Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
подпись инициалы, фамилия
«___» _____2019г.

БАКАЛАВРСКАЯ РАБОТА

Проектирование и разработка мобильного приложения отслеживания космических объектов

09.03.04 Программная инженерия

Руководитель	_____ подпись, дата	старший преподаватель должность, ученая степень	А.С. Михалёв инициалы, фамилия
Выпускник	_____ подпись, дата		Е.В. Костенко инициалы, фамилия
Консультант	_____ подпись, дата	доцент, к.т.н должность, ученая степень	А.С. Кузнецов инициалы, фамилия
Нормоконтролер	_____ подпись, дата		О. А. Антамошкин инициалы, фамилия

Красноярск 2019

Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
подпись инициалы, фамилия
«___» _____2019г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы

Студент Костенко Егор Вячеславович

фамилия, имя, отчество

Группа КИ15-17Б Направление (специальность) 09.03.04

номер код

Программная инженерия

наименование

Тема выпускной квалификационной работы: Проектирование и разработка мобильного приложения отслеживания космических объектов.

Перечень разделов ВКР: введение, анализ предметной области, выбор средств разработки, проектирование программного обеспечения, описание приложения.

Перечень графического материала: Презентационные слайды PowerPoint.

Руководитель ВКР А. С. Михалев, старший преподаватель, ИКИТ СФУ

инициалы, фамилия, должность, ученое звание и место работы

подпись инициалы и фамилия

Задание принял к исполнению

подпись, фамилия и инициалы студента

РЕФЕРАТ

Выпускная квалификационная работа по теме «Проектирование и разработка мобильного приложения отслеживания космических объектов» содержит 48 страниц текстового документа, 23 иллюстрации, 15 использованных источников.

КЛЮЧЕВЫЕ СЛОВА: КОСМИЧЕСКИЕ ОБЪЕКТЫ, ANDROID, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ «COSMOTRACKER».

Цель выпускной квалификационной работы: проектирование и разработка мобильного приложения в виде библиотеки космических объектов с возможностью их отслеживания.

Для достижения поставленной цели необходимо реализовать следующие задачи:

- исследовать наиболее значимую литературу в выбранной области разработки;
- проанализировать предметную область;
- проанализировать рынок мобильных приложений, связанных с выбранной тематикой;
- спроектировать и разработать мобильное приложение.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Ни для кого не секрет, что космос – невероятно красив и огромен. Отчасти из-за этого человечество всегда тянулось к звездам, ведь любопытство всегда было его отличительной чертой. Однако, у некоторых людей как раз таки из-за такого масштаба и угасает интерес, им становится важнее то, что происходит на самой планете земле. Но их позиция может быть ошибочна. Ведь именно интерес к космосу позволил получить человеку столько преимуществ – благодаря космонавтике существуют такие понятия, как навигационные системы, прогноз погоды и телевидение. Да и как еще человеку можно было понять, как прекрасна планета с высоты нескольких сотен километров? Да, не стоит забывать, что людям нравится красота. Помимо безумно красивой планеты Земля, существуют метеоритные дожди, солнечные затмения, пролетающие кометы и многие другие красивые явления.

Поскольку космос – область крайне сложная и обширная, появилась потребность в создании такого приложения, в котором абсолютно любой человек мог бы с легкостью получить доступ такой интересной части мироздания. Ну и, конечно, не стоит забывать, что современный человек часто испытывает трудности в планировании, поэтому необходимо включить в функционал приложения не только возможность просмотреть всю подробную информацию о космических объектах, но и, по желанию, получать оповещения. Благодаря такой функции пользователь может отметить интересующее его событие, например, солнечное затмение, и в день его свершения получить оповещение.

Целью данной работы является проектирование и разработка мобильного приложения в виде библиотеки космических объектов с возможностью их отслеживания. Задачи, возникающие в ходе реализации поставленной цели, следующие:

- исследовать наиболее значимую литературу в выбранной области разработки;
- проанализировать предметную область;

- проанализировать рынок мобильных приложений, связанных с выбранной тематикой;
- спроектировать и разработать мобильное приложение.

1 Инициация проекта

1.1 Анализ предметной области

В XXI веке с приходом в повседневную жизнь современных технологий обывателю стало намного проще приобщиться к астрономии, даже не обладая какими-то особыми навыками и не получая даже минимального образования в этом разделе науки. И от этого можно получить реальное удовольствие, стоит одна проблема – проблема популяризации науки. Нужно дать людям простой путь, который потребует, как можно меньше времени и усилий, чтобы начать интересоваться космосом. Ведь, как известно, самое сложное – это начать.

Астрономия изучает Солнце и другие звёзды, планеты Солнечной системы и их спутники, экзопланеты, астероиды, кометы, метеороиды, межпланетное вещество, межзвёздное вещество, пульсары, чёрные дыры, туманности, галактики и их скопления, квазары и многое другое [1]. Большое количество небесных тел, комет, астероидов пролетают по траектории, достаточно близкой к Земле, чтобы их увидеть, а некоторые из них можно разглядеть и без специальной сложной оптической техники. Иногда хватит мощности любительского телескопа с двухсоткратным увеличением, иногда бинокля, а иногда можно наблюдать за явлениями невооружённым глазом.

В современном ритме жизни сложно следить за всеми космическими явлениями, хоть они происходят не так уж и часто. Необходимо узнать, когда пролетит комета, либо другое космическое тело, можно ли будет её увидеть в своём регионе, из своей области проживания. К тому же детям было бы интересно посмотреть на космос не в планетарии на картинках, а самим выехать на природу, посмотреть в телескоп, или, в некоторых случаях, просто посмотреть на затмение солнца с балкона.

Поскольку космос – область крайне сложная и обширная, появилась потребность в создании такого приложения, в котором абсолютно любой человек мог бы с легкостью получить доступ такой интересной части мироздания. Что

касается платформы, то было принято решение разрабатывать приложение на операционной системе Android. Это одно из самых перспективных направлений в индустрии, так как с каждым годом возрастает количество купленных смартфонов, и, соответственно, пользователей. Так, по статистике [2], доля занимаемого рынка устройств на данной операционной системе составляет около 75%. Так же, немаловажным фактором является относительно низкая стоимость устройств, что может означать, что у более молодой аудитории мобильные устройства именно на платформе Android.

Так как в космосе существует бесчисленное количество объектов и событий, создать такое приложение, которое вмещало бы все это в себя – невозможно. Поэтому было принято решение брать информацию из разных источников. Основная часть информации берется с сайта Смитсоновской астрофизической обсерватории, под эгидой Отдела F Международного астрономического союза (MAC). Все операционные средства MPC поступают из гранта программы НАСА по наблюдению за околоземными объектами [3].

Теперь, с появлением мобильного приложения, всё, что требуется – это скачать приложение с Google Play, потратить несколько минут на то, чтобы разобраться в интуитивно понятном эргономичном интерфейсе, посмотреть, какие явления интересны, и за которыми можно будет понаблюдать в своём его регионе. Также это приложение напомнит, чтобы пользователь не пропустил интересующее событие.

1.2 Обзор существующих аналогов

Существует несколько приложений близких по данной тематике. Ниже предоставлен обзор схожих приложений, кратко обозначен их функционал, а также сравнение с разрабатываемым приложением.

Solar Walk Lite [4] – Атлас космоса: Планеты и Спутники. Данное приложение реализует 3D модель Солнечной системы, а также нашей Галактики. Позволяет в реальном времени отслеживать движение искусственных

спутников. Также содержит атлас по множеству различным объектам. При наличии 3D-очков позволяет наблюдать объёмную картинку на смартфоне. Solar Walk Lite больше напоминает 3D атлас, нежели чем позволяет отследить конкретные небесные тела.

Sky Map [5]. Приложение позволяет определять звезды, планеты, туманности при наведении камеры смартфона на небо. Sky Map работает как приложение дополненной реальности в реальном времени, не подразумевая оповещение о событиях, которые будут спустя определенное время.

SkyView [6]. Данное приложение также реализует дополненную реальность. Позволяет отслеживать в реальном времени движение искусственных спутников. При наведении на созвездие соединяет звезды, входящие в него, а также накладывает на него образ. Приложение не позволяет отслеживать объекты в долгосрочной перспективе и получать уведомления.

1.3 Сравнение аналогов

Для наглядного сравнения аналогов была составлена таблица 1. Таблица включает такие характеристики как отсчет времени до перигелия, оповещение о событиях, краткая информация и возможность фильтрации объектов.

Таблица 1 – Сравнение аналогов и разрабатываемого приложения

	Отсчет времени до перигелия	Оповещение о событиях	Краткая информация	Возможность фильтрации объектов
Разрабатываемое приложение	+	+	+	+
SolarWalkLite	-	-	+	+
SkyMap	-	-	+	-
SkyView	-	-	-	-

Таким образом разрабатываемое приложение имеет свои функциональные особенности по сравнению с приложениями, уже представленными на рынке.

Вывод по первой главе

В результате всего вышесказанного можно сделать вывод о том, что в настоящее время существует большая потребность в приложении для наблюдения за космическими объектами с низким порогом вхождения. Ведь именно сложность отпугивает многих людей от такой интересной и важной области.

Из расчёта анализа и прогнозирования был сделан вывод о том, кто будет пользоваться приложением. Особенное внимание стоит уделить на то, чтобы любой ребенок смог воспользоваться приложением. Это именно та аудитория, которая, в приоритете, будут задействовать приложение.

Также, после сравнения с другими конкурирующими программными средствами был сделан вывод, какой уникальный функционал должно иметь разрабатываемое приложение. Плюсом ко всему, удалось выяснить ключевое преимущество – не во всех программных системах приведены решения для простого пользователя, присутствующие в одной программной системе.

2 Обоснование выбора методов и технических средств разработки

2.1 Выбор модели жизненного цикла и методы разработки

Во время разработки приложения использовалась каскадная модель жизненного цикла программного обеспечения и технология разработки Kanban.

Каскадная модель является одной из самых старых моделей жизненного цикла. Суть каскадной модели заключается в том, что каждая стадия должна полностью завершиться до начала следующей (Рисунок 1).

Рисунок 1 – Схема каскадной модели

Каскадная модель даст отличные результаты только в проектах, в которых четко определены требования и методы их реализации. Невозможно сделать шаг назад, тестирование начинается только после того, как разработка завершится или будет близка к завершению. Продукты, разработанные без разумного обоснования выбора этой модели, будут иметь недостатки (список требований не может быть изменен в любое время), которые станут известны благодаря строгой последовательности действий. Изменения дорого обойдутся, потому что

существует необходимость ожидания завершения всего проекта. Однако имеет смысл использовать каскадную модель для легкого управления проектом. Из-за своей жесткости разработка происходит быстро, а затраты и время заранее определены.

Эта модель была выбрана потому, что требования к приложению полностью понятны на этапе проектирования. Поскольку усложнения модели не является необходимыми, была взята самая простая и проверенная временем технология разработки.

Что касается методов разработки, то в качестве основного была выбрана методика Kanban, являющийся продуктом философии гибкой разработки. Суть данной методики заключается в визуализации хода разработки. Для этого используются доски и задачи.

