МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

Выпускная квалификационная работа

Разработка мобильного приложения для ЦБС Петроградского района

Обучающегося 4 курса
Скоробогатова Кирилла Денисовича

Научный руководитель:
профессор, доктор педагогических наук
Готская Ирина Борисовна

Санкт-Петербург
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня библиотека выполняет больше функций, чем просто предоставление доступа к книгам и объектам культурного наследия. В стенах современной библиотеки проводятся занятия, концерты, кинопоказы и другие формы мероприятий. Приспособленное пространство дает возможность организации образовательных встреч самим читателям. Таким образом, в настоящее время библиотека преобразуется в своего рода информационный центр предоставления услуг современному пользователю [постнаука].

Для информирования посетителей о предстоящих мероприятиях, обеспечения обратной связи, доступа к каталогу и оказания других услуг современные библиотеки используют собственные web-сайты.

В наши дни наличие только web-сайта уже не является достаточным показателем клиентоориентированности. Когда количество смартфонов в мире превысило общее число персональных и планшетных компьютеров вместе взятых [statcounter], и когда 10 из 11 минут, проведенных за использованием телефона, среднестатистический пользователь затрачивает на обращение к мобильным приложениям и только 9% времени на просмотр веб-страницы [vc.ru], необходимо задуматься над созданием и поддержкой мобильного приложения.

Также сайты и мобильные приложения выполняют разные задачи. Первые нужны для разового или редкого получения информации. Мобильные приложения необходимы для для постоянного взаимодействия и регулярного получения необходимой информации.

Все вышесказанное подтверждает актуальность темы выпускной квалификационной работы «Разработка мобильного приложения для ЦБС Петроградского района».

Выпускная квалификационная работа выполнена по заказу Санкт-Петербургского государственного бюджетного учреждения «Централизованная библиотечная систем Петроградского района» (ЦБС

Петроградского района) на разработку мобильного приложения, реализующего часть функционала официального сайта системы.

Объект исследования: создание мобильного приложения для государственной библиотеки

Предмет исследования: обеспечение пользователей необходимой актуальной информацией

Целью данной ВКР является разработка мобильного приложения для операционной системы Android в соответствии с требованиями заказчика. Достижение цели потребовало решения следующих **задач**:

1) проанализировать возможные существующие реализации мобильных приложений для библиотек;
2) ознакомиться с особенностями взаимодействия с системой автоматизации библиотек ИРБИС64;
3) выбрать оптимальные инструментальные средства для создания Android-приложения;
4) спроектировать интерактивный макет приложения, имитирующий работу реального приложения с помощью платформы Marvel;
5) разработать мобильное приложение для ОС Android.

В результате решения последней задачи разработанное мобильное приложение было внедрено в работу Централизованной библиотечной системы Петроградского района. Акт о внедрении представлен в приложении 1.

ГЛАВА 1. ПОДГОТОВКА К СОЗДАНИЮ ПРИЛОЖЕНИЯ

1.1. Анализ существующих решений

Сегодня в магазине программного обеспечения Play Маркет для операционной системы Android существует немало приложений для библиотек.

Анализ приложений проводился по следующим критериям:

1) Наличие свободного доступа к электронному каталогу.
2) Возможность просмотра предстоящих мероприятий.
3) Официальный статус приложения.
4) Оценка на основе отзывов пользователей больше четырех баллов из пяти возможных

На первом этапе для анализа было выбрано 15 мобильных приложений.

В результате анализа было выбрано три приложения для американских библиотек и одно для английской, анализ которых проводился на втором этапе:

- Arlington Public Library, США.
- Naperville Public Library, США.
- LPL Mobile, США.
- DerbyLibrary, Великобритания.

Публичные библиотеки в США пользуются огромной популярностью, предоставляя множество услуг, помимо традиционных. Американцы посещают их чаще, чем кинотеатры или спортивные мероприятия. [link] Поэтому не удивительно, что большинство исследуемых приложений созданы для библиотек данной страны.

Подобных решений для отечественных библиотек найдено не было. Возможно предположить, что если у библиотеки нет необходимого финансирования для разработки отдельного мобильного приложения, то, возможно, необходимый функционал для мобильного обслуживания посетителей будет реализован на специально адаптированной под этот вид устройств версии

обычного сайта. Мобильные версии сайтов двух российских библиотек будут рассмотрены после обзора отобранных мобильных приложений.

Arlington Public Library

Приложение создано для публичной библиотеки Арлингтон, находящейся в американском штате Вирджиния, и опубликовано в 2017 году. Последнее обновление зафиксировано в конце 2019 года. Насчитывает больше пяти тысяч скачиваний и имеет оценку 4,6 балла из пяти возможных.

Помимо уже описанного обязательного набора возможностей, данное мобильное приложение предоставляет следующие:

- личный кабинет читателя;
- доступ к базе знаний вопросов и ответов;
- контактная информация для всех филиалов библиотеки;
- поиск ближайших филиалов на основе местоположения пользователя;
- форма для обратной связи с библиотекой;
- сканирование штрих-кода для быстрого поиска издания;
- возможность управления уведомлениями, например об истекающем сроке использования книжного экземпляра;

На рисунке 1.1 представлены основные активности приложения. Графический интерфейс интуитивно понятен. Цветовое решение и расположение элементов в целом соответствуют основным трендам современного дизайна. Весь функционал приложения находятся в выдвигающемся боковом меню. Элемент поиска по каталогу для удобства продублирован в главном окне приложения. Тут же расположены вкладки новых поступлений, событий и новостей. Каждое мероприятие содержит достаточно подробное описание с указанием местоположения библиотеки-организатора.

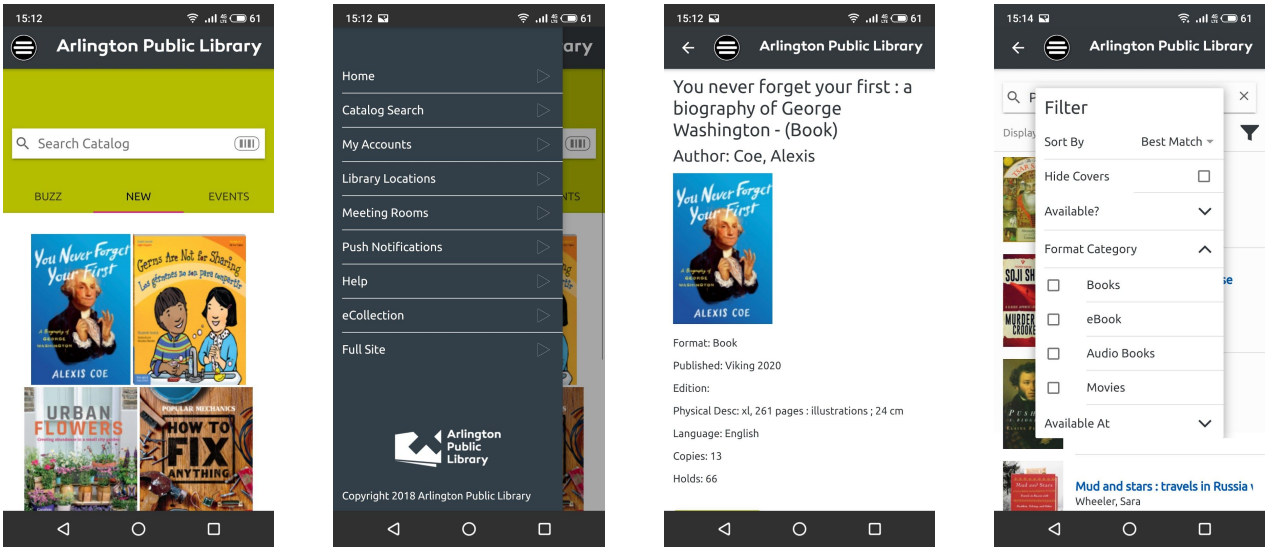

Рисунок 1.1 — Интерфейс приложения Arlington Public Library

В качестве основного недостатка приложения можно выделить отсутствие возможности поиска по всем мероприятием, включая прошедшие.

Naperville Public Library

Приложение разработано для публичной библиотеки Нейпервилл, расположенной в одноименном городе на севере США. Выход в прокат состоялся в 2016 год, а последнее обновление датируется 2019 годом. Пользовательская оценка приложения в момент анализа составила 4,3 балла из пяти возможных. Количество скачиваний приложений составляет более пяти тысяч.

Включая поиск по каталогу и возможность просмотра предстоящих мероприятий, приложение предоставляет пользователю следующие функции:

- Личный кабинет читателя, дающий возможность мониторинга информации о взятых книгах, их продления и бронирования;
- Контактная информация и расположение филиалов библиотеки;
- Быстрый поиска по каталогу, используя сканирование ISBN камерой смартфона;
- Доступ к бесплатным загрузкам, предоставляемым библиотекой для авторизированных пользователей;

- Обратная связь с библиотекой посредством телефонного звонка, СМС-сообщения, электронной почты или социальных сетей;
- Ссылки на полезные обучающие ресурсы для детей;
- Оповещения о мероприятиях с использованием стандартного пользовательского приложения «Календарь» или с помощью push-уведомлений;

Графический интерфейс приложения частично представлен на рисунке 1.2. Все элементы расположены логично, а количество шагов для получения необходимой информации не превышает трех. Внешнее оформление приложение не соответствует современному представлению мобильного дизайна.

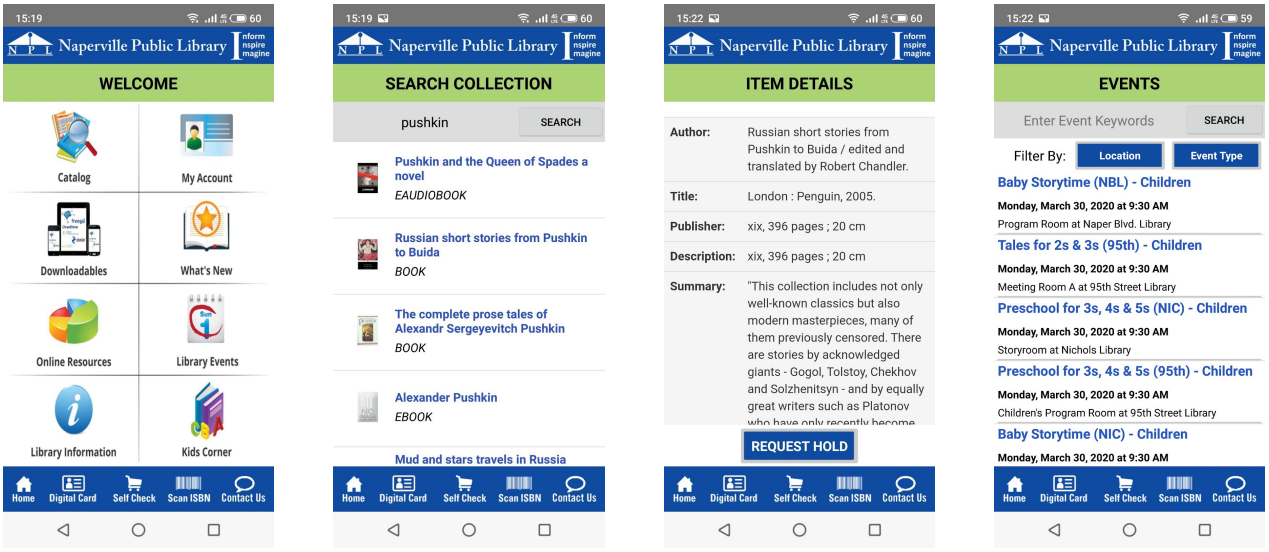

Рисунок 1.2 — Интерфейс приложения Naperville Public Library

LPL Mobile

Приложение помогает в реализации услуг публичной библиотеки Лафайета, расположенной в американском штате Луизиана. Опубликовано в 2019 году, тогда же вышло последнее на данный момент обновление. Оценка на основе отзывов пользователей составляет 4,9 балла из пяти возможных. Количество скачиваний приложения перешагнуло отметку в одну тысячу.

Предоставляется ряд возможностей:

- Просмотр данных читателя, включая списки взятых книг;

- Поиск ближайших библиотек и их контактной информации;
- Доступ к бесплатным загрузкам электронных ресурсов для авторизированных пользователей;
- Фильтр для поиска по каталогу;
- Поиск по уникальному идентификационному номеру издания (ISBN) с использованием камеры смартфона;

Снимки экрана с открытым приложением представлены на рисунке 1.3. Графический интерфейс не осложнен большим количеством элементов. Присутствует удобная навигация по разделам. Стилевое оформление в целом соответствует современному дизайну.

