Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

РАЗРАБОТКА Мобильного приложения ДЛЯ ПРОСМОТРА ПРОГНОЗА ПОГОДЫ

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

МДК.01.03 Разработка мобильных приложений

НАТКиГ.200100.010.000ПЗ

Разработал:

студент группы ПР–21.102

Антонов Е.А.

2024

**Содержание**

[Введение 3](#_Toc161149391)

[1 Исследовательский раздел 4](#_Toc161149392)

[1.1 Описание предметной области 4](#_Toc161149393)

[1.2 Образ клиента 5](#_Toc161149394)

[1.3 Сценарии 5](#_Toc161149395)

[1.4 Сбор и анализ прототипов 7](#_Toc161149396)

[2 Проектирование приложения 8](#_Toc161149397)

[2.1 UI/UX дизайн приложения 8](#_Toc161149398)

[2.2 Выбор технологии, языка и среды программирования 11](#_Toc161149399)

[3 Разработка мобильного приложения 12](#_Toc161149400)

[3.1 Выбор API 12](#_Toc161149401)

[3.2 Разработка мультимедийного контента 13](#_Toc161149402)

[3.3 Описание используемых плагинов 15](#_Toc161149403)

[3.4 Описание разработанных процедур и функций 16](#_Toc161149404)

[4 Тестирование 27](#_Toc161149405)

[4.1 Протокол тестирования дизайна приложения 27](#_Toc161149406)

[4.2 Протокол тестирования функционала приложения 30](#_Toc161149407)

[Заключение 33](#_Toc161149408)

[Библиография 34](#_Toc161149409)

[Приложение А (обязательное) Техническое задание 35](#_Toc161149410)

Введение

В современном мире, где быстрые и удобные технологические решения становятся неотъемлемой частью повседневной жизни, мобильные приложения играют ключевую роль в обеспечении информацией и комфортом.

Одним из самых популярных и востребованных приложений являются приложения для прогноза погоды. Независимо от того, находитесь ли вы в пути, планируете отдых или просто хотите знать, какая погода ожидается сегодня и завтра, мобильные приложения прогноза погоды обеспечивают надежный и актуальный доступ к информации о погоде в любом месте и в любое время.

Целью курсового проекта является создание мобильного приложения для просмотра прогноза погоды.

Задачами курсового проекта в связи с указанной целью являются:

* изучение предметной области;
* рассмотрение приложения с точки зрения пользователя для выявления функций приложения;
* разработка и реализация дизайна приложения;
* написание кода приложения;
* тестирование полученного продукта.

Объект исследования – мобильное приложение для просмотра прогноза погоды.

Предмет исследования – изучение принципов функционирования и инструментов приложения.

# Исследовательский раздел

## Описание предметной области

Современные сервисы измерения и предсказания погоды стали более точными и доступными благодаря использованию современных методов и технологий благодаря данным, поступающими от местных метеостанций непрерывно измеряя параметры, такие как температура, осадки, направление и скорость ветра, а также влажность.

Собранные данные обрабатываются для анализа и интерпретации метеорологических показателей, учитывая различные факторы, включая географические особенности, изменения климата. Такой подход позволяет предоставлять наиболее актуальную и точную информацию о текущих и будущих погодных условиях.

Важным аспектом является также постоянное обновление данных в режиме реального времени, что обеспечивает возможность моментального доступа к действующей информации о погоде. Такое взаимодействие с данными и использование передовых алгоритмов прогнозирования позволяют создать приложение, способное более точно предсказывать погодные условия, повышая уровень удовлетворенности пользователя и повышая уровень доверия к прогнозу погоды.

Пользователи получают информацию о погодных условиях, включая температуру, влажность, скорость ветра, а также прогноз на будущие дни. Эти данные необходимы для планирования и регулирования повседневных дел пользователями.

В современном мире применение новых информационных технологий в сфере измерения и прогнозирования погоды привело к созданию уникальных сервисов, которые оперативно и точно предоставляют всю необходимую информацию о погодных условиях. Современные методы обработки данных и использование специализированных алгоритмов позволяют не только осуществлять измерение погодных параметров с высокой точностью, но и предсказывать будущие изменения с учетом различных влияющих факторов. В сфере метеорологии взаимодействуют с нашим повседневным опытом, обеспечивая актуальной и достоверной информацией, что, в конечном итоге, улучшает качество нашей жизни и позволяет эффективнее планировать повседневные занятия.

## Образ клиента

Сервисы для отслеживания погодных условий созданы для всех, кто ценит точную и оперативную информацию о погоде. Пользователи, находясь в различных районах, получают достоверный источник данных о погоде.

Люди, которые проверяют информацию о погоде ежедневно, стремятся планировать свои повседневные дела и принимать такие решения, чтобы погодные условия не нарушали распорядок дня, в результате, быть готовыми к переменам в окружающей среде. Эта целевая аудитория ценит удобство использования и доступность приложения в любой точке, где имеется интернет-соединение.

Прогнозирование ориентировано на широкий круг пользователей, предоставляя всем возможность получения актуальных данных о погоде, что делает его инструментом для тех, кому важна своевременная и достоверная информация, независимо от их местонахождения.

## Сценарии

В мире современных технологий, предоставление точных и своевременных данных о погоде становится важным элементом повседневной жизни. Прогноз погоды не только ориентирован на предоставление актуальной информации, но также становится надежным спутником в различных сценариях, способствуя принятию осмысленных решений в зависимости от погодных условий.

Пикник

В теплый воскресный день группа друзей решила устроить пикник на свежем воздухе, но они не проверили прогноз погоды заранее. В результате, они оказались совершенно промокшими из-за неожиданного ливня, который намочил пледы, вещи и еду. Если бы они внимательно следили за прогнозом погоды, они либо перенесли бы пикник на другой день, либо выбрали бы место с крышей, чтобы избежать неприятных сюрпризов.

Поездка

Семья, готовясь к длительному путешествию, упустила из виду важность безопасности на дороге и не проверила прогноз погоды по маршруту. Этот недостаток внимания сказался на их поездке, когда они застали грозу, размывшую песчаную дорогу в лесу, что привело к аварии. Семье пришлось ждать помощи от проезжавшей машины, чтобы выбраться из сложившейся ситуации. Если бы они уделили должное внимание безопасности и проверили прогноз погоды, они могли бы избежать этих проблем на дороге.

Неудавшаяся свадьбы из-за дождя и порывистого ветра

Важные моменты, такие как свадьба, требуют тщательного планирования. Пара, готовящаяся к этому особенному дню, учла множество деталей, но упустила из виду прогноз погоды. Непредвиденные погодные условия испортили начало церемонии: порывистый ветер развеял украшения, а дождь намочил музыкальное и видеооборудование, а также всех гостей. Это привело к тому, что многие ушли до начала венчания, оставив пару в разочаровании. Если бы они учли прогноз заранее, они могли бы переместить церемонию в помещение и избежать неприятных сюрпризов со стороны погоды.

Наводнение

В весенний сезон жители прибрежного района игнорировали прогнозы погоды и не были предупреждены о возможных наводнениях, что привело к серьезным последствиям. Наводнение разрушило строения у берега и значительно размыло сам берег, вызвав разочарование среди жителей. Этот случай подчеркивает важность внимания к прогнозам погоды и принятия мер предосторожности для защиты от природных бедствий.

## Сбор и анализ прототипов

В процессе разработки приложения для прогноза погоды, был осуществлен глубокий анализ лучших практик. Изучив интерфейсы, функции и интерфейс пользователя существующих местных приложений в данной сфере.

Анализируя дизайн и структуру приложений, были определены оптимальные методы представления ключевых параметров, таких как температура, влажность и скорость ветра, для обеспечения удобства использования.

Подробно исследовав функциональность прототипов, включая возможности предоставления текущей погоды и прогнозов, были выявлены дополнительные функции, такие как уведомления о смене погоды и интерактивные карты. В результате, анализ взаимосвязи пользователей с другими приложениями помог выявить потребности в комбинированных функциональностях, что послужило направляющим фактором при разработке приложения.