Во-первых, необходимо проанализировать рабочий процесс и разделить доску на столбцы, отражающие этапы создания продукта (Рисунок 2). Нередко он делится на следующие этапы: «Это необходимо сделать», «В процессе», «Готово» и другие.

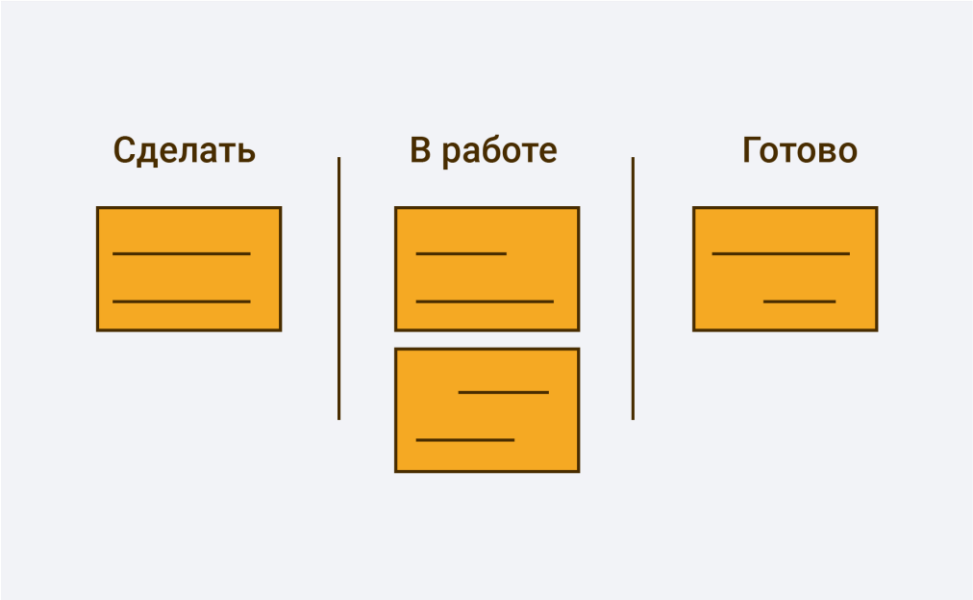

Рисунок 2 – Пример Kanban-доски

Создав и разделив доску, можно приступить к формулированию и выполнению задач, которые необходимо перемещать по доске в соответствии с их состоянием. Необходимо написать название задания на карточке или наклейке и прикрепляют его на доску.

Используя этот подход, можно сделать рабочий процесс открытым и легким для понимания, независимо от состава команды. Это очень важно для любого программного инструмента.

2.2 Выбор средств разработки

2.2.1 Выбор среды разработки и языка программирования

Так как основной платформой приложения была выбрана операционная система Android, то выбор Android Studio в качестве среды разработки был очевиден, так как данная среда является официальным средством разработки для данной платформы.

Android Studio имеет возможность записывать тесты пользовательского интерфейса во время работы приложения. Эти тесты можно редактировать или повторно запускать (в Firebase Test Lab или локально). Эта среда также включает анализатор кода, который выполняет подробные проверки написанной программы. Разработчики могут также проверить разработанную программу на предмет уменьшения размера файла.

Для тестирования написанного приложения Android Studio позволяет эмулировать устройство на базе ОС Android. Это очень удобно, потому что появляется возможность увидеть, как программа выглядит на разных устройствах. Стоит отметить, что эмулированное устройство достаточно быстрое и имеет хорошо проработанный интерфейс с соответствующими набором сервисов, камерами и GPS (Рисунок 3).

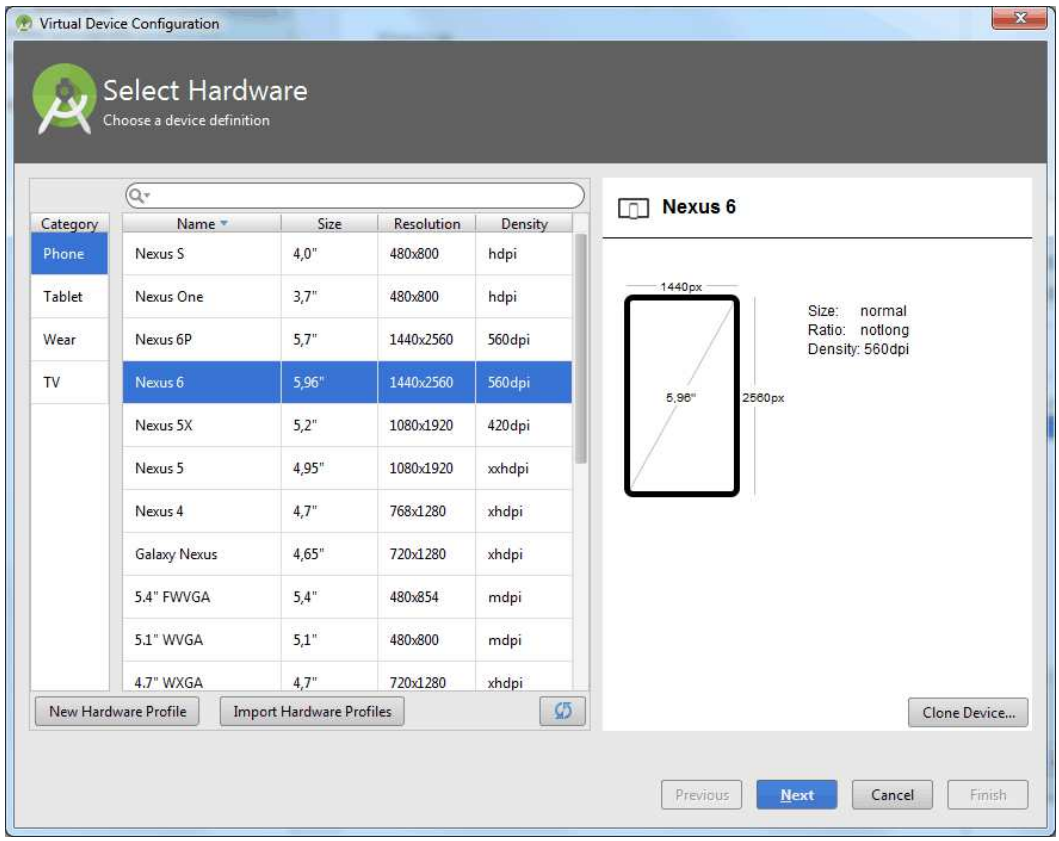

Рисунок 3 – Создание эмулятора

Преимущества Android Studio:

- отличный дизайнер пользовательского интерфейса, облегчающий визуальное проектирование приложений;
- удобный редактор XML файлов;
- поддержка системы контроля версий;
- эмуляция устройств;
- богатая база примеров проектирования;
- возможность проводить тестирование и анализ кода;
- скорость сборки приложения.

Недостатки Android Studio:

- англоязычный интерфейс;
- для разработки требуются навыки в программировании.

Так как количество преимуществ преобладает над недостатками было решено использовать именно эту среду для разработки приложения.

В данной среде разработки также есть возможность реализовывать задуманное на нескольких языках программирования, таких как Java, Kotlin и C++. Но, мало того, что Java имеет очень дружественный синтаксис, кроссплатформенность и очень большою группу почитателей, вся документация и большая часть библиотек написаны именно на этом языке. Поэтому, в данной работе большая часть кода была написана именно на языке Java.

2.2.2 Выбор базы данных

База данных необходима для хранения больших объемов данных, которые не будут потеряны после закрытия приложения в течение длительного периода времени. Кроме того, базы данных на основе SQL позволяют проводить множество манипуляций, например, таких как поиск ближайшего солнечного затмения, поиск только комет или только планет.

SQLite зарекомендовал себя как высоконадежная система баз данных, используемая во многих потребительских продуктах и программах, включая некоторые MP3-плееры, iPhone, iPod Touch, Mozilla Firefox и другие [7].

SQLite позволяет создать независимые реляционные базы данных для мобильного приложения. По умолчанию все базы данных закрыты, и только приложение, создавшее их, может получить к ним доступ. Также SQLite не должно каким-либо образом подключаться или запрашивать разрешение у пользователя. На Android он присутствует в виде библиотеки.

2.2.3 Выбор системы контроля версий

С помощью контроля версий можно вернуться к более старой версии проекта, выполнить сравнения, провести анализ, объединить изменения и многое другое. Без контроля версий выполнять более или менее полноценную разработку становится очень трудной задачей, поэтому важно правильно

выбрать набор консольных утилит для отслеживания и записи изменений в файлах (в основном это касается исходного кода программы).

Основное различие между Git и другими системами контроля версий (такими как Subversion и ей подобных) заключается в том, как Git смотрит на данные. В принципе, большинство других систем сохраняют информацию в виде списка изменений (исправлений) для файлов. Эти системы (CVS, Subversion, Perforce, Bazaar и другие) относятся к хранимым данным как к набору файлов и изменений, сделанных для каждого из этих файлов во времени. Git не сохраняет данные в этом формате. Вместо этого Git обрабатывает сохраненные данные как набор слепков небольшой файловой системы. Каждый раз, когда текущая версия проекта фиксируется, Git сохраняет слепок того, как все файлы в проекте выглядят в данный момент. Для эффективности, если файл не был изменен, Git создает ссылку на ранее сохраненный файл вместо сохранения файла снова.

Это важное различие между Git и почти всеми другими системами контроля версий. Git больше похож на небольшую файловую систему с невероятно мощными инструментами, работающими поверх неё, чем на просто система контроля версий. Благодаря этой возможности пространство, занимаемое проектом в удаленном хранилище, уменьшается в несколько раз и работа с ним, наоборот, происходит быстрее.

GitHub был выбран в качестве удаленного репозитория. GitHub – крупнейший веб-сервис для размещения ИТ-проектов и их совместной разработки [8]. Помимо личных предпочтений и технических характеристик, есть одна важная причина: поскольку очень много разработчиков используют GitHub, это отличная платформа для работы в сети. С развитием Git другие системы контроля версий стали менее популярными. Разработчики GitHub приложили немало усилий, чтобы удовлетворить все потребности программистов. Помимо открытого исходного кода, многие разработчики также размещают частные репозитории на GitHub из-за удобства платформы.

2.2.4 Выбор программного обеспечения для организации разработки приложения

Программное обеспечение для управления проектами или системы задач используются для продуктивной и понятной работы над проектом. Такие программы помогают управлять проектом и связывают менеджеров, дизайнеров, программистов и тестировщиков, помогающая им взаимодействовать друг с другом и отслеживать вклад и действия каждого [9].

Эта система бесплатна, имеет приятный интерфейс и интуитивно понятна. Trello позволяет распределять задачи максимально удобно и гибко. Интерфейс программы выполнен в виде досок, содержащие список карточек (Рисунок 4). Каждая карточка является отдельной задачей для выполнения. Поэтому очень удобно визуально понять необходимый перечень работ, который необходимо сделать.

