**Weather tomorrow app**

**Краткое описание:**

Приложение выводит минимальную и максимальную температуру на завтра на произвольное количество городов (об этом позже). Данные берутся из сайта [https://openweathermap.org](https://openweathermap.org/) в формате JSON.

**Работа с приложением:**

При запуске приложения и открытия главного окна, начинают загружаться данные.

*ProgressBar*Пока данные не загрузились, отображается ProgressBar.

*EmptyView*  
В програме реализован EmptyView для следующих целей:

1. Если нет интернет соединения, на экран выведется сообщение “No internet connection.”
2. Если соединение есть, но данные отсутствуют, на экран выведется сообщение “No forecasts found.”

*General*В главном окне отображается список городов для которых делается прогноз. Каждый элемент списка несет в себе информацию о назвиния города, 2-значного коды страны, минимальную и максимальную температуры. **При нажатии на любой элемент**, пользователь переходит на страницу [https://openweathermap.org](https://openweathermap.org/) где отобразится прогноз погоды для выбраного города.

*SharedPreferences*Также в програме реализован простой функционал SharedPreferences. Он позволяет выбирать единицы измерения температуры (градусы Цельсия и Фаренгейта). При выборе новых параметров список элементов обновляется.

**Техническая информация:**

В програме реализованы следующие классы:

1. QueryUtils – вспомогательный класс, в котором описана процедура обращения к API OpenWeatherMap и формирования объекта Weather
2. SettingsActivity – класс в котором реализован функционал SharedPreferences
3. Weather – класс в котором описаны методы и конструктор для работы с прогнозом для одного города.
4. WeatherActivity – класс в котором заложена основная логика приложения. В этом же классе находится список в котором захардкожены отображаемые города. Описывает формирование URI для парсинга информации, получение обработаных и подготовленых для отображения элементов списка и отображения самого списка прогнозов.
5. WeatherAdapter – класс-адаптер, который готовит каждый элемент списка прогнозов для конечного отоббажения.
6. WeatherLoader – класс-загрузчик.

*Краткий принцип работы приложения:*При запуске приложения, запускается метод onCreateLoader, в котором подгружаются SharedPreferences, создается список городов и заполняется оными. Для того чтобы отобразить больше прогнозов на экране, достаточно в List<String> cities добавить город и 2-значный код страны в котором он находится. Эту информацию можно найти в архиве city.list.json.gz по ссылке <http://bulk.openweathermap.org/sample/> или же зайти на сайт [https://openweathermap.org](https://openweathermap.org/) где ввести в поиск нужный город, где и можно будет получить необходимую информацию.

Для каждого города в списке формируется URI , в котором передаются параметры описаные API документации сайта [https://openweathermap.org](https://openweathermap.org/).

По поводу работы этого API нужно знать несколько вещей:

1. Доступ к таким API не бесплатное удовольствие. Для того чтобы обращаться к серверу данных, нужно обзавестись специальным ключом, который передается 4-м параметром при формировании URI для парсинга (а именно параметр “appid”). Такой ключ на разных ресурсах дается в trial-версии на период времени до полугода. Я же выбрал именно OpenWeatherMap API т.к. он бесплатный и мне не придется каждые несколько месяцев регистрироваться на сайте для получения нового ключа (а ведь его потом и в приложение нужно вшить). Правда совершенно безболезненно мой выбор мне не обошелся.
2. В мой триал-пакет входит следующий тариф. По моему ключу можно обращаться к серверу не больше 600-сот раз в течении 10 минут, и не больше 50 000 раз в течении дня. Так же данные которые я могу получать от сервера это прогноз на 5 дней с интервалом в 3 часа. То есть в JSON ответе приходит список из 40 элементов, каждый из которых делает прогноз на последующие 3 часа.
3. Обращаться к серверу с просьбой показать мне сразу несколько городов с их прогнозами я тоже не могу. Только 1 город, и время указаное в пункте 2.

Что ж, хочешь удобств – плати за них.

После формирования URI для каждого города, эти URI записываются в список, который передается в новосозданный объект WeatherLoader’а. WeatherLoader создает список List<Weather> forecasts а его заполнение поручает методу fetchForecastData(List<String> url) из класса QueryUtils. В данном методе по каждому URI пришедшему извне, делается HTTP запрос на получение данных в формате JSON. Пришедшие данные обрабатываются методом extractFeatureFromJson.

В этом методе достается информация из JSON. Id города (для формирования ссылки перехода на прогноз на сайте при нажатии на элемент списка на главном экране), имя города и код страны.

Так же, достается информация по температуре. Помните я говорил на каких условиях я могу получать доступ к информации?

Попробую описать ситуацию: для того чтобы получить корректный прогноз на завтра, мне нужно выбрать все 3-часовые интревалы завтрашней даты и пропустить те же интервалы сегодняшней даты. API настроен таким образом, что прогноз выдает с начала текущего 3-часового интервала. То есть, если сейчас 19:35, то первым элементом в пришедшем от сервера JSON списке будет прогноз на период с 18:00 до 21:00. Соответсвенно, чтобы взять время на завтрашную дату, надо пропустить первые 2 интервала (18:00-21:00 ; 21:00 – 00:00) и взять последующие 8.

Именно следуя такой логике, в зависимости от текущего часа, мы отсекаем ненужные интервалы switch’ом на основании результатов выполения которого, мы получаем переменные counterFrom counterTo, которые отвечают за границы цикла, который пробегает по нужным нам интервалам.

В этом цикле мы пробегаем по всем итервалам завтрашнего дня и выбираем максимальную и минимальную температуры. После получения всех необходимых данных, формируем объект Weather и возвращаем его в пердыдущий метод. И так для всех городов.

После того как все объекты Weather сформировались – возвращаемся в fetchForecastData, где формируем список таких объектов.

Как только WeatherLoader закончил свою работу, вызывается метод onLoadFinished в котором наш список привязыывается к адаптеру. В WeatherAdapter каждый элемент из списка объектов Weather соотносится к разметке weather\_list\_item.xml после чего, в виде View возвращается для формирования окончательного списка элементов, таким как мы будем его видеть. Единственно что стоит здесь еще упомянуть, так это то, что в зависимости от знака температуры, текст в TextView красится в разные цвета: «+» - оранжевый, «-» - голубой, а для «0» цвет будет темно серым.