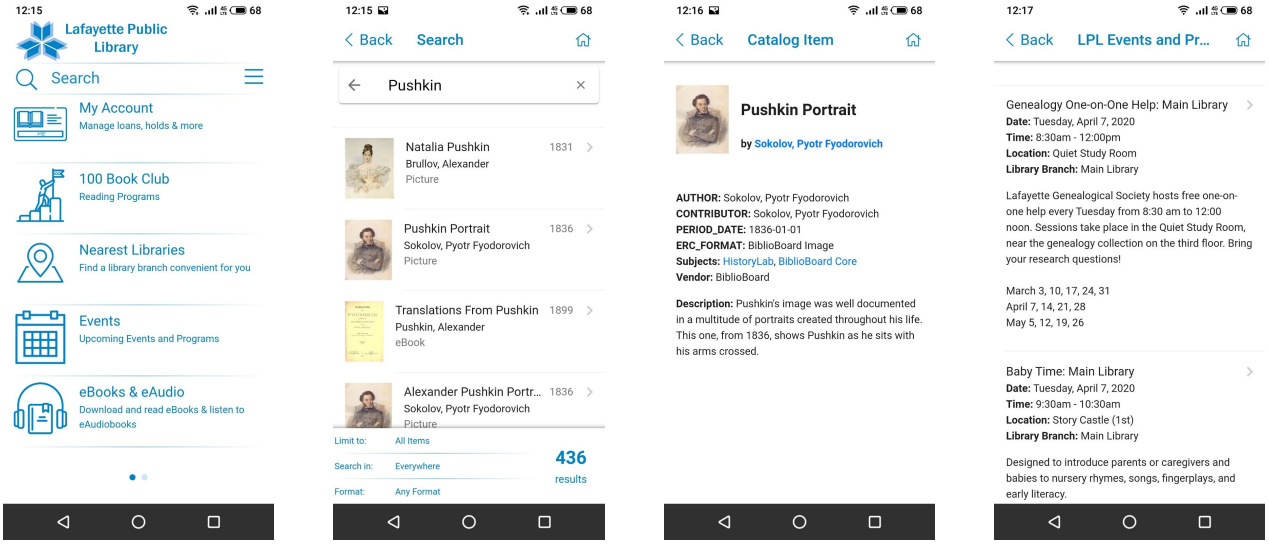

Рисунок 1.3 — Интерфейс приложения LPL Mobile

В качестве основных минусов можно отметить отсутствие:

1. Поиска по мероприятиям;
2. Удобного для восприятия списка мероприятий;
3. Возможности напомнить о предстоящем событии;

DerbiLibrary

Приложение предназначено для публичной библиотеки Дерби, расположенной в одноименном английском городе. Опубликовано в 2012 году, последние обновление зафиксировано летом 2014 года. Несмотря на большой

промежуток времени, прошедший с момента последнего обновления, приложение активно рекламируется на официальном сайте библиотеки, следовательно, не утратило актуальность. Рейтинг на момент анализа составляет 4,3 балла из пяти возможных. Количество скачивание превышает одну тысячу.

Предоставляемые возможности включают:

- Мгновенный поиск по каталогу по ключевым словам;
- Контактную информацию;
- Возможность найти книгу по фотографии штрих-кода;
- Поиск по предстоящим событиям;
- Личный кабинет читателя;
- Каталог электронных ресурсов, доступный даже не авторизованному пользователю;
- Возможность отправки напоминания о мероприятии на почту;

Примеры графического интерфейса приложения проиллюстрированы на рисунке 1.4. Сразу же бросается в глаза наличие элементов с разным оформлением, находящимся в одном месте. К примеру, центральная часть главного экрана, включая нижнее меню, представлена элементами с современным дизайном, а оставшееся пространство экрана обладает сильно устаревшим. Можно предположить, что основной функционал приложения берет на себя особый компонент платформы Android WebView, позволяющий встраивать web-страницы прямо в приложение. Невозможность использовать данное программное обеспечение без доступа к сети Интернет, только подтверждает выдвинутое предположение.

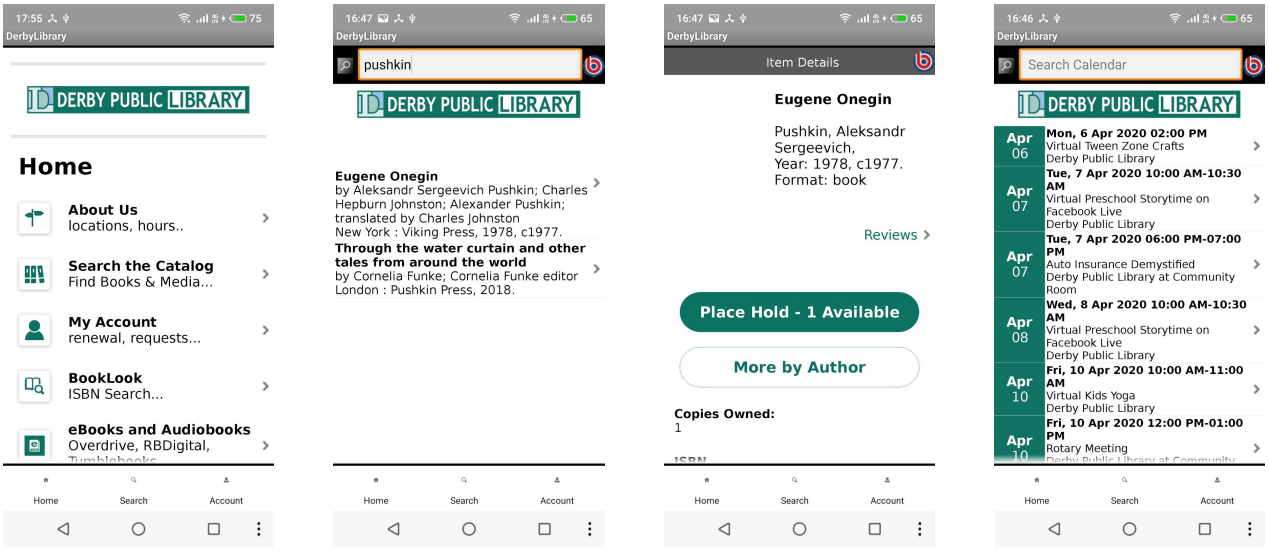

Рисунок 1.4 — Интерфейс приложения DerbiLibrary

Можно выделить основные недостатки:

1. Необходимость постоянного доступа к Интернету;
2. Отсутствие фильтра для поиска по каталогу и событым;
3. Трудная для зрительного восприятия афиша мероприятий;
4. Отсутствие единого стиля оформления;

Результаты проведенного анализа представлены в таблице 1.

Для рассмотрения мобильных версий сайтов отечественных библиотек были выбраны сайты Владивостокской и Калининградской ЦБС в соответствии со следующими критериями:

- наличие свободного доступа к просмотру электронного каталога (многие библиотеки делают просмотр каталога возможным только авторизированным пользователям);
- использования системы автоматизации ИРБИС;
- государственная библиотечная система, подразумевающая совокупность нескольких библиотек;
- организация библиотечной системой различного рода мероприятий (встречи, лекции, экскурсии и т.п.);

Сайт Владивостокской ЦБС

Сайт данной библиотечной системы имеет почти полностью оптимизированную мобильную версию, частично отображенную на рисунке 1.5.

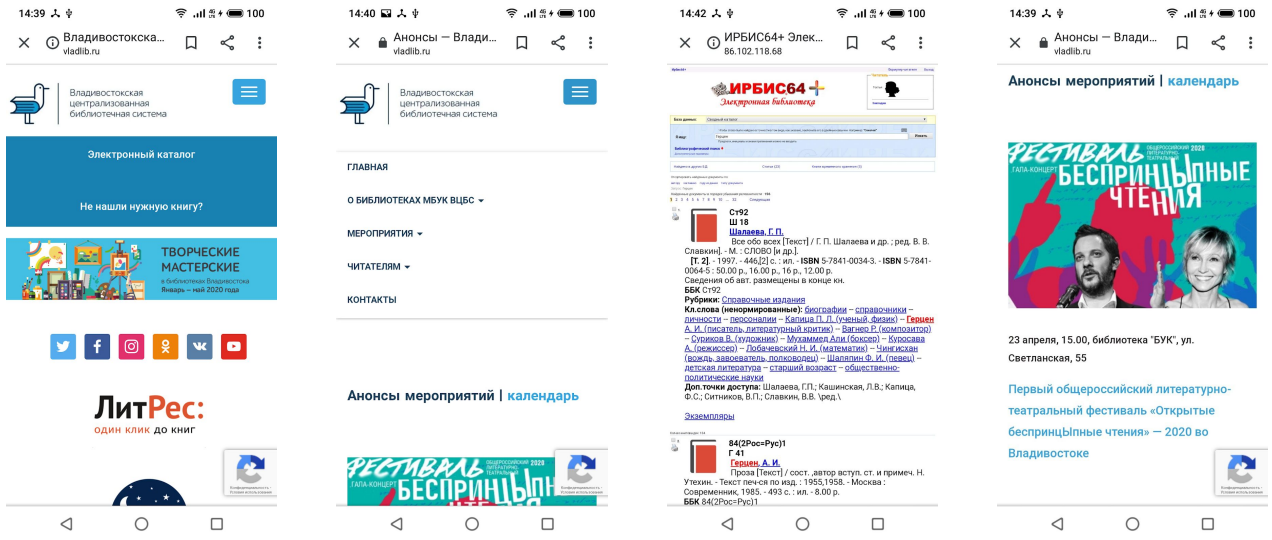

Рисунок 1.5 — Мобильная версия сайта vladlib.ru

Некоторые элементы неудобно просматривать на мобильном устройстве. Например, часть подробной информации о предстоящих мероприятиях представлена в виде таблиц, требующих прокрутки, в таком же виде представлен календарь этих событий. Возможности поиска по всем мероприятиям, включая прошедшие, не предусмотрено, но есть архив новостей в виде вертикального списка.

Страница электронного каталога представлена стандартным интерфейсом системы автоматизации ИРБИС64+ и не имеет мобильной адаптации. При этом шрифт текста информации о каждом найденном книжном экземпляре удобен для чтения без предварительного увеличения масштаба экрана. Но такая информация содержит много служебных символов, используемых системой автоматизации, что затрудняет восприятие для обычного пользователя. Можно проверить наличие экземпляров в филиалах библиотеки.

Сайт Калининградской ЦБС

На рисунке 1.6. представлены снимки экрана с разделами открытого сайта.

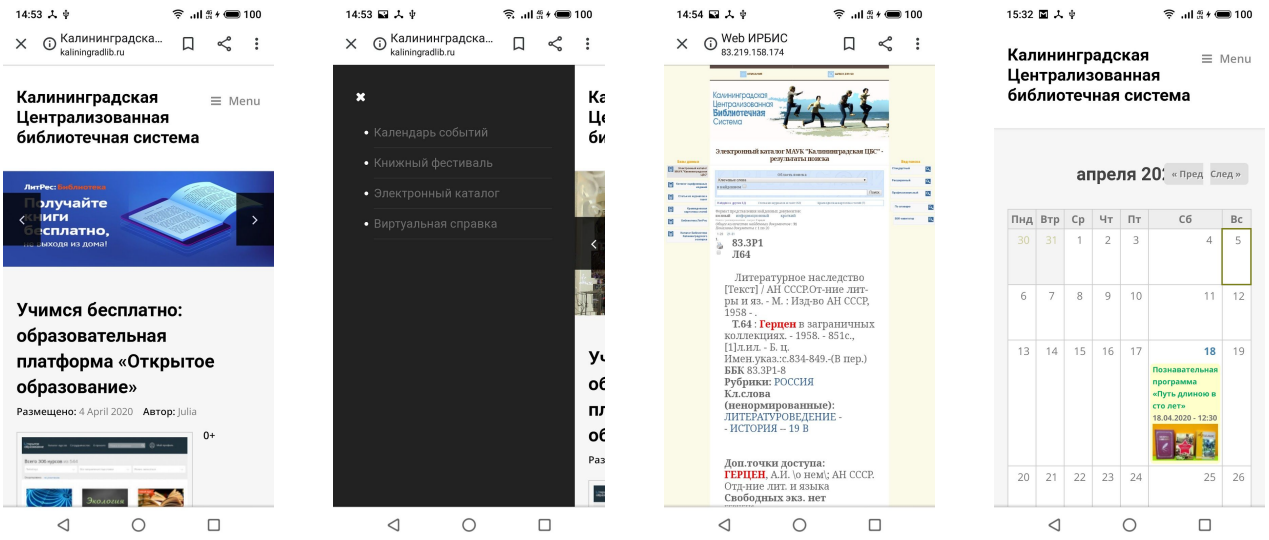

Рисунок 1.6 — Мобильная версия сайта www.kaliningradlib.ru

В разделе с информацией о мероприятиях представлен календарь, содержащий ссылку на подробное описание каждого событии. К сожалению, такая таблица плохо адаптирована для мобильного экрана. В виде списка предстоящие события не представлены, только новости.

Электронный каталог представлен интерфейсом Web ИРБИС с возможностью расширенного и профессионального поиска. Нет полной мобильной адаптации, но информация об экземпляре каталога читается. Возможно проверить наличие в библиотеках системы.

Результаты проведенного анализа представлены в таблице 2.