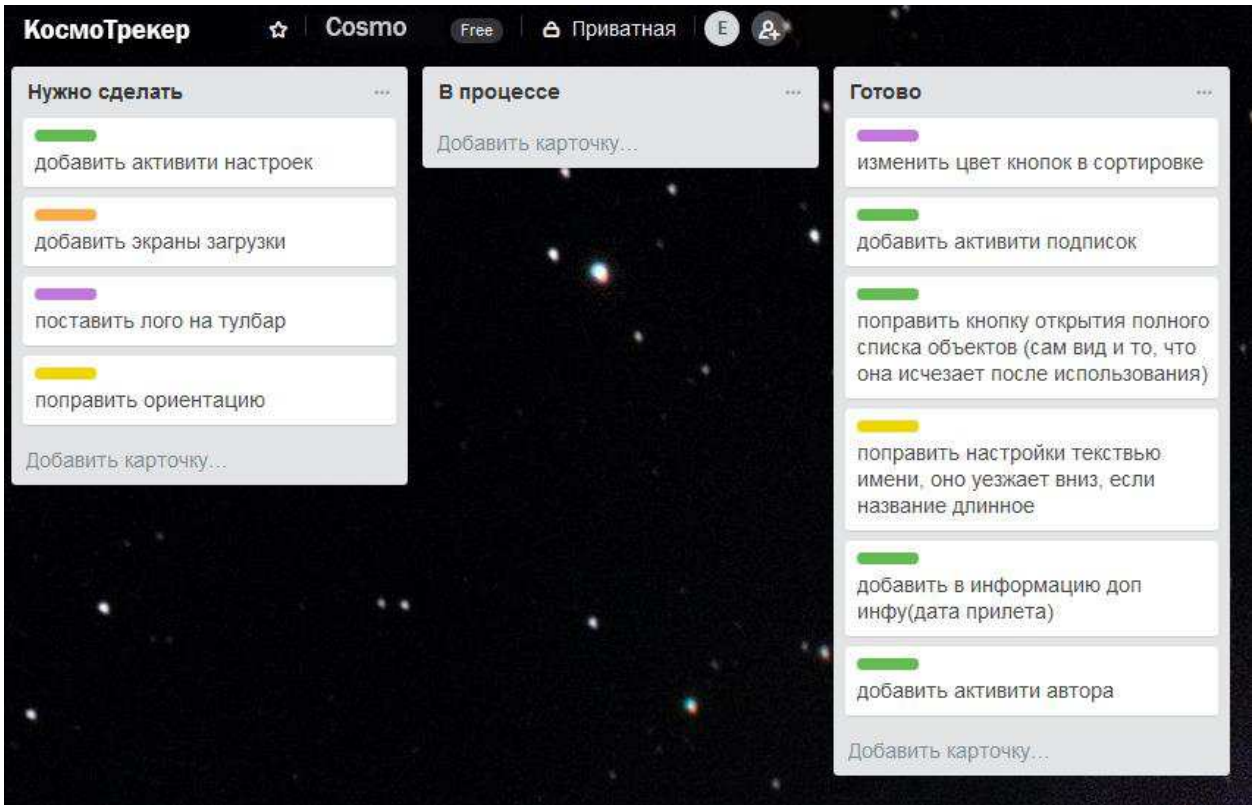

Рисунок 4 – Доска Trello.

Кроме того, каждая карта имеет цветную метку (Рисунок 5), чтобы помочь определить, к какому отделу разработки (разработчикам, тестировщикам или дизайнерам) относится эта задача. Trello позволяет настраивать цвета, добавляя собственные метки.

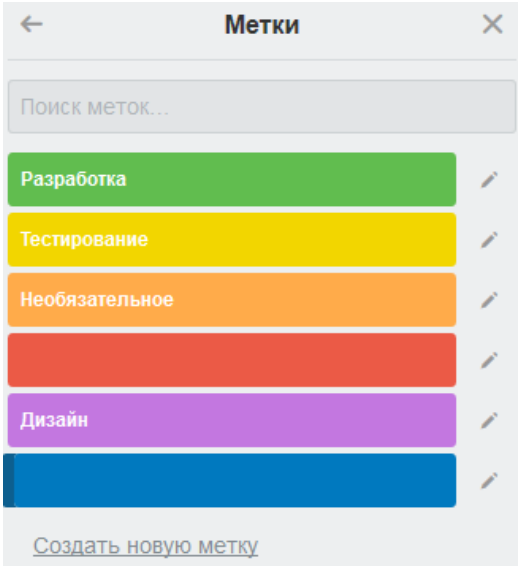

Рисунок 5 – Метки в Trello.

Выводы по второй главе

В связи с тем, что требования проекта достаточно просты и не требуют доработок во время разработки, была выбрана каскадная модель жизненного цикла с добавлением методики Kanban, позволяющая сделать рабочий процесс открытым и понятным. Сделано это было с помощью веб-сервиса Trello.

В качестве среды разработки и основным языком программирования были выбраны Android Studio и Java соответственно, так как Java является нативным языком программирования для платформы Android, и Android Studio позволяет с легкостью воспользоваться всеми возможностями этого языка. SQLite выступил в качестве базы данных за счет удобного использования и простого внедрения в приложение.

Системой контроля версий был выбран Git за его уникальные возможности в комбинации с хостингом репозиториев GitHub.

3 Проектирование системы

3.1 Описание логической структуры

Для описания логической структуры программы разработаны диаграммы вариантов использования, наглядно позволяющие описать работу пользователя с приложением.

Диаграмма вариантов использования работы пользователя с модулем списка космических объектов представлена на рисунке 6. Пользователь может полностью отобразить весь список космических объектов, подписаться на объект и выбрать какое-то конкретное событие для подробного просмотра. Нажав на событие, пользователь сможет перейти на удаленный источник информации о космическом объекте, подписаться на интересующее событие или объект, а также посмотреть подробную информацию, куда входит дата следующего перигелия в числовом формате и текст, информирующий о подробностях события.

Рисунок 6 – Диаграмма вариантов использования для работы с модулем списка космических объектов

Диаграмма вариантов использования работы пользователя с модулем меню представлена на рисунке 7. Пользователь может перейти к списку

подписок, а также просмотреть справочную информацию, куда входит информация по обозначениям в приложении и ссылки для связи с автором.

Рисунок 7 – Диаграмма вариантов использования для работы с меню

Диаграмма вариантов использования работы пользователя с модулем сортировки представлена на рисунке 8. Пользователь может отсортировать список космических объектов и события по видимости, типу и времени до следующего перигелия.

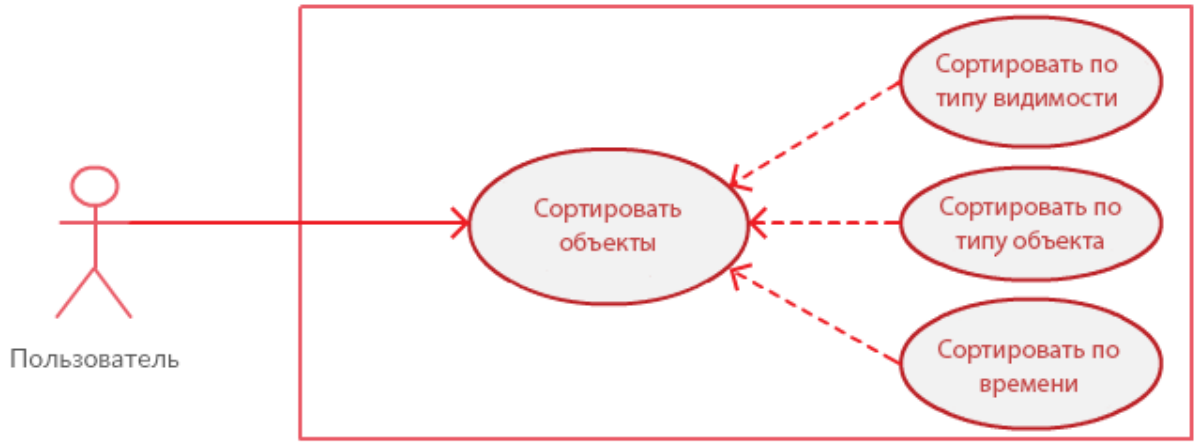

Рисунок 8 – Диаграмма вариантов использования для работы с сортировкой

3.2 Архитектура проекта

Для разработки приложения была использована архитектура на основе Model-View-ViewModel (MVVM). Это шаблон архитектуры клиентского приложения, предложенный Джоном Госсманом [10] в качестве альтернативы шаблонам MVC и MVP при использовании технологии привязки данных. Идея состоит в том, чтобы отделить логику представления данных от бизнес-логики путем вынесения её в отдельный класс для более четкого разграничения.

Диаграмма классов приложения представлена на рисунке 9.

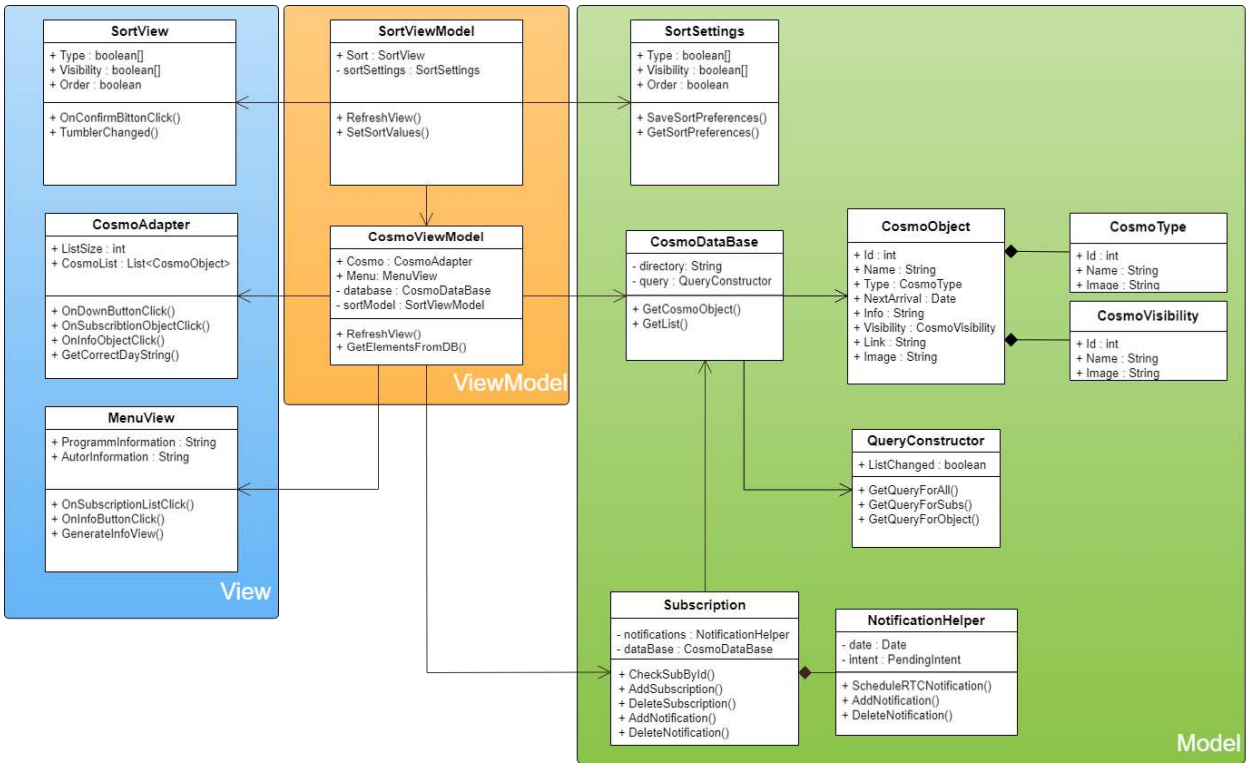

Рисунок 9 – Диаграмма классов приложения

Использование MVVM даёт следующие преимущества [11]:

– гибкость разработки. Этот подход повышает удобство работы в команде, так как пока один член команды работает над интерфейсом – другой может описывать логику получения данных и их обработки;

– тестирование. Такая архитектура упрощает процесс создания тестов. Кроме того, в большинстве случаев модульные тесты могут быть обернуты

самой ViewModel, что устраняет необходимость автоматического тестирования пользовательского интерфейса;

– разграничение логики. Мало того, что код становится более читабелен, но он также более гибок и прост в обслуживании за счет большего разграничения. Каждый модуль отвечает только за свою конкретную функцию.