# Проектирование приложения

## UI/UX дизайн приложения

Дизайн проекта разработан в программе Figma, поскольку она предоставляет удобные инструменты для создания прототипов, работы с векторной графикой в реальном времени, а также работе в команде. Кроме того, Figma обеспечивает возможность работы непосредственно в облаке, что позволяет удобно обмениваться и сохранять проекты, а также легко работать с ними из любого места и на любом устройстве.

Для проекта были определены основные экраны:

* главный экран с прогнозом погоды;
* настойки;
* окно о разработчике;
* техническая поддержка.

Для приложения определены следующая цветовая схема. Тема реализуется в светлых тонах, поэтому в ней основными цветами являются: синий, черный, белый, ярко зеленый, представленных на рисунке 1.



Рисунок 1 – Цветовая схема

Но стоит заметить, что данные цвета, разбавленные изображениями, не выглядят скучно, а весьма лаконично.

Логотип приложения представлен в ярких и приятных тонах, пользователь без затруднений найдет приложение в меню установленных приложений, логотип для мобильного приложения прогноза погоды представлен на рисунке 2.



Рисунок 2 – Логотип прогноза погоды

Цветовая схема логотипа состоит из: персикового, розового, голубого, зелёного и черного, представленных на рисунке 3. Это броские цвета, которые могут привлечь внимание пользователя к приложению.

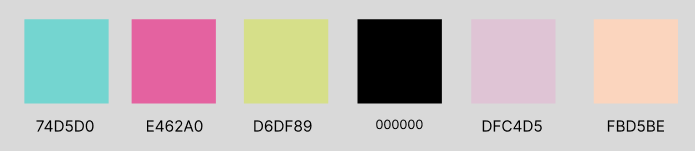


Рисунок 3 – Цветовая схема логотипа

После создания логотипа и выбора цветовой схемы приложения, мной был разработан дизайн для основных экранов, представленный на рисунке 4.

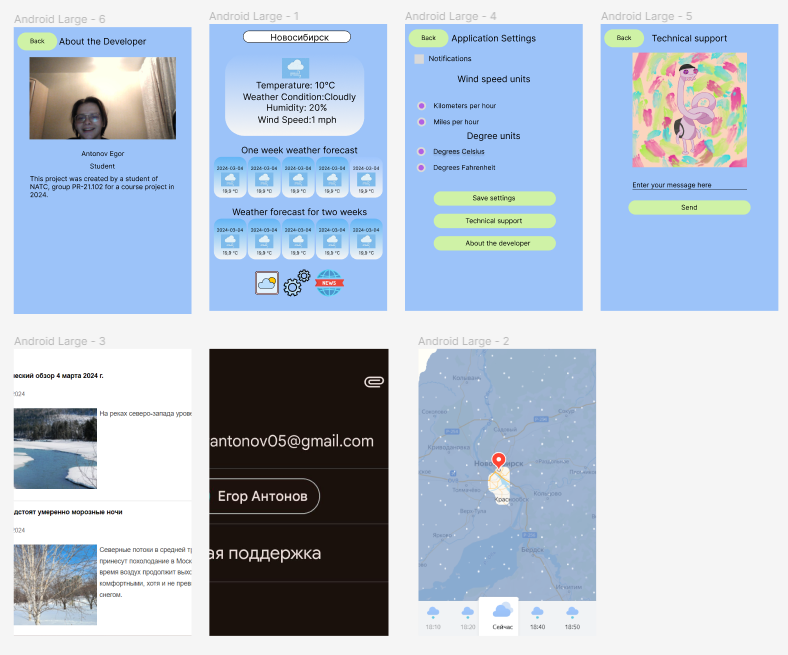


Рисунок 4 – Дизайн приложения

На рисунке 5 показано перемещение пользователя в приложении.

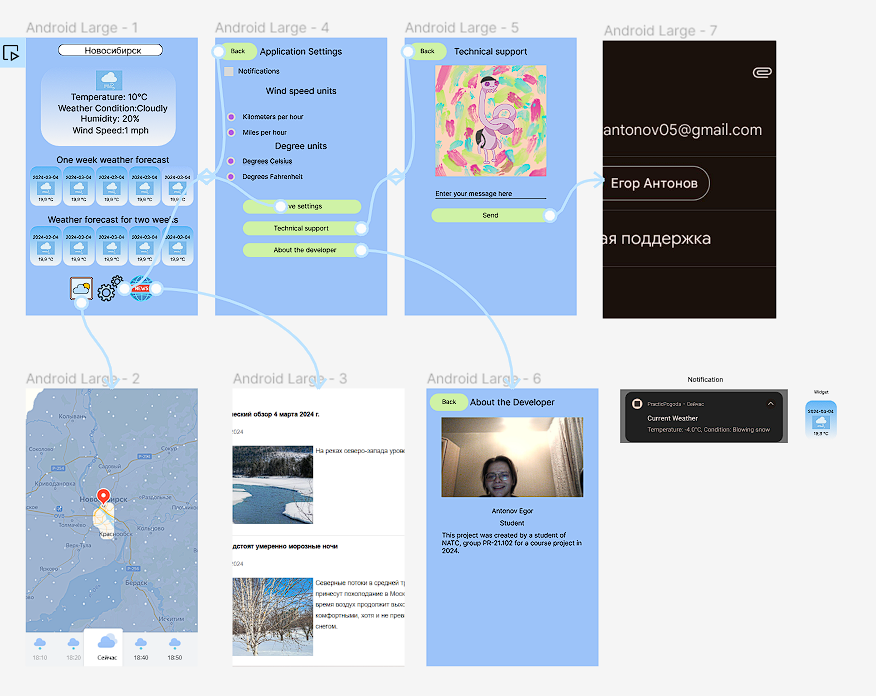


Рисунок 5 – Перемещение пользователя в приложении

При первом запуске приложения пользователь попадает на главный экран с прогнозом погоды. С каждого экрана можно перейти в любую часть приложения, однако на рисунке 5 обозначены только действия при нажатии кнопок, это сделано для того, чтобы не нагружать рисунок одинаковыми действиями. При нажатии на кнопку «Настройки» пользователь перейдёт на экран с настройками приложения, в настройках при нажатии на кнопку «Сохранения настроек» пользователь вернется на главный экран, также на экране настроек пользователь может отправить сообщение в техническую поддержку и ознакомиться с создателем приложения на экране «О разработчике». Нажимая на кнопку «Новости» откроется сайт погодных новостей в браузере. При нажатии на «Карты», пользователь сможет увидеть погодную карту.

## Выбор технологии, языка и среды программирования

Средой программирования для создания приложения выбрана Android Studio, так как она является одной из наиболее удобных и популярных сред разработки для мобильных приложений под платформу Android. Ее широкая распространенность обеспечивает доступ к обширному сообществу разработчиков, что упрощает поиск решений при возникновении ошибок.

Язык программирования Java был выбран для разработки приложения из-за его широкой популярности и устоявшейся позиции на рынке разработки программного обеспечения. Java обладает богатым набором функций и постоянно обновляется с добавлением новых возможностей, что делает его отличным выбором для разработки мобильных приложений.

Для получения данных о погоде в приложении использован сервис API – WeatherAPI. Этот сервис был выбран из-за своей бесплатности, подходящей функциональности и простоты использования. WeatherAPI предоставляет данные о погоде без ограничений, имеет понятную документацию и легко интегрируется в приложение, что делает его отличным выбором для данной задачи.

# Разработка мобильного приложения

## Описание API

В качестве API выбрана WeatherAPI, интегрируемая в AndroidStudio. В ней данные хранятся в формате JSON. Пример данных передаваемых с помощью API представлен на рисунке 6.