В соответствии с результатами анализа можно сделать **вывод**, что пока отечественные реализации мобильных приложений представлены недостаточно. Не все мобильные версии сайтов рассмотренных ЦБС позволяют с комфортом взаимодействовать с ними. Существующие зарубежные мобильные приложения предоставляют пользователям примерно одинаковый набор возможностей, включающий личный кабинет, контактные данные, адрес, режим работы библиотеки, виртуальную справочную службу, возможность поиска книги по фотографии штрих-кода и настройки уведомлений о будущих мероприятиях.

Разработчики делают акцент в первую очередь на функционал и на удобство использования.

1.2. Информационные технологии в библиотеке

1.2.1. Автоматическая библиотечная информационная система

Приоритетная задача современных библиотек заключается в автоматизации различных процессов комплектования, обработки, создания электронного каталога, оказания услуг читателям.

Благодаря появлению новых технологий часть шаблонной работы сотрудников была делегирована специализированному программному обеспечению. В результате уровень оказания услуг заметно повысился.

Изменение библиотечной системы начинается с создания электронного каталога (ЭК), предполагающего предоставление читателю электронного билета с возможностью бронирования литературы и отслеживания уже взятых книжных изданий. [ПО авт в России]

Система автоматизации библиотек (САБ), является основным компонентом традиционной автоматической библиотечной информационной системы (АБИС) и обеспечивает наиболее удобный доступ читателей к содержимому библиотечных фондов. Именно благодаря САБ возможны функции пополнения, обслуживания и предоставления пользователям каталога, поддерживаются библиотечные форматы и стандарты.

Как правило, автоматизированные библиотечные системы включают реляционную базу данных, программные средства для взаимодействия с этой базой данных и два графических интерфейса пользователя (один для читателей, другой для персонала) [wiki].

Большинство АБИС разделяют свои функции на отдельные программы, называемые модулями, каждый из которых интегрирован с единым интерфейсом. Примерами модулей могут быть:

- Каталогизация (классификация и индексация материалов).

- Контроль выдачи и возврата экземпляров.
- Мониторинг периодических издания (газет и журналов).
- Доступный для каждого ЭК и интерфейс для взаимодействия с ним.

Существует немало отечественных и зарубежных систем автоматизации библиотек. В российских библиотеках пользуются популярностью: ИРБИС, MARK, Руслан, Aleph, SQL и другие. [ПО авт в России]

Особого внимания заслуживает отечественная система автоматизации библиотек ИРБИС, так как она используется ЦБС Петроградского района — заказчиком программного продукта, описываемого в данной работе. Система разработана ГПНТБ России, являющейся исследовательским институтом и крупнейшей отечественной научно-технической библиотекой. Способна взаимодействовать с популярными иностранными и библиографическими форматами, такими как российский RUSMARC или международный UNIMARC. [elnit]

Библиографический формат используется для формирования библиографических записей, их удобного обмена и читается специальными компьютерными программами. Такой формат представляет собой совокупность данных и связывающих их взаимоотношения. Взаимоотношения и правила для заполнения данных являются важными элементами для создания национального или международного библиографического описания, зависящего от специализации формата.

Первые версии ИРБИС были установлены на компьютеры под управлением MS-DOS. Можно предположить, что за все время существования этой АБИС было учтено и исправлено множество ошибок, внесено немало полезных функций. Система соответствует общепризнанным мировым стандартам и обладает следующими возможностями:

- Позволяет создавать неограниченное количество баз данных.
- Универсальная для библиотек любого типа и профиля.

- Обладает низкими требованиями к ресурсам.
- Дает возможность описывать любые типы файлов (аудиофайлы, видеоматериалы, графические файлы и т.п.).
- Предлагает единую каталогизацию для всех филиалов библиотеки.
- Позволяет использовать штрих-коды на экземплярах и читательских билетах.
- Обеспечивает быстрый поиск за счет автоматического создания словарей.
- Обладает широким набором сервисных средств, повышающих уровень удобства пользовательского интерфейса.
- Исключает дублирование информации и многое другое (плагиат).

1.2.2. Особенности работы с системой автоматизации библиотек ИРБИС64

Система автоматизации библиотек ИРБИС представляет собой совокупность программных решений. В их число входит система ИРБИС64, предназначенная для ведение электронного каталога, система ИРБИС64+ для организации электронной библиотеки, содержащей ссылки на полные тексты изданий, компонента, основанного на веб-технологиях Web-ИРБИС, предоставляющего доступ к базам данных ИРБИС и многих других не менее важных продуктов.

Мобильное приложение, описываемое в данной работе взаимодействует с системой автоматизации ИРБИС64 посредством специальной библиотеки подпрограмм (1.3.3.).

Чтобы правильно получать и интерпретировать данные от сервера, необходимо разобраться с несколькими особенностями работы с этой системой автоматизации.

Чтобы начать взаимодействие с сервером, нужно подключиться к нему. Для этого требуется указать сетевой адрес компьютера, на котором развернут сервер, сетевой порт, имя нужной базы данных, а также логин и пароль пользователя.

Каждая запись, хранящаяся в базе данных, содержит уникальный номер файла документа — MFN, совокупность данных полей и подполей, а также другую служебную информацию. Запись можно экспортировать в специфическом текстовом обменном формате, не поддерживаемом другими библиотечными системами. Пример представления одной из записей отображен на рисунке 1.7. Каждое поле документа расположено на новой строке (за исключением поля с меткой #461, так как оно перенесено на другую строку из-за большого количества символов). Здесь номера 920, 102, … , 905 — числовые метки полей. Они обозначают, какую именно информацию хранит поле: название произведения, данные об авторе, серии и др. Некоторые поля содержат подполя, разделяемые с помощью специальных символов и буквы латинского алфавита.

```
#920: SPEC
#102: RU
#101: rus
#606: ^АХУДОЖЕСТВЕННАЯ ЛИТЕРАТУРА (ПРОИЗВЕДЕНИЯ)
#919: ^Arus^N02  ^KPSBO^Gca
#60: 10
#961: ^ZДА^АШукшин^ВВ. М.^GВасилий Макарович
#210: ^D1998
#610: РУССКАЯ ЛИТЕРАТУРА
#610: РАССКАЗЫ
#1119: 7ef4c9af-f1d3-4adc-981b-5012463155a1
#900: ^B03^C11a^Xm^Ta
#215: ^A528^3в пер.
#200: ^VКн. 3^AСтранные люди
#10: ^A5-86150-048-7^D80
#907: ^СПК^A20180613^BNovikovaIA
#461: ^CСобрание сочинений: в 6 кн.^FВ. М. Шукшин^GНадежда-
1^H1998^Z1998^ХШукшин, Василий Макарович^DМосква^U1
#903: -051259089
#910: ^A0^B1759089^DФ104^U2018/45^Y45^C20180607
#905: ^21^D1^J1^S1
*****
```

Рисунок 1.7 — Файл в обменном формате ИРБИС

Числовые метки полей могут отличаться для записей разных видов элементов каталога. Периодические, многотомные издания, аудио- и видеоматериалы имеют разные критерии описания. Это необходимо учитывать, чтобы избежать «пустых» значений при запросе данных из определенных полей.

Сервер предоставляет возможность экспортировать необходимую информацию о записи в формате ISO 2709 и в обменном формате ИРБИС.

На основе имеющихся записей в базе данных, ИРБИС64 составляет поисковой словарь системы. Он состоит из отдельных частей, для поиска по которым используются специальные префиксы. Например, *A* — для автора, *K* — для ключевых слов, *T* — для заглавий. Осуществлять поиск можно не только по префиксам словаря, но по уникальному номеру записи MFN, или по конкретным значениям полей и подполей.

Удобство поиска в автоматизированной системе ИРБИС64 обеспечивается специальным языком. Существует два возможных варианта поиска: прямой и обратный. Прямой поиск является очень быстрым, так как он основан на словарях. Последовательный наоборот — медленный. Это вызвано перебором всех записей базы данных [линк]. В описываемом в данной работе мобильном приложении целесообразнее использовать прямой поиск, так как для всех необходимых поисковых параметров уже существуют словари и подходящие для них термины.

Выражение для прямого поиска формируется с помощью терминов, префиксов словаря и логических операторов булевой алгебры. Основные операторы: *ИЛИ* (+), *И* (*), *НЕ* (^). Порядок выполнения операторов зависит от их приоритета. Первыми всегда выполняются операции *И* и *НЕ*, затем *ИЛИ*. Разрешается использование круглых скобок. Термины — это конкретные значения для поиска, например, название книги. Так как термин может содержать пробелы, итоговый запрос в виде строки два раза заключается в кавычки.

1.3. Выбор инструментальных средств

1.3.1. Анализ языков программирования

На сегодняшний день разработчикам приложений для операционной системы Android предлагается несколько языков программирования, которые они могут внедрить в свои проекты. Наиболее популярными являются официальные для системы языки Java и Kotlin. Последний в 2019 году был признан компанией Google приоритетным официальным языком программирования для Android-приложений [I/O 2019]. Также компания допускает использование C/C++ языков в отдельных частях приложения. На этом список возможных языков программирования не заканчивается, создавать приложения можно с помощью простого для восприятия Basic, стандартного языкового набора web-разработчика (HTML, CSS, JavaScript) с применением фреймворка PhoneGap, популярного языка Python в связке с фреймворком Kivy и множества других инструментальных средств. [link] Но они официально не поддерживаются Android ОС, не обладают широкой популярностью у разработчиков и как следствие, достаточным набором инструкций и разобранных решений, возможностью оперативно получить ответ на заданный вопрос в соответствующих тематических форумах и т. д. В связи с этим, выбор ограничился языками: Java, Kotlin, C/C++. Ниже представлена краткая информация по ним.

Java

Объектно-ориентированный язык, начавший свою историю с 1995 года. Входит в тройку самых популярных языков программирования на протяжении нескольких лет по данных компании TIOBE. [link] Логично предположить, что такая популярность во многом связана с возможностью создавать приложения для Android ОС. Существует огромное количество всевозможных теоретических материалов, библиотек подпрограмм, курсов для данного языка. Его выбирают за:

- **Стабильность.** Java пользуется почти неизменной популярность длительное время. Благодаря компании Oracle язык непрерывно улучшается, выпускаются новые версии.
- **Крупнейшее сообщество** программистов в Интернете [link]. Вероятность найти разобранного решения проблемы крайне велика, всегда есть возможность узнать мнение у специалиста по интересующему вопросу
- **Специальные возможности языка**, такие как автоматическое управление памятью (что очень важно, так как переполненная неиспользуемыми объектами память сильно влияет на производительность приложения), возможность многопоточности и многие другие.

Kotlin

Как и Java является статически-типизированным и объектно-ориентированным языком программирования. Новый, быстро развивающийся язык появился в 2010 году в Петербургском офисе международной компании JetBrains. [link] В 2016 году состоялся официальный релиз. Kotlin полностью совместим с Java, так как может компилироваться в аналогичный с последним байткод. [link] Язык призван сократить объем текста программы, повысить тем самым его удобство восприятия и, как следствие, уменьшить вероятность совершения ошибок. Сравнение классов для хранения данных в Kotlin и Java представлено в приложении 0.0. [link] Разработчики особенно подчеркивают безопасность языка: по умолчанию объекты не могут иметь значение Null. Это важно, так как если в процессе выполнения программы какая-то из её частей попытается обратиться к объекту со значением Null, поднимется исключение NullPointerException. Данное исключение также известно как «ошибка на миллион долларов», так как убытки компаний, связанные с его неожиданным возникновением, оцениваются в миллионы американских долларов. [link]. В связи новизной, язык пока не может сравниться с популярностью Java и

не обладает таким огромным сообществом и объемом справочных документов. Но как уже было отмечено, Google позиционирует Kotlin как приоритетный язык программирования для Android ОС с 2019 года. Поэтому, есть вероятность, что ситуация еще поменяется в сторону данного языка.

Таким образом, разработчики выбирают Kotlin за следующие особенности:

- Приоритетный статус для разработки приложения под Android ОС
- Полная совместимость с Java, и библиотеками, написанными на этом языке
- Компактный текст программы
- Возможность быстрого прототипирования приложения
- Null-защищенность по умолчанию

C/C++

Данные языки используются как правило при написании игр, где необходимы сложные графические вычисления. [link] Так как функциональная часть операционной системы Android почти полностью написана на C++ и C, то программы, написанные на этих языках, теоретически должны работать быстрее, чем аналогичные на Java или Kotlin. [link] Использовать их в своем проекте позволяет официальный набор инструментов NDK. Как правило, приложения не разрабатываются только используя эти языки программирования. В основном они идут в связке с Java, чтобы добавить, к примеру, функционал сторонней библиотеке написанной на C или C++. [link] Google рекомендует использовать данные языки только при необходимости.

Так как в разрабатываемом приложении планируется использовать библиотеку подпрограмм для взаимодействия с сервером, написанную на Java и не предполагается серьёзных вычислений, то нет необходимости в использовании языков C и C++.

