Refresh, renew, refactor

Modernising a large Android app with many users

Andrey Liashuk
Android Engineer
andrey-liashuk-607602164
AndrewLiashuk

Maia Grotepass
Android Engineer
@maiatoday
maiatoday

LUNO

Luno app

Intro and stats*

Luno makes it **safe and easy** to buy, store and learn about cryptocurrencies

~1.35M active Android users

40 countries

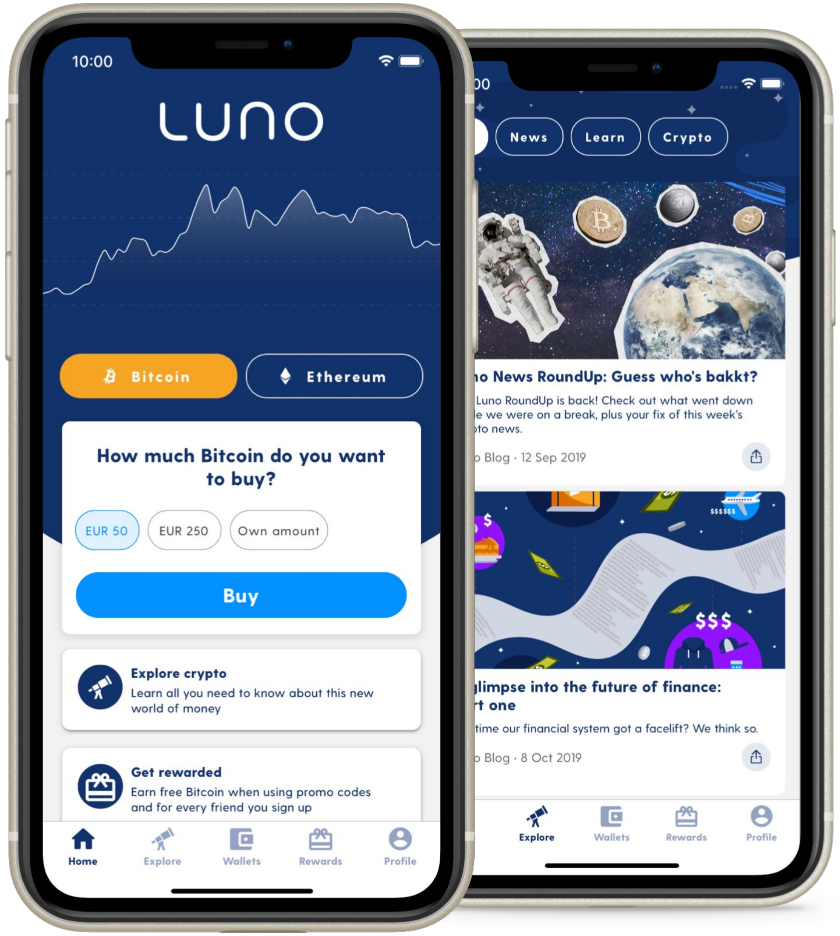
First commit Friday **25 July 2014**

329 fragment layouts

8918 lines of **Java** code

250912 lines of **Kotlin** code

51941 lines of **xml**


10:00
LUNO
Bitcoin
Ethereum
How much Bitcoin do you want to buy?
EUR 50
EUR 250
Own amount
Buy
Explore crypto
Learn all you need to know about this new world of money
Get rewarded
Earn free Bitcoin when using promo codes and for every friend you sign up
Home
Explore
Wallets
Rewards
Profile
News
Learn
Crypto
o News RoundUp: Guess who's bakkt?
Luno RoundUp is back! Check out what went down e we were on a break, plus your fix of this week's to news.
Blog · 12 Sep 2019
limpse into the future of finance: t one
time our financial system got a facelift? We think so.
Blog · 8 Oct 2019
Explore
Wallets
Rewards
Profile

* as of 1 Oct 2020

Past

Java
MVVM no Jetpack/Burrito🌯
Activity + Fragments
Asynctask
Json REST + Retrofit
Eventbus

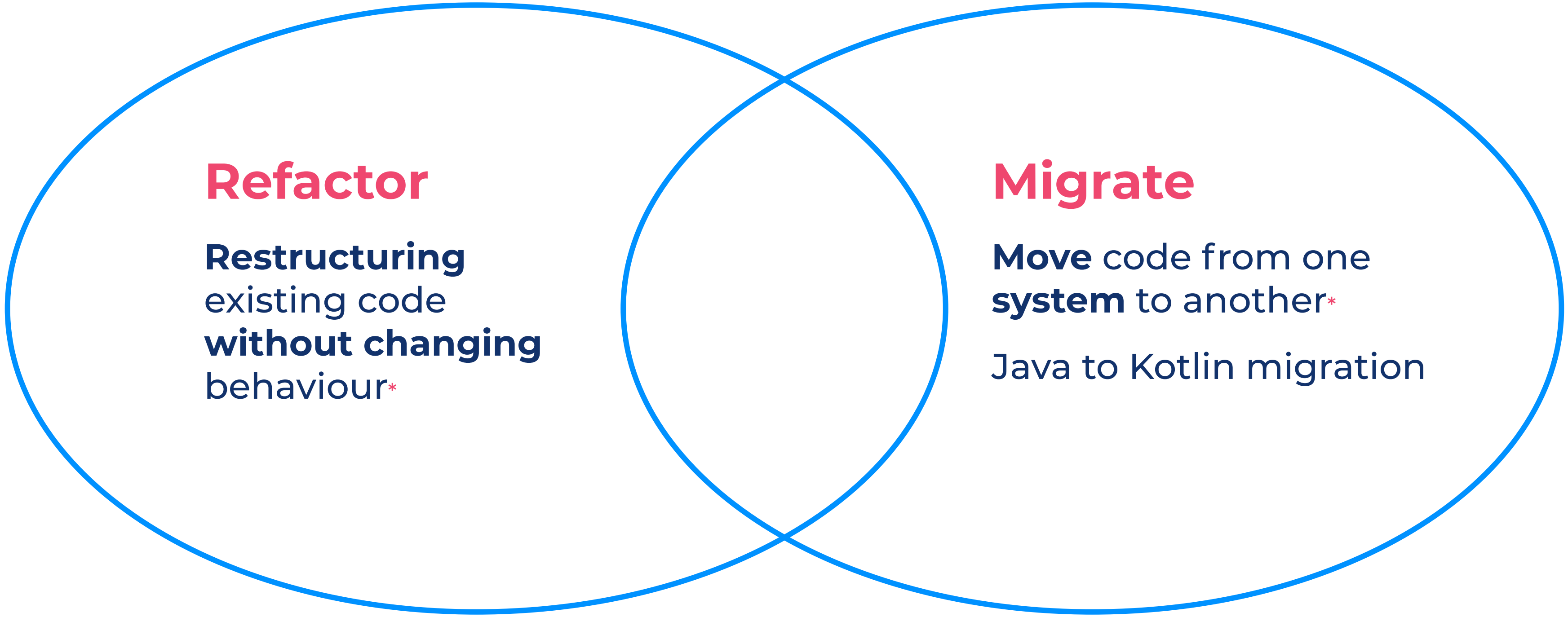
tests?