Рисунок 6 – Страница WeatherAPI

Выбранное API предоставляет доступ к широкому спектру погодных данных, включая прогнозы, текущие условия, обеспечивает простой и понятный интерфейс для запроса погодных данных, что является удобным для разработчиков любого уровня опыта, обеспечивает надежный и стабильный доступ к погодным данным, что важно для приложений и сервисов, которые зависят от актуальной информации о погоде, сервис обрабатывать большое количество запросов, что является подходящим как для небольших, так и крупных проектов, предоставляет различные параметры и опции для настройки запросов, такие как географические координаты, временные интервалы и типы погодных данных, что позволяет разработчикам получать именно те данные, которые им необходимы, сопровождается подробной документацией и технической поддержкой, что облегчает процесс интеграции и решение возможных проблем.

## Разработка мультимедийного контента

Весь контент разрабатывался с помощью языка разметки XML. Вёрстка выполнялась по дизайну, разработанному ранее в приложении Figma. Однако, по мере разработки, в дизайн были внесены изменения.

Иконки, кнопки и другие элементы приложения, были импортированы в проект в виде XML–файлов и фотографий (Рисунок 7). Такой способ хранения уменьшает вес приложения, а также, избавляет от проблем с потерей качества мультимедийного контента. Все ресурсы хранятся в папке «drawable»

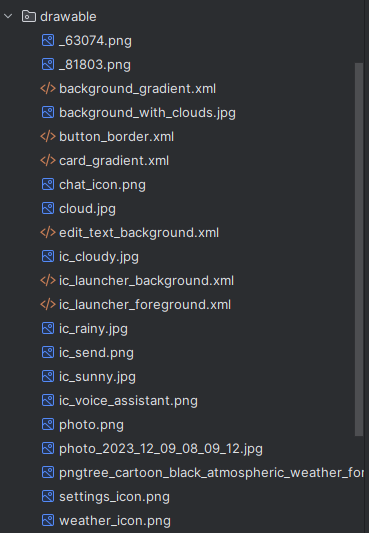


Рисунок 7 –Контент

Так как в приложении следовало сменить цвет navigationBar и statusBar с фиолетового и черного соответственно, то необходимо хранить цвета в файле colors.xml, далее представлен фрагмент кода:

<color name="black">#FF000000</color>

<color name="white">#FFFFFFFF</color>

<color name="title">#BB2196F3</color>

Цвет title используются для изменения цветов navigationBar и statusBar, поскольку эти элементы не имеет свойств в окне редактирования экранов следует изменять их в двух файлах: themes.xml–для светлой темы, а themes.xml (night) – для тёмной темы, далее представлен фрагмент кода, где представлены тёмная и светлая тема и были изменены цвета для этих двух элементов.

<style name="Base.Theme.PracticPogoda" parent="Theme.Material3.DayNight.NoActionBar"><item name="android:windowContentOverlay"> @null  
 </item>  
 <item name="android:statusBarColor"> @color/title  
 </item>  
 <item name="android:navigationBarColor"> @color/title  
 </item></style>

<style name="Base.Theme.PracticPogoda" parent="Theme.Material3.DayNight.NoActionBar"><item name="android:windowContentOverlay"> @null  
 </item>  
 <item name="android:statusBarColor"> @color/title  
 </item>  
 <item name="android:navigationBarColor"> @color/title  
 </item>  
</style>  
<style name="Theme.PracticPogoda" parent="Base.Theme.PracticPogoda" />

Это было сделано с целью эстетической привлекательности, поскольку фиолетовая и черные полосы не сочеталась с концепцией приложения.

В качестве основного фона был выбран градиент, переходящий из светло–синего в белый цвет, далее представлен фрагмент кода основного заднего фона.

<shape xmlns:android="http://schemas.android.com/apk/res/android">  
 <gradient  
 android:startColor="#F5F5F5"  
 android:centerColor="#64B5F6"  
 android:endColor="#2196F3"  
 android:angle="45"  
 android:type="linear"/>  
</shape>

Данное решение для заднего фона хорошо сочетается с основными элементами интерфейса, не отвлекает пользователя от элементов, а также не бросается в глаза, не затмевая основное впечатление при использовании приложения.

## Описание используемых плагинов

В проекте используются библиотеки с различными компонентами и функциями. Список всех библиотек, а также их описание представлен в таблице 2.

Таблица 2 – Библиотеки и их описание

|  |  |
| --- | --- |
| Библиотека | Описание |
| com.google.android.gms:play-services-location:18.0.0 | Эта библиотека предоставляет различные функции местоположения для приложений Android, такие как доступ к текущему местоположению устройства, геокодирование и другие сервисы местоположения от Google. |
| androidx.cardview:cardview:1.0.0" | Библиотека предоставляет компонент CardView, который упрощает создание карточек в пользовательском интерфейсе приложения, обеспечивая им тень и скругленные углы. |
| com.squareup.retrofit2:retrofit:2.9.0 | Библиотека для работы с сетевыми запросами в приложениях на Android. Упрощает выполнение HTTP–запросов и обработку ответов. |
| com.squareup.retrofit2:converter-gson:2.9.0 | Этот модуль Retrofit предоставляет конвертер для работы с JSON–ответами, упрощая процесс преобразования данных между JSON и Java объектами. |
| androidx.appcompat:appcompat:1.6.1 | Обеспечивает обратную совместимость с новыми функциями и возможностями Android для более старых версий платформы. |

Продолжение таблицы 2

|  |  |
| --- | --- |
| com.google.android.material:material:1.11.0 | Эта библиотека содержит компоненты пользовательского интерфейса, такие как кнопки, поля ввода, тулбары и другие, чтобы помочь создать красивое и современное приложение. |
| com.squareup.picasso:picasso:2.71828 | Библиотека для загрузки и отображения изображений в приложениях на Android. Она облегчает процесс работы с изображениями из сети или локального хранилища. |
| androidx.constraintlayout:constraintlayout:2.1.4 | Библиотека, предоставляющая гибкую систему размещения элементов в пользовательском интерфейсе, позволяя устанавливать ограничения между элементами для создания динамических макетов. |
| junit:junit:4.13.2 | Библиотека для написания и запуска модульных тестов в Java. |
| androidx.test.ext:junit:1.1.5 | Библиотека, расширяющая возможности JUnit для тестирования Android–приложений. |
| androidx.test.espresso:espresso-core:3.5.1 | Библиотека для автоматизированного UI–тестирования Android–приложений с использованием Espresso фреймворка. |

Данные библиотеки являются неотъемлемой частью приложения, обеспечивая его правильное функционирование и реализацию всех задуманных возможностей. Без этих библиотек приложение не сможет работать в полной мере, так как они предоставляют необходимые инструменты и функции, которые необходимы для его работы.

## Описание разработанных процедур и функций

В приложении реализованы следующие методы (таблица 3):

Таблица 3 – Методы приложения

|  |  |
| --- | --- |
| Метод | Описание |
| fetchDefaultCityByLocation() | Метод получает местоположение пользователя через датчики геопозиции. Если есть разрешение, получает координаты и определяет город с помощью Geocoder. Обновляет город в интерфейсе и запускает запросы погоды для этого города. |
| onRequestPermissionsResult() | Метод перехватывает результат запроса разрешений. Если запрос относится к разрешению на местоположение и разрешение предоставлено, вызывается метод fetchDefaultCityByLocation(), чтобы получить местоположение пользователя. В случае отказа от разрешения выводится сообщение Toast о его отклонении. |
| fetchWeatherData() | Метод отправляет запрос на получение данных о погоде с использованием API. После получения ответа, если запрос успешен, метод обновляет пользовательский интерфейс, обновляет город в виджете, выводит уведомление о погоде и сохраняет название последнего полученного города в SharedPreferences. В случае ошибки при запросе, метод выводит сообщение об ошибке в журнале. |
| showWeatherNotification() | Метод отображает уведомление о текущей погоде на основе данных, полученных из API. Он получает температуру и условия погоды из объекта weatherResponse. Перед отображением уведомления, метод проверяет, включены ли уведомления в настройках приложения. Если уведомления отключены, он отменяет все предыдущие уведомления и завершает выполнение. Если уведомления включены, метод создает канал уведомлений, а затем создает и отображает уведомление с помощью NotificationManagerCompat. |
| updateWidgetCity() | Метод обновляет город в виджете, сохраняя новое название города в SharedPreferences. Затем метод создает и отправляет широковещательное сообщение с действием ACTION\_UPDATE\_WIDGET и дополнительной информацией о новом названии города. |