3.2.1 Классы компонента View

Компоненты View обычно представляет собой интерфейс, который видит пользователь. Пользователь может взаимодействовать с ее элементами, но, когда какое-нибудь событие интерфейса будет затрагивать логику модели, компонент, реализующий шаблон проектирования «наблюдатель», будет отсылать уведомление подписчикам, а уже ViewModel будет их перехватывать.

Классы View модели представлены на рисунке 10.

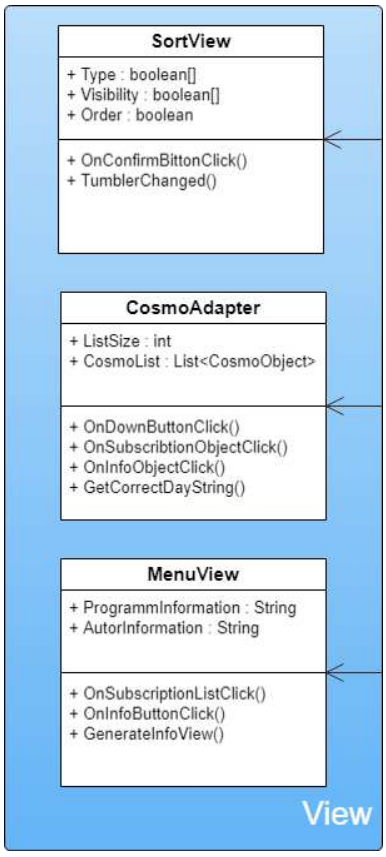

Рисунок 10 – Классы компонента View

Класс SortView отвечает за отображение настроек сортировки списка космических объектов. Пользователь может выбрать интересующий тип объектов или явлений, тип их видимости и порядок отображения списка, зависящий от даты следующего перигелия.

Класс имеет поля:

- Type (публичное свойство, тип boolean[]) – значения типа, по которым будут сортироваться космические объекты или явления(комета, планета, солнечное затмение или метеорный поток);
- Visibility (публичное свойство, тип boolean[]) – значения видимости, по которым будут сортироваться космические объекты или явления(глаз, бинокль, телескоп);
- Order (публичное свойство, тип boolean[]) – порядок, в котором будут сортироваться космические объекты или явления(по убыванию и возрастанию, в зависимости от следующего перигелия).

Также класс имеет методы:

- OnConfirmBittonClick (публичный, не принимает аргументов, возвращает void) – отвечает за обработку нажатия на кнопку подтверждения сортировки, что приводит выбор пользователя к исполнению;
- TumblerChanged(публичный, не принимает аргументов, возвращает void) – вызывается при изменении положения тумблеров сортировки для сохранения значений(необходимо, чтобы пользователь каждый раз не изменял положения тумблеров);

Класс CosmoAdapter отвечает за отображение самого космического объекта. Реализован посредством переопределения базового Android класса Baseadapter [12]. Сделано это с целью кастомизации классического списка объектов.

Класс имеет поля:

- ListSize (публичное свойство, тип int) – размер списка, который будет отображаться пользователю;

– CosmoList (публичное свойство, тип List<CosmoObject>) – список космических объектов.

Класс имеет методы:

– OnDownButtonClick(публичный, не принимает аргументов, возвращает void) – отвечает за обработку нажатия кнопки отображение следующей части списка космических объектов;

– OnSubscribtionObjectClick(публичный, не принимает аргументов, возвращает void) – вызывается при нажатии кнопки подписки на космический объект;

– OnInfoObjectClick(публичный, не принимает аргументов, возвращает void) – вызывается при нажатии на панель космического объекта, и реагирует на всю панель, за исключением кнопки подписки;

– GetCorrectDayString(публичный, принимает Date, возвращает String) – считает разницу между датой следующего перигелия и сегодняшней, и переводит ее в читабельный вид – к примеру, вместо числа 432000 (секунд) получается строка – “5 дней”.

Класс MenuView отвечает за отображение интерфейса меню, куда входит справочная информация о программе и список подписок конкретного пользователя.

– ProgrammInformation (публичное свойство, тип String) – дополнительная информация о программе (типы комет, видимости);

– AutorInformation (публичное свойство, тип String) – информация об авторе.

Класс имеет методы:

– OnSubscriptionListClick(публичный, не принимает аргументов, возвращает void) – вызывается при нажатии на кнопку вызова списка подписок;

– OnInfoButtonClick(публичный, не принимает аргументов, возвращает void) – вызывается при нажатии на кнопку справочной информации;

– GenerateInfoView(публичный, не принимает аргументов, возвращает void) – генерирует страницу со справочной информацией.

3.2.2 Классы компонента ViewModel

ViewModel [13] или модель представления связывает модели и представления с помощью механизма привязки данных. Когда значения свойств изменяются в модели, если модель реализует шаблон проектирования наблюдатель, отображаемые данные в представлении автоматически изменяются, хотя модель и представление не связаны напрямую. ViewModel также содержит логику для получения данных из модели. Затем эти данные передаются в представление.

VewModel также определяет логику обновления данных в модели. Представление взаимодействуют с ViewModel через команды, потому что элементы представления, то есть визуальные компоненты (например, кнопки), не используют события.

Классы VewModel модели представлены на рисунке 11.

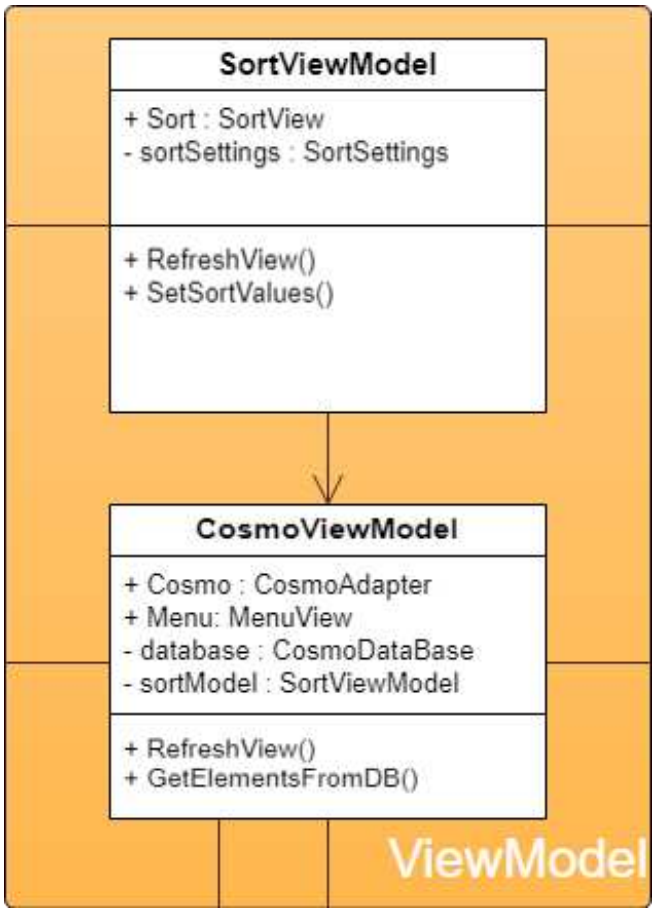

Рисунок 11 – Классы компонента View

Класс SortViewModel отвечает за обработку введенных пользователем значений в интерфейсе сортировки.

Класс имеет поля:

- Sort (публичное свойство, тип SortView) – ссылка на элемент View (в данном случае, список тумблеров сортировки);
- sortSettings (приватное, тип SortSettings) – настройки сортировки, введенные пользователем.

Класс имеет методы:

- RefreshView(публичный, не принимает аргументов, возвращает void) – обновляет элемент View;
- SetSortValues(публичный, не принимает аргументов, возвращает void) – устанавливает сохраненные значения тумблеров. Необходим для того, чтобы пользователь каждый раз не вводил настройки сортировки по новой.

Класс CosmoViewModel отвечает за обработку введенных пользователем значений в списке космических объектов. Предоставляет интерфейсу информацию из базы данных по запросу.

Класс имеет поля:

- Cosmo (публичное свойство, тип CosmoAdapter) – ссылка на элемент View (в данном случае, список космических объектов);
- Menu (публичное свойство, тип MenuView) – ссылка на элемент View (в данном случае, список меню);
- database (приватное, тип CosmoDataBase) – ссылка на базу данных;
- sortModel (приватное, тип SortViewModel) – настройки сортировки.

Класс имеет методы:

- RefreshView(публичный, не принимает аргументов, возвращает void) – обновляет элемент View;
- GetElementsFromDB(публичный, не принимает аргументов, возвращает CosmoObject[]) – достает из базы данных некоторое количество космических объектов. Размер зависит от значения поля ListSize, указанном в элементе View.

3.2.3 Классы компонента Model

Компонент Model или модели описывает данные, используемые в приложении. Модель может содержать логику, непосредственно связанную с этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных или взаимодействием с визуальными элементами управления.

Классы компонента Model представлены на рисунке 12.

