



Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

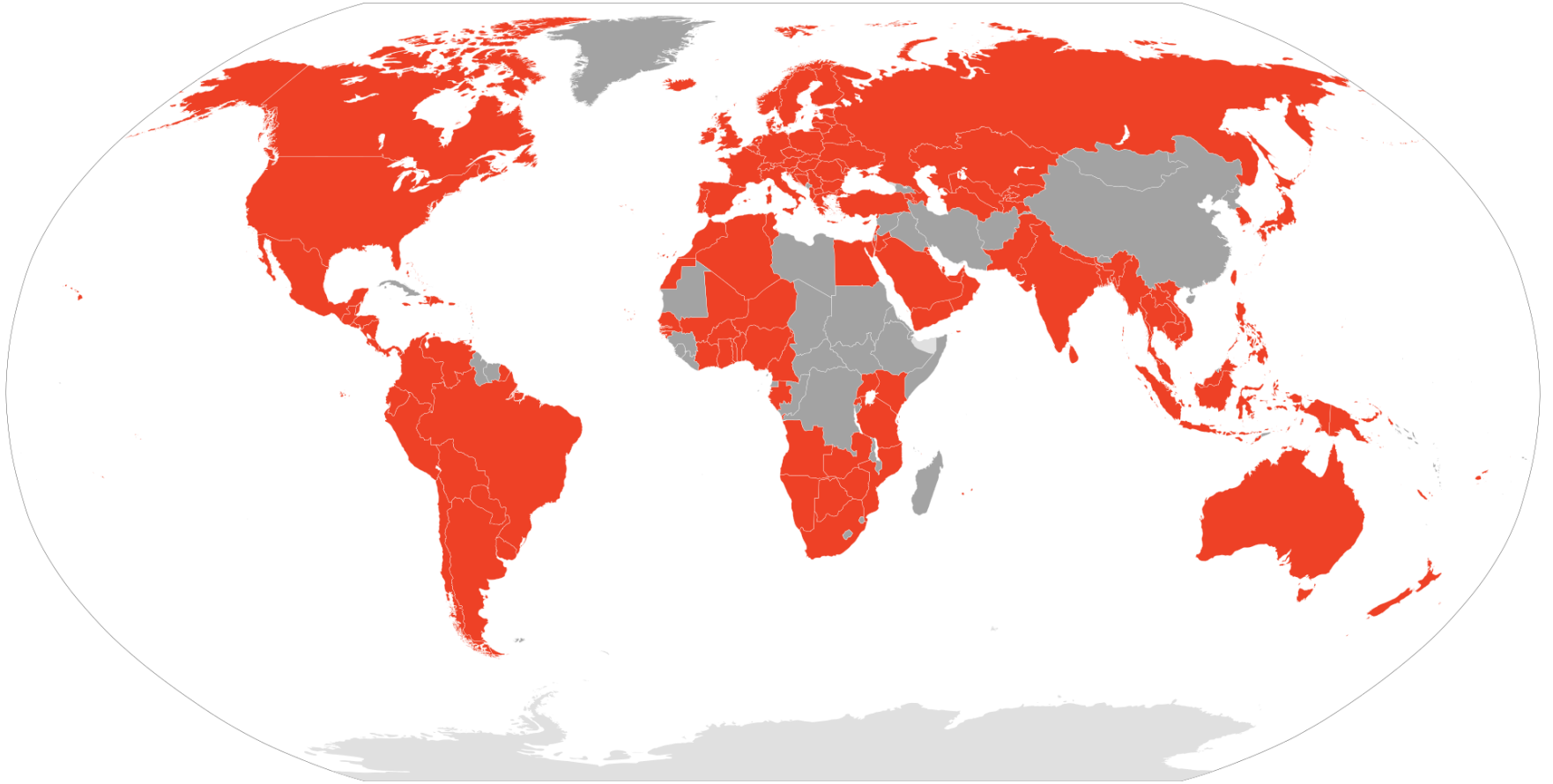
Android Development

Google Play

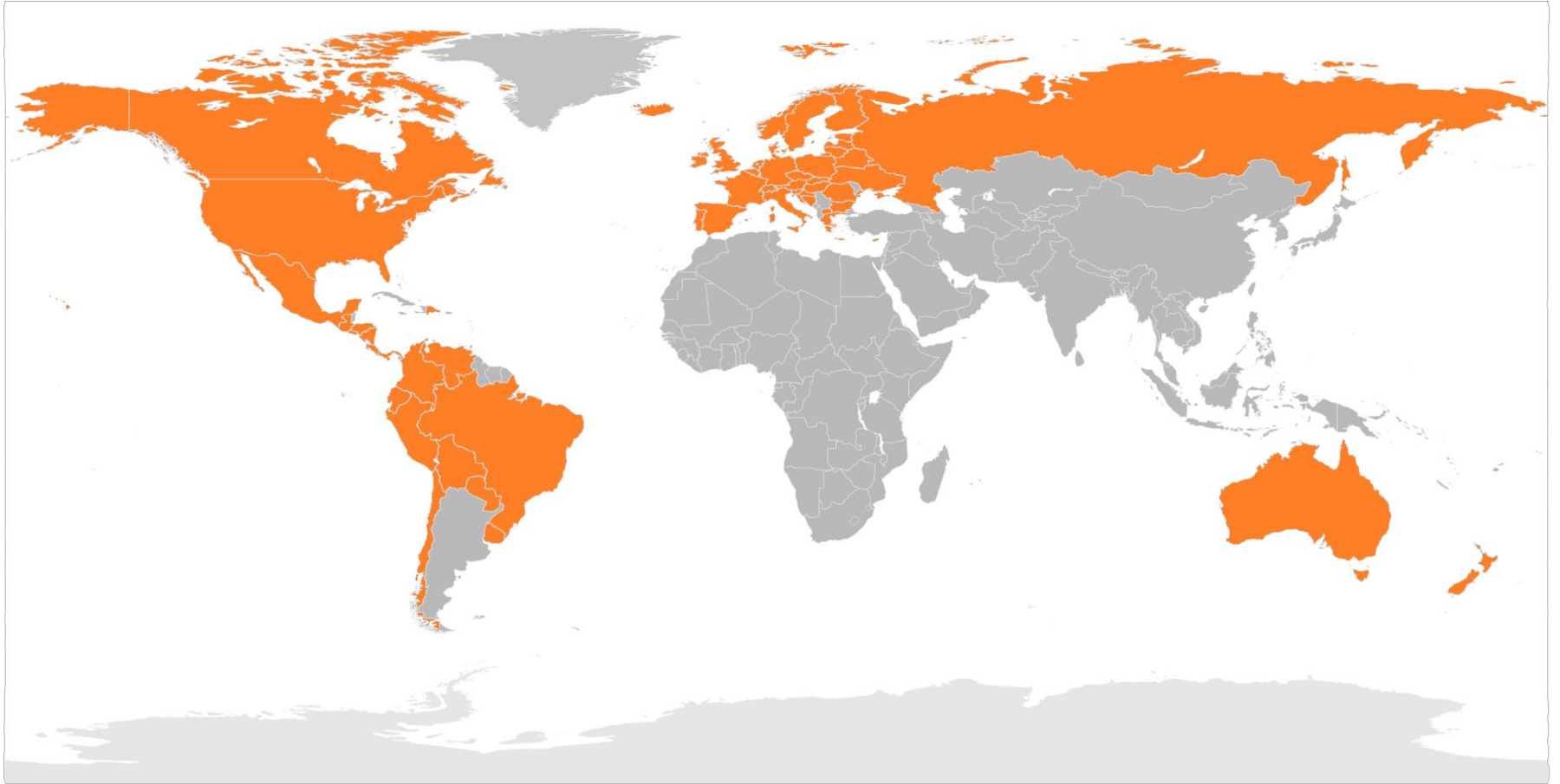
Google Play

- Official and most widespread way to distribute Android applications
- The Google Play Application is pre-installed on all devices
 - However, not on emulator
- It can be accessed with web browser as well, from desktop
- Not only for applications
 - Google Play Games
 - Play Pass
 - Google Play Movies and TV shows
 - Google Play Books
 - Google Play Music
 - Google Play Newsstand
 - News, magazines
 - Google Play Devices
 - Tablets, Phones, etc.