В результате анализа были отобраны два наиболее подходящих языка программирования: Kotlin и Java. Было принято решение использовать оба языка:

Kotlin для взаимодействия с пользовательским интерфейсом (UI), Java — для основной логики приложения. Такое разделение связано со следующими причинами:

1) Текст программы для работы с элементами интерфейса выглядит намного компактней и понятней, если он написан на Kotlin. Например, нет необходимости каждый раз искать кнопку или текстовый блок, достаточно добавить одну специальную строчку и обращаться к элементам в максимально сокращенном виде.
2) Синтаксис Kotlin далеко не самый простой, как принято считать в современной литературе и обзорах. Минималистичность оформления не означает прозрачность принципов работы. Поэтому для основной логики был выбран язык Java.

1.3.2. Выбор интегрированной среды разработки

Интегрированная среда разработки (IDE) представляет набор инструментов для разработки программного обеспечения, взаимодействия с которыми происходит посредством графического интерфейса. Как правило, IDE состоит из следующих компонентов:

- **Редактор исходного кода**, обеспечивающий удобство и более быстрое написания текста программы: подсветка синтаксиса, подсказки, интеллектуальный поиск ошибок, автодополнение команд и др.
- **Утилиты для автоматизации процесса сборки**, необходимые для автоматизации простых повторяющихся задач в рамках создания локальной сборки программного обеспечения, таких как компиляция исходного текста программы в двоичный код, сборка из этого кода исполняемого файла и запуск автоматических тестов.

- **Отладчик**, представляющий собой программу для тестирования, с помощью которой можно поэтапно исследовать написанную программа наличие ошибок. [link]

Начиная с 2014 года компания Google рекомендует использовать интегрированную среду разработки Android Studio для создания мобильных приложений под Android ОС. [link] Это совместный продукт технологического гиганта и компании JetBrains, разработавшей IntelliJ IDEA, которая послужила основой Android Studio. До приоритетной и официально поддерживаемой для Android-разработки IDE являлась Eclipse.

Среди популярных интегрированных сред для мобильной разработки под Android ОС можно выделить следующие:

- Android Studio;
- Eclipse;
- IntelliJ IDEA.

Ниже представлена обзор каждой из этих сред в порядке их появления.

IntelliJ IDEA

Данная среда разработки была представлена компанией JetBrains в начале 2001 года. [wiki] По сравнению с уже существующими на тот момент IDE, IntelliJ IDEA обладала интеллектуальными функциями, сделать процесс программирования максимально комфортным и быстрым. В результате среда стала пользоваться большой популярностью. настоящий момент регулярно обновляется и имеет качественную службу технической поддержки. Не специализирована под конкретный язык и дает возможность создавать свой проект с использованием таких языков, как Java, Kotlin и многих других. Интерфейс программы представлен в приложении 0.

Достоинства:

- Интеллектуальное дополнение текста программы. Среда анализирует уже написанный код и на его основе предлагает возможные варианты улучшения;
- Мгновенная проверка на наличие ошибок в тексте программы;
- Возможность сразу увидеть результат выполнения программа на эмуляторе устройства или на подключенном к компьютеру реальном смартфоне;
- Доступна для Microsoft Windows, Linux и Mac OS;
- Позволяет работать с системой Git на локальной машине, или добавить аккаунт GitHub, чтобы взаимодействовать с ним напрямую из программы.

Недостатки:

- Требовательна к ресурсам системы;
- Полный функционал еды доступен только в платной версии;
- Нередки случаи ошибок, связанных с обновлением среды или плагинов. Но данные ошибки оперативно исправляются разработчиками в следующих обновлениях.

Eclipse

Проект компании Eclipse Foundation с открытым исходным кодом. [link] Универсальная интегрированная среда разработки, позволяющая использовать разные языки программирования. Была представлена во второй половине 2001 года, но до сих пор получает обновления и имеет службу поддержки. [link] Интерфейс программы представлен в приложении 0.

Достоинства:

1. Возможность вести разработку проекта с разных компьютеров с применением облачных технологий; [link]

2. Доступно огромное число плагинов для расширения возможностей и кастомизации данной IDE;
3. Возможность писать на Kotlin; [link]
4. Интуитивно понятный интерфейс;
5. Бесплатная полноценная IDE;
6. Доступна для Linux, Mac OS, Microsoft Windows;
7. Низкие системные требования.

Недостатки:

- Низкая скорость сборки проекта, составляющая в среднем 1-2 минуты; [link]
- Отсутствие интеллектуального автозаполнения и подсказок.

Android Studio

В 2013 году компания Google представила замену Eclipse — Android Studio. [link] Как уже говорилось, компания взяла за основу популярную IntelliJ IDEA. Поэтому не удивительно, что Android Studio на первый взгляд почти полностью похожа на IntelliJ IDEA в плане интерфейса программы. Разработчики Google унифицировали новую среду для мобильной разработки, предоставив все необходимое для максимально комфортного процесса создания Android-приложений. Интерфейс среды приведен в приложении 0. Android Studio вбирает в себя достоинства и недостатки IntelliJ IDEA, а также включает свои собственные, связанные с доработками компании Google. [link]

Достоинства:

- Современная система сборки проекта Gradle;
- Визуальный конструктор приложения, позволяющий создавать интерфейс не обращаясь к языку разметки;
- Настраиваемый, быстрый эмулятор;
- Возможность тестирования приложения на реальном устройстве;
- Содержит базовые макеты приложений;

- Предоставляет функции оптимизации готового приложения, автоматического обнаружения утечек памяти; [link]
- Наличие официального руководства для начинающих разработчиков.

Недостатки:

- Стабильная и быстрая работа эмулятора возможна только на системах с процессором от Intel;
- Высокие системные требования.

Таким образом, в результате анализа популярных интегрированных сред разработки была выбрана Android Studio в связи с официальной поддержкой компании Google, огромным спектром предоставляемых возможностей, а также опытом работы в программах от разработчиков JetBrains.

1.3.3. Подбор библиотек подпрограмм

Перед созданием мобильного приложения логично проанализировать существующие библиотеки подпрограмм, применение которых повысит эффективность приложения, разнообразит его интерфейс и упростит процесс разработки.

Так как описываемое в данной работе приложение должно взаимодействовать с системой автоматизации библиотек ИРБИС64, возник вопрос о существовании программных продуктов, реализующих взаимодействие с этой системой. Такие решения были найдены на платформе для хранения, обмена исходного кода программного обеспечения GitHub. Это библиотеки подпрограмм главы отдела автоматизации Иркутского национального исследовательского технического университета А.В. Миронова [link]. Они написанные на языках: Java, Python, C#, PHP и C++. Также были иные решения от других авторов, они либо не имели документации, либо давно не обновлялись разработчиком.

Из проектов А.В. Миронова был выбран наиболее подходящий под названием «JavaIrbis». Это библиотека подпрограмм, написанная на языке Java.

Она отлично подойдет для описываемого в данной работе приложения, так как она обладает полной совместимостью с операционной системой Android. JavaIrbis имеет MIT-лицензию. Это означает свободное использование данного ПО без ограничений, при этом автор не несет ответственности за возможные ошибки. [link] Последнее можно нивелировать богатым опыт работы автора с системой автоматизации библиотек ИРБИС. Библиотека подпрограмм имеет краткую документацию. Последнее обновление репозитория проекта на момент написания работы датируется 13 январем 2020 года. [link] JavaIrbis обладает необходимыми возможностями:

- Поиском записей по электронному каталогу с применением поискового словаря;
- Представлением записей в разных форматах, включая обменный формат ИРБИС.

Запись базы данных может быть представлена в виде фрагмента HTML-страницы. Например, для базы данных, хранящих информацию о мероприятиях, это один из самых удобных форматов для её извлечения. Для этих целей была выбрана популярная Java-библиотека подпрограмм jsoup, позволяющая работать со всеми видами HTML. [link] Имеет MIT-лицензию.

Некоторые книжные издания и мероприятия содержат ссылку на изображение: на обложку или постер соответственно. Чтобы загрузить картинку по ссылке и отобразить его в специальном компоненте можно использовать Picasso или Glide. Это хорошо протестированные библиотеки подпрограмм, обладающие примерно одинаковым принципом работы и синтаксисом. Для использования в проекте была выбрана библиотека Glide, поскольку эффективнее использует память устройства. [link]

ВЫВОДЫ К ГЛАВЕ I

1) Проведен анализа существующих реализаций Android-приложений для библиотек. В результате анализа было найдено несколько зарубежных решений, выявлены их достоинства и недостатки. Также были проанализированы мобильные версии сайтов отечественных библиотек, отмечены их сильные и слабые стороны.
2) Рассмотрены основные особенности взаимодействия с системой автоматизации ИРБИС64: форматы представления данных, структура отдельной записи и правила поиска по базе данных.
3) Были выбраны оптимальные инструментальные средства для разработки мобильного приложения: языки программирования Java и Kotlin для основной логики и взаимодействия с пользовательским интерфейсом соответственно, в качестве интегрированной среды разработки Android Studio, а также библиотеки-подпрограмм, применение которых должно повысить эффективность приложения.

Во второй главе данной выпускной квалификационной работы описан процесс проектирования и разработки программного продукта с помощью отобранных средств разработки.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. Описание организации и установленных требований

2.1.1. Сведения о заказчике

Централизованная библиотечная система Петроградского района является государственным учреждением, представлена пятью взрослыми, одной юношеской и двумя детским библиотеками. Основана в 1978 году. [link] В настоящее время библиотека не перестаёт пользоваться популярностью. Так, за прошедший 2019 год её услугами воспользовалось более 30 тысяч человек. [link] В этом же году современное оснащение получила библиотека им. В. И. Ленина, а также был представлен обновленный сайт системы.

Библиотечная система предоставляет доступ к книжному каталогу, содержащему более 500 тысяч экземпляров. [link] Представляет собой современное культурно-информационное пространство, для поддержания которого проводит множество мероприятий, таких как концерты, презентации, творческие встречи и д.р. Система имеет аккаунты в социальных сетях, где публикуются последние новости, осуществляется дополнительная обратная связь с читателями. Помимо внутренних событий, библиотечная система инициирует городские и районные мероприятия с участием образовательных заведений, книжных издательств и творческих общественных организаций [link]

2.1.2. Требования заказчика

На основании обсуждения вопросов необходимого функционала мобильного приложения с главой ЦБС Петроградского района, сформулировано техническое задание (Приложение). Было было принято решение о создании минимально жизнеспособного продукта (Minimum Viable Product, MVP). Такое программное обеспечение предоставляет набор минимальных, но при этом достаточных возможностей для первых пользователей. [link] На основе отзывов и

популярности приложения, руководство библиотечной системы сможет сделать вывод о дальнейшей поддержке и развитии данного программного продукта.

Были обозначены следующие требуемые возможности приложения:

1. Поиск по электронному каталогу, включающий возможность применения специального фильтра (по автору, серии, году издания и т. д.)
2. Просмотр подробной информации о каждой найденной записи каталога (библиографическое описание, доступность в библиотеках и т.д.)
3. Просмотр предстоящих мероприятий с описанием (дата, время проведения, описание, место проведения и т. д.)
4. Поиск по уже прошедшим мероприятиям
5. Просмотр информации о читаемых книгах по номеру читательского билета
6. Наличие краткой контактной информации для каждой из библиотек системы

Требования к оформлению …(может не нужно)

2.2. Проектирование интерактивного прототипа

Перед программированием продукта, имеющего графический интерфейс, логично разработать его прототип. Прототип мобильного приложения представляет собой макет, отображающий расположение элементов интерфейса, взаимодействие между ними, а также демонстрирует основную логику приложения. Благодаря нему можно понять, насколько правильно разработчик понял техническое задание. Очень важно убедиться, что разработка идет в правильном направлении с поставленными условиями заказчика. Незамеченные на начальном этапе отклонения могут привести к незапланированным изменениям готового программного продукта или даже срыву всего проекта.

Существует множество вариантов представления прототипов: от наброска на обычной бумаге до составленного в профессиональной программе анимированного макета. [link] Также прототип может быть интерактивным, то есть взаимодействие между его элементами приближено к реальной версии продукта. Такой прототип нагляднее, чем неинтерактивный и не требует спецификации, а значит удобнее как для исполнителя, так и для заказчика.

В данной работе было принято решение использовать онлайн-платформу для прототипирования Marvel. Во-первых, Marvel обладает собственным редактором, включающим базовые элементы интерфейса для разных операционных систем. Во-вторых, позволяет быстро вносить правки в любой момент, так как макет хранится на стороне компании. В-третьих, сделанным интерактивным прототипом легко поделиться — сервис предоставляет ссылку на специальную страницу с для взаимодействия с ним.

Скриншот интерфейса платформы с разрабатываемым макетом для главной страницы приложения представлен на рисунке 2.1. Имеющийся инструментарий позволяет создавать макеты различной детализированности: содержит как базовые геометрические фигуры и текст, так и элементы, характерные для

определенных операционных систем, такие как панель навигации, строка состояния и д. р.