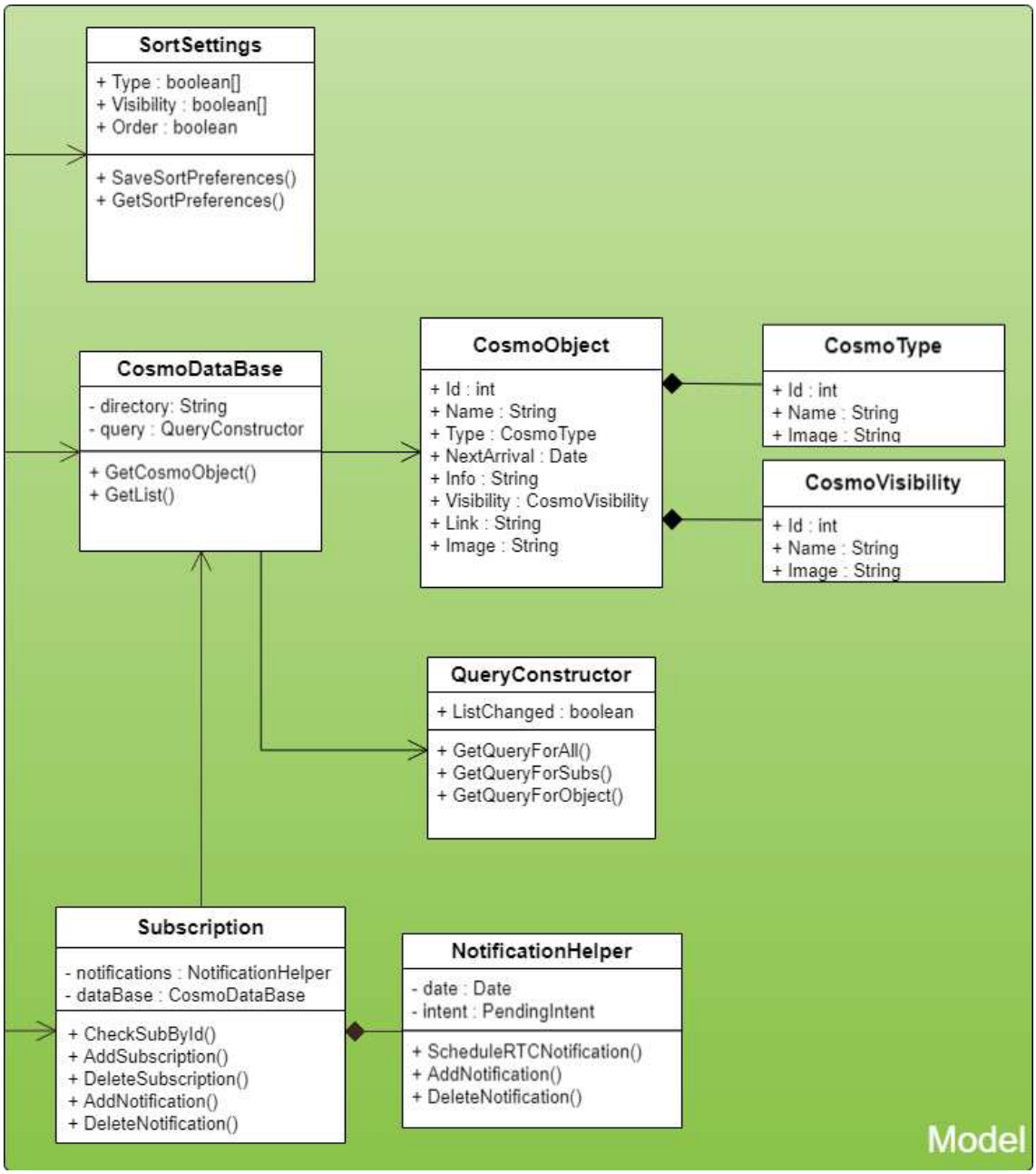

Рисунок 12 – Классы компонента Model

Класс SortSettings отвечает за удобный доступ к настройкам сортировки и сохранением их в локальные настройки приложения.

Класс имеет поля:

- Type (публичное свойство, тип boolean[]) – значения типа, по которым будут сортироваться космические объекты или явления;
- Visibility (публичное свойство, тип boolean[]) – значения видимости, по которым будут сортироваться космические объекты или явления;
- Order (публичное свойство, тип boolean[]) – порядок, в котором будут сортироваться космические объекты или явления.

Класс имеет методы:

- SaveSortPreferences(публичный, не принимает аргументов, возвращает void) – сохраняет значения настроек сортировки, введённые пользователем, в локальные настройки приложения;
- GetSortPreferences(публичный, принимает int, возвращает boolean[]) – достает сохраненные значения настроек сортировки, введенные пользователем из локальных настроек приложения.

Класс CosmoDataBase облегчает доступ к базе данных и позволяет по запросу получать любые данные. Также является фабрикой класса CosmoObject. Связан с классом QueryConstructor отношением типа ассоциация.

Класс имеет поля:

- directory (приватное, тип string) – путь к базе данных;
- query (приватное, тип QueryConstructor) – ссылка на объект класса QueryConstructor, представляющий из себя сформированный запрос к базе данных.

Класс имеет методы:

- GetCosmoObject(публичный, принимает int, возвращает CosmoObject) – возвращает космический объект из базы данных;
- GetList(публичный, принимает int, возвращает List<CosmoObject>) – возвращает список космических объектов из базы данных.

Класс QueryConstructor отвечает за создание самих запросов к базе данных.

Класс имеет поля:

- ListChanged (публичное свойство, тип boolean) – поле, не дающее лишний вызвать метод GetList класса CosmoDataBase, так как если значения настроек сортировки не были изменены, то менять список космических объектов не нужно.

Класс имеет методы:

- GetQueryForAll(публичный, принимает SortSettings, возвращает String) – возвращает сформированный запрос для базы данных, суть которого заключается в вызове всех космических объектов, опираясь на настройки сортировки;
- GetQueryForSubs(публичный, принимает SortSettings, возвращает String) – возвращает сформированный запрос для базы данных, суть которого заключается в вызове только добавленных в подписки космических объектов;
- GetQueryForObject(публичный, принимает SortSettings, возвращает String) – возвращает сформированный запрос для базы данных, суть которого заключается в вызове только одного космического объекта.

Класс CosmoObject. Представляет из себя типичный POJO объект. Содержит поля для отображения данных о космических объектах. Практически полностью дублирует одноименную таблицу CosmoObject базы данных.

Класс имеет поля:

- Id (тип int) – идентификационный номер космического объекта;
- Name (тип string) – имя объекта;
- Type (тип CosmoType) – тип объекта (комета, планета и т.д.);
- NextArrival (тип Date) – дата следующего перигелия;
- Info (тип string) – информация об объекте;
- Visibility (тип CosmoVisibility) – тип наблюдения (телескоп, невооруженным глазом и т.д.);
- Link (тип string) – ссылка на внешний источник информации;
- Image (тип string) – адрес изображения космического объекта.

Класс CosmoType представляет из себя типичный POJO объект. Содержит поля для отображения данных о типе космических объектов. Практически полностью дублирует одноименную таблицу CosmoType базы данных. Класс связан с классом CosmoObject отношением типа композиция.

Класс имеет поля:

- Id (тип int) – идентификационный номер типа космического объекта;
- Name (тип integer) – название типа (планета, комета, солнечное затмение или метеорный поток);
- Image (тип String) – директория изображения типа космического объекта.

Класс CosmoVisibility представляет из себя типичный POJO объект. Содержит поля для отображения данных о типе космических объектов. Практически полностью дублирует одноименную таблицу CosmoVisibility базы данных. Класс связан с классом CosmoObject отношением типа композиция.

Класс имеет поля:

- Id (тип int) – идентификационный номер видимости космического объекта;
- Name (тип integer) – название видимости (глаз, бинокль или телескоп);
- Image (тип String) – директория изображения видимости космического объекта.

Класс Subscription отвечает за чтение и запись подписок на космические объекта, а также за включение уведомлений.

Класс имеет поля:

- notifications (приватное, тип NotificationHelper) – ссылка на класс, суть которого заключается в удобном доступе к настройке уведомлений для платформы Android;
- dataBase (приватное, тип CosmoDataBase) – доступ к классу CosmoDataBase для удобной записи подписок в базу данных.

Класс имеет методы:

- CheckSubById(публичный, принимает int, возвращает void) – проверяет космический объект на статус подписки;
- AddSubscription(публичный, принимает CosmoObject, возвращает void) – добавляет космический объект в подписки;
- DeleteSubscription(публичный, принимает CosmoObject, возвращает void) – удаляет космический объект из списка подписок;
- AddNotification(публичный, принимает CosmoObject, возвращает void) – добавляет уведомление – когда наступит время следующего перигелия, телефон уведомит пользователя об этом;
- DeleteNotification(публичный, принимает CosmoObject, возвращает void) – отключает уведомление конкретного космического объекта.

Суть класса NotificationHelper заключается в предоставлении удобного доступа к настройке уведомлений для платформы Android. Класс связан с классом Subscription отношением типа композиция.

Класс имеет поля:

- date (приватная, тип Date) – дата ближайшего перигелия космического объекта, добавленного в подписки;
- intent (приватная, тип PendingIntent) – настройки уведомления (место, где будет отображаться, как будет отображаться и так далее).

Класс имеет методы:

- ScheduleRTCNotification(публичный, принимает CosmoObject, возвращает void) – настраивает уведомление. Работает по принципу очереди, самый ранний перигелий записывает в статичное поле date;
- AddNotification(публичный, принимает CosmoObject, возвращает void) – добавляет уведомление в локальные настройки приложения;
- DeleteNotification(публичный, принимает CosmoObject, возвращает void) – удаляет уведомление из локальных настроек приложения.

3.3 Хранение данных

Для хранения данных была выбрана стандартная база данных SQLite, которая специально разрабатывалась для работы с языком Java.

Структура базы данных состоит из четырех взаимосвязанных таблиц. Главная таблица «CosmoObjects» содержит все отображаемые объекты и данные о них. Так как может быть несколько комет, метеоритных потоков и других одинаковых объектов было решено вынести тип этих объектов в отдельную таблицу «Types». Таблица «Visibility» содержит тип видимости объекта, можно ли его увидеть невооруженным глазом или только с помощью специальных устройств. Таблица «Subscriptions» содержит список космических объектов, на которых подписался пользователь.

Таблица CosmoObjects (Рисунок 13) имеет поля:

- _id (тип integer, первичный ключ) – идентификационный номер космического объекта;
- Name (тип text) – название космического объекта;
- TypeId (тип integer, внешний ключ) – тип объекта (комета, планета и т.д.);
- NextArrival (тип datetime) – дата следующего перелегия;
- Info (тип text) – информация об объекте;
- VisibilityId (тип integer, внешний ключ) – тип наблюдения (телескоп, невооруженным глазом и т.д.);
- Link (тип text) – ссылка на внешний источник информации.

CosmoObjects	
_id [PK]	int
Name	text
TypeId [FK]	int
NextArrival	datetime
Info	text
VisibilityId [FK]	int
Link	text

Рисунок 13 – Таблица CosmoObjects

Таблица Types (Рисунок 14) имеет поля:

– _id (тип integer, первичный ключ) – идентификационный номер типа космического объекта;

– Name (тип integer) – название типа (планета, комета, солнечное затмение или метеорный поток).

Type	
_id [PK]	int
Name	text

Рисунок 14 – Таблица Type

Таблица Visibility (Рисунок 15) имеет поля:

– _id (тип integer, первичный ключ) – идентификационный номер типа видимости;

– Name (тип integer) – название видимости (глаз, бинокль, телескоп).

Visibility	
_id [PK]	int
Name	text

Рисунок 15 – Таблица Visibility

Таблица Subscriptions (Рисунок 16) имеет поля:

– _id (тип integer, первичный ключ) – идентификационный номер подписки;

– CosmoId (тип integer, внешний ключ) – идентификационный номер космического объекта.

Subscriptions	
_id [PK]	int
CosmoId [FK]	int

Рисунок 16 – Таблица Subscriptions

Полная схема данных со всеми связями представлена на рисунке 17.