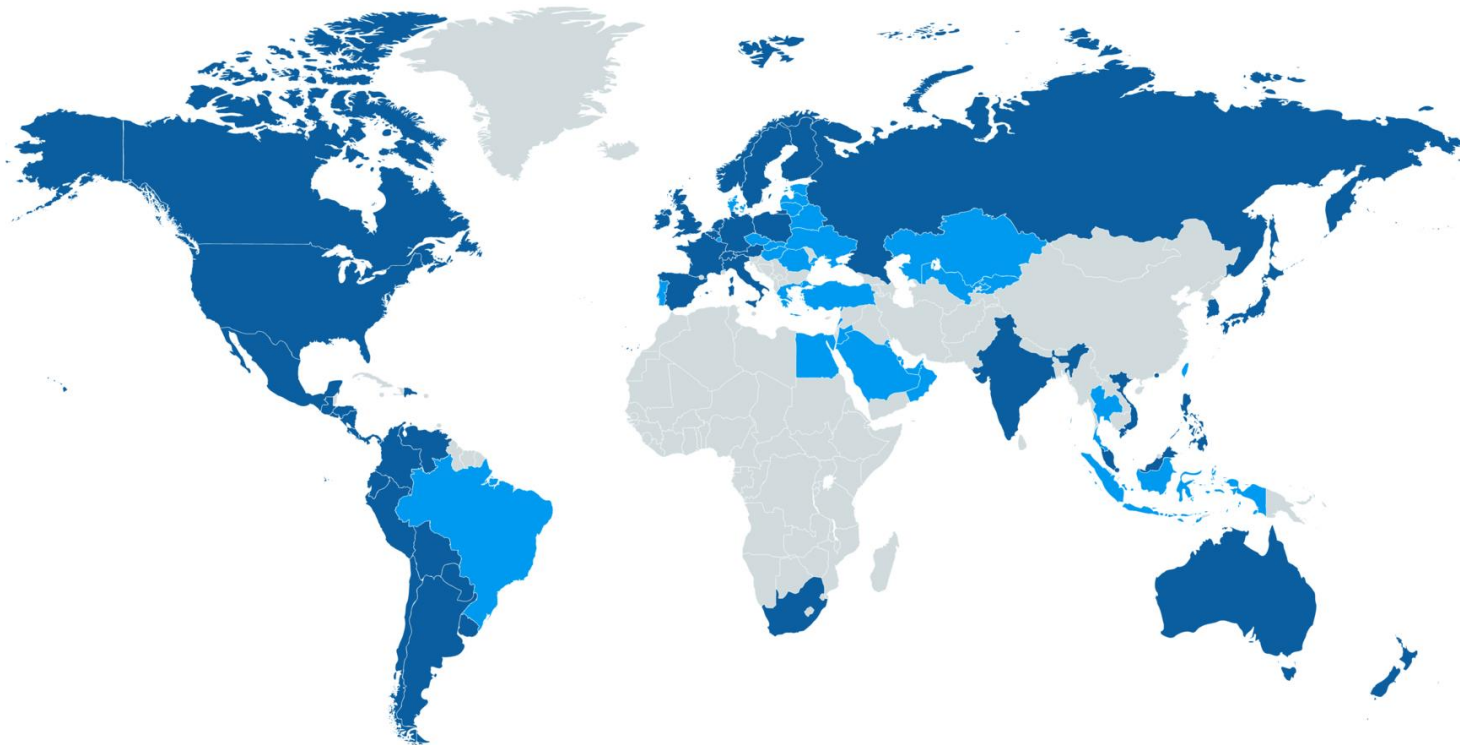
Google Play



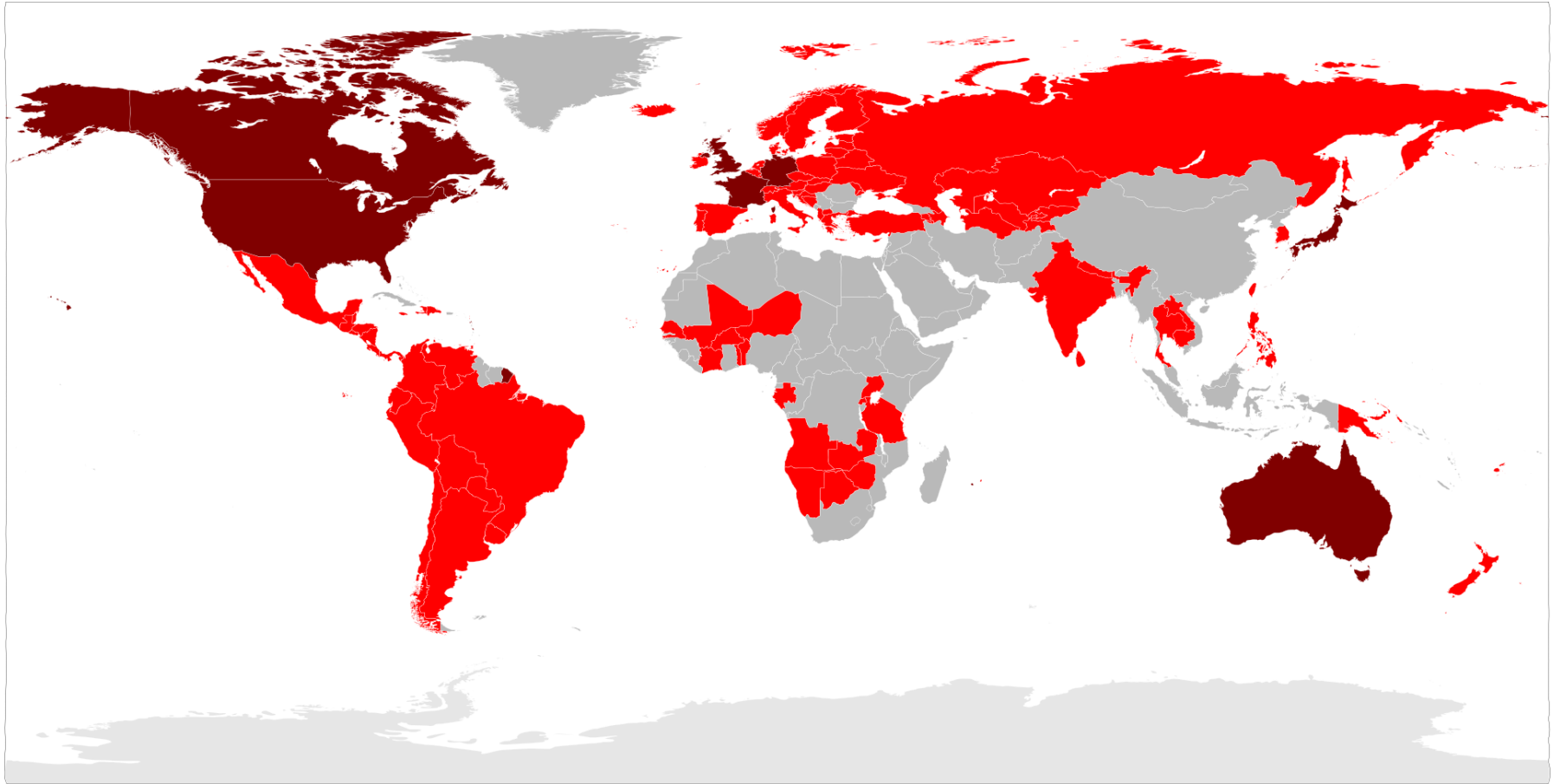
Google Play Music








Google Play Books



Google Play Movies & TVs

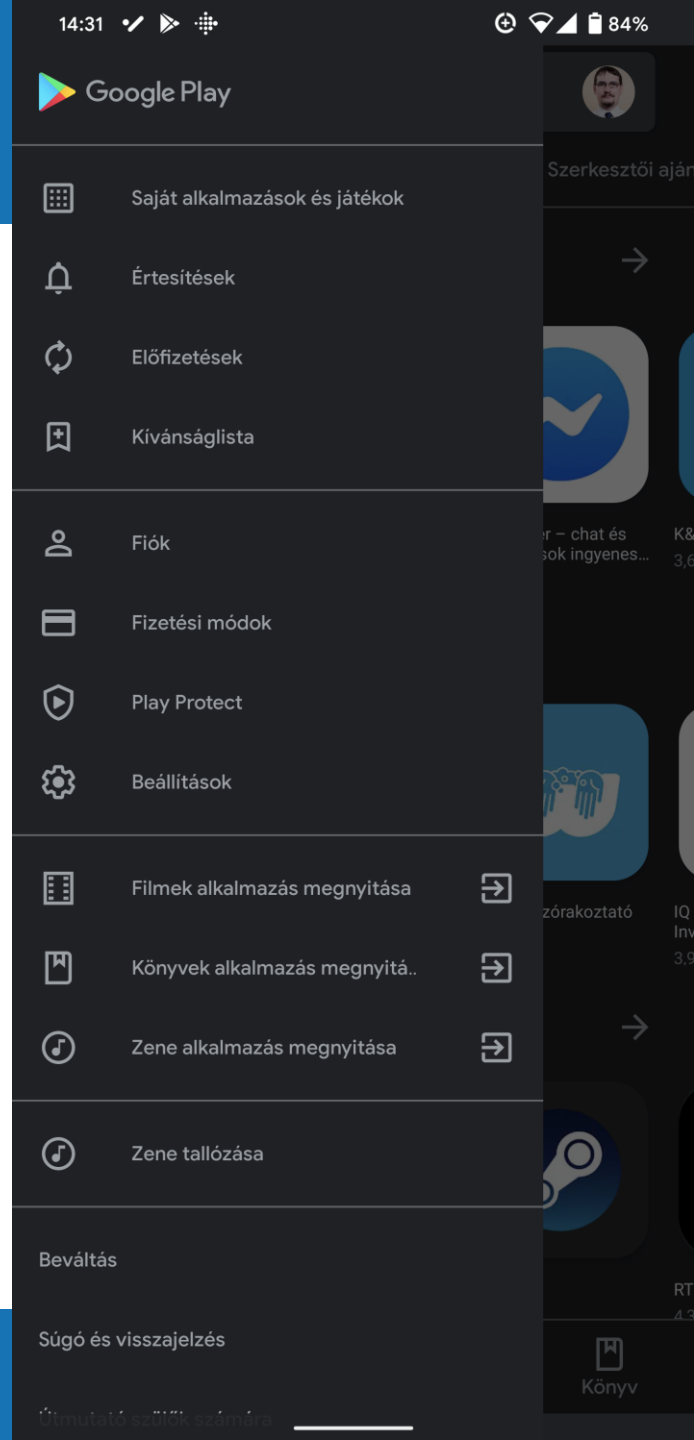


Google Play Hungary

Country/Region ⇅	Paid apps and games		Devices ^[114] ⇅	Magazines ^[44] ⇅	Books ^[44] ⇅	Movies & TV ^[44]		Music ^[44]	
	Customers can purchase ^[115] ⇅	Developers can sell ^[116] ⇅				Movies ⇅	TV shows ⇅	Standard ⇅	All Access ⇅
 Honduras	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes
 Hong Kong	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
 Hungary	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes
 Iceland	Yes	No	No	No	Yes	Yes	No	Yes	Yes
 India	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No