Продолжение таблицы 3

|  |  |
| --- | --- |
| fetchWeatherForecast7Days() | Метод запрашивает прогноз погоды на 7 дней для указанного города через API. После получения ответа, если запрос успешен, метод обновляет данные прогноза погоды на 7 дней в соответствующем адаптере для отображения в пользовательском интерфейсе. В случае ошибки при запросе, метод выводит сообщение об ошибке в журнале. |
| fetchWeatherForecast14Days() | Запрашивает прогноз погоды на 14 дней для указанного города через API. После получения ответа, если запрос успешен, метод обновляет данные прогноза погоды на 14 дней в соответствующем адаптере для отображения в пользовательском интерфейсе. В случае ошибки при запросе, метод выводит сообщение об ошибке в журнале. |
| openWeatherMap() | Открытие погодной карты в браузере. |
| openWeatherNews() | Открытие ленты новостей в браузере. |
| updateUI() | Метод обновляет пользовательский интерфейс приложения согласно полученным данным о погоде и настройкам, сохраненным в SharedPreferences. Он извлекает текущую погоду из объекта weatherResponse и применяет ее к соответствующим элементам интерфейса, таким как температура, скорость ветра, условия погоды и влажность. Метод также загружает и отображает иконку погоды с помощью библиотеки Picasso, используя URL–адрес иконки из полученных данных о погоде. |
| sendEmail() | Метод позволяет отправить сообщение по электронной почте на адрес технической поддержки, определяет получателя, тему письма и текст сообщения на основе введенного пользователем текста. Затем метод создает Intent для отправки электронного письма и добавляет получателя, тему и текст сообщения в Intent. После этого метод запускает диалоговое окно выбора приложения для отправки электронного письма. Если на устройстве нет приложения для отправки почты, выводится сообщение с просьбой установить приложение для отправки почты. |

Продолжение таблицы 3

|  |  |
| --- | --- |
| updateWidgetCity() | Метод обновляет город в виджете, используя полученное название города. Он получает экземпляр AppWidgetManager, создает RemoteViews для обновления макета виджета, устанавливает новое название города в TextView виджета, а затем обновляет виджет с помощью AppWidgetManager. |
| loadWeatherIcon() | Метод загружает иконку погоды из указанного URL–адреса с использованием библиотеки Picasso. После успешной загрузки, метод устанавливает изображение иконки в ImageView виджета с помощью RemoteViews и обновляет виджет с обновленным изображением. В случае возникновения ошибки при загрузке, метод записывает сообщение об ошибке в журнал. |
| setAlarm() | Метод устанавливает сигнал для обновления виджета погоды с указанным интервалом времени, создает PendingIntent, который будет вызываться при срабатывании сигнала, и использует AlarmManager для установки повторяющегося сигнала с заданным интервалом. |
| updateAppWidget() | Метод обновляет содержимое виджета погоды с использованием данных о погоде, полученных из weatherResponse, настраивает содержимое виджета, устанавливая текущую температуру, название города и текущую дату. Затем метод загружает и устанавливает иконку погоды, используя URL–адрес изображения. После настройки виджета метод вызывает appWidgetManager.updateAppWidget() для обновления виджета с новым содержимым. Если данные о текущей погоде отсутствуют, метод записывает сообщение об ошибке в журнал. |
| onUpdate() | Метод обновляет виджет погоды, запускает процесс получения данных о погоде с помощью fetchWeatherData() и устанавливает повторяющийся сигнал с интервалом обновления, используя setAlarm(). Затем метод устанавливает обработчик кликов для виджета и обновляет его макет. После этого он планирует задачу для обновления погоды через определенный интервал времени. |

Продолжение таблицы 3

|  |  |
| --- | --- |
| disableAndEnableWidget() | Метод отключает и затем снова включает виджет погоды. Он получает экземпляр AppWidgetManager и список идентификаторов виджетов для данного компонента. Затем метод обновляет макет виджета, чтобы временно отключить его. После этого он приостанавливает выполнение потока на 1 секунду, а затем снова обновляет макет, чтобы включить его. |

Так же в приложении есть классы, которые не имеют методов (таблица 4):

Таблица 4 – Классы приложения

|  |  |
| --- | --- |
| Класс | Описание |
| Preferences | Класс Preferences представляет собой компонент приложения, отвечающий за управление настройками, содержит переменные для элементов пользовательского интерфейса, такие как переключатели и кнопки, а также методы для их инициализации и обработки событий, позволяет пользователю выбирать единицы измерения для температуры и скорости ветра, а также включать или выключать уведомления, после сохранения настроек происходит переход к главному экрану приложения |
| AppSettings | Класс управляет настройками приложения, позволяя получать и устанавливать состояние уведомлений, а также состояние элементов через объект SharedPreferences. |
| CurrentWeather | Модель, представляющая текущие погодные данные, включая температуру, состояние погоды, влажность, скорость ветра и название города. Использует аннотации SerializedName для соответствия полям JSON–ответа. |
| Developer | Активность, отображающая информацию о разработчике приложения, содержит кнопку «Назад», которая завершает активность при нажатии. |
| WeatherApiService | Интерфейс Retrofit для выполнения запросов к API погоды. Он содержит методы для получения текущей погоды и прогноза на 7 и 14 дней. Каждый метод представляет собой асинхронный вызов, который возвращает объект типа Call для выполнения запроса. |

Продолжение таблицы 4

|  |  |
| --- | --- |
| WeatherForecastResponse | Содержит классы для преобразования JSON–ответа от API погоды. Класс WeatherForecastResponse представляет ответ на прогноз погоды. Он включает класс Forecast, который содержит список прогнозов ForecastDay. Каждый объект ForecastDay представляет прогноз на конкретный день и содержит информацию о дате и погодных условиях в объекте Day. Внутри класса Day содержится информация о максимальной температуре в градусах Цельсия и Фаренгейта, а также объект WeatherCondition, описывающий текущие погодные условия. |
| WeatherResponse | Класс используется для хранения данных о текущей погоде, полученных из API. Он содержит объект currentWeather, представляющий информацию о текущих погодных условиях. |
| WeatherCondition | Класс, представляющий погодные условия, содержит текстовое описание погоды и URL иконки, отображающей текущие погодные условия, использует аннотации SerializedName для сопоставления полей JSON–ответа с полями класса при преобразовании данных. |

Метод позволяет приложению получать местоположение пользователя с помощью датчиков геопозиции. Если разрешение на доступ к местоположению еще не предоставлено, приложение запросит его у пользователя, затем определяется текущее местоположение пользователя, получает город, используя геокодирование, и обновляет информацию о нем. После этого происходит запрос погодных данных для этого города, а также прогноза на 7 и 14 дней, далее представлен фрагмент кода.