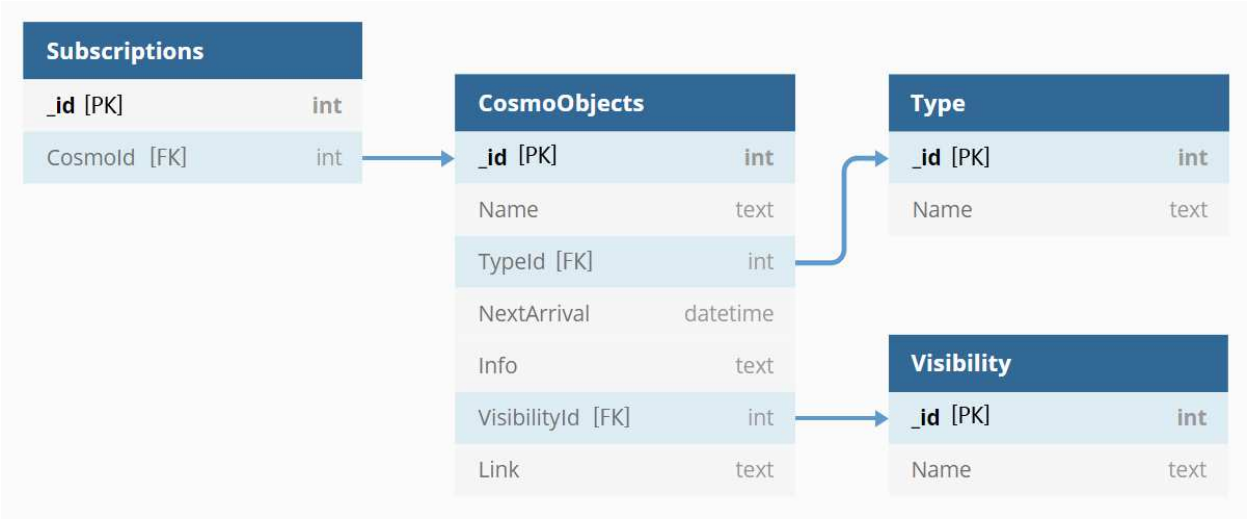

Рисунок 17 – Полная схема базы данных

На рисунке 18 представлена часть заполненной данными таблицы объектов и событий, отсортированной по времени до следующего перигелия.

_id	Name	Type	NextArrival	Info	Visibil	Link
38	Солнечное затмение 02.07.2019	2	2019-07-02	Солнечное затмение 2 июля 201...	1	https://ru.wikipedia.org/wiki/%D0%A1%D0%...
60	Орионида	3	2019-10-02	Рой метеорных тел, образующи...	1	https://ru.wikipedia.org/wiki/%D0%9E%D1%8...
29	Комета Уэста-Когоутека	1	2019-10-26	Комета Уэста — Когоутека — Ик...	3	https://ru.wikipedia.org/wiki/76P/%D0%A3%...
56	Леониды	3	2019-11-06	Леони́ды — метеорный поток с ...	1	https://ru.wikipedia.org/wiki/%D0%9B%D0%...
27	Комета Клемола	1	2019-11-09	Комета Клемола (68P/Klemola) ...	3	https://ru.wikipedia.org/wiki/68P/%D0%9A%...
63	Фениициды	3	2019-11-28	Фениициды — метеорный поток ...	1	https://ru.wikipedia.org/wiki/%D0%A4%D0%...
57	Геминиды	3	2019-12-04	Геминиды — один из самых мо...	1	https://ru.wikipedia.org/wiki/%D0%93%D0%B...
62	Урсиды	3	2019-12-17	Урсиды — метеорный поток с р...	1	https://ru.wikipedia.org/wiki/%D0%A3%D1%...
58	Квадрантиды	3	2019-12-28	Квадрантиды — метеорный пот...	1	https://ru.wikipedia.org/wiki/%D0%9A%D0%...
64	Эта-Аквариды	3	2020-04-19	Эта-Аквариды - метеорный пото...	1	https://ru.wikipedia.org/wiki/%D0%AD%D1%...
59	Ариетиды	3	2020-05-22	Ариетиды (от лат. Aries — Овен) ...	1	https://ru.wikipedia.org/wiki/%D0%90%D1%8...
2	Комета Энке	1	2020-06-26	Комета Э́нке (официальное обоз...	1	https://ru.wikipedia.org/wiki/%D0%9A%D0%...
61	Пегасиды	3	2020-07-07	Пегасиды (англ. Pegasids) — мет...	1	https://ru.wikipedia.org/wiki/%D0%9F%D0%B...
55	Персеиды	3	2020-07-17	Персе́иды — метеорный поток, ...	1	https://ru.wikipedia.org/wiki/%D0%9F%D0%B...
8	Комета Темпеля-Свифта	1	2020-11-26	11P/Темпеля — Свифта — LINEA...	3	https://ru.wikipedia.org/wiki/11P/%D0%A2%...
39	Солнечное затмение 14.12.2020	2	2020-12-14	Солнечное затмение 14 декабря...	1	https://ru.wikipedia.org/wiki/%D0%A1%D0%...
11	Комета Холмса	1	2021-02-19	Комета Холмса (официальное н...	2	https://ru.wikipedia.org/wiki/17P/%D0%A5%...
30	Комета Когоутека	1	2021-03-07	75D/Когоутека (75D/Kohoutek) —...	3	https://ru.wikipedia.org/wiki/75D/%D0%9A%...
10	Комета Понса-Виннеке	1	2021-05-27	7P/Понса-Виннеке — короткопе...	3	https://ru.wikipedia.org/wiki/7P/%D0%9F%D0...
47	Солнечное затмение 10.06.2021	2	2021-06-12	Солнечное затмение 10 июня 20...	1	https://ru.wikipedia.org/wiki/%D0%A1%D0%...
7	Комета Финлея	1	2021-07-13	Комета 15P/Финлея — комета, о...	3	https://ru.wikipedia.org/wiki/15P/%D0%A4%...
4	Комета Туттля	1	2021-08-27	9 января 1790 года Пьер Мешéн ...	3	https://ru.wikipedia.org/wiki/8P/%D0%A2%D...
3	Комета Фая	1	2021-09-08	Комета Фая (официальное назва...	1	https://ru.wikipedia.org/wiki/4P/%D0%A4%D...
5	Комета Кодзимы	1	2021-11-03	Комета Кодзимы (70P/Kojima) —...	3	https://ru.wikipedia.org/wiki/70P/Кодзимы
13	Комета Борелли	1	2022-02-01	21 сентября 2001 г. космический...	3	https://ru.wikipedia.org/wiki/19P/%D0%91%...
9	Комета Темпеля	1	2022-03-04	Во время обнаружения период ...	3	https://ru.wikipedia.org/wiki/9P/%D0%A2%D...
6	Комета Чурюмова-Герасимен...	1	2022-03-13	Комета Чурюмова — Герасимен...	3	https://ru.wikipedia.org/wiki/67P/%D0%A7%...
16	Комета Копфа	1	2022-03-18	Комета 22P/Копфа открыта 23 ав...	3	https://ru.wikipedia.org/wiki/22P/%D0%9A%...
12	Комета Хонда-Мркоса	1	2022-04-22	Комета Хонда — Мркоса — Пай...	3	https://ru.wikipedia.org/wiki/%D0%9A%D0%...
32	Комета Швассмана-Вахмана	1	2022-08-25	Комета Швассмана — Вахмана 3...	3	https://ru.wikipedia.org/wiki/73P/%D0%A8%...
22	Комета Туттля-Джакобини	1	2022-09-12	Комета Туттля — Джакобини — ...	2	https://ru.wikipedia.org/wiki/41P/%D0%A2%...
46	Солнечное затмение 25.10.2025	2	2022-10-25	Солнечное затмение 25 октября ...	1	https://ru.wikipedia.org/wiki/%D0%A1%D0%...

Рисунок 18 – Заполненная данными таблица

Таким образом была описана структура базы данных, используемой в разрабатываемом приложении.

4 Описание программы

4.1 Установка приложения

Для установки мобильного приложения необходимо устройство с ОС Android версией от 4.1, так как, по статистическим данным Google, версии Android ниже 4.1 уже практически неактуальны [14].

Загрузка и запуск программы осуществляется с помощью установки apk-файла на устройство и открытие его из файловой системы. Другой способ для установки приложения через приложение «Play market», используя поисковую строку по запросу «Космотрекер» [15].

4.2 Работа с приложением

Разработанное приложение является библиотекой космических объектов и явления, и, соответственно, при заходе в приложение, пользователь должен видеть главный экран, на котором расположен список объектов, которые должны в скором времени можно увидеть с земли.

Схематичное изображение главного экрана представлено на рисунке 19.

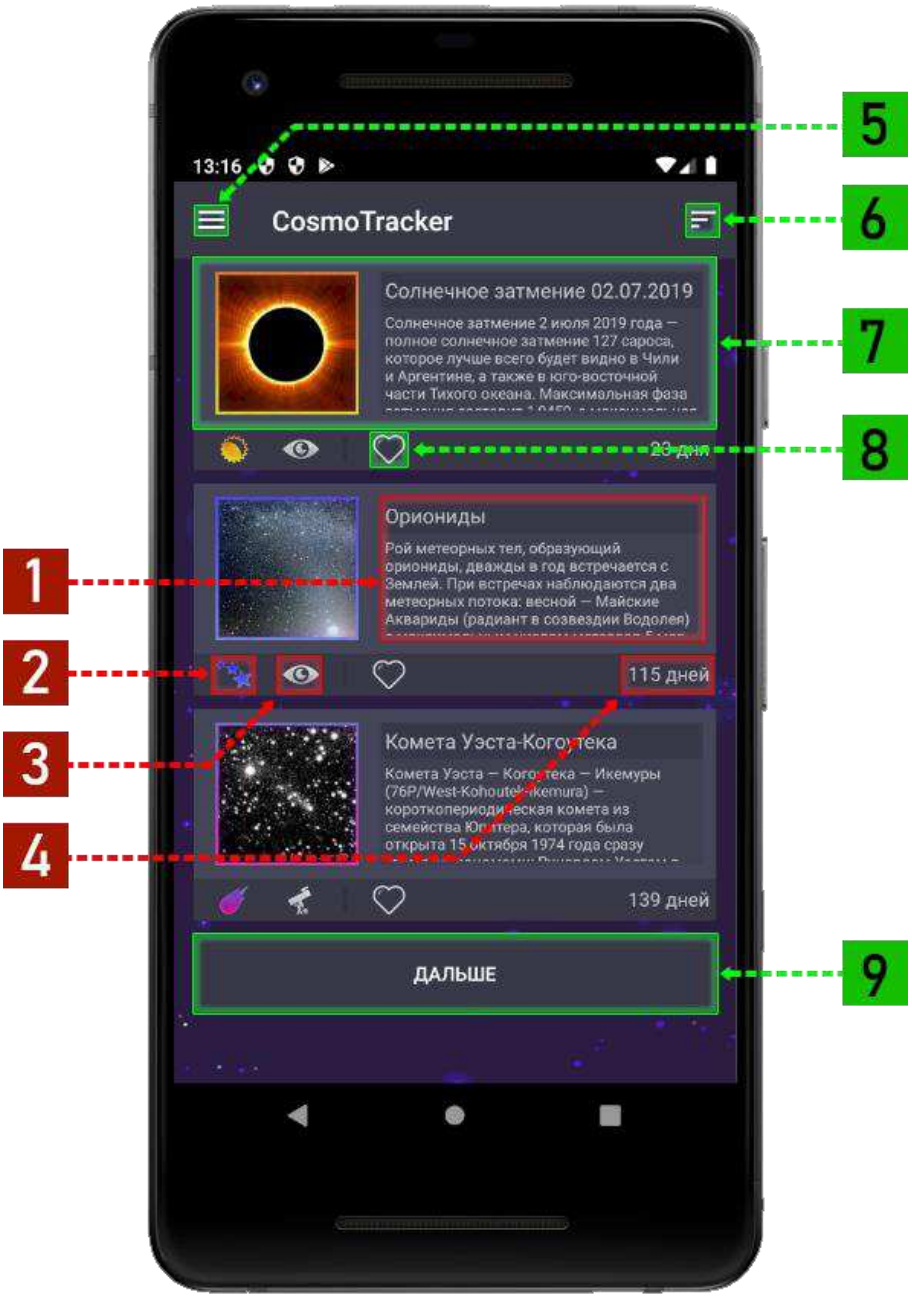

Рисунок 19 – Главный экран

Главный экран состоит из следующих компонентов:

- номер 1 – краткое описание космического объекта;
- номер 2 – иконки типов объектов: комета/планета/солнечное затмение/метеоритный поток;
- номер 3 – иконки типов видимости объектов: невооруженным глазом/через бинокль/через телескоп;
- номер 4 – таймер до следующего перигелия;

− номер 5 – кнопка меню, при нажатии на которую открывается окно с доступом ко списку подписок и справочной информации;

− номер 6 – кнопка настроек сортировки;

− номер 7 – при нажатии на космический объект открывается отдельное окно с подробным описанием, таймером, возможностью подписаться на событие;

− номер 8 – зеленое/серое сердечко – возможность подписаться/отписаться от события и, соответственно, включение и отключение уведомления;

− номер 9 – стрелка, направленная вниз, внизу центра экрана – показывает больше событий.

При нажатии на космический объект открывается экран с подробной информацией об этом космическом объекте. Схематичное изображение экрана с подробной информацией представлено на рисунке 20.

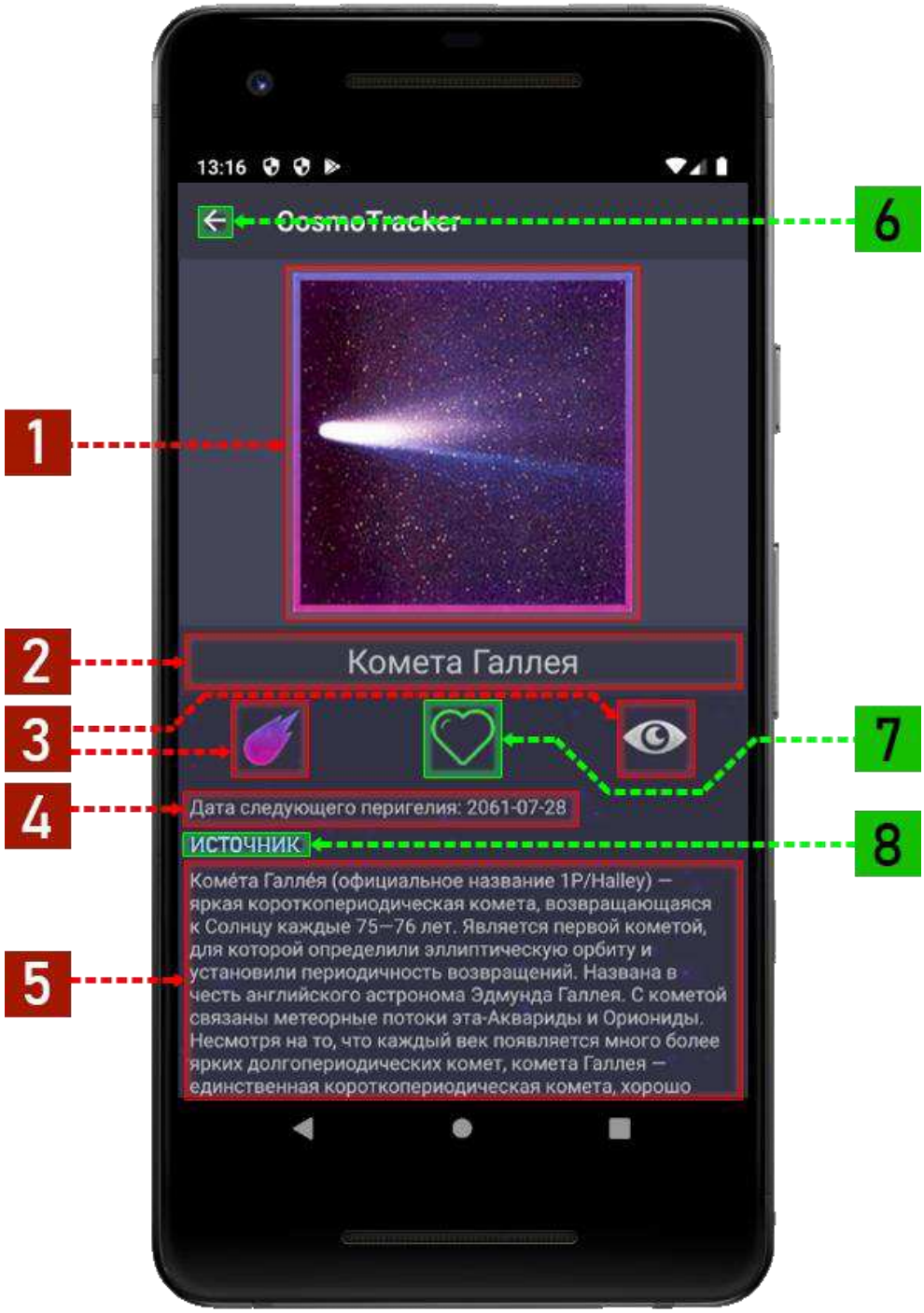

Рисунок 20 – Экран подробной информации о космическом объекте

Экран состоит из следующих компонентов:

- номер 1 – изображение космического объекта;
- номер 2 – название космического объекта;
- номер 3 – иконки типа космического объекта и типа его видимости;
- номер 4 – дата следующего перигелия;

– номер 5 – вся информация об объекте;

– номер 6 – кнопка выхода из экрана просмотра подробной информации;

– номер 7 – подписка/отписка от объекта;

– номер 8 – активная ссылка на источник информации, при нажатии на которую откроется браузер и сайт источника.

Экран меню носит в себе всего две функции – открыть список подписок и открыть экран со справочной информацией. Схематичное изображение меню представлено на рисунке 21.

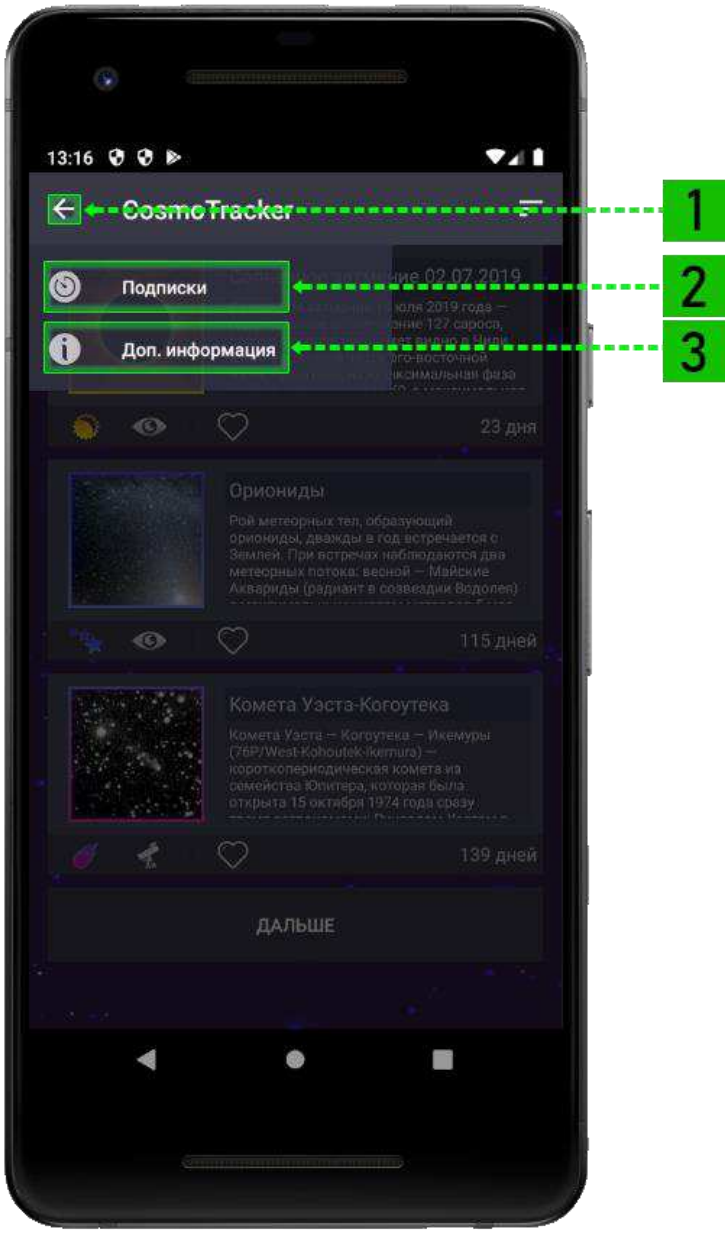

Рисунок 21 – Экран меню

Меню состоит из следующих компонентов:

- номер 1 – закрытие экрана подменю;
- номер 2 – иконка списка подписок открывает экран со всеми объектами, на которые подписан пользователь;
- номер 3 – иконка дополнительной информации открывает экран со справочной информацией.

При нажатии на кнопку дополнительной информации откроется экран, представленный на рисунке 22.