Google Play

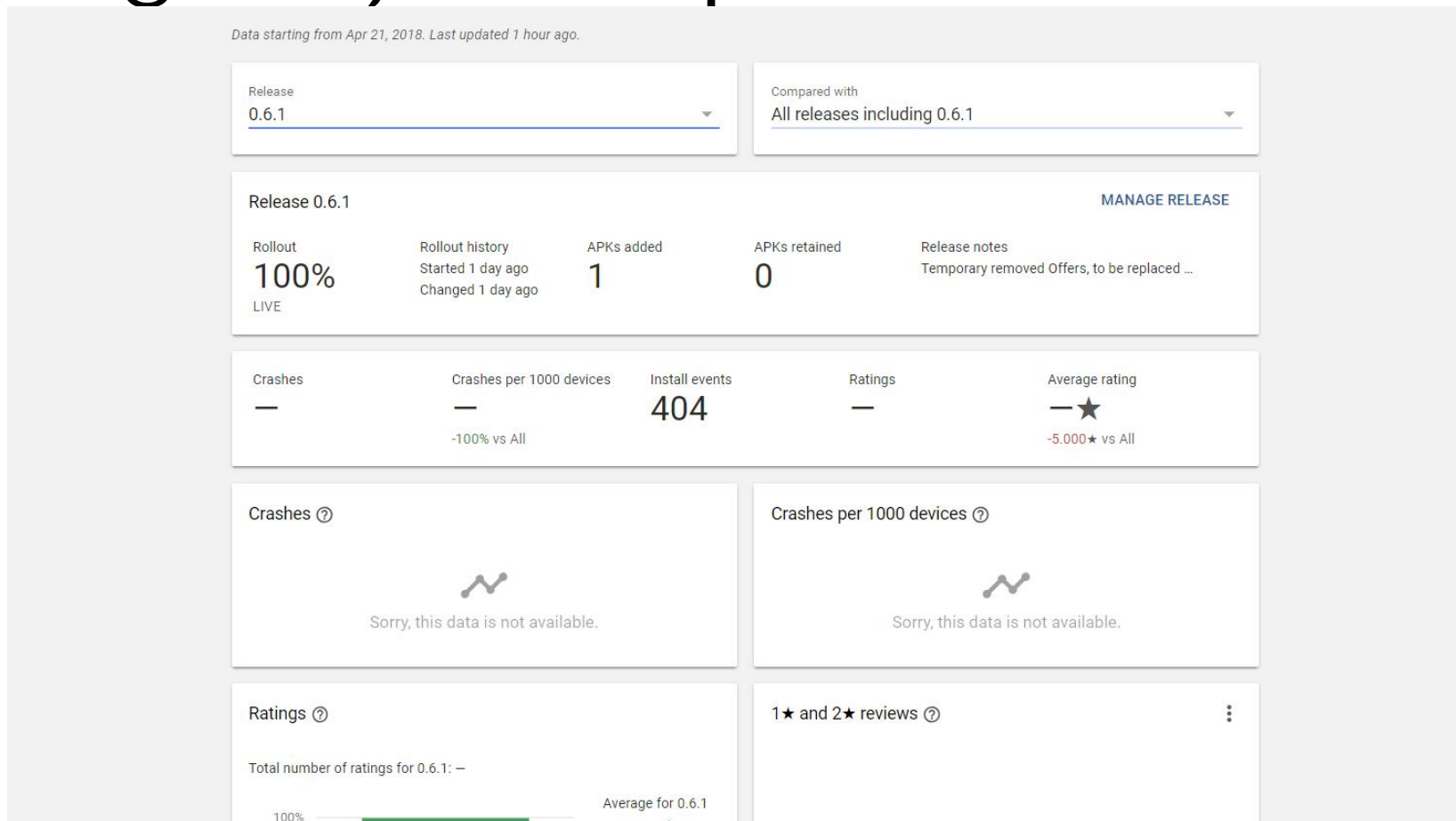
- Ratings of applications, stars
- Browsing categories
- Search
- Toplists
- Featured lists
 - App collections
 - Editor's Choice
 - Top Developer
- Detailed information about applications
 - Screen shots
 - Videos
 - Description
 - Opinions



Google Play developer access

- Before publishing a registration is required
 - 1. Open [Google Play Developer Console](#)
 - 2. Fill basic personal data (name, email, etc.)
 - 3. Accept the terms and conditions (license) (May differ by countries)
 - 4. Pay 25 USD (Using Google Wallet)
 - 5. After registration is accepted, the applications can be uploaded
-
- Merchant account
 - For applications with fees
 - 1. Open [Google Play Developer Console](#)
 - 2. Financial reports tab
 - 3. „Setup a Merchant Account now”

Google Play Developer Console




Lists of uploaded applications

☰ All applications

🔍 Search for apps



🔼 Filter ▼

▲ App name	Active / Total installs Ⓜ	Avg. rating / Total #	Last update	Status
 Pakli io.yomba.pakli	996 / 2,381	★ 4.81 / 67	Apr 21, 2018	Published

Page 1 of 1

Developer account

- More than one Google account can be connected to each developer account
 - In a company several person is entitled to manage the uploaded applications
 - The 25 USD registration fee must be paid only once

ACCOUNT DETAILS

Saved

DEVELOPER PROFILE

Developer name *

Company or developer name...

The developer name will appear to users under the name of your application.

Email address *

your.address@gmail.com

Website

http://your.site.com

Phone Number *

+1-650-555-1212

Include plus sign, country code and area code.
For example, +1-650-253-0000.

Email updates

☐ I'd like to get occasional emails about development and Google Play opportunities.

Editing the description of an application

Store listing

Pakli
Published

Product details

ENGLISH (UNITED STATES) – EN-US Languages (2) Manage translations

Fields marked with * need to be filled before publishing.

Title *

English (United States) – en-US

Pakli

5/50

Short description *

English (United States) – en-US

Meet people, play board games, make new friends!

48/80

Full description *

English (United States) – en-US

Have you ever been one player short of playing your favourite game? Have you ever been afraid of being thrown out of a pub for playing board games? Or you are simply open for meeting new people, and trying out new things? Pakli is here for you.

Pakli helps you find people nearby to play board games with! You can organize game nights at home, or at board game friendly pubs and cafes. Invite your friends and/or wait for new people to join your event. Come, and be part of a fast growing community, filled with awesome people of the golden age of board gaming!

563/4000

Please check out our [Metadata policy](#) to avoid some common violations related to app metadata. Also, please make sure to review all the other [program policies](#) before you submit your apps.

Description

- Mandatory fields
 - Name of the application
 - Short description
 - Full (detailed) description
 - Feature graphic
 - Screen shots
 - At least two
 - Video (YouTube)
 - Icon
 - Tablet, Android TV ...
 - Further screenshots

Device exclusion rules

- You can view the catalog of available devices and review which devices are compatible with your app.
 - To help ensure the widest availability of your app, review your supported and excluded devices lists regularly.
- App can be
 - Supported
 - Your app is compatible with the device.
 - Partially supported
 - If a device has multiple models, you'll see this status when only some of the models are supported by your app's manifest criteria.
 - Unsupported
 - Your app includes a feature or property (e.g. screen size, SDK level, etc.) not available on the device
- Rules can be defined to exclude devices

Distribution channels

- Internal test

- Quickly distribute your app for internal testing and quality assurance checks.
- You can create a list of internal testers by email address. An internal test can have up to 100 testers per app.
- Note
 - Payment: For paid apps, testers can install your internal test version for free.
 - Device exclusion rules: don't apply to internal testers.
 - Policy and security reviews: Internal tests may not be subject to the usual Play policy or security reviews.

- Closed test

- Create a closed release to test pre-release versions of your app with a larger set of testers.
- Once you've tested with a smaller group of employees or trusted users, you can expand your test to an open release.
- An Alpha track will be available as your initial closed test.
 - Additional closed groups can be created.