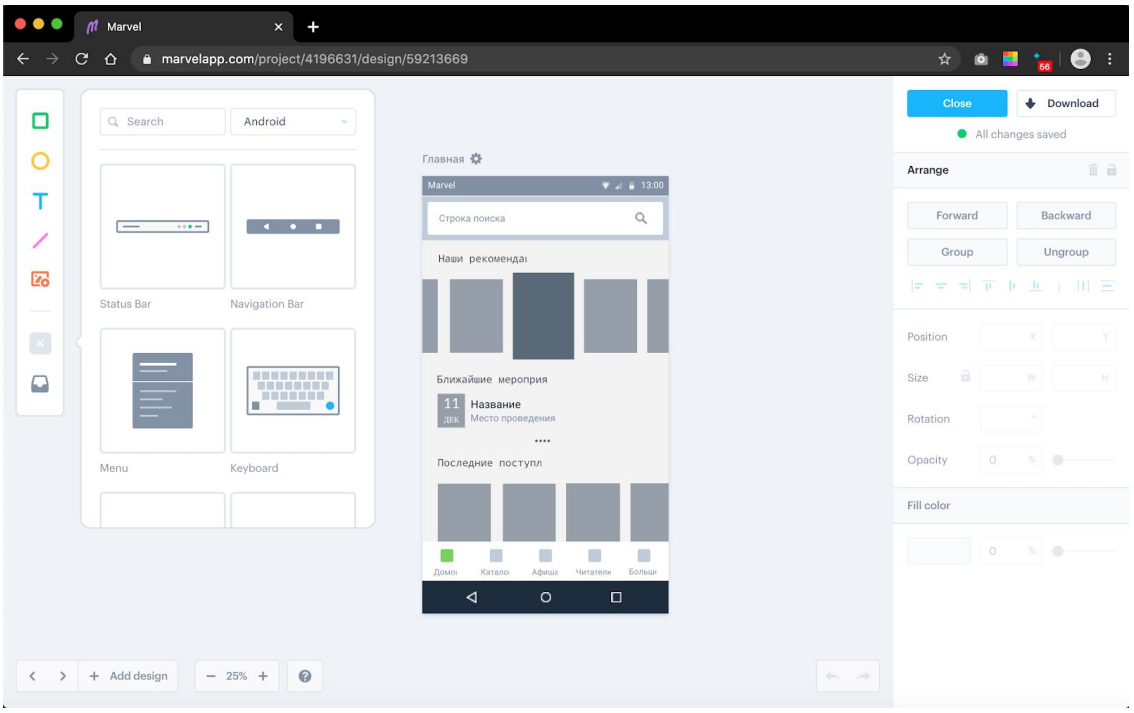

Рисунок 2.1 — Процесс создания макета

Разработанный прототип не содержит конечного стилевого оформления приложения: отсутствуют изображения, цвета и т. д. Необходим для понимания общей логики интерфейса.

Прототип приложения состоит из 12 экранов. Связь между ними отображена в приложении 0. В соответствии с техническим заданием, приложение должно иметь 5 основных разделов, поэтому в качестве наиболее подходящего способа переключения между ними рекомендуется использовать нижнюю навигационную панель (Bottom Navigation Bar) [link]

Стартовый экран

Загрузочный экран, во время отображения которого осуществляется загрузка актуальных данных с сервера.

Главный экран

Содержит прокручиваемые карточки рекомендованных и недавно появившиеся в библиотечной системе книжных изданий. Также отображает

панель быстрого поиска по электронному каталогу и список нескольких ближайших по дате мероприятий.

Каталог

Экран для поиска по электронному каталогу, найденные записи которого отображаются в горизонтально прокручиваемых карточках. Карточки содержат обложку, краткие фрагменты из библиографического описания, а также информацию о факте наличия в одной из библиотек системы. При нажатии на карточку открывается экран с подробным описанием записи. Для уточнения поиска на экране Каталог присутствует специальная кнопка, перенаправляющая на экран расширенного поиска.

Информация о книге

Содержит полное библиографическое описание записи электронного каталога, а также наличие экземпляров в каждой их библиотек системы

Расширенный поиск

Экран предоставляющий возможность сократить результаты поиска с использованием специальных параметров: наличия в одной или нескольких библиотеках, года выпуска, серии и т. д. Присутствуют поля для рукописного ввода и блоки с выбором необходимого элемента.

Афиша

Экран с горизонтально прокручиваемыми карточками ближайших мероприятий, отсортированных по дате. Каждая карточка содержит название события, аннотацию, наименование библиотеки-организатора, контактные данные, а также кнопку для перехода к подробному описанию. Экран также содержит строку поиска. Найденные мероприятия отображаются в том же месте и в том же виде, что и ближайшие. Для возвращения актуальных событий предусмотрена кнопка, расположенная рядом со строкой поиска.

Информация о мероприятии

Предоставляет подробную информацию о мероприятии.

Читателю

Имеет два представления. Отличаются карточкой, связанной с актуальной информацией взятых из библиотеки книг. Сначала пользователь вводит номер своего читательского билета. Затем, после нажатия подтверждающей кнопки, карточка заменяется другой, содержащей номер введенного номера билета, количеством читаемых книг и кнопкой, ведущей на экран со списком этих книг.

Список взятых книг

Содержит набор вертикально расположенных карточек с информацией о читаемых книгах: дате получения в библиотеке, прошедшем с этого момента количестве дней, а также библиографическое описание.

Как стать читателем

Представляет собой список раскрываемых и скрываемых областей, содержащих ответы на возможные вопросы читателя библиотеки.

Контакты

Данный экран приложения содержит общую контактную информацию для библиотечной системы, а также горизонтально прокручиваемые карточки с минимально необходимой информацией для каждой библиотеки системы, включая актуальные для временя открытия приложения часы работы.

Чтобы придать интерактивность готовым макетам экранов в платформе Marvel, необходимо открыть специальную страницу и с помощью выделения необходимых областей настроить ссылки на желаемые страницы.

2.3. Структура приложения

Любое Android-приложение включает файл-манифест, набор ресурсов, а также исходный код. Файл-манифест содержит информацию о приложении, передаваемую операционной системе устройства, на котором оно запускается. Это данные о минимальной поддерживай версии системы устройства, требуемых разрешениях, объявленных компонентах, библиотеках и д. р. [link link]. Ресурсы приложения представлены графическими элементами, макетами, строковыми константами, цветовой схемой, стилями и д. р.

В рамках описываемого приложения в качестве компонентов были использованы активности (activities) и намерения (intents). Активность — обязательный компонент, представляющий собой единый экран с пользовательским интерфейсом. Каждая активность обладает собственным макетом. Намерение — объект, с помощью которого происходит взаимодействие между всеми компонентами Android: обеспечивается переход от одной активности к другой, передача необходимых данных и д. р. [link, link]

Помимо активностей в проекте описываемого мобильного приложения используются фрагменты (fragments). Это несамостоятельные элементы, которые встраиваются в активности и выполняют схожую с ними роль. Так главная активность приложения MainActivity.kt содержит пять фрагментов, переключаемых с помощью нижней панели навигации. Без использования фрагментов пришлось бы делать пять активностей с одинаковой панелью навигации, при переключении которых наблюдалось бы заметное мерцание всего экрана приложения.

Чтобы оптимизировать работу Android-приложения и избежать возможных ошибок, необходимо учесть, что любая активность имеет жизненный цикл. Это важно, так как, к примеру, попытка обратиться к контексту уже не существующей активности приведет к серьезной ошибке. Жизненный цикл активности — это

процесс от ее создания до уничтожения операционной системой устройства. Состоит из семи этапов, соответствующих вызовам следующих методов:

- **onCreate()** — вызывается при создании активности; в методе выполняется базовая логика активности, которая должна произойти только один раз за весь срок её существования
- **onStart()** — делает активность видимой для пользователя
- **onResume(** — перед началом взаимодействия пользователя с активностью
- **onPause()** — при смене активности
- **onStop()** — в момент, когда активность больше не видна пользователю
- **onDestroy()** — перед уничтожением операционной системой устройства [Head]

Фрагмент также обладает жизненным циклом, но в отличии от активности основная логика содержиться в onViewCreated(). Методы onResume() и onStop() в описываемом приложении используются для скрытия и показа панели действий главной активности, содержащей блок поиска. Метод onDestroy() очищает результаты поиска по каталогу в соответствующем для фрагменте.

Классы описываемого приложения можно разделить на несколько групп:

- Активности;
- Фрагменты;
- Классы, реализующие основную логику (получение и форматирование данных с сервера);
- Модели данных;
- AsyncTask;
- Класс базы данных.

Приведенные группы классов будут рассмотрены в последующих параграфах главы.

2.4. Интерфейс приложения

Создание графического пользовательского интерфейса можно назвать первым этапом реализации приложения. IDE Android Studio обладает визуальным редактором, при использовании которого нет необходимости прибегать к разметке. Однако в рамках текущего проекта удобнее оказалось ручное редактирование XML-кода макета. Открытое окно предварительного просмотра синхронно с вносимыми изменениями обновляет представление макета. На рисунке 2.2. представлен скриншот процесса создания макета для экрана расширенного поиска по электронному каталогу.

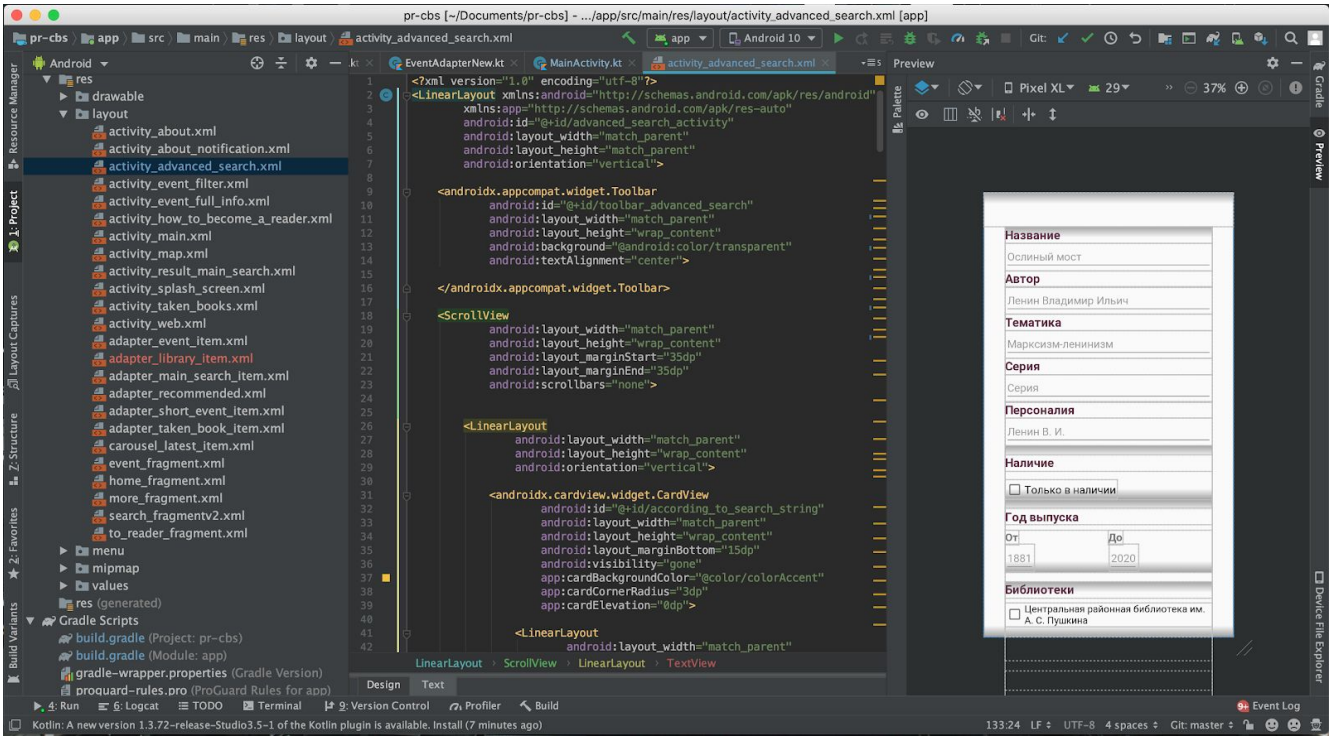

Рисунок 2.2 — Процесс создания макета активности в Android Studio

Каждый макет представляет собой структуру вложенных элементов, образованных от базовых классов View и ViewGroups. [li] Вложенные элементы — контейнеры и готовые к использованию компоненты. В описываемом приложении для разных целей применяется несколько видов контейнеров:

- **LinearLayout** — простой вид контейнера для горизонтального или вертикального расположения элементов внутри него.

- **ReiativeLayout** — внутренние элементы располагаются относительно родительского контейнера или других компонентов; контейнер допускает перекрытие элементов между собой, что можно использовать на экранах, где одни блоки интерфейса должны меняться на другие, например, в фрагменте поиска по каталогу, где в зависимости от его результатов, наличия подключения к Интернету изменяется отображаемая информация.
- **FrameLayout** — контейнер с самой простой разметкой, в нем можно разместить фрагменты. Так приложение содержит пять фрагментов, переключаемых в таком контейнере в главной активности.
- **ConstraintLayout** — современный способ организовать элементы, задавая каждому как минимум одно ограничение по горизонтали и одно по вертикали. Такой контейнер используется, например, для размещения элементов внутри карточки с информацией о конкретном мероприятии.
- **ScrollView** — позволяет прокручивать находящиеся в нем элементы

Почти все макеты состоят из комбинации приведенных контейнеров. Внутри них располагаются такие компоненты, как текстовые блоки, кнопки, графические объекты, поля для ввода и многие другие.

Изменение графического интерфейса происходит из класса активности или фрагмента, к которому прикреплен необходимый макет.

Как уже упоминалось, некоторые компоненты необходимо скрывать и показывать в зависимости от ситуации. Для этого меняется свойство android:visibility на:

- VISIBLE — компонент отображается на экран;
- INVISIBLE — компонент не отображается на экране;
- GONE — компонент не отображается на экране, но и не занимает места в макете.

Разработанный интерфейс мобильного приложения отображен в приложении 0.

2.5. Обеспечение основного функционала приложения

2.5.1. Работа с потоками

Операционная система Android устроена таким образом, что каждое ее приложение запускается в отдельном процессе. Это необходимо, в первую очередь, для предотвращения нежелательного влияния одного приложения на другое, если в работе первого, к примеру, возникли критические ошибки. В процессе по-умолчанию запускается главный поток (main thread). В нем запускаются все компоненты Android-приложения. При запуске трудоемких операций в главном потоке происходит его блокирование и, как следствие, нарушается работа всего приложения. Это сетевые операции, взаимодействия с базой данных и д. р. [link, link]

В описываемом мобильном приложении необходимо взаимодействовать с сервером, чтобы получить от него нужные данные — это сетевая операция. Некоторые полученные данные сохраняются в базу данных, а затем считываются — это обращения к базе данных. Чтобы не мешать работе главного потока, эти процессы должны выполняться в отдельных потоках.

Описываемое приложение должно показать, что загрузка началась, а потом отобразить полученную информацию на экране. Следовательно, необходим класс для работы с потоками, имеющий возможность взаимодействовать с пользовательским интерфейсом (UI) приложения. Такой особенностью обладает класс AsyncTask. Это простой и удобный способ справится с поставленной задачей. Данный класс имеет три основных метода:

1) **onPreExecute()** — выполняет подготовительные действия перед началом фоновой операции, например, показывает индикатор начала ее выполнения (progress bar);
2) **doInBackground(Params)** — главный метод, не имеющий доступ к UI. Запускает на исполнение трудоемкую операцию;

3) **onPostExecute(Result)** — необходим для обновления пользовательского интерфейса в соответствии с результатами выполненной операцией.

Несмотря на доступ некоторых методов класса AsyncTask к элементам UI, Android Studio не рекомендует передавать в качестве его аргументов ссылки на сами объекты интерфейса: активность или фрагмент, на элементы которых будет ссылаться AsyncTask после выполнения, могут не существовать к этому моменту, что приведет к серьезной ошибке приложения. В качестве возможного решения данной проблемы может быть использован метод обратного вызова (callback). Такой метод нужно вызвать на финальной из рассмотренных стадий выполнения AsyncTask. Для его создания используется интерфейс (interface), определяющий функционал, но не имеющий конкретной реализации. Затем интерфейс применяется к классу фрагмента или активити, а метод этого интерфейса переопределяется. Как только AsyncTask закончит свое выполнение, вызовется соответствующий переопределенный метод. Данный способ реализуется в приложении во время внесения и считывания информации из базы данных с помощью трех потоков. Чтобы не допустить таких ситуаций, когда в базу данных одновременно записывается и считывается информация, потоки с этими операциями запускаются один за другим.

На рисунке 2.3. Изображен один из используемых в проекте классов AsyncTask. Вызывается в фрагменте, где необходимо сделать доступной для нажатия кнопку перехода на активити, содержащую список числящихся за читателем книг в случае их наличия. В методе onPreExecute() происходит удаление существующих экземпляров класса-модели взятой книги. Далее в doInBackground(vararg) выполняется метод для загрузки данных, а результат выполнения присваивается специальной переменной. Наконец это переменная передается callback-функции allTakenBooksLoaded(resCode) в методе onPostExecute(result) и становится доступна в классе, где он был переопределен. Далее у кнопки вызывается специальный метод для её активации.

```
class LoadTakenBooksAsyncTask(
        private val callback: LoadTakenBooksFinished,
        var ticketNumber: String,
        private val internetConnection: Boolean
) : AsyncTask<Unit, Unit, Unit>() {
    private var hasResult: Int = 0
    override fun onPreExecute() {
        super.onPreExecute()
        TakenBookStorage.Instance().clear()
    }
    override fun doInBackground(vararg p0: Unit?) {
        this.hasResult = TakenBookStorage.Instance().downloadAllTakenBooks(ticketNumber,
internetConnection)
    }
    override fun onPostExecute(result: Unit?) {
        super.onPostExecute(result)
        callback.allTakenBooksLoaded(hasResult)
    }
    interface LoadTakenBooksFinished {
        fun allTakenBooksLoaded(resCode: Int)
    }
}
```

Рисунок 2.3 — Пример класса AsyncTask

2.5.2. Получение и форматирование данных

Основная задача описываемого приложения — получение и форматирование данных от сервера. Для её достижения используются два класса: модель и класс для основной логики. Модель представляет собой описание необходимой информации для отдельно взятой записи базы данных. Это шаблон, массив экземпляров которого будет формироваться в классе с основными методам.

Описанное решение используется для получения записей электронного каталога, данных о предстоящих мероприятиях и книгах, числящихся читателями. Классы для реализации этих задач различаются моделями,

возможность порционной загрузки, необходимостью сохранять полученные данные.

Класс BookStorage частично представлен в приложении 0. В данном классе используется библиотека подпрограмм JavaIrbis, которая уже упоминалась в первой главе. BookStorage содержит следующие основные методы для работы с электронным каталогом:

- **getIrbisConnection()** — осуществляет подключение к необходимой базе данных;
- **fetchMFNsByQuery**(String query, Boolean internetConnection, Boolean is Advanced) — формирует массив из идентификаторов для каждой найденной записи по введенному запросу. Тут же идет обработка возможного исключения при подключении к базе данных, факта наличия записей по запросу или подключения устройства пользователя к сети Интернет;
- **loadNextPage**(int position) — добавляет в массив экземпляров модели загруженные с помощью другого метода экземпляры порционно по 10 штук. База данных электронного каталога состоит из огромного числа записей. Чтобы не затрачивать время и ресурсы устройства пользователя, данные загружаются частями, только в случае, если количество найденных записей меньше или равно 15, они скачиваются сразу;
- **downloadBookRecord**(int mfn, IrbisConnection connection) — наполняет данными экземпляр класса модели, полученной и отформатированной информаций в сотвествии с еднтификатором записи. Форматирование в большей степени происходит с помощью библиотеки JavaIrbis: запись можно получить в обменном формате IRBIS, отдельно сформировать полное библиографическое описание или получить значение поля записи по его номеру. Полученные данные имеют строковый тип данных. Так как каталог содержит записи разных типов, поля для них также разные,

вследствие чего, появляются их Null-значения. Значения для ряда таких параметров (автор, название и год издания) извлекаются из библиографического описания с помощью отдельных методов. Также с помощью библиотеки JavaIrbis нет возможности сразу получить все необходимые данные. Так, например, чтобы получить информацию о наличии книжных изданиях в библиотеке вызывается метод **getCopiesInfo**(String description), извлекающий ей из записи в обменном формате IRBIS.

На рисунке 2.4 представлено взаимодействие описанных методов класса BookStorage между собой, а также с классами AsyncTask.

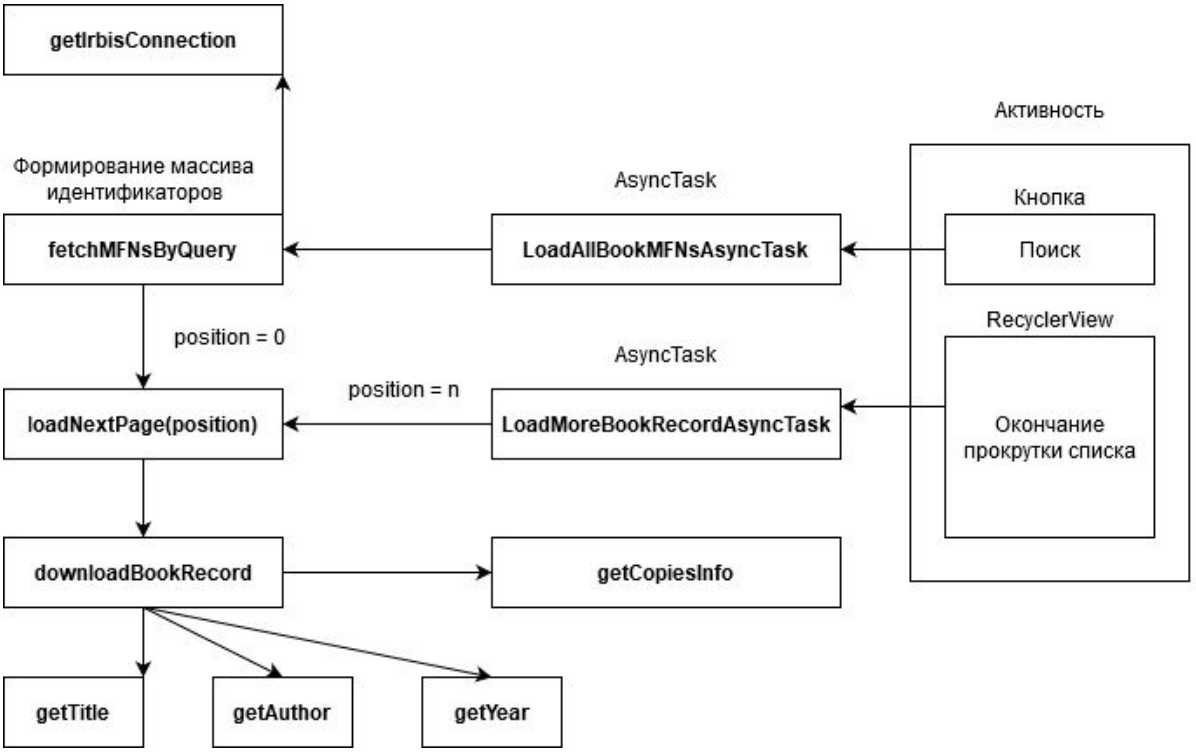

Рисунок 2.4 — Схема частичного взаимодействия методов класса BookStorage между собой и с классами AsyncTask

Устройство класса для работы с данными о мероприятиях почти аналогичное. За исключением того, что данные получаются не с помощью указания конкретных полей, а также присутствует необходимость их хранения. Библиотека JavaIrbis дает возможность получить данные с сервера в

оптимизированном формате, который он сам подберет на основе вида записи. Для базы данных с мероприятиями это фрагмент HTML-страницы, содержащий всю необходимую информацию. Для процесса его парсинга используется библиотека подпрограмм jsoup. Участок текста программы, отвечающего за получение данных из HTML-страницы приведен в приложении 0.

На устройство пользователя помещается информация о ближайших мероприятиях во время работы загрузочного экрана каждый раз, когда с даты последнего обращения к серверу прошло 24 часа. Это сделано для оптимизации ресурсов пользователя (расход Интернет-трафика на каждое скачивание и д. р.) и уменьшения нагрузки на сервер.

Так как приложение должно осуществлять поиск по всем мероприятиям, включая прошедшие, класс обладает методом для получения данных без их сохранения.

Сервер не дает возможность получить отсортированные по дате записи. Процесс сортировки реализован с помощью базы данных SQLite. При выборе всех строк заполненной таблицы указывается столбец по которому необходимо сделать сортировку.

На рисунке 2.5. изображена схема взаимодействия основных методов класса EventStorage для получения и форматирования данных о мероприятиях . Первое взаимодействие с ним начинается на экране загрузки приложения. В этот момент с помощью метода loadAllActualEvents(...) проверяется разница между текущей датой на устройстве пользователя и сохраненной датой последнего обновления. Если хранящаяся в БД информация актуальна, то на ее основе в методе loadEventFromDatabase(...) формируется массив из экземпляров специального для мероприятия класса модели. Если же информации о последнем обновлении нет, или же она устарела, то вызывается метода downloadActualEventRecord(...) происходит получение и добавление данных в БД. В остальном работа класса

похожа с работой класса BookRecord для взаимодействия с электронным каталогом.

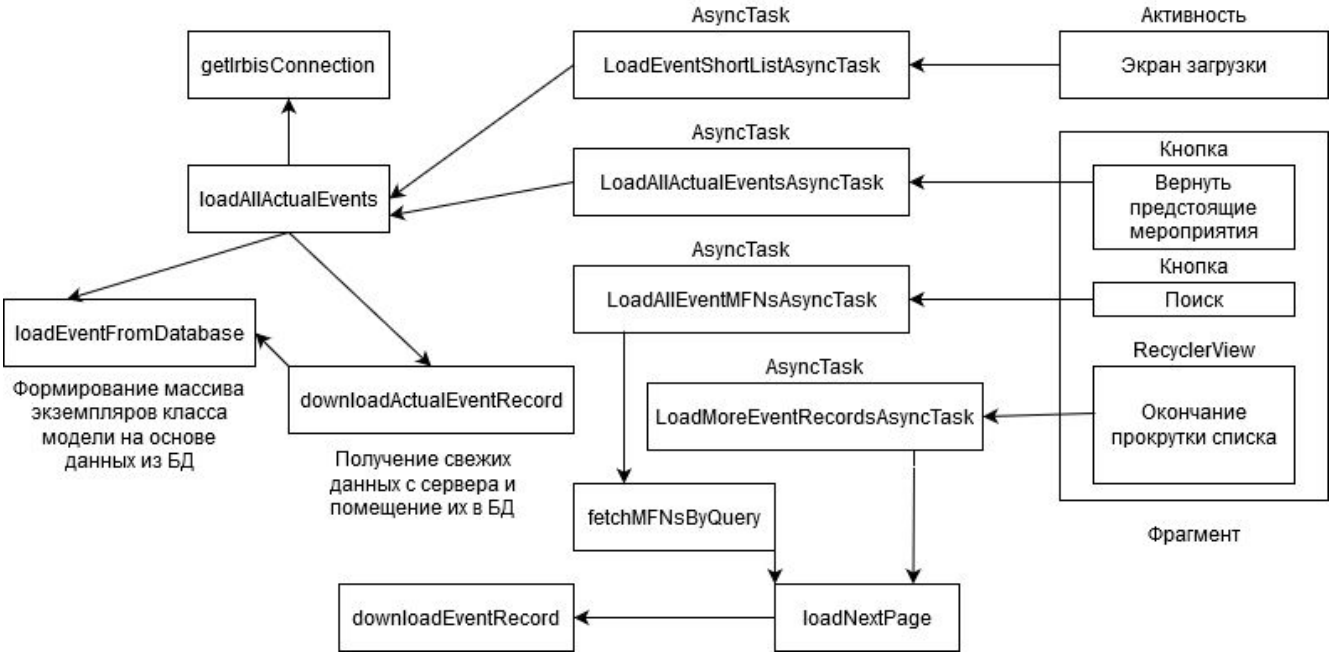

Рисунок 2.5 — Схема работы главных методов класса EventRecord

Классы для получения данных о числящихся за читателем позициях каталога, о рекомендованных и последних поступлениях книжных изданий рассмотрены не будут, так представляют собой урезанные версии уже рассмотренных классов с разными поисковыми выражениями.

2.5.3. Хранение информации

Данные о предстоящих мероприятиях обновляются не чаще одного раза в день, поэтому загружать их каждый раз, когда пользователь открывает приложение нецелесообразно. Информация о рекомендованных книжных изданиях также редактируется не часто — раз в неделю. Поэтому, полученные данные необходимо было где-то хранить. В качестве решения для хранения была выбрана база данных SQLite, которая встроена в операционную систему Android по умолчанию.

Выбор SQLite обусловлен прежде всего тем, что данная БД:

- Будет гарантированно работать на всех устройствах, под управлением Android ОС.
- Эффективно использует ресурсы устройства: база данных использует вычислительную мощность процессора только в момент работы с ней. Это помогает сберечь заряд аккумулятора устройства.
- Легковесная и быстродействующая. Текст программы, который считывает и записывает данные, переводится на язык программирования С, что и обеспечивает быстроту выполнения этих операций. [Head First Android Development]

Для начала работы с SQLite необходимо создать класс, наследуемый от SQLiteOpenHelper. Фрагмент созданного для этой цели класса DBHelper представлен в приложении 0. Класс содержит два обязательных метода:

1) **onCreate()** — вызывается при первом создании БД. Содержит описания всех таблиц базы: их названия, наименования столбцов и типов данных, которые в них будут хранится;
2) **onUpgrade()** — вызывается при изменении БД. В классе DBHelper этом методе реализовано удаление существующих таблиц и повторный вызов метода onCreate().

Для управления созданной БД используется класс SQLiteDatabase. Среди используемых методов класса можно выделить следующие:

- **execSQL()** — позволяет выполнить код на языке SQL;
- **query()** — отвечает за чтение данных из конкретной таблицы. В качестве параметров можно указать необходимые столбцы, условие выборки, группировки, сортировки и д. р. Совокупность полученных данных возвращается в объект Cursor;
- **insert()** — добавляет строку в выбранную таблицу БД. Строка должна быть представлена экземпляром класса ContentValues. [link]

На рисунке 2.6. представлен фрагмент из класса EventRecord позволяющий добавлять и извлекать данные из созданной БД DBHelper. В методе query() используется сортировка по дате мероприятия от ближайших к более поздним

```
//внесение данных
SQLiteDatabase database = DBHelper.getInstance(context).getWritableDatabase();
database.execSQL("delete from events");
ContentValues contentValues = new ContentValues();
...
contentValues.put(DBHelper.KEY_START_TIME, start_time);
...
database.insert(DBHelper.TABLE_EVENTS, null, contentValues);

//извлечение данных
SQLiteDatabase database = dbHelper.getWritableDatabase();
Cursor cursor = database.query(DBHelper.TABLE_EVENTS, null, null, null, null, null,
"start_date" + " ASC", null);
if (cursor.moveToFirst()) {
   int startDateIndex = cursor.getColumnIndex(DBHelper.KEY_START_DATE);
   ...
   do {
       EventRecord eventRecord = new EventRecord();
       eventRecord.start_date = cursor.getString(startDateIndex);
       ...
   } while (cursor.moveToNext());
}
cursor.close();
```

Рисунок 2.6 — Осуществление взаимодействия с БД SQLite

Файл созданной базы данных легко найти с помощью встроенного в Android Studio менеджера файлов устройства, а затем открыть в любой программе, умеющей читать файлы с расширением .sql.

Как уже упоминалось, взаимодействие с БД происходит только после оценки даты последнего обновления. В расширенном поиске по каталогу указанные параметры не должны обнуляться после его применения. Вводить номер читательского билета каждый раз, когда в этом есть необходимость, крайне неудобно для пользователя. Все перечисленное является простыми данными и требует сохранения. Для этого был выбран встроенный в Android ОС интерфейс SharedPreferences, реализующий методы для модификации данных в формате ключ-значение в специальном файле. [link]

Обращение к интерфейсу должно происходит из разных частей приложения, чтобы не дублировать фрагменты текста программы, в классе главной активности MainActivity были созданы основные статические методы для взаимодействия с интерфейсом. Они изображены на рисунке 2.7. Это три метода, которые отвечают добавление, получение и проверку наличия данных в SharedPreferences.

```
companion object {
    fun putInSharedPreferences(name: String, value: String, context: Context) {
        val sharedPreferences = context.getSharedPreferences("pref_settings",
Context.MODE_PRIVATE)
        val editor = sharedPreferences.edit()
        editor.putString(name, value)
        editor.apply()
    }
    fun getFromSharedPreferences(name: String,  context: Context): String {
        val sharedPreferences = context.getSharedPreferences("pref_settings",
Context.MODE_PRIVATE)
        return sharedPreferences.getString(name, "").toString()
    }
    fun checkSharedPreferenceAvailability(name: String, context: Context): Boolean {
        val sPref = context.getSharedPreferences("pref_settings", Context.MODE_PRIVATE)
        if (!sPref.contains(name)) return false
        return true
    }
}
```

Рисунок 2.7 — Методы для взаимодействия с интерфейсом SharedPreferences.

2.5.4. Отображение полученных данных

Почти все получаемые с сервера данные отображаются в приложении в виде списка прокручиваемых карточек, представленными элементом CardView. Для единичной организации такого вида представления данных необходимы:

- **Макеты**: один с RecyclerView для активности или фрагмента, другой для элемента списка.

- **Диспетчер разметки RecyclerView**, содержащий список элементов. Рекомендуется использовать при необходимости отображения большого массива информации и данных, которые часто меняются. [link]
- **Адаптер** — специальный класс для поддержания связи отдельного элемента списка с массивом данных. Содержит метод, в котором описываются значения для компонентов макета элемента списка.
- **Массив данных**. В рамках описываемого проекта представляет собой массив экземпляров класса модели.

На рисунке 2.8 представлена схема, кратко описывающая процесс отображения данных в списке элемента RecyclerView.

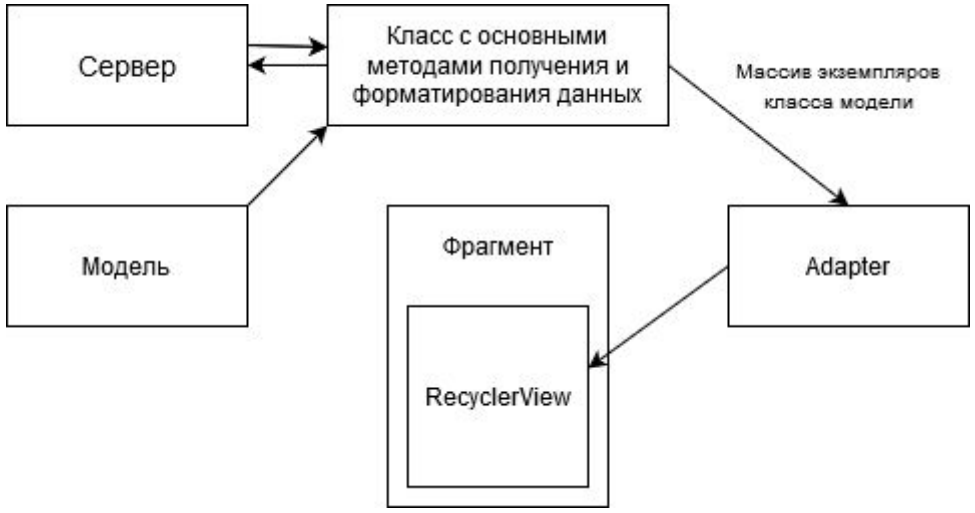

Рисунок 2.8 — Процесс отображения данных в RecyclerView

2.6. Тестирование

Тестирования является завершающим этапом процесса разработки программного обеспечения. Для того, чтобы убедиться в правильной работоспособности приложения необходимо протестировать его на нескольких устройствах. Среда разработки Android Studio позволяет использовать встроенный эмулятор с настраиваемыми конфигурациями. Так можно выбрать эмулятор уже существующего смартфона и загрузить на него любую из доступных версий Android ОС или создать эмулятор с собственными настройками. На рисунке 2.9. представлен снимок экрана с запущенным эмулятором.

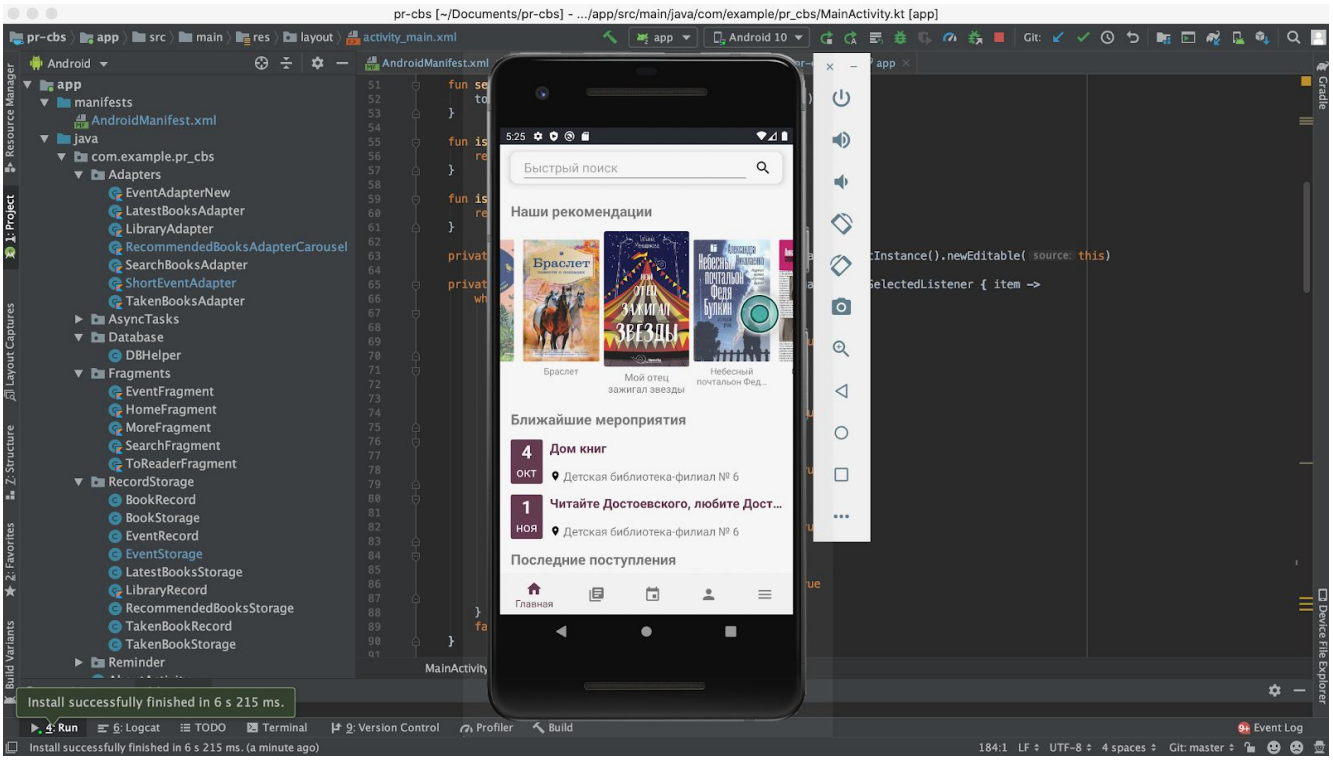

Рисунок 2.9 — Тестирование приложения на эмуляторе Android Studio

Тестирование производилось на нескольких эмуляторах с разными диагоналями и разрешениями экранов, а также версиями операционной системы Android. Данные об этих устройствах отображены таблице 1.

Таблица 1 — Виртуальные устройства для тестирования

Устройство	Pixel 2	Nexus 5X	Pixel 3a
Диагональ экрана	5.0"	5.2"	5.6"
Разрешение экрана	1920x1080	1920x1080	2220x1080
Плотность пикселей	420 dpi	420 dpi	440 dpi
Версия ОС	5.0 (API 21)	6.0 (API 23)	7.1.1 (API 25)

Тестирование на эмуляторе происходит на «чистой» версии ОС, т. е. без дополнительных оболочек и функций, добавляемых большинством производителей мобильных устройств. Из-за таких изменений может нарушиться работа приложения, поэтому было принято решение протестировать продукт дополнительно на нескольких реальных устройствах, некоторые данные о конфигурации которых приведены в таблице 2.

Таблица 2 — Реальные устройства для тестирования

Устройство	Meizu 16X	Samsung Galaxy	Huawei Mate S
Диагональ экрана	6"	6.4"	5.5"
Разрешение экрана	2160x1080	2340x1080	1920x1080
Плотность пикселей	402 ppi	403 ppi	401 ppi
Версия ОС	8.1 (API 27)	9.0 (API 28)	5.1 (API 22)

В результате тестирования были найдены и устранены следующие ошибки:

1. Отсутствие загрузки изображений из Интернета на устройствах с версией ОС Android выше 8.1

2. Некорректное отображение интерфейса на устройствах с экраном формата 18,5:9
3. Невозможность установить приложение на смартфоны марки Huawei

ВЫВОДЫ К ГЛАВЕ II

1) Создан интерактивный макет в онлайн-платформе для прототипирования Marvel, имитирующий деятельность реального приложения.
2) Осуществлена разработка мобильного приложения на основании технического требования заказчика с использованием оптимальных проанализированных инструментальных средств.
3) Проведено тестирование и внедрение программного продукта.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной выпускной квалификационной работы было разработано мобильное приложения для операционной системы Android «ЦБС Петроградского района»

В процессе выполнения работы были решены следующие задачи:

1) проанализированы возможные существующие реализации мобильных приложений для библиотек;
2) рассмотрены основные особенности взаимодействия с системой автоматизации библиотек ИРБИС64;
3) отобраны оптимальные инструментальные средства для создания Android-приложения;
4) спроектирован интерактивный макет приложения, имитирующий работу реального приложения с помощью платформы Marvel;
5) разработано мобильное приложение для ОС Android.

ЛИТЕРАТУРА

1) Какой должна быть современная библиотека? [Электронный ресурс]. - Режим доступа: https://postnauka.ru/video/30233 (Дата обращения: 22.05.2020)
2) Desktop vs Mobile vs Tablet Market Share Worldwide [Электронный ресурс]. - Режим доступа: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide (Дата обращения: 22.05.2020)
3) Интернет 2020 в России и мире: статистика и тренды [Электронный ресурс]. - Режим доступа: https://vc.ru/future/109699-internet-2020-v-rossii-i-mire-statistika-i-trendy (Дата обращения: 22.05.2020)
4) Крауз М. Г. Популярнее, чем когда-либо: восприятие библиотек в США и будущие тенденции их развития // Научные и технические библиотеки. - 2015. №7. - С. 20 или 60-80
5) Гололобова А. В. Программное обеспечение автоматизированных библиотечных систем в России [Электронный ресурс] // Культурная жизнь Юга России. - 2011. - №3. - Режим доступа: https://cyberleninka.ru/article/n/programmnoe-obespechenie-avtomatizirovannyh-bibliotechnyh-sistem-v-rossii **(дата обращения: 22.05.2020)**
6) Integrated library system [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Integrated_library_system (Дата обращения: 22.05.2020)
7) Основные характеристики системы ИРБИС64 [Электронный ресурс]. - Режим доступа:

http://elnit.org/index.php?option=com_content&view=article&id=65&Itemid=451 **(Дата обращения: 22.05.2020)**

8) Справка ИРБИС64. Язык последовательного поиск [Электронный ресурс]. - Режим доступа: http://sntnarciss.ru/irbis/spravka/irbis64.html?pril01701002000.htm (Дата обращения: 22.05.2020)
9) 10 языков для Android-разработчика [Электронный ресурс]. - Режим доступа: https://geekbrains.ru/posts/android_dev_langs (Дата обращения: 22.05.2020)
10) TIOBE Index for May 2020 [Электронный ресурс]. - Режим доступа: https://www.tiobe.com/tiobe-index/ (Дата обращения: 22.05.2020)
11) Плюсы и минусы программирования на Java [Электронный ресурс]. - Режим доступа: https://clck.ru/NbgHw (Дата обращения: 22.05.2020)
12) За что Kotlin так полюбили в Google и кому нужны две тысячи языков программирования [Электронный ресурс]. - Режим доступа: https://news.itmo.ru/ru/science/it/news/6683/ (Дата обращения: 22.05.2020)
13) Солонько Максим Константинович ЯЗЫК ПРОГРАММИРОВАНИЯ KOTLIN // Вестник науки и образования. 2020. №7-1 (85). URL: https://cyberleninka.ru/article/n/yazyk-programmirovaniya-kotlin-1 (дата обращения: 22.05.2020).
14) Google объявила Kotlin приоритетным языком программирования для разработки Android-приложений [Электронный ресурс]. - Режим доступа: https://vc.ru/dev/66728-google-obyavila-kotlin-prioritetnym-yazykom-program/ (Дата обращения: 22.05.2020)
15) Разработка Android-приложений на Kotlin. Nullability [Электронный ресурс]. - Режим доступа: https://stepik.org/lesson/63226/step/1?unit=40331 (Дата обращения: 22.05.2020)

16) C++ в Android. Часть 1 - Введение [Электронный ресурс]. - Режим доступа: https://spark.ru/startup/mobiledev/blog/33292/s-v-android-chast-1-vvedenie (Дата обращения: 22.05.2020)
17) Android (operating system) [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Android_(operating_system) (Дата обращения: 22.05.2020)
18) What is an IDE? [Электронный ресурс]. - Режим доступа: https://www.redhat.com/en/topics/middleware/what-is-ide (Дата обращения: 22.05.2020)
19) What is an IDE? [Электронный ресурс]. - Режим доступа: https://www.redhat.com/en/topics/middleware/what-is-ide (Дата обращения: 22.05.2020)
20) Сравнительный анализ Eclipse и Android Studio [Электронный ресурс]. - Режим доступа: https://files.scienceforum.ru/pdf/2019/5c180bf503208.pdf (Дата обращения: 22.05.2020)
21) IntelliJ IDEA [Электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/IntelliJ_IDEA (Дата обращения: 22.05.2020)
22) Васильева К. Н., Хусаинова Г. Я. ОБЗОР ПРОГРАММНЫХ СРЕДСТВ ДЛЯ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ // Colloquium-journal. 2020. №2 (54). URL: https://cyberleninka.ru/article/n/obzor-programmnyh-sredstv-dlya-razrabotki-mobilnyh-prilozheniy (дата обращения: 22.05.2020).
23) About the Eclipse Foundation [Электронный ресурс]. - Режим доступа: https://www.eclipse.org/org/ (Дата обращения: 22.05.2020)
24) Васильева К. Н., Хусаинова Г. Я. ОБЗОР ПРОГРАММНЫХ СРЕДСТВ ДЛЯ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ // Colloquium-journal. 2020. №2 (54). URL:

https://cyberleninka.ru/article/n/obzor-programmnyh-sredstv-dlya-razrabotki-mobilnyh-prilozheniy (дата обращения: 22.05.2020).

25) Getting Started with Eclipse IDE [Электронный ресурс]. - Режим доступа: https://kotlinlang.org/docs/tutorials/getting-started-eclipse.html (Дата обращения: 22.05.2020)

26) Знакомьтесь: Android Studio [Электронный ресурс]. - Режим доступа: http://developer.alexanderklimov.ru/android/studio/androidstudio.php (Дата обращения: 22.05.2020)

27) Meet Android Studio [Электронный ресурс]. - Режим доступа: https://developer.android.com/studio/intro?hl=fa (Дата обращения: 22.05.2020)

28) Android Studio 3.6 [Электронный ресурс]. - Режим доступа: https://android-developers.googleblog.com/2020/02/android-studio-36.html (Дата обращения: 22.05.2020)

29) The MIT License (MIT) [Электронный ресурс]. - Режим доступа: https://mit-license.org/ (Дата обращения: 22.05.2020)

30) Документация библиотеки-подпрограмм JavaIrbis в GitHub-репозитории [Электронный ресурс]. - Режим доступа: https://github.com/amironov73/JavaIrbis/blob/master/docs/chapter1.md (Дата обращения: 22.05.2020)

31) jsoup: Java HTML Parser (MIT) [Электронный ресурс]. - Режим доступа: https://jsoup.org/ (Дата обращения: 22.05.2020)

32) Glide vs. Picasso (MIT) [Электронный ресурс]. - Режим доступа: https://medium.com/@multidots/glide-vs-picasso-930eed42b81d (Дата обращения: 22.05.2020)

33) Библиотеки Петроградской стороны [Электронный ресурс]. - Режим доступа: https://pr-cbs.ru/about (Дата обращения: 22.05.2020)

34) Юношеская библиотека имени А.П. Гайдара [Электронный ресурс]. - Режим доступа: http://www.library-gaidara.ru/rus/address.htm (Дата обращения: 22.05.2020)

35) Библиотеки Петроградской стороны в социальной сети ВКонтакте [Электронный ресурс]. - Режим доступа: https://vk.com/libpetrograd (Дата обращения: 22.05.2020)

36) Minimum Viable Product (MVP) [Электронный ресурс]. - Режим доступа: https://www.productplan.com/glossary/minimum-viable-product/ (Дата обращения: 22.05.2020)

37) Прототипирование мобильного приложения: от идеи до рабочего экрана [Электронный ресурс]. - Режим доступа: https://habr.com/ru/company/mobile_dimension/blog/327452/ (Дата обращения: 22.05.2020)

38) Bottom Navigation Bar Android Tutorial [Электронный ресурс]. - Режим доступа: https://blog.iamsuleiman.com/bottom-navigation-bar-android-tutorial/ (Дата обращения: 22.05.2020)

39) Application Fundamentals [Электронный ресурс]. - Режим доступа: https://developer.android.com/guide/components/fundamentals?hl=uk (Дата обращения: 22.05.2020)

40) Лекция 1: Введение в разработку Android-приложений [Электронный ресурс]. - Режим доступа: https://www.intuit.ru/studies/courses/4462/988/lecture/14988?page=2 (Дата обращения: 22.05.2020)

41) Android Activity and its Lifecycle [Электронный ресурс]. - Режим доступа: https://blog.mindorks.com/android-activity-lifecycle (Дата обращения: 22.05.2020)

42) Dawn Griffiths, David Griffiths Head First Android Development: A Brain-Friendly Guide/Dawn Griffiths, David Griffiths. - 2017. - 928 с.

43) Android: Потоки [Электронный ресурс]. - Режим доступа: http://developer.alexanderklimov.ru/android/theory/thread.php (Дата обращения: 22.05.2020)
44) SQLite на Android [Электронный ресурс]. - Режим доступа: http://developer.alexanderklimov.ru/android/sqlite/android-sqlite.php (Дата обращения: 22.05.2020)
45) SharedPreferences [Электронный ресурс]. - Режим доступа: https://developer.android.com/reference/android/content/SharedPreferences (Дата обращения: 22.05.2020)
46) Create a List with RecyclerView [Электронный ресурс]. - Режим доступа: https://developer.android.com/guide/topics/ui/layout/recyclerview (Дата обращения: 22.05.2020)

ПРИЛОЖЕНИЕ 1