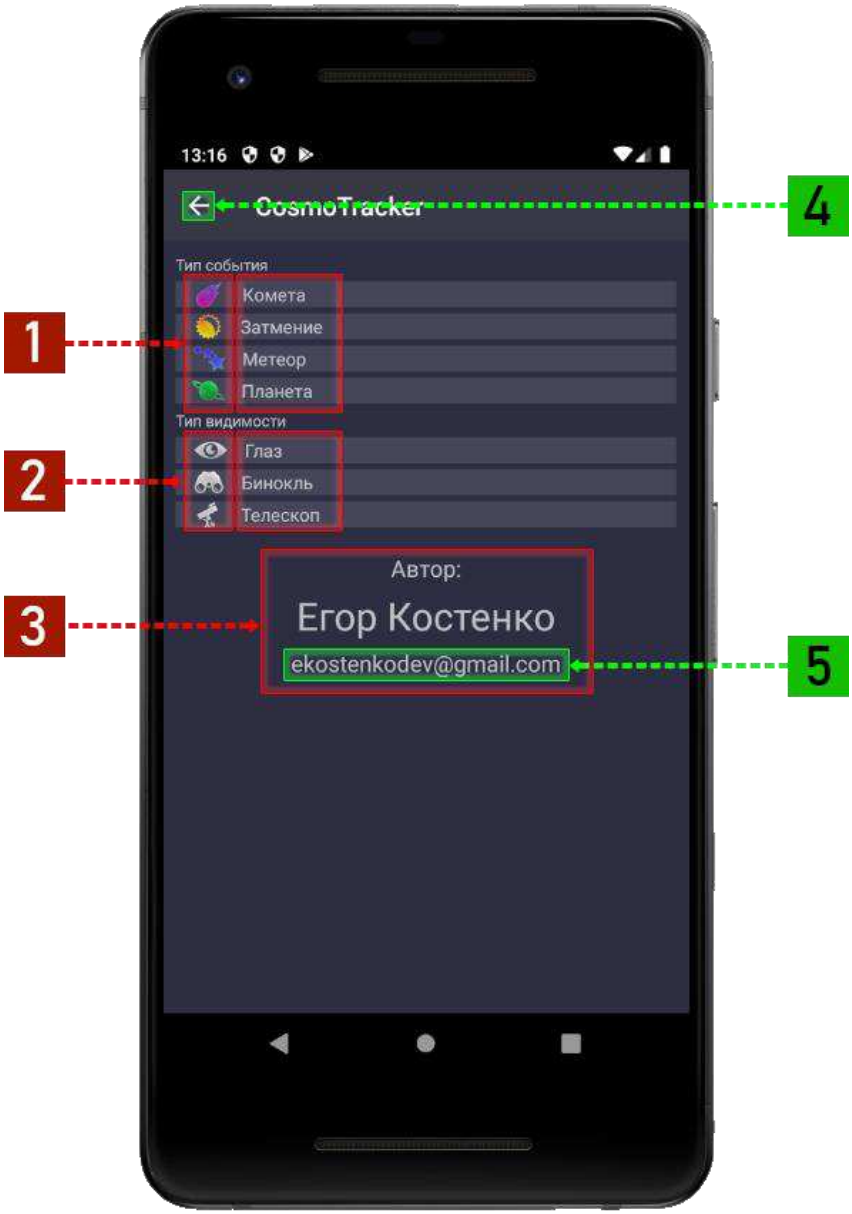

Рисунок 22 – Экран справочной информации

Экран справочной информации состоит из следующих компонентов:

− номер 1 – иконки и названия типов космических объектов (комета, затмение, метеорный поток, планета);

− номер 2 – иконки и названия типов видимости космических объектов (глаз, бинокль, телескоп);

− номер 3 – сведения об разработчике мобильного приложения;

− номер 4 – кнопка выхода из экрана справочной информации;

− номер 5 – активная кнопка для связи с разработчиком мобильного приложения.

Экран сортировки содержит список типов космических объектов, тип их видимости с земли и порядок сортировки. Для каждого типа объекта и его видимости соответствует один тумблер, нажатие которого включает или выключает его из списка космических объектов. Порядок сортировки зависит от даты до следующего перигелия. Схематичное изображение экрана сортировки представлено на рисунке 23.

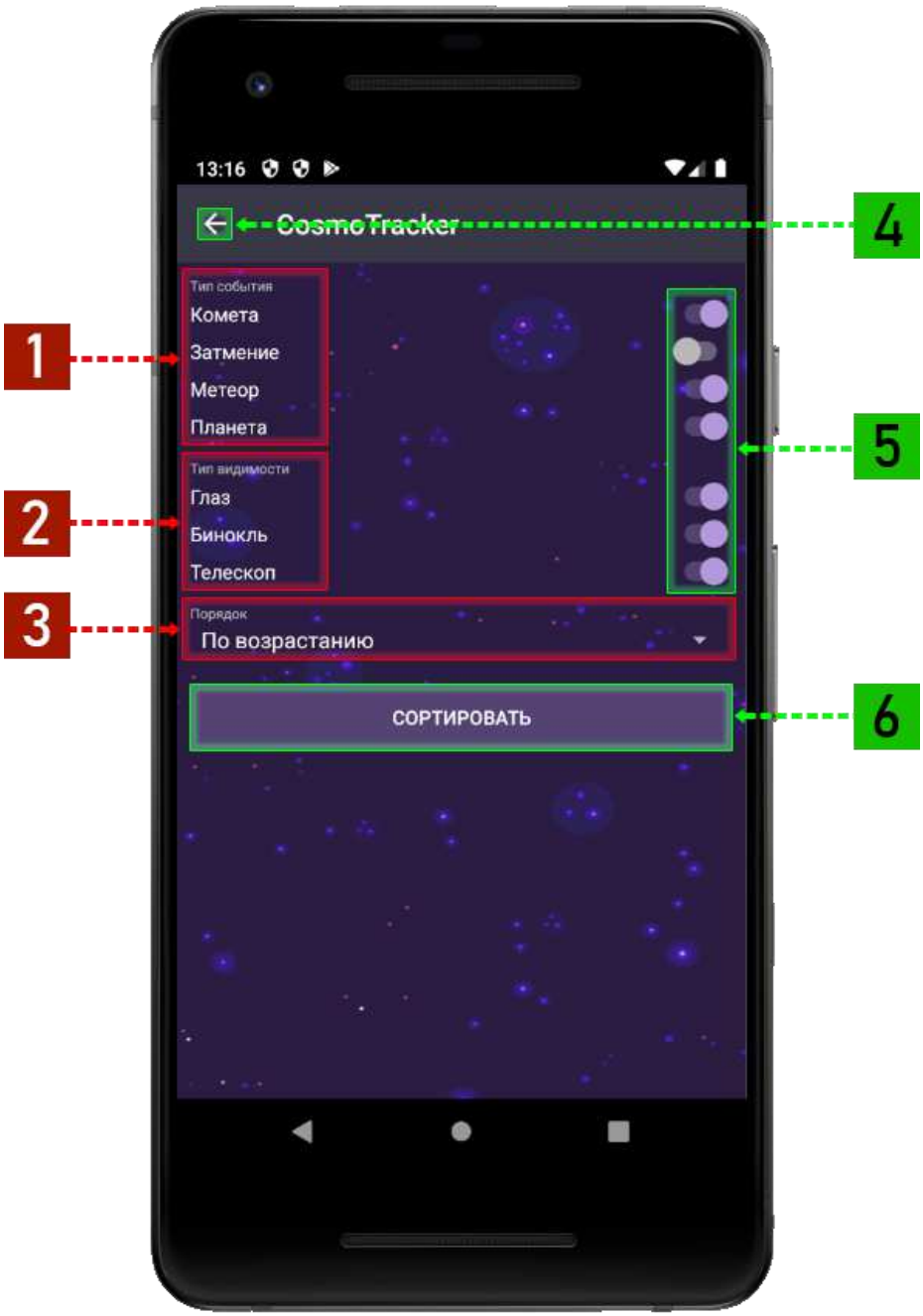

Рисунок 23 – Экран настройки сортировки объектов на главном экране

Экран сортировки состоит из следующих компонентов:

- номер 1 – список типов космических объектов;
- номер 2 – список типов видимости космических объектов;
- номер 3 – выпадающий список для сортировки по возрастанию/убыванию списка объектов на главном экране, зависящий от даты следующего перигелия;

- номер 4 – кнопка выхода из экрана сортировки;
- номер 5 – ряд тумблеров для изменения отображения списка;
- номер 6 – кнопка подтверждения.

ЗАКЛЮЧЕНИЕ

Приложение содержит библиотеку космических объектов и событий, и позволяет пользователю получать уведомление о наступлении выбранного события, на которое тот в свою очередь подписался.

В отчете был представлен и описан весь цикл разработки от инициации проекта заканчивая описанием работы с приложением. Отчет включает и описывает такие аспекты как анализ предметной области и сравнение аналогов. Описан и рассмотрен инструментарий для разработки.

В ходе разработки мобильного приложения была выявлена такая перспектива, как дальнейшее развитие с помощью разработки кроссплатформенного мобильного приложения. Одной из самых популярных ОС является iOS, поэтому данная перспектива является приемлемой для разработанного приложения. Она не была реализована в процессе разработки, так как такая разработка потребует несколько программистов и больше времени на разработку.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Значение слова «Астрономия». [Электронный ресурс]: карта слов и выражений русского языка. – Режим доступа: https://kartaslov.ru/ значение-слова/астрономия.

2. Статистика операционных систем смартфонов [Электронный ресурс]: веб-сайт, который представляет собой инструмент для анализа веб-трафика. – Режим доступа: http://gs.statcounter.com/os-market-share/mobile/worldwide.

3. Источник данных о космических объектах [Электронный ресурс]: сайт Смитсоновской астрофизической обсерватории. – Режим доступа: https://minorplanetcenter.net.

4. Solar Walk [Электронный ресурс]: 3D модель Солнечной системы. – Режим доступа: http://www.vitotechnology.com/solar-walk.html.

5. Sky Map [Электронный ресурс]: приложение дополненной реальности для определения созвездий. – Режим доступа: http://sky-map-team.github.io/stardroid/.

6. SkyView [Электронный ресурс]: приложение для отслеживания в реальном времени движение искусственных спутников и планет. – Режим доступа: https://www.terminaleleven.com/skyview/iphone/.

7. Введение в базы данных SQLite [Электронный ресурс]: базы данных в мобильных приложениях. – Режим доступа: https://docplayer.ru/74372998-8-lekciya-baza-dannyh-v-mobilnyh-prilozheniyah-8-1-vvedenie-v-bazy-dannyh-sqlite.html.

8. Определение ресурса Github [Электронный ресурс]: свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/GitHub.

9. Как организовать работу над интернет-проектом в Trello [Электронный ресурс]: коллективный блог, посвященный IT индустрии. – Режим доступа: https://habr.com/ru/company/carrotquest/blog/291964.

10. Введение в шаблон MVVM [Электронный ресурс]: официальный сайт новостей и заметок компании Microsoft. – Режим доступа:

https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps.

11. Реализация MVVM в Android [Электронный ресурс]: сайт о различных методах разработки приложений. – Режим доступа: https://stfalcon.com/ru/blog/post/android-mvvm.

12. Документация класса BaseAdapter [Электронный ресурс]: документация по разработке приложения под Android. – Режим доступа: https://developer.android.com/reference/android/widget/BaseAdapter.

13. Определение паттерна MVVM [Электронный ресурс]: сайт, посвященный различным языкам и технологиям программирования. – Режим доступа: https://metanit.com/sharp/wpf/22.1.php.

14. Статистика используемых версий операционной системы Android [Электронный ресурс]: официальный сайт разработчиков Android. – Режим доступа: https://developer.android.com/about/dashboards.

15. Приложение «Космотрекер» [Электронный ресурс]: магазин приложений на операционную систему Android. – Режим доступа: https://play.google.com/store/apps/details?id=com.ekostenkodev.cosmotracker.

Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических информационных технологий

Кафедра «Информатика»

УТВЕРЖДАЮ

Заведующий кафедрой

_____ А.С. Кузнецов

подпись инициалы, фамилия

«09» 07 2019г.

БАКАЛАВРСКАЯ РАБОТА

Проектирование и разработка мобильного приложения отслеживания космических объектов

09.03.04 Программная инженерия

Руководитель	08.07.19	старший преподаватель	А.С. Михалёв
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	08.07.19		Е.В. Костенко
	подпись, дата		инициалы, фамилия
Консультант	08.07.19	доцент, к.т.н	А.С. Кузнецов
	подпись, дата	должность, ученая степень	инициалы, фамилия
Нормоконтролер	07.07.19		О. А. Антамошкин
	подпись, дата		инициалы, фамилия

Красноярск 2019