Distribution channels

- Open groups also can be defined
 - Create an open release to run a test with a large group and surface your app's test version on Google Play.
 - If you run an open test, anyone can join your testing program and submit private feedback to you
 - An URL have to be visited
- Staged rollout
 - Public
 - You can define the amount of users (in %) who gets the update
 - Update can be stopped in case of serious problems
 - Installed updates cannot be revoked

Distribution channels

- Staged rollout – Public
 - New and existing users are eligible to receive updates from staged rollouts and are chosen at random for each new release rollout.
 - If you discover an issue, you can halt a staged rollout to help minimize the number of users who experience the issue with your app.
 - When you halt and then resume the rollout of your release, you'll be affecting the same set of users.
 - Your app update will be available to the percentage of users in your staged rollout
 - It may take time for the full group to receive the update.
 - Users won't be notified if they receive a version of your app in a staged rollout.

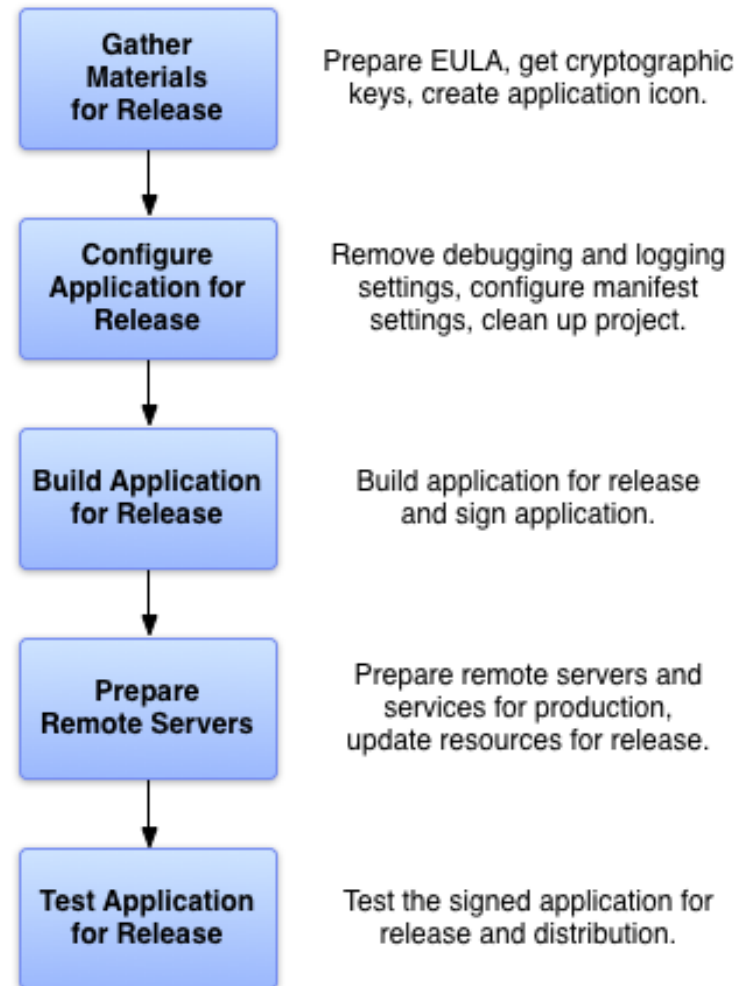
Application requires payments

- A free application cannot be transformed into a paying application
 - The other direction is available
 - In-App Products are also available
- Different prices can be set for different countries
 - Can be changed
 - Minimum price for Hungary is 125 Ft
 - Maximum price is 133 700 Ft
- Most of the cases the free app is a different application with different id
 - Or the „premium” one is an in-app product

Restrictions

- Restrictions can be set
 - Countries
 - Regions
 - Carriers
 - Android version
 - OpenGL ES properties
 - Device properties

How to release an application



Version number

- `versionCode`
 - Integer
 - It must be increasing
 - Not displayed for the user
 - Google Play uses for upgrading
- `versionName`
 - Text
 - For users
 - May be correlated with the `versionCode`
 - You should use the MAJOR.MINOR.PATCH format
 - Semantic version numbering system

Signing the application

- A key has to be generated first
 - Easy with Android Studio
- Build/Generate Signed APK

Signing the application

```
android {  
    ...  
    defaultConfig { ... }  
    signingConfigs {  
        release {  
            storeFile file("myreleasekey.keystore")  
            storePassword "password"  
            keyAlias "MyReleaseKey"  
            keyPassword "password"  
        }  
    }  
    buildTypes {  
        release {  
            ...  
            signingConfig signingConfigs.release  
        }  
    }  
}
```

- \$ gradlew assembleRelease

gradle-play-publisher plugin

- Non-official: [here](#)
- Uses the [Android Publisher API](#)
- A Google Service account has to be set up
 - And linked to the Play Developer account
 - Permissions has to be granted
 - Then you are identified with the Service account and p12 file
- Gradle tasks
 - For upload application
 - Upload descriptions
- You do not have to use GUI
- Data can be accessed from repository
- A Continuous Integration server can be integrated

Multiple APK

- Multiple APK support allows you to publish different APKs for your application that are each targeted to different device configurations.
 - Each APK is a complete and independent version of the application
 - They share the same application listing on Google Play
 - They must share the same package name
 - Must be signed with the same release key.
- This feature is useful for cases in which your application cannot reach all desired devices with a single APK.

Multiple APK

- Android-powered devices may differ in several ways
- Android applications usually run on most compatible devices with a single APK
 - By supplying alternative resources for different configurations and the Android system can select the appropriate resources for the device at runtime.
- In a few cases a single APK is unable to support all device configurations
 - Because alternative resources make the APK file too big
 - Or other technical challenges prevent a single APK from working on all devices.

Multiple APK

- Google Play allows you to publish multiple APKs under the same application listing.
- Google Play then supplies each APK to the appropriate devices based on configuration support you've declared in the manifest file of each APK.

Multiple APK

- By publishing your application with multiple APKs, you can:
 - Support different OpenGL texture compression formats with each APK.
 - Support different screen sizes and densities with each APK.
 - Support different device feature sets with each APK.
 - Support different platform versions with each APK.
 - Support different CPU architectures with each APK
 - such as for ARM or x86, when your app uses the Android NDK)
 - Optimize for entry-level devices such as those running Android
 - Go edition

Multiple APK

- Creating multiple APKs
 - It is probably needed to create separate Android projects for each APK
 - Can be developed separately.
 - Each APK for the same application must have a unique version code, specified by the `android:versionCode` attribute.

Android App Bundle

- An Android App Bundle is a publishing format
 - It includes all your app's compiled code and resources
 - defers APK generation and signing to Google Play.
- Google Play uses your app bundle to generate and serve optimized APKs for each device configuration
 - only the code and resources that are needed for a specific device are downloaded to run your app.
 - No longer have to build, sign, and manage multiple APKs to optimize support for different devices
 - Users get smaller, more-optimized downloads.

Android App Bundle

- Dynamic feature modules can be added
 - These modules contain features and assets that you can choose not to include when users first download and install your app.
 - Using the Play Core Library, your app can later request to download those modules.
 - Google Play will serve only the code and resources for that module to the device.
 - When you combine this with support for uncompressed native libraries, larger apps (such as games) can reduce their storage requirements and increase user retention.