Future

Kotlin
MVVM
Activity + Fragments
Coroutines
Protobuf REST + Retrofit

Tests! 🤩

Refactor / Migrate



Refactor

Restructuring existing code **without changing** behaviour*

Migrate

Move code from one **system** to another*

Java to Kotlin migration

* paraphrased from Wikipedia

Why?

Agenda

~~Overview~~

Andrey

- Kotlin migration
- Migration tips
- Event bus and coroutines
- Protobuf migration

Maia

- Architecture
- Team
- CI / CD
- What's next

How to start migration?

1. **Prepare product request**

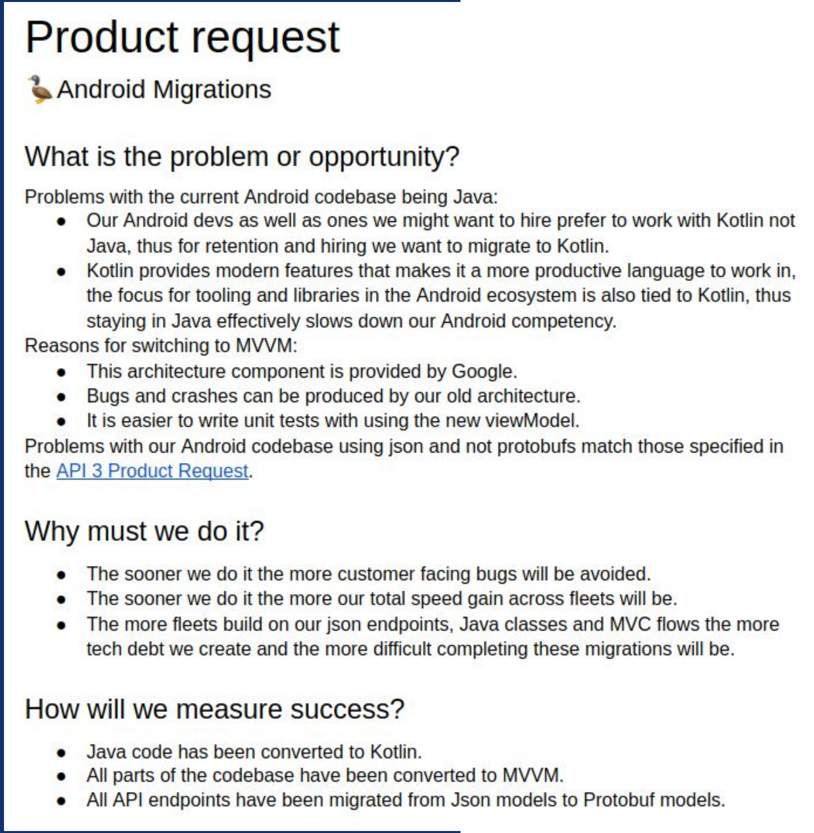
   Product request is a document describing the problems, goal and measure of success.

   Don't convert everything to Kotlin

   ⛔ Legacy classes

   ⛔ Classes to be deleted

   ⛔ Complicated third party classes



Product request

🦤Android Migrations

What is the problem or opportunity?

Problems with the current Android codebase being Java:

- Our Android devs as well as ones we might want to hire prefer to work with Kotlin not Java, thus for retention and hiring we want to migrate to Kotlin.
- Kotlin provides modern features that makes it a more productive language to work in, the focus for tooling and libraries in the Android ecosystem is also tied to Kotlin, thus staying in Java effectively slows down our Android competency.

Reasons for switching to MVVM:

- This architecture component is provided by Google.
- Bugs and crashes can be produced by our old architecture.
- It is easier to write unit tests with using the new viewModel.

Problems with our Android codebase using json and not protobufs match those specified in the API 3 Product Request.

Why must we do it?

- The sooner we do it the more customer facing bugs will be avoided.
- The sooner we do it the more our total speed gain across fleets will be.
- The more fleets build on our json endpoints, Java classes and MVC flows the more tech debt we create and the more difficult completing these migrations will be.

How will we measure success?

- Java code has been converted to Kotlin.
- All parts of the codebase have been converted to MVVM.
- All API endpoints have been migrated from Json models to Protobuf models.

2. Define champion

Champion

a person **responsible:**

- know the **status**
- track **progress**