private void fetchDefaultCityByLocation() {  
 if (ContextCompat.*checkSelfPermission*(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) != PackageManager.*PERMISSION\_GRANTED*) {  
 ActivityCompat.*requestPermissions*(this, new String[]{Manifest.permission.*ACCESS\_FINE\_LOCATION*}, *LOCATION\_PERMISSION\_REQUEST\_CODE*);  
 } else {  
 FusedLocationProviderClient fusedLocationClient = LocationServices.*getFusedLocationProviderClient*(this);  
 fusedLocationClient.getLastLocation()  
 .addOnSuccessListener(this, new OnSuccessListener<Location>() {  
 @Override  
 public void onSuccess(Location location) {  
 if (location != null) {  
 double latitude = location.getLatitude();  
 double longitude = location.getLongitude();  
 Geocoder geocoder = new Geocoder(MainActivity.this, Locale.*ENGLISH*);  
 try {  
 List<Address> addresses = geocoder.getFromLocation(latitude, longitude, 1);  
 if (!addresses.isEmpty()) {  
 CITY\_NAME = addresses.get(0).getLocality();  
 Log.*d*("Location", "City Name: " + CITY\_NAME);  
 updateCityNameEditText(CITY\_NAME);  
 fetchWeatherData();  
 fetchWeatherForecast7Days(CITY\_NAME);  
 fetchWeatherForecast14Days(CITY\_NAME);  
 } else {  
 Log.*e*("Location", "No address found for the location");  
 }  
 } catch (IOException e) {  
 Log.*e*("Location", "Error getting city name from location", e);  
 }  
 } else {  
 Log.*e*("Location", "Location is null");  
 }  
 }  
 });  
 }  
}

Следующий метод отвечает за получение информации о погоде в указанном городе. Подобный метод используется для получения прогноза погоды на 7 и 14 дней соответственно. Этот метод выполняет асинхронный запрос к сервису погоды с использованием API–ключа и имени города. При получении успешного ответа, информация о погоде обновляется на пользовательском интерфейсе, также обновляется виджет погоды и отправляется уведомление о погоде. Кроме того, последний выбранный город сохраняется в локальном хранилище приложения для последующего использования. В случае возникновения ошибки, соответствующие сообщения об ошибках записываются в журнал логов, далее представлен фрагмент кода:

private void fetchWeatherData() {  
 Call<WeatherResponse> call = service.getWeather(*API\_KEY*, CITY\_NAME);  
 call.enqueue(new Callback<WeatherResponse>() {  
 @Override  
 public void onResponse(Call<WeatherResponse> call, Response<WeatherResponse> response) {  
 if (response.isSuccessful()) {  
 WeatherResponse weatherResponse = response.body();  
 updateUI(weatherResponse,sharedPreferences);  
 updateWidgetCity(CITY\_NAME);  
 showWeatherNotification(weatherResponse);  
 SharedPreferences preferences = getSharedPreferences("MyPreferences", Context.*MODE\_PRIVATE*);  
 SharedPreferences.Editor editor = preferences.edit();  
 editor.putString(*PREF\_LAST\_CITY*, CITY\_NAME);  
 editor.apply();  
 } else {  
 Log.*e*("WeatherAPI", "Failed to get weather data");  
 }  
 }  
 @Override  
 public void onFailure(Call<WeatherResponse> call, Throwable t) {  
 Log.*e*("WeatherAPI", "Error fetching weather data", t);  
 }  
 });  
}

Следующий метод предназначен для отображения уведомления о текущей погоде при запуске приложения. Этот метод создает уведомление о текущей погоде с использованием данных о температуре и условиях погоды, полученных из ответа на запрос о погоде. Перед отображением уведомления проверяется, разрешено ли отображение уведомлений в настройках приложения. Если уведомления отключены, все текущие уведомления отменяются. В случае использования версии Android SDK 26 и выше, создается канал уведомлений, если его еще не существует. Затем уведомление строится с использованием полученных данных и отображается пользователю. В случае возникновения ошибки, они выводится в журнал логов, далее представлен фрагмент кода:

private void showWeatherNotification(WeatherResponse weatherResponse) {  
 String temperature = String.*valueOf*(weatherResponse.getCurrentWeather().getTemperatureCelsius());  
 String condition = weatherResponse.getCurrentWeather().getWeatherCondition().getConditionText();  
 boolean isNotificationEnabled = appSettings.isNotificationEnabled();  
 if (!isNotificationEnabled) {  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(this);  
 notificationManager.cancelAll();  
 return;  
 }  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = getString(R.string.*channel\_name*);  
 String description = getString(R.string.*channel\_description*);  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel(*CHANNEL\_ID*, name, importance);  
 channel.setDescription(description);  
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(this, *CHANNEL\_ID*)  
 .setSmallIcon(R.drawable.*weather\_icon*)  
 .setContentTitle("Current Weather")  
 .setContentText("Temperature: " + temperature + "°C, Condition: " + condition)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*);  
 try {  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(this);  
 notificationManager.notify(1, builder.build());  
 } catch (SecurityException e) {  
 e.printStackTrace();  
 }  
}

Этот метод отображает данные о погоде в соответствии с настройками приложения пользователя, обрабатывает единицы измерения для температуры полученные из SharedPreferences. В зависимости от выбора пользователя данные форматируются и выводятся соответственно. Подобное форматирование используется не только для карточек на 7 и 14 дней, но и для карточки текущей погоды, но помимо единиц измерения температуры учитывается и скорости ветра, для скорости ветра предполагается, что в настройках могут быть выбраны одна из единиц измерения: «m/s» или «km/h», далее представлен фрагмент кода:

@Override  
public void onBindViewHolder(@NonNull ForecastViewHolder holder, int position) {  
 WeatherForecastResponse.ForecastDay forecastDay = forecastDays.get(position);  
 holder.forecastDateTextView.setText(forecastDay.getDate());  
 SharedPreferences sharedPreferences = PreferenceManager.*getDefaultSharedPreferences*(holder.itemView.getContext());  
 String temperatureUnit = sharedPreferences.getString("temperature\_unit", "Celsius");  
 double maxTemperature;  
 if (temperatureUnit.equals("Celsius")) {  
 maxTemperature = forecastDay.getDay().getMaxTempCelsius();  
 holder.forecastTemperatureTextView.setText(String.*valueOf*(maxTemperature) + "°C");  
 } else {  
 maxTemperature = forecastDay.getDay().getMaxTempFahrenheit();  
 holder.forecastTemperatureTextView.setText(String.*valueOf*(maxTemperature) + "°F");  
 }

Метод openWeatherMap(), который отвечает за переход на погодную карту в браузере при нажатии пользователем на кнопку. Этот метод создает новый Intent для просмотра содержимого по указанному URL–адресу. В данном случае, URL–адрес указывает на погодную карту. После этого Intent отправляется системе, и она открывает браузер по умолчанию с указанным URL–адресом. Подобный код используется для открытия ленты новостей в браузере, далее представлен фрагмент кода:

private void openWeatherMap() {  
 String url = "https://yandex.ru/pogoda/maps/nowcast";  
 Intent intent = new Intent(Intent.*ACTION\_VIEW*);  
 intent.setData(Uri.*parse*(url));  
 startActivity(intent);  
}

Метод sendEmail(), который позволяет отправлять почтовые сообщения в техническую поддержку с указанием суть ошибки из EditText. Этот метод формирует Intent для отправки электронной почты. Он определяет получателя (в данном случае, адрес технической поддержки), тему письма (в данном случае, «Техническая поддержка»), а также текст сообщения, который пользователь вводит в EditText. После этого Intent отправляется системе, и пользователю предоставляется выбор приложения для отправки почтового сообщения. Если на устройстве отсутствует подходящее приложение для отправки почты, выводится уведомление через Toast о необходимости установить такое приложение, далее представлен фрагмент кода:

private void sendEmail() {  
 String recipient = "egorantonov1804@gmail.com";  
 String subject = "Техническая поддержка";  
 String message = messageEditText.getText().toString();  
 Intent emailIntent = new Intent(Intent.*ACTION\_SEND*);  
 emailIntent.setType("plain/text");  
 emailIntent.putExtra(Intent.*EXTRA\_EMAIL*, new String[] { recipient });  
 emailIntent.putExtra(Intent.*EXTRA\_SUBJECT*, subject);  
 emailIntent.putExtra(Intent.*EXTRA\_TEXT*, message);  
 try {  
 startActivity(Intent.*createChooser*(emailIntent, "Отправка письма..."));  
 } catch (android.content.ActivityNotFoundException ex) {  
 Toast.*makeText*(this, "Установите приложение для отправки почты.", Toast.*LENGTH\_SHORT*).show();  
 }  
}

