Цель: разработать приложение, которое:

1) при запуске обращается к Firebase (realtime database / remote config / cloud firestore)
2) от Firebase получает домен, через который обращается к нашему серверу
3) в зависимости от ответа нашего сервера:
    a) если сервер выдал ответ **error** - открывает MainActivity с текстом Hello World по центру
    b) если сервер выдал ответ в виде домена - открываем данный домен в Webview (Custom Chrome Tabs, далее CCT), примечание: в Webview / CCT открываем домен полученный от НАШЕГО сервера, первый домен, полученный от Firebase используется только для обращения к нашему серверу

Язык: java / kotlin - на ваш выбор

ВАЖНО:

1) Нужно использовать как можно МЕНЬШЕ кода, пишите проще, НЕ используйте паттерны, сложную архитектуру и тд.
2) minSdk - API 21, targetSdk - самый актуальный на текущий момент (сейчас API 33)
3) В build.gradle уровня приложения использовать
   classpath "com.android.tools.build:gradle:7.0.2"
4) проджект менеджер выдаст вам google-services.json, Firebase **уже настроен** за вас, вам ничего создавать и настраивать **НЕ нужно**
5) сервер **уже настроен** и работает, вам нужно реализовать лишь обращение к серверу и обработку, получаемой информации

Название проекта: server_v1 (либо v2 / v3 и тд, если уже выполняли подобный проект)
Название пакета: com.template именно так. НЕЛЬЗЯ com.template.еще_что_то

Нейминг:

- LoadingActivity - так должна называться первая активити, из которой происходит обращение к серверу.
- MainActivity - так должна называться пустая активити-заглушка (просто Hello World в центре экрана)
- WebActivity - так должна называться активити с Webview.

Что используется:

- Firebase Analytics - просто подключить и инициализировать на старте
- Firebase Cloud Messaging - просто подключить и инициализировать на старте
  ВАЖНО: необходимо учесть, что для корректной работы пушей на андроид 13+ необходимо запрашивать пермишен

Запросить разрешение на уведомление во время выполнения на Android 13+

Android 13 представляет новое разрешение среды выполнения для отображения уведомлений. Это влияет на все приложения, работающие на Android 13 или более поздней версии, которые используют уведомления FCM.

По умолчанию FCM SDK (версия 23.0.6 или выше) включает разрешение `POST_NOTIFICATIONS` , определенное в манифесте. Однако вашему приложению также потребуется запросить версию этого разрешения во время выполнения с помощью константы `android.permission.POST_NOTIFICATIONS` . Вашему приложению не будет разрешено показывать уведомления, пока пользователь не предоставит это разрешение.

Чтобы запросить новое разрешение среды выполнения:

Как правило, вы должны отображать пользовательский интерфейс, объясняющий пользователю функции, которые будут включены, если они предоставят приложению разрешения на публикацию уведомлений. Этот пользовательский интерфейс должен предоставлять пользователю возможность согласиться или отклонить, например кнопки **«ОК»** и **«Нет, спасибо»** . Если пользователь выбирает **ОК** , запросите разрешение напрямую. Если пользователь выбирает **Нет, спасибо** , разрешить пользователю продолжить без уведомлений.

Дополнительные сведения о том, когда ваше приложение должно запрашивать разрешение `POST_NOTIFICATIONS` у пользователя, см. в разделе Разрешение на выполнение уведомлений.

- Webview **или** Chrome Custom Tabs - **уточнять у своего проджект менеджера**
- Firebase Realtime Database **или** Firebase Remote Config **или** Cloud Firestore - **уточнять у своего проджект менеджера**

Логика LoadingActivity (ориентация только портрет):

1) Обращаемся к Firebase (firestore / realtime database / remote config)
2) В Firebase хранится домен, при помощи которого мы обращаемся к серверу.

   Примечание: google-services.json вам выдаст проджект менеджер, firebase уже настроен и все работает. НЕЙМИНГИ смотрите в самом конце документа, конкретно для ВАШЕГО сдк (firestore / realtime database / remote config)
3) Теперь нужно обработать следующие ситуации:
   - Firebase НЕ хранит ничего (пустой) => сразу открываем MainActivity. И при будущих открытиях приложения сразу открываем MainActivity, то есть не нужно проверять даже наличие ссылки в Firebase

- Firebase отдает вам домен вида https://site.com тогда переходим к пункту 4

4) После получения домена, сформируем ссылку для обращения к нашему серверу.