Android App Bundle

- To build app bundles:
 - Android Studio 3.2 is required
 - Add support for Dynamic Delivery by including a base module, organizing code and resources for configuration APKs, and, optionally, adding dynamic feature modules.
 - Build an Android App Bundle using Android Studio.
 - You can also deploy your app to a connected device from an app bundle by modifying your run/debug configuration
 - Test your Android App Bundle by using it to generate APKs that you deploy to a device.
 - Enroll into app signing by Google Play. Otherwise, you can't upload your app bundle to the Play Console.
 - Publish your app bundle to Google Play.

Android App Bundle

- Format

- An Android App Bundle is a file (with the .aab file extension) that you upload to Google Play.
- App bundles are signed binaries that organize
 - app's code
 - resources into
- Code and resources for each module are organized similarly to what you would find in an APK
- Google Play uses the app bundle to generate the various APKs that are served to users.



Android App Bundle (*.aab)

BUNDLE-METADATA/

BundleConfig.pb

Base Module (base/)

assets.pb
resources.pb
native.pb

manifest/
AndroidManifest.xml

dex/
classes.dex
classes2.dex
...

root/
com/...

res/
drawable/...
drawable-hdpi/...
drawable-xhdpi/...
...

values/...
values-en/...
values-fr/...
...

layout/...
xml/...
...

assets/
textures/...
sounds/...
materials/...
shaders/...
licenses.txt

lib/
x86/native-lib.so
x86_64/...
armeabi/...
armeabi-v7a/...
arm64-v8a/...

Dynamic Feature 1 (feature1/)

[Placeholder boxes for Dynamic Feature 1 assets]

Dynamic Feature 2 (feature2/)

[Placeholder boxes for Dynamic Feature 2 assets]

Asset Pack 1 (asset_pack_1/)

manifest/
AndroidManifest.xml

assets/
textures/...
sounds/...
materials/...
shaders/...

Asset Pack 2 (asset_pack_2/)

manifest/
AndroidManifest.xml

assets/
textures/...
sounds/...
materials/...
shaders/...

Android App Bundle

- base/, feature1/, and feature2/:
 - Each of these top-level directories represent a different module of your app
 - The base module for your app is always contained in a base directory of the app bundle.
 - The directory for each dynamic feature module is given the name specified by the split attribute in the module's manifest.
- asset_pack_1/ and asset_pack_2/:
 - For large, graphically-demanding apps or games, you can modularize assets into asset packs.
 - Asset packs are ideal for games due to their large size limits.

Android App Bundle

- BUNDLE-METADATA/:
 - This directory includes metadata files that contain information useful for tools or app stores.
 - Such metadata files may include ProGuard mappings and the complete list of your app's DEX files.
 - Files in this directory are not packaged into your app's APKs.
- Module Protocol Buffer (*.pb) files:
 - These files provide metadata that helps describe the contents of each app module to app stores
 - BundleConfig.pb provides information about the bundle itself.

Android App Bundle

- manifest/:
 - app bundles store the AndroidManifest.xml file of each module in this separate directory
- dex/:
 - app bundles store the DEX files for each module in this separate directory
- res/, lib/, and assets/:
 - These directories are identical to those in a typical APK.
 - When you upload your app bundle, Google Play inspects these directories and packages only the files that satisfy the target device configuration, while preserving file paths.

Android App Bundle

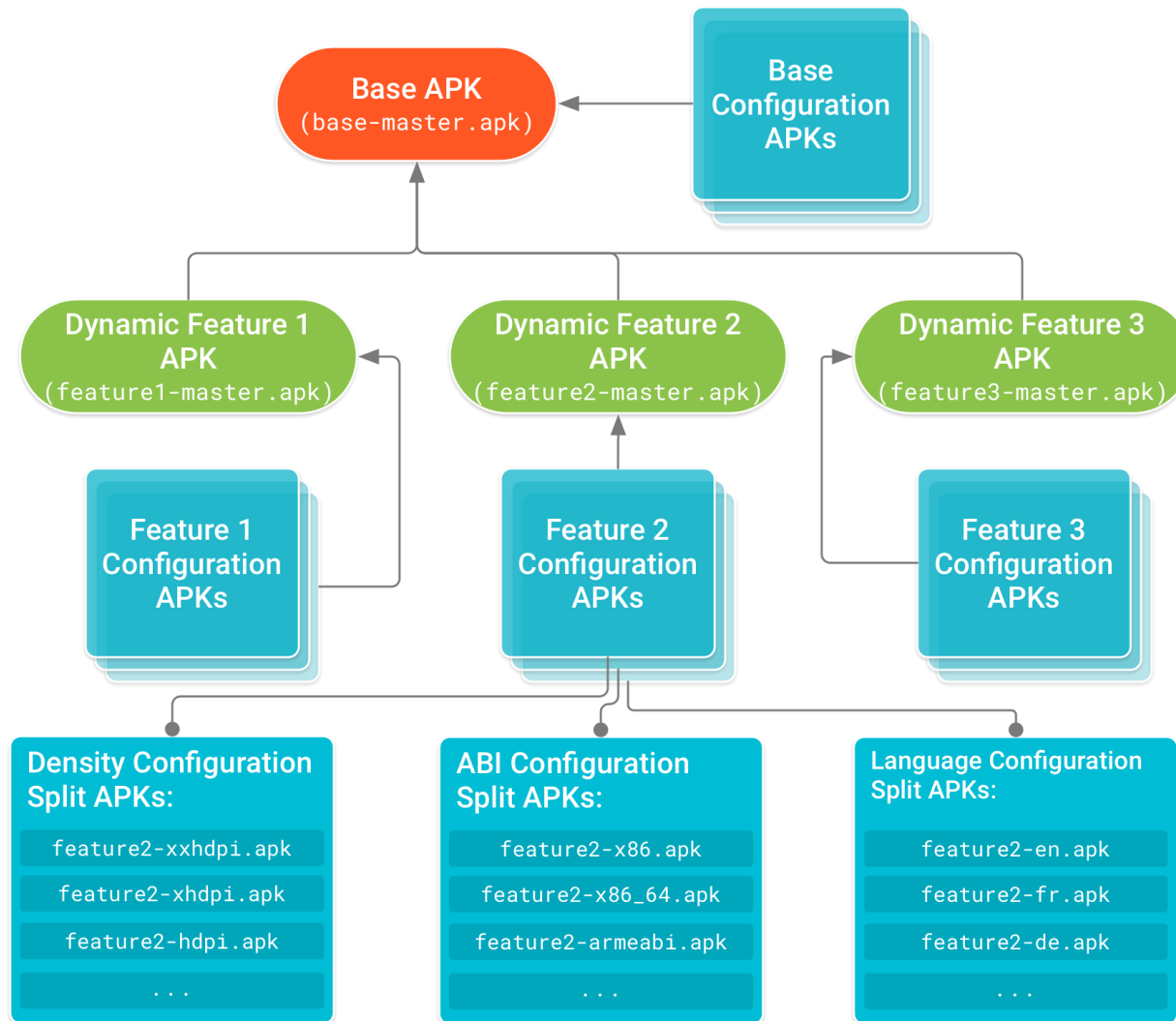
- root/:
 - This directory stores files that are later relocated to the root of any APK that includes the module that this directory is located in.
 - For example, the base/root/ directory of an app bundle may include Java-based resources that your app loads using `Class.getResource()`.
 - Those files are later relocated to the root directory of your app's base APK and every multi-APK that Google Play generates.
 - Paths within this directory are also preserved.
 - That is, directories (and their subdirectories) are also relocated to the root of the APK.

Dynamic Delivery

- Google Play's app serving model – Dynamic Delivery,
 - Uses Android App Bundles to generate and serve optimized APKs
- Most app projects won't require much effort to build app bundles that support serving optimized split APKs with Dynamic Delivery.
 - If you already organize your app's code and resources according to established conventions, simply build signed Android App Bundles using Android
- To support advanced capabilities of Dynamic Delivery, such as configuring certain features of your app to be delivered conditionally or downloaded on demand, read the section on how to customize feature delivery.

Dynamic Delivery

- split APKs
 - A fundamental component of Dynamic Delivery is the split APK mechanism
 - very similar to regular APKs - they include
 - compiled DEX bytecode
 - Resources
 - Android manifest.
 - the Android platform is able to treat multiple installed split APKs as a single app.
 - The benefit of split APKs is the ability to break up a monolithic APK into smaller, discrete packages that are installed on a user's device as required.
 - split APK may include the code and resources for an additional feature that only a few of your users need



Dynamic Delivery

- Base APK:
 - This APK contains code and resources that all other split APKs can access and provides the basic functionality for your app.
 - When a user requests to download your app, this APK is downloaded and installed first.
 - That's because only the base APK's manifest contains a full declaration of your app's services, content providers, permissions, platform version requirements, and dependencies on system features.

Dynamic Delivery

- Configuration APKs:
 - Each of these APKs includes native libraries and resources for a specific screen density, CPU architecture, or language.
 - When a user downloads your app, their device downloads and installs only the configuration APKs that target their device.
 - Each configuration APK is a dependency of either a base APK or dynamic feature APK.
 - That is, they are downloaded and installed along with the APK they provide code and resources for.
 - Unlike the base and dynamic feature modules, you don't create a separate module for configuration APKs.

Dynamic Delivery

- Dynamic feature APKs:
 - Each of these APKs contains code and resources for a feature of your app that you modularize using dynamic feature modules.
 - Through Dynamic Delivery, you can then customize how and when that feature is downloaded onto a device.
 - Consider a chat app that downloads and installs the ability to capture and send photos only when the user requests to use that functionality.
 - Because dynamic features may not be available at install time, you should include any common code and resources in the base APK

Google Analytics

- The usage of applications can be monitored
 - Crash and Exception
 - Event tracking
 - Flow visualization
 - Real-time reporting
 - Custom reports
- Can be integrated to Google Play

Setting up Google Analytics

- <https://developers.google.com/analytics/devguides/collection/android/v4>
- Gradle
 - Buildsript:
 - classpath 'com.google.gms:google-services:3.0.0'
 - Dependency:
 - compile 'com.google.android.gms:play-services-analytics:10.2.4'

Setting up Google Analytics

- Manifest

- `<uses-permission android:name="android.permission.INTERNET" />`
- `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />`

- Tracker-ID

- `<?xml version="1.0" encoding="utf-8"?>`
`<resources>`
`<string name="ga_trackingId" translatable="false">${YOUR_TRACKING_ID}</string>`
`</resources>`

- google-services.json file

- Previous link

Google Analytics – Application class

```
public class AnalyticsApplication extends Application {  
    private Tracker mTracker;  
  
    /**  
     * Gets the default {@link Tracker} for this {@link Application}.  
     * @return tracker  
     */  
    synchronized public Tracker getDefaultTracker() {  
        if (mTracker == null) {  
            GoogleAnalytics analytics = GoogleAnalytics.getInstance(this);  
            // To enable debug logging use: adb shell setprop log.tag.GAv4 DEBUG  
            mTracker = analytics.newTracker(R.xml.global_tracker);  
        }  
        return mTracker;  
    }  
}
```

Sending a Screen event

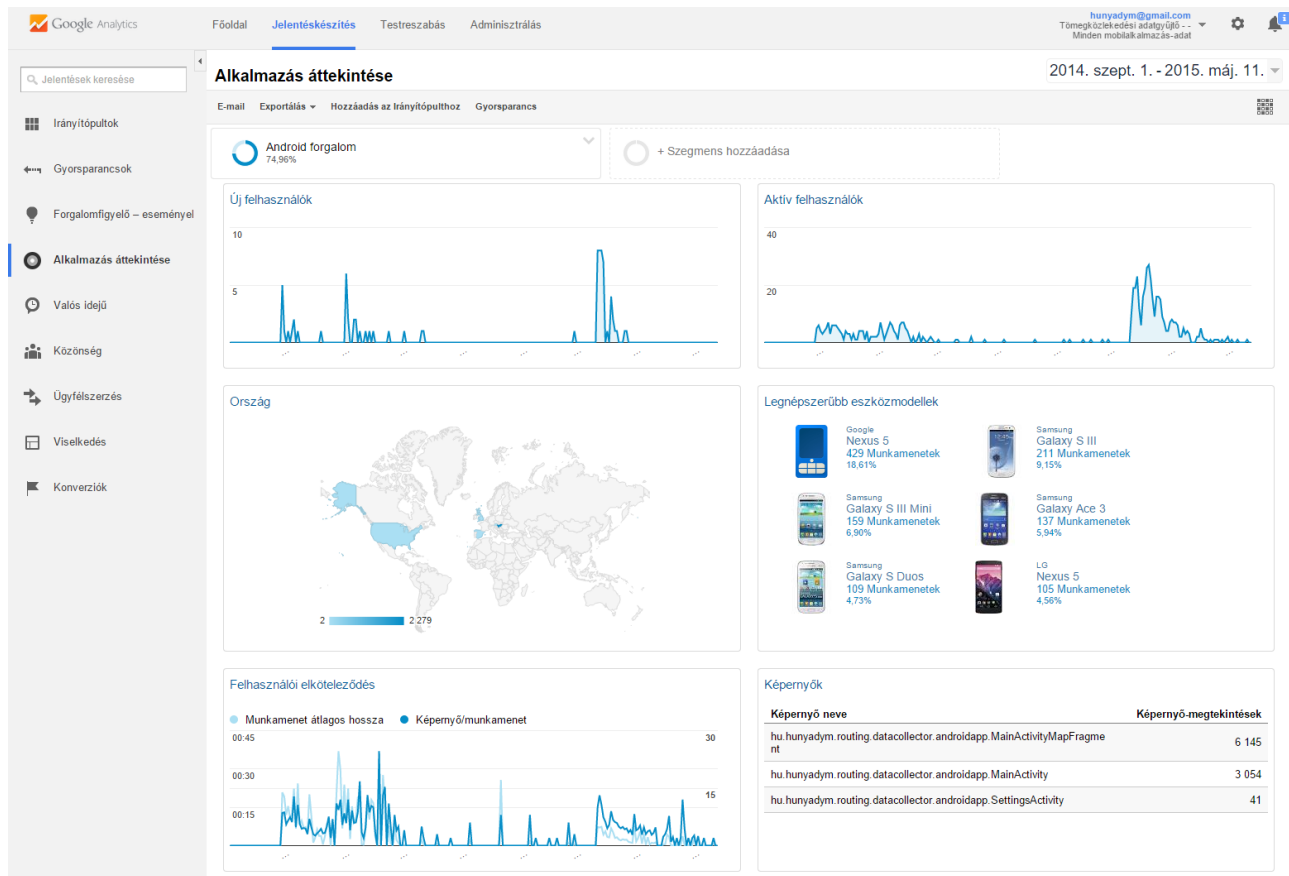
- Activity, Fragment

- *// Get tracker.*
Tracker t = ((App) getActivity()).getApplication().getTracker();

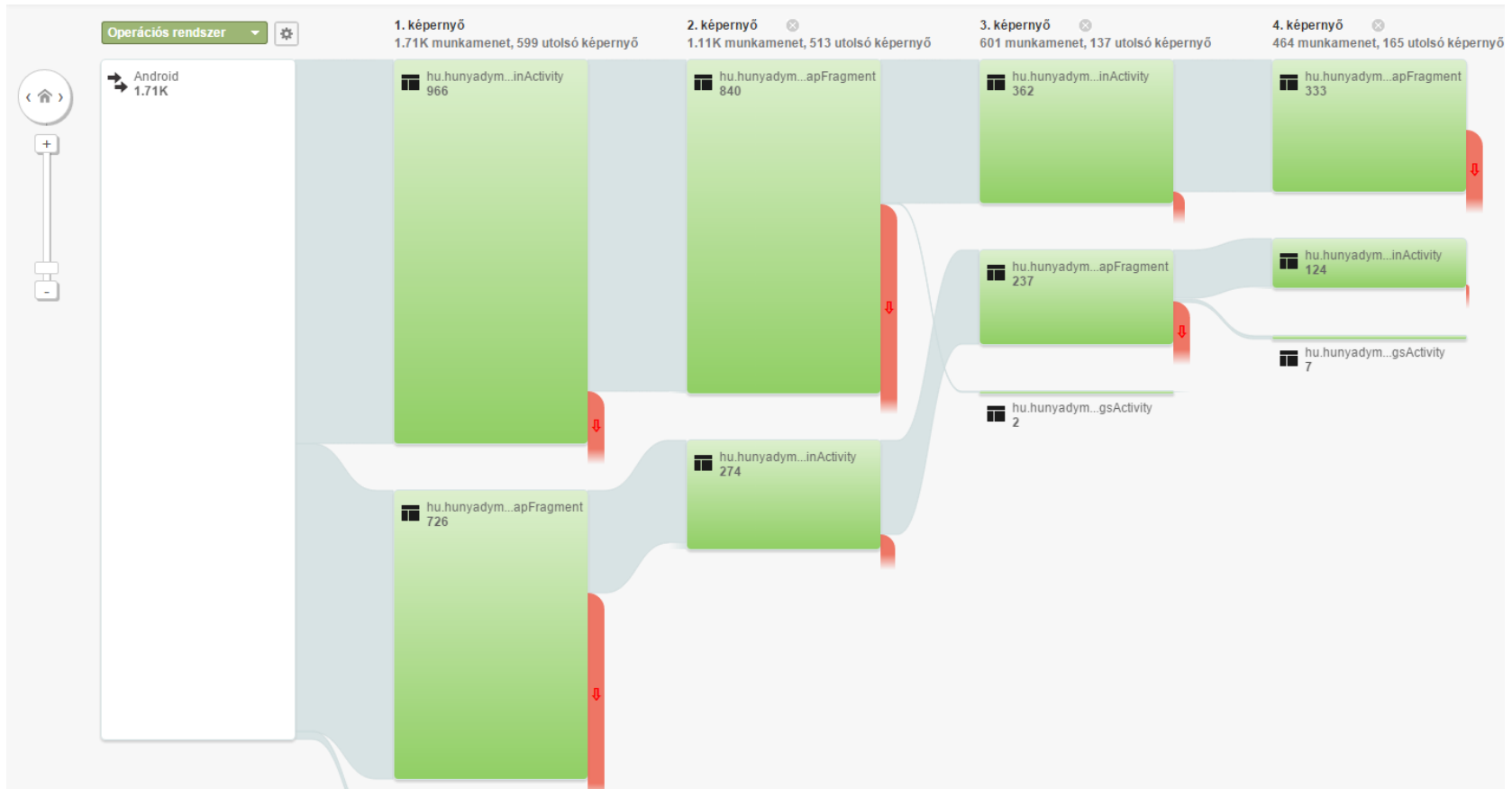
// Set screen name.
t.setScreenName(screenName);

// Send a screen view.
t.send(new HitBuilders.ScreenViewBuilder().build());

Google Analytics dashboard



Google Analytics Activity transitions



Google Analytics Activity transitions



Google Analytics crash reports

<input type="checkbox"/>	Kivétel leírása ?	Operációs rendszer verziója ?	Összeomlások ? ↓
	Android forgalom		68 % a teljesből: 100,00% (68)
<input type="checkbox"/>	1. NullPointerException (@PhoneWindow.onKeyUpPanel:1004) {main}	4.1.2	8 (11,76%)
<input type="checkbox"/>	2. IllegalArgumentException (@a:<init>:-1) {AsyncTask #1}	5.0.1	4 (5,88%)
<input type="checkbox"/>	3. IllegalStateException (@Activity.onBackPressed:260) {main}	4.4.4	4 (5,88%)
<input type="checkbox"/>	4. IllegalStateException (@MasterDetailScreenFragment.openDetail:43) {main}	4.1.1	4 (5,88%)
<input type="checkbox"/>	5. NullPointerException (@ToolbarPreference\$1.onClick:33) {main}	4.4.4	4 (5,88%)
<input type="checkbox"/>	6. NullPointerException (@MusicPlayService.prepareDirectory:193) {main}	2.3.7	3 (4,41%)
<input type="checkbox"/>	7. StackOverflowError (@SyncEvent:<init>:25) {AsyncTask #1}	5.1	3 (4,41%)
<input type="checkbox"/>	8. AssertionError (@a:<init>:-1) {AsyncTask #1}	5.0.1	2 (2,94%)
<input type="checkbox"/>	9. ClassCastException (@Activity.checkLastSyncTime:331) {main}	4.4.4	2 (2,94%)
<input type="checkbox"/>	10. ClassNotFoundException (@BaseDexClassLoader.findClass:56) {main}	5.0.1	2 (2,94%)



Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Android System

Next week