Метод устанавливает сигнал для обновления виджета погоды с указанным интервалом времени. Он создает PendingIntent, который будет вызываться при срабатывании сигнала, и использует AlarmManager для установки повторяющегося сигнала с заданным интервалом UPDATE\_INTERVAL, далее представлен фрагмент кода:

private void setAlarm(Context context) {  
 Intent intent = new Intent(context, WeatherWidgetProvider.class);  
 intent.setAction(AppWidgetManager.*ACTION\_APPWIDGET\_UPDATE*);  
 PendingIntent pendingIntent = PendingIntent.*getBroadcast*(context, 0, intent, PendingIntent.*FLAG\_MUTABLE* | PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
 AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.*ALARM\_SERVICE*);  
 if (alarmManager != null) {  
 alarmManager.setRepeating(AlarmManager.*RTC*, System.*currentTimeMillis*(), *UPDATE\_INTERVAL*, pendingIntent);  
 }  
}

Представленные методы являются неотъемлемой часть мобильного приложения, обеспечивая его правильное функционирование и реализацию всех задуманных возможностей. Без этих методов приложение не сможет работать в полной мере, так как они предоставляют необходимые инструменты и функции, которые необходимы для его работы.

# Тестирование

## Протокол тестирования дизайна приложения

Тестирование дизайна приложения проводится на Android SDK 24 и на более позднем Android SDK 34 с различной диагональю экранов для проверки разметки страниц и вёрстки приложения.

Примеры проверок отображения элементов на экране представлены на рисунках 8–17, слева расположен экран с Android SDK 24, а справа Android SDK 34, на рисунке 8 представлен главный экран приложения.

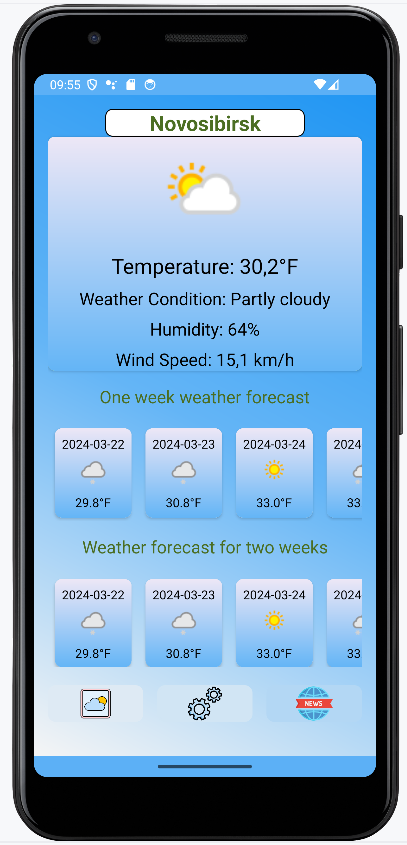
 

Рисунок 8 – Главный экран

Элементы интерфейса главного экрана в обоих случаях отображаются корректно. NavigationBar и statusBar нормальных размеров. Кнопки настроек, карты и браузера также стоят на своих местах.

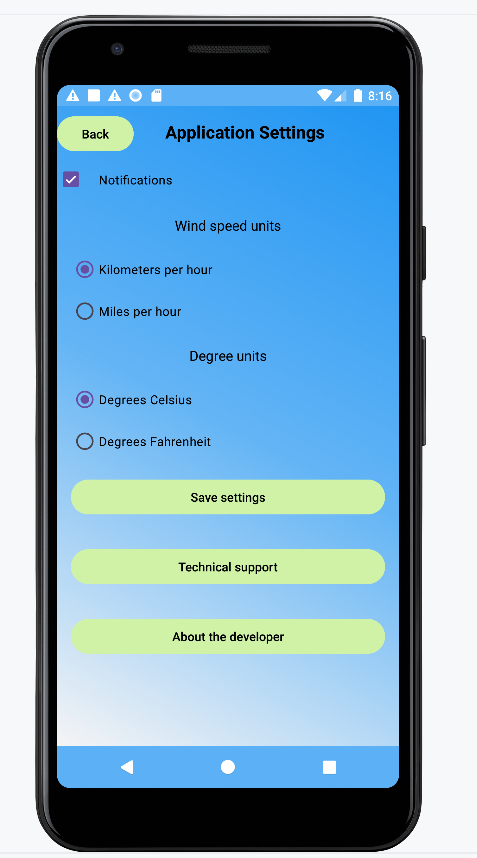
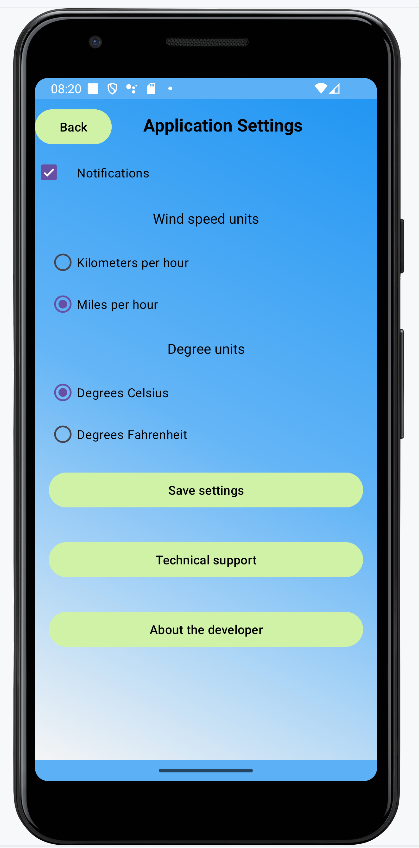
**** 

Рисунок 14 – Экраны настроек

Экран настроек так же отображается корректно на обоих устройствах, все элементы интерфейса расположены на своих местах.

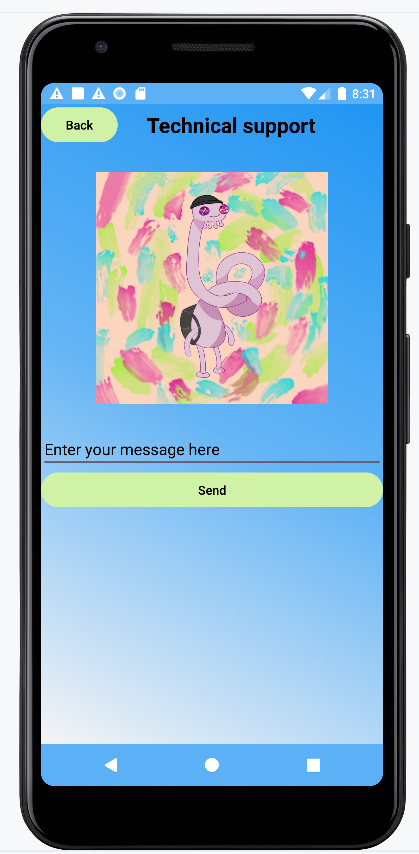
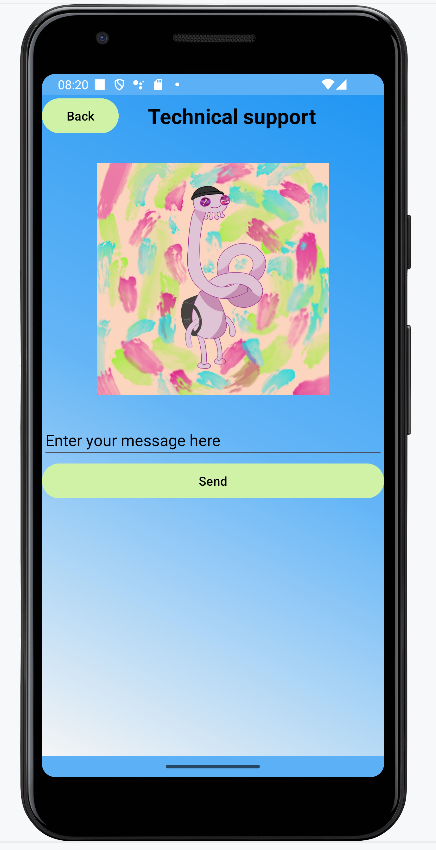
 

Рисунок 15 –Экраны технической поддержки

Экраны технической поддержки тоже отображаются корректно, поля и кнопки не сдвинуты, логотип и кнопка назад стоят на своих местах.

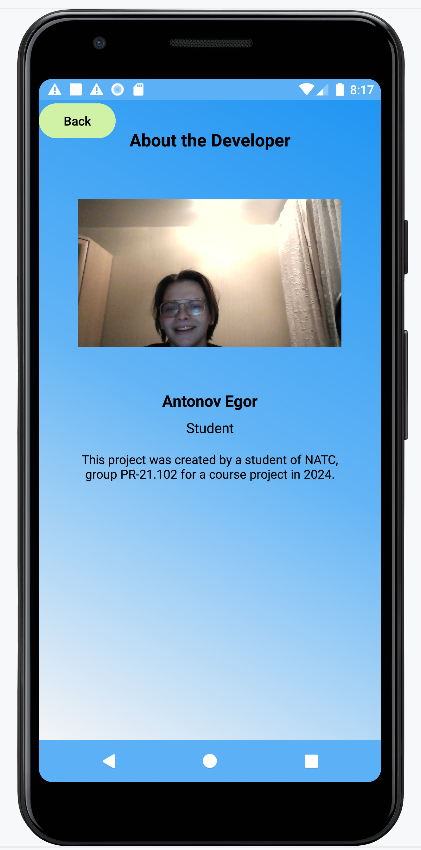
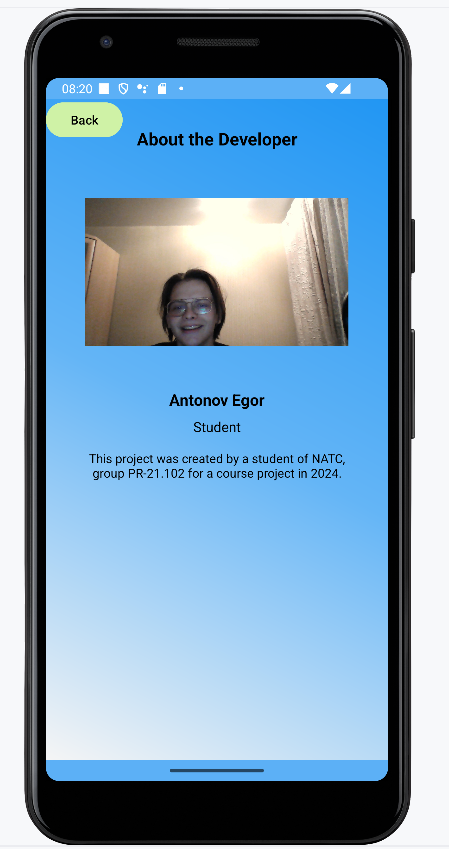
 ****

Рисунок 16 – Экран о разработчике

На экране о разработчике так же не обнаружено ошибок. Масштаб элементов сохранен. NavigationBar и statusBar находятся на своём месте и имеют нормальные размеры.

В процессе тестирования дизайна приложения на Android SDK 24 и более поздних версиях, в том числе SDK 34, были выполнены проверки отображения основных элементов интерфейса на различных диагоналях экранов. Результаты тестирования показали, что все страницы приложения, включая главный экран, экраны настроек, технической поддержки и информации о разработчике, отображаются корректно и без существенных отклонений на обоих версиях SDK. Элементы интерфейса, такие как NavigationBar и statusBar, сохраняют нормальный размер и расположение. Кнопки, поля ввода и другие элементы на страницах приложения также остаются на своих местах и не имеют смещений или искажений. Масштабирование элементов также осуществлено правильно, что обеспечивает удобное использование приложения на разных устройствах. Таким образом, на основании результатов тестирования можно сделать вывод о качественной работе дизайна приложения.

## Протокол тестирования функционала приложения

Так же необходимо проверить функционал приложения. Для этого, для каждой функции были разработаны TestCase.

В таблице 5 представлено тестирование функции вывода прогноза погоды на главном экране на сегодня, на 7 и 14 дней

Таблица 5 – Тестирование функции вывода прогноза погоды на главном экране на сегодня, на 7 и 14 дней

|  |  |
| --- | --- |
| Идентификатор | TC001 |
| Приоритет | Наивысший |
| Название | Вывод прогноза погоды на главном экране на сегодня, на 7 и 14 дней |
| Актеры | Пользователь |
| Предусловия | Открыто приложение с заполненным полем города или полученным местоположением |
| Шаги воспроизведения и альтернативные сценарии. | 1.Ввести название города при первом запуске  1a.Название города получено по местоположению и/или приложение запущено не первый раз и название города сохранено в предыдущем запуске  2.Прогноз погоды получен. |
| Ожидаемый результат | Прогноз погоды получен на сегодня, на 7 дней и на 14 дней. |
| Фактический результат | Прогноз погоды получен на сегодня, на 7 дней и на 14 дней. |
| Постусловие | Приложение получило прогноз погоды и пользователь может ознакомиться с ним. |

В таблице 6 представлено тестирование функции настроек приложения.

Таблица 6 – Тестирование функции настроек приложения.

|  |  |
| --- | --- |
| Идентификатор | TC002 |
| Приоритет | Наивысший |
| Название | Настройки приложения |
| Актеры | Пользователь |
| Предусловия | Открыто приложение с экраном настроек |
| Шаги воспроизведения и альтернативные сценарии. | 1.Произвести настройки, выбрав подходящие, используя checkbox и radiobutton  1a. Не менять действующие настройки  2.Нажать на кнопку сохранения настроек. |
| Ожидаемый результат | Настройки сохранились и приложение работает в соответствии с настройками. |
| Фактический результат | Настройки сохранились и приложение работает в соответствии с настройками. |
| Постусловие | Приложение произвело настройки, которые сохранились и приложение вернулось на главный экран и работает в соответствии с ними. |

В таблице 7 представлено тестирование функции отправки сообщения в техническую поддержку.

Таблица 7 – Тестирование функции отправки сообщения в техническую поддержку.

|  |  |
| --- | --- |
| Идентификатор | TC003 |
| Приоритет | Высокий |
| Название | Отправка сообщения в техническую поддержку |
| Актеры | Пользователь |
| Предусловия | Открыто приложение с экраном технической поддержки |
| Шаги воспроизведения и альтернативные сценарии. | 1.Заполнить EditText (поле ввода) сообщением  2.Нажать на кнопку отправки сообщения.  3.Выбрать сервис через который будет отправлено сообщение.  4.Удостовериться, что произошло автоматическое заполнение заголовка письма, адреса получателя, а также заполнился текст, введенный в поле ввода. |
| Ожидаемый результат | Пользователь перешёл в сервис по отправке письма, сообщение сформировалось и в последующем отправлено в техническую поддержку. |
| Фактический результат | Пользователь перешёл в сервис по отправке письма, сообщение сформировалось и в последующем отправлено в техническую поддержку. |
| Постусловие | Пользователь перешёл в сервис по отправке письма, сообщение сформировалось и в последующем отправлено в техническую поддержку, а после отправки вернулся на экран технической поддержки в приложении. |

В таблице 8 представлено тестирование функций перехода на погодную карту и ленту погодных новостей в браузере.