   Пример ссылки, по которой мы будем обращаться уже к нашему серверу:

   domenFromFirebase/?packageid=packageName&usserid=XXXXXXXXXXXX&getz=timeZone&getr=utm_source=google-play&utm_medium=organic

   Здесь:

   domenFromFirebase - это то, что вы получите из Firebase (relatime / firestore/ remote config). Домен будет иметь вид: https://site.com

   /? - ставится перед packageid

   packageid=packageName - вместо packageName будет пакет приложения. Важно: НЕЛЬЗЯ прописывать пакет вручную.

   usserid=XXXXXXXXXXXX - вместо иксов подставляем java UUID (random id)

   getz=timeZone - здесь нужно заменить timeZone на таймзону юзера (примеры: Europe/Moscow, Asia/Yekaterinburg и тд, нужна таймзона конкретного девайса)

   getr=utm_source=google-play&utm_medium=organic - данный кусок добавляем в конце ссылке “как есть”

   Обратите внимание, что перед packageid ставится **/?**
   А далее используется **&** в качестве разделителя

   Открывая сформированную ссылку, устройство должно отправлять свой настоящий User-Agent
   Самая популярная **ошибка** передавать в заголовок браузера
   - okhttp/3.14.9
   - android

   ВНИМАНИЕ: нужно делать запрос к ссылке с НАСТОЯЩИМ заголовком устройства пользователя, чтобы **было видно версию андроида и модель**.

5) Теперь, когда ссылка сформирована, обращаемся к серверу. Сервер настроен таким образом, что будет выдавать:

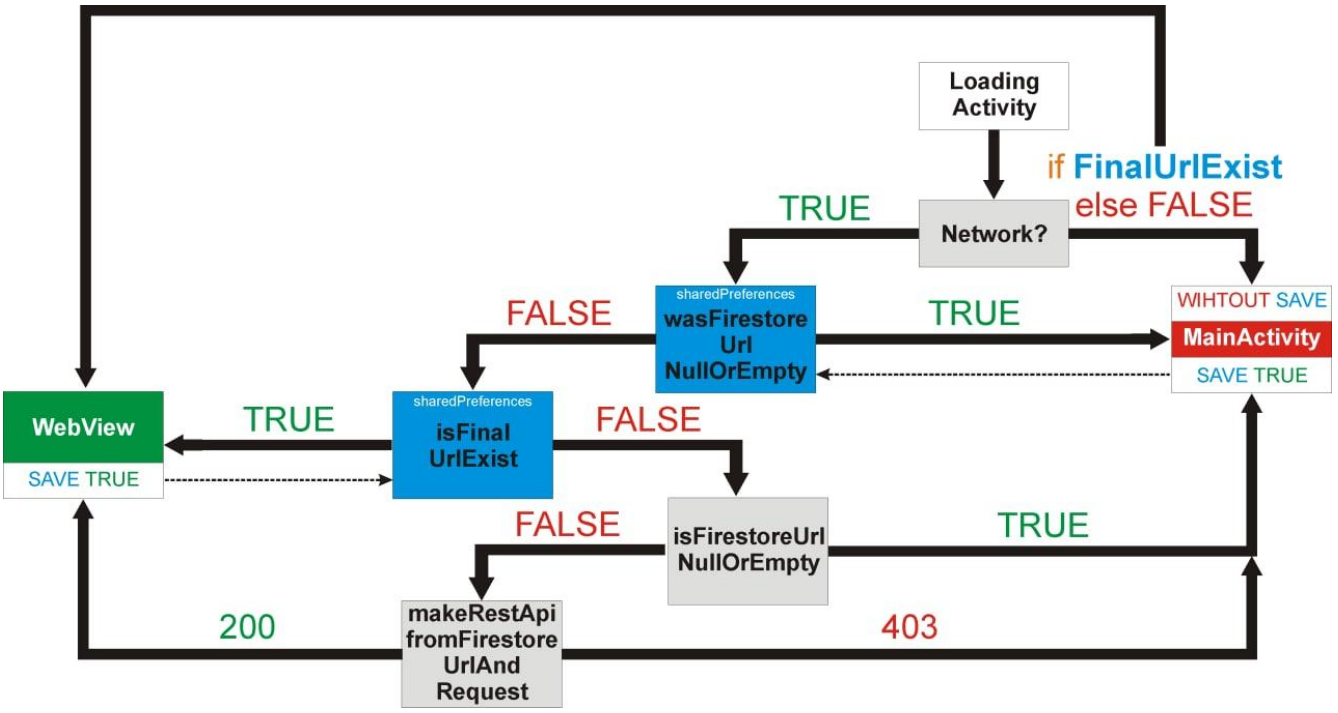
   - ошибку 403 с текстом error => открываем MainActivity
   - статус 200 => будет показ ответ в виде url, который мы запоминаем (в Preferences) и открываем в Webview / CCT

6) Не важно, что мы получили 403 или 200, мы запоминаем результат и в будущем (на 2, 3,...N открытие приложения) показываем, то что показали в первый раз (MainActivity или Webview / CCT)

   К серверу больше НЕ обращаемся. Обращение к firebase и нашему серверу происходить только 1 раз, при первом открытии приложения (first_open). То есть пока юзер не снесет приложение обращаться к серверу больше не требуется.

Примечание: Если у пользователя нет интернета, то открываем MainActivity и не сохраняем результат => при след запуске заново проверка на интернет и, в случае его наличия - обращение к серверу

Для лучшего понимания логики, ниже приводим блок-схему того, что нужно сделать. Обратите внимание, что в блок-схеме используется WebView и Firestore. В вашем же случае возможно будет что-то другое (Chrome Tabs вместо вебвью, remote config / realtime database вместо firestore), но логика при этом НЕ меняется.



Требования к верстке LoadingActivity

Далее будут описаны разные виды верстки для данной Activity, вы будете **реализовать только 1, какую именно вам скажет ваш проджект менеджер**.

1) По центру иконка приложения, снизу стандартный горизонтальный progress bar, который зациклен и НЕ отображает текущего прогресса в % (прогресс бар выровнен по горизонтали)
2) По центру: иконка и под ней название приложения, снизу круглый progress bar (выровнен по горизонтали)
3) По центру круглый progress bar, снизу надпись Loading… (выровнена по горизонтали)
4) По центру круглый progress bar
5) По центру иконка и под ней круглый progress bar

Примечание:
- от всех краев должны быть отступы
- все элементы должны быть выровнены (по центру / вертикали / горизонтали)
- элементы (картинки / текст / прогресс бар) должны иметь отступы друг от друга

WebActivity (Chrome Custom Tabs) - что нужно использовать именно вам, уточняйте у вашего проджект менеджера

Требования к Webview:

- Webview должно быть настроено для корректной отработки JS, работы с куками, local storage
- Сохранить куки в приложении. Именно куки браузера вебвью, то есть, если человек где то регистрировался то он остается залогинен там
- Должны корректно работать всплывающие окна на сайтах
- В manifest нужно добавить android:usesCleartextTraffic="true"
- Кнопка назад (на телефоне юзера): когда некуда возвращаться по страницам - НЕ закрывает приложение и НЕ перегружает страницу - Просто ничего не происходит, если некуда возвращаться. Если есть куда вернуться, то кнопка “назад” работает в стандартном режиме.
- В манифесте, для вебвью необходимо добавить параметр: android:windowSoftInputMode="adjustPan"
- Вебвью должно быть полноэкранным (без тулбара)
- Ориентация и portrait, и landscape (при этом при смене ориентации должно сохраняться состояние)

Требования к Chrome Custom Tabs

- Запускаем Chrome Tabs и параллельно закрываем приложение (finish)
- Цвет тулбара должен быть черным

- В manifest нужно добавить android:usesCleartextTraffic="true"
- Для Chrome Custom Tabs **не нужно** создавать WebActivity
- Инструкция по настройке Chrome Custom Tabs (обратите внимание, что домен нужно брать НЕ из инструкции, а тот, что **получен от нашего сервера**): https://static.gamezop.com/docs/guides/CCT-Implementation.html

Firebase Realtime Database

https://firebase.google.com/docs/database

Name: db
Name: link

Cloud Firestore

https://firebase.google.com/docs/firestore

Collection ID: database
Document ID: check
Field: link

Firebase Remote Config

https://firebase.google.com/docs/remote-config
Название параметра: check_link