2. Принципы организации командной разработки и распределения ролей среди членов команды 6](#_Toc158025281)

[3. Разработка программы Аудиоплеер 8](#_Toc158025282)

[Листинг кода 9](#_Toc158025283)

[Заключение 10](#_Toc158025284)

[Список литературы 11](#_Toc158025285)

[Приложение А 12](#_Toc158025286)

## **Введение**

В современном информационном обществе мобильные платформы становятся неотъемлемой частью повседневной жизни, предоставляя широкий спектр функциональных возможностей для пользователей. Сейчас в мире практически не осталось людей, которые бы не пользовались мобильными телефонами. Причинами их популярности можно назвать: функционал, доступ ко всем способам связи и, конечно же, небольшой размер. Большинство последних передовых моделей электронных устройств не уступают по функционалу и характеристикам громоздким мобильным телефонам и компьютерам.

Разработка программного обеспечения для мобильных устройств играет ключевую роль в обеспечении их эффективного функционирования и соответствия потребностям современных пользователей.

Данная контрольная работа посвящена изучению и практическому применению принципов разработки программного обеспечения для мобильных платформ, а также оценке полученных навыков и знаний. В рамках контрольной работы рассмотрены ключевые аспекты проектирования и реализации мобильных приложений, а также проведен анализ эффективности примененных подходов. Полученные результаты позволят сделать выводы о степени освоения материала и готовности к решению задач в области разработки программного обеспечения для мобильных платформ.

## **1. Способы диагностики качества кода**

Диагностика качества кода играет важную роль в процессе разработки программного обеспечения, поскольку от этого зависит эффективность, надежность и поддерживаемость создаваемых приложений. Существует несколько способов диагностики качества кода:

1. Статический анализ кода: Этот метод включает в себя проверку и анализ кода без его фактического выполнения. Специальные инструменты анализа кода, такие как SonarQube, ESLint, Pylint и другие, проводят проверку синтаксиса, стилевых соглашений, выявляют потенциальные ошибки, идентифицируют код, не соответствующий лучшим практикам.
2. Тестирование кода: Это включает в себя написание и выполнение тестов для проверки функциональности и корректности работы кода. Модульные тесты, интеграционные тесты и автоматизированные тесты помогают выявить ошибки и гарантировать, что изменения в коде не приведут к нежелательным побочным эффектам.
3. Анализ производительности: Оценка времени выполнения кода, использование ресурсов (память, процессорное время) позволяет выявить эффективность кода и оптимизировать его для более быстрой и эффективной работы.
4. Code Reviews (Рецензирование кода): Процесс, при котором другие разработчики анализируют код на предмет соблюдения стандартов, правильности решения задачи, идентификации потенциальных уязвимостей и ошибок. Code reviews способствуют обмену знаний и повышению качества кода.
5. Использование метрик кода: Определение и отслеживание метрик, таких как цикломатическая сложность, коэффициент поддерживаемости кода и других параметров, помогает оценить качество кода и выявить его уязвимости.
6. Анализ зависимостей: Понимание взаимосвязей между различными компонентами кода, а также зависимости от внешних библиотек и фреймворков, помогает предотвратить проблемы совместимости и несоответствия требованиям.
7. Использование инструментов автоматической сборки и непрерывной интеграции (CI/CD): Интеграция кода в общий репозиторий с последующей автоматической сборкой и тестированием позволяет оперативно выявлять ошибки и проблемы, а также гарантирует стабильность кодовой базы.
8. Автоматизированные системы анализа уязвимостей: Использование инструментов, способных выявлять потенциальные уязвимости и проблемы безопасности в коде, обеспечивает безопасность приложения и предотвращает возможные атаки.
9. Мониторинг производительности в реальном времени: Отслеживание работы приложения в процессе эксплуатации позволяет выявлять узкие места, неэффективные участки кода и недостаточную производительность, что важно для обеспечения качественного пользовательского опыта.
10. Рефакторинг кода: Периодическое улучшение структуры и читаемости кода, а также оптимизация его производительности с использованием современных подходов и паттернов способствует повышению его качества и поддерживаемости.
11. Обучение и обновление команды: Проведение тренингов и обучение членов команды новым технологиям и лучшим практикам помогает поддерживать высокий уровень профессионализма и качества кода.
12. Использование методологий разработки: Применение агильных методов разработки, таких как Scrum или Kanban, способствует более гибкой и эффективной разработке кода, а также позволяет оперативно реагировать на изменения требований заказчика.

Интеграция и объединение этих методов обеспечивает системный и всесторонний подход к диагностике качества кода, позволяет разработчикам создавать высококачественный, надежный и легко поддерживаемый код, что в свою очередь способствует успешной разработке и поддержке программного обеспечения.

## **2. Принципы организации командной разработки и распределения ролей среди членов команды**

Организация командной разработки – это сложный и многогранный процесс, который требует соблюдения определенных принципов для эффективного сотрудничества и достижения высокого качества продукта.

Распределение ролей в команде разработки играет ключевую роль в эффективности и успехе проекта. Явное определение обязанностей и ответственности помогает снизить конфликты, повысить производительность и обеспечить равномерное покрытие всех аспектов разработки.

Принципы распределения ролей должны быть гибкими и адаптивными, учитывая изменения в проекте и развитие команды. Распределение ролей – это процесс, который требует внимания к динамике команды и постоянной оптимизации для достижения максимальной эффективности.

Ключевые принципы:

1. Ясное определение целей проекта:

* Определите конечные цели проекта и общее видение.
* Убедитесь, что каждый член команды полностью понимает эти цели и их вклад в успех проекта.

1. Адекватный выбор членов команды:

* Выбирайте членов команды на основе их навыков, опыта и предыдущих достижений.
* Обеспечьте разнообразие навыков, чтобы покрыть все аспекты проекта.

1. Четкое определение ролей и обязанностей:

* Определите, какие задачи и обязанности возлагаются на каждого члена команды.
* Избегайте перекрытия обязанностей и конфликтов между ролями.

1. Формирование комплектов навыков:

* Обеспечьте, чтобы у каждого члена команды были уникальные навыки, способные дополнять друг друга.
* Поощряйте обмен знаний и навыков внутри команды.

1. Коммуникация и прозрачность:

* Обеспечьте открытую и эффективную коммуникацию внутри команды.
* Поддерживайте регулярные обсуждения, встречи и отчеты о ходе проекта.

1. Лидерство и мотивация:

* Назначьте лидера команды, который будет координировать деятельность и принимать стратегические решения.
* Обеспечьте мотивацию каждого члена команды через признание и вознаграждение за достижения.

1. Гибкость и адаптивность:

* Будьте готовы к изменениям в проекте и адаптируйте распределение ролей в соответствии с новыми требованиями.
* Поддерживайте гибкие методологии разработки, такие как Agile.

1. Обучение и развитие:

* Предоставляйте возможности для обучения и развития членов команды.
* Содействуйте повышению квалификации и освоению новых технологий.

1. Регулярные обзоры и обратная связь:

* Проводите регулярные обзоры производительности и предоставляйте обратную связь для улучшения работы команды.
* Учитывайте мнение каждого члена команды в процессе обратной связи.

1. Сбалансированный подход:

* Соблюдайте баланс между индивидуальной свободой и коллективным вкладом.
* Стремитесь к формированию команды, где каждый член может проявить свой потенциал.

Эти принципы помогут создать эффективную и гармоничную команду разработки, повысят ее производительность и поспособствуют достижению высоких результатов. Такая команда способна эффективно справляться с задачами и достигать поставленных целей в проектах различной сложности.

## **3. Разработка программы Аудиоплеер**

В современном цифровом мире аудиоплееры стали неотъемлемой частью нашей повседневной жизни, предоставляя нам возможность наслаждаться музыкой, подкастами и аудиокнигами в любое время и в любом месте. В данном контексте, вопросы, связанные с разработкой аудиоплеера, охватывают широкий спектр аспектов, начиная от выбора функциональных возможностей до обеспечения удобного пользовательского интерфейса

Данное приложение было написано на языке Java в среде разработки Android Studio. Целью данного процесса было создание функционального и интуитивно-понятного приложения для воспроизведения аудиофайлов.

Подготовка среды разработки: Разработка началась с установки Android Studio и освоения основных концепций разработки под Android. Был создан новый проект, выбран пустой шаблон и заданы все необходимые параметры.

Работа с интерфейсом: Для создания макетов экранов приложения использовались файлы XML. В них были размещены все кнопки, списки песен, изображения и другие элементы интерфейса, предусмотрены их стилизация и расположение на экране.

Обработка событий: Для всех кнопок были прописаны обработчики событий, чтобы при нажатии выполнялись соответствующие действия. Также были прописаны функции для расчета времени и анимации SeekBar. Была анимирована картинка при переключении между треками. Добавлены сторонние библиотеки для визуализации проигрывания музыки.

Работа с медиаплеером: Была изучена документация по классу MediaPlayer для управления аудиофайлами и внедрения его в проект. Также были подключены функции для отправки запросов на установку необходимых разрешений пользователя для использования приложения, таких как: разрешение на доступ к медиафайлам устройства и разрешение на доступ к микрофону устройства.

Визуализация на смартфоне: Для визуализации приложения к компьютеру через USB был подключен смартфон. Для этого в настройках телефона было необходимо включить режим разработчика и отладку по USB.

Тестирование на реальном устройстве: Интерфейс, функциональность и производительность приложения оценивалось и тестировалось в реальных условиях.

## **Листинг кода**

public class MainActivity extends AppCompatActivity {  
ListView listView;  
String[] items;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 listView = findViewById(R.id.*listViewSong*);  
 runtimePermission();}  
 public void runtimePermission(){  
Dexter.*withContext*(this).withPermissions(Manifest.permission.*READ\_EXTERNAL\_STORAGE*, Manifest.permission.*RECORD\_AUDIO*)  
 .withListener(new MultiplePermissionsListener() {  
 @Override  
 public void onPermissionsChecked(MultiplePermissionsReport multiplePermissionsReport) {  
 displaySongs();}  
 @Override  
 public void onPermissionRationaleShouldBeShown(List<PermissionRequest> list, PermissionToken permissionToken) {  
 permissionToken.continuePermissionRequest();}  
 }).check();}  
 @SuppressLint("Range")  
 void displaySongs() {  
 ContentResolver contentResolver = getContentResolver();  
 Uri songUri = MediaStore.Audio.Media.*EXTERNAL\_CONTENT\_URI*;  
 Cursor cursor = contentResolver.query(songUri, null, null, null, null);  
 if (cursor != null && cursor.moveToFirst()) {  
 int titleColumn = cursor.getColumnIndex(MediaStore.Audio.Media.*TITLE*);  
 int artistColumn = cursor.getColumnIndex(MediaStore.Audio.Media.*ARTIST*);  
  
 items = new String[cursor.getCount()];  
 ArrayList<String> mySongs = new ArrayList<>();  
 int i = 0;  
 do {  
 String title = cursor.getString(titleColumn);  
 String artist = cursor.getString(artistColumn);  
 String songInfo = title + " - " + artist;  
mySongs.add(cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.*DATA*))); *// Абсолютный путь к файлу;* items[i] = songInfo;  
 i++;} while (cursor.moveToNext());  
 cursor.close();customAdapter customAdapter = new customAdapter();  
 listView.setAdapter(customAdapter);  
 listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int i, long id) {  
 String songName = (String) listView.getItemAtPosition(i);  
 startActivity(new Intent(getApplicationContext(), PlayerActivity.class)  
 .putExtra("songs", mySongs)  
 .putExtra("songname", songName)  
 .putExtra("pos", i));}});}}

## **Заключение**

В ходе выполнения контрольной работы было проведено глубокое исследование теоретических аспектов, связанных с разработкой мобильных приложений на платформе Android. Изучены основы программирования на языке Java, принципы работы с Android Studio, а также основные компоненты пользовательского интерфейса и взаимодействия с медиа-ресурсами.

Разработка приложения на основе полученных теоретических знаний позволила применить практические навыки в создании функционального аудиоплеера. В процессе работы были решены задачи по визуализации интерфейса, обработке событий, внедрению медиаплеера, а также взаимодействию с реальным устройством через USB.

Опыт разработки приложения не только расширил технические знания, но и обеспечил понимание важности соблюдения принципов командной разработки, гибкости в адаптации к изменениям и внимания к деталям. Полученные навыки предоставляют прочную основу для дальнейшего изучения и применения в области мобильной разработки, а также подчеркивают важность баланса между теоретическими знаниями и практическим опытом в современной области информационных технологий.

## **Список литературы**

1. Глухова, Л. А. Технология разработки программного обеспечения: учеб. пособие для студентов специальности 1-40 01 01 «Программное обеспечение информационных технологий» / Л. А. Глухова. – Минск: БГУИР, 2007. – 178 с.

2. Бахтизин, В. В. Технология разработки программного обеспечения: учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2010. – 267 с.

3. Бахтизин, В. В. Стандартизация и сертификация программного обеспечения: учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2006. – 200 с.

4. Орлов, С. А. Программная инженерия. Учебник для вузов / С. А. Орлов. – 5-е изд., обнов. и доп. – СПб.: Питер, 2017. – 640 с.

5. Хьюз, Дж. Структурный подход к программированию / Дж. Хьюз, Дж. Мичтом. – М.: Мир, 1980. – 278 с.

6. Липаев, В. В. Программная инженерия. Методологические основы: учеб. / В. В. Липаев. – М.: ТЕИС, 2006. – 608 с.

7. Stephens, R. Beginning Software Engineering / R. Stephens. – Indianapolis, IN: John Wiley & Sons, Inc., 2015. – 482 с.

8. Черемных, С. В. Моделирование и анализ систем. IDEF-технологии: практикум / С. В. Черемных, И. О. Семенов, В. С. Ручкин. – М.: Финансы и статистика, 2006. – 192 с.

## **Приложение А**