Таблица 8­– Тестирование функций перехода на погодную карту и ленту погодных новостей в браузере.

|  |  |
| --- | --- |
| Идентификатор | TC004 |
| Приоритет | Средний |
| Название | Перехода на погодную карту и ленту погодных новостей в браузере. |
| Актеры | Пользователь |
| Предусловия | Открыто приложение с главным экраном |
| Шаги воспроизведения и альтернативные сценарии. | 1.Открыть главный экран приложения.  2.Нажать на кнопку новостной ленты или погодной карты.  3.Удостовериться, что в браузере открылся выбранный сервис. |
| Ожидаемый результат | Пользователь перешёл в выбранный сервис. |
| Фактический результат | Пользователь перешёл в выбранный сервис. |
| Постусловие | Пользователь перешёл в выбранный сервис в браузере и может изучить погодную карту или новостную ленту. |

В таблице 9 представлено тестирование функции перехода на экран о разработчике.

Таблица 9­– Тестирование функции перехода на экран о разработчике.

|  |  |
| --- | --- |
| Идентификатор | TC005 |
| Приоритет | Низкий |
| Название | Перехода на экран «О разработчике» |
| Актеры | Пользователь |
| Предусловия | Открыто приложение с экраном настроек. |
| Шаги воспроизведения и альтернативные сценарии. | 1.Открыть экран настроек.  2.Нажать на кнопку, которая отвечает за переход на экран о разработчике.  3.Удостовериться в открытии экрана «О разработчике» |
| Ожидаемый результат | Пользователь перешёл на экран «О разработчике» |
| Фактический результат | Пользователь перешёл на экран «О разработчике» |
| Постусловие | Пользователь перешёл на экран «О разработчике» и может ознакомиться с информацией |

Разработанные TestCase демонстрируют корректную работу функций приложений. Так же в процессе тестирования не выявлено ошибок в дизайне приложения и его логике. Каждый экран прошёл проверку на разных API на корректное отображение элементов соответствующих экранов.

Заключение

В ходе разработки мобильного приложения прогноза погоды были учтены потребности пользователей в быстром доступе к актуальным метеорологическим данным.

Приложение обеспечивает надежный прогноз погоды на текущий момент и на ближайшие дни, позволяя пользователям быстро и удобно получать необходимую информацию.

Интерфейс приложения разработан с учетом простоты использования и минимального количества действий для получения прогноза погоды, позволяет пользователям быстро ознакомиться с текущей температурой, условиями погоды и прогнозом на будущее.

Важным элементом концепции приложения является своевременное обновление данных, что обеспечивает достоверность информации. Поддержка функций просмотра прогноза на несколько дней вперед дает пользователям возможность планировать свои активности с учетом ожидаемой погоды.

Тестирование приложения помогло убедиться в его корректной работе и соответствии ожиданиям пользователей. Разработанное приложение является надежным инструментом для получения актуальной информации о погоде в любой точке мира.

В целом разработка мобильного приложения прогноза погоды позволила создать удобный и функциональный инструмент, отвечающий потребностям пользователей. Проделанная работа является основой для дальнейшего развития и совершенствования приложения с учетом обратной связи пользователей, в том числе из технической поддержки, а также изменений в метеорологических условиях.

Библиография

1 WeatherAPI [Электронный ресурс]: Документация к API – Режим доступа к руководству: <https://www.weatherapi.com/docs/>

2 DevelopersAndroid[Электронный ресурс]: Полная документация по Android–Режим доступа к руководству: <https://developer.android.com/>

3 METANIT[Электронный ресурс]: Документация по языку программирования Java – Режим доступа к руководству: [https://metanit.com/java/android/](https://metanit.com/kotlin/)

Приложение А

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж

имени Б.С. Галущака»

**РАЗРАБОТКА Мобильной версии ДЛЯ ПРОСМОТРА ПРОГНОЗА ПОГОДЫ**

Техническое задание

НАТКиГ.202000.010.000ПЗ

Выполнил:

Студент группы ПР–21.102

Антонов Е.А.

2023

**Содержание**

Введение 43

1 Назначения разработки 44

2 Требования к мобильному приложению 45

2.1 Требования к функциональным характеристикам 45

2.2 Требования к надёжности 45

2.3 Условия эксплуатации 45

2.4 Требования к составу и параметрам технических средств 45

2.5 Требования к информационной и программной совместимости 46

2.6 Требования к защите информации 46

2.7 Требования к маркировке и упаковке 46

3 Требования к программной документации 47

4 Технико–экономические показатели 48

5 Стадии и этапы разработки 49

6 Порядок контроля и приёмки 50

**Введение**

Настоящее техническое задание распространяется на разработку мобильного приложения «Разработка мобильной версии прогноза погоды», используемого для просмотра погоды,

Наименование приложения: «PrognozPogody».

Краткая характеристика области применения: мобильное приложение предоставляет возможность просмотра видеороликов, загруженных на видеохостинг, а также ведения списка загруженных видеороликов самого пользователя и наиболее понравившегося ему контента.

Основанием для проведения разработки является Протокол №6 от 21 февраля 2024 года.

Наименование темы разработки − «Разработка мобильного приложения для просмотра прогноза погоды».

Условное обозначение темы разработки – «PrognozPogody».

**1 Назначение разработки**

Основное назначение приложения заключается в:

* обеспечении удобного и понятного интерфейса приложения для пользователя;
* предоставление текущего прогноза погоды, на неделю, на две недели.

Лица, которые могут работать с данной системой:

администратор – управляет полностью всем приложением, следит за его работоспособностью, следит за ящиком технической поддержки, по истечению срока действия ключа к API, продлевает или заменяет;

пользователь приложения – может смотреть прогноз погоды, следить за новостной лентой, а также изучать погодные карты.

**2 Требования к мобильному приложению**

**2.1 Требования к функциональным характеристикам**

Требования к составу выполняемых функций:

–– выбор города для просмотра погоды;

–– настройки для приложения;

–– просмотр прогноза текущей погоды, на 7 и 14 дней;

–– просмотр актуальных значений погоды.

**2.2 Требования к надёжности**

Обеспечение устойчивого функционирования должно выполняться несколькими действиями:

* организация стабильного интернет–соединения и корректной работы службы местоположения.

Приложение должно контролировать входную информацию:

* соблюдение типов данных при заполнении поля названия города.

**2.3 Условия эксплуатации**

Пользователь должен иметь практические навыки использования мобильного устройства под управлением операционной системы Android.

**2.4 Требования к составу и параметрам технических средств**

Для работы приложения необходимо мобильное устройство с установленной операционной системой Android не ниже версии 8.0.

**2.5 Требования к информационной и программной совместимости**

Проектирование взаимодействия с файловой системой должно быть выполнено в рамках разработки курсового проекта. При разработке взаимодействия с файловой системой должен быть использован язык программирования Java.

**2.6 Требования к защите информации**

Доступ к информации API предоставляется только администратору базы данных.

**2.7** **Требования к маркировке и упаковке**

Требования к маркировке и упаковке не предъявляются.

**3 Требования к программной документации**

Состав программной документации должен включать в себя:

* техническое задание;
* пояснительная записка.

**4 Технико–экономические показатели**

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

**5 Стадии и этапы разработки**

Таблица 1 – Стадии разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | **Этапы разработки КП** | **Сроки выполнения** | **Отчётность** |
| 1 | Определение цели и задач, объекта и предмета исследования | 24.02.2024 | Пояснительная записка |
| 2 | Описание предметной области | 26.02.2024 | Пояснительная записка |
| 3 | Выбор технологии, языка и средыпрограммирования | 01.03.2024 | Пояснительная записка |
| 4 | Оформление технического задания | 05.02.2024 | Техническое задание |
| 5 | Проектирование UI/UXдизайна | 10.02.2024 | Спецификации программного обеспечения |
| 6 | Разработка мобильного приложения | 22.04.2024 | Схема структурная системы и спецификации компонентов |
| 7 | Выбор API | 02.05.2024 | Программный  продукт |
| 8 | Отладка и тестирование приложения | 13.05.2024 | Тексты программных компонентов |
| 9 | Оформление документации | 22.05.2024 | Программная  документация |
| 10 | Защита | 09.04.2024 |  |

**6 Порядок контроля и приёмки**

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

* техническое задание;
* пояснительная записка;
* программный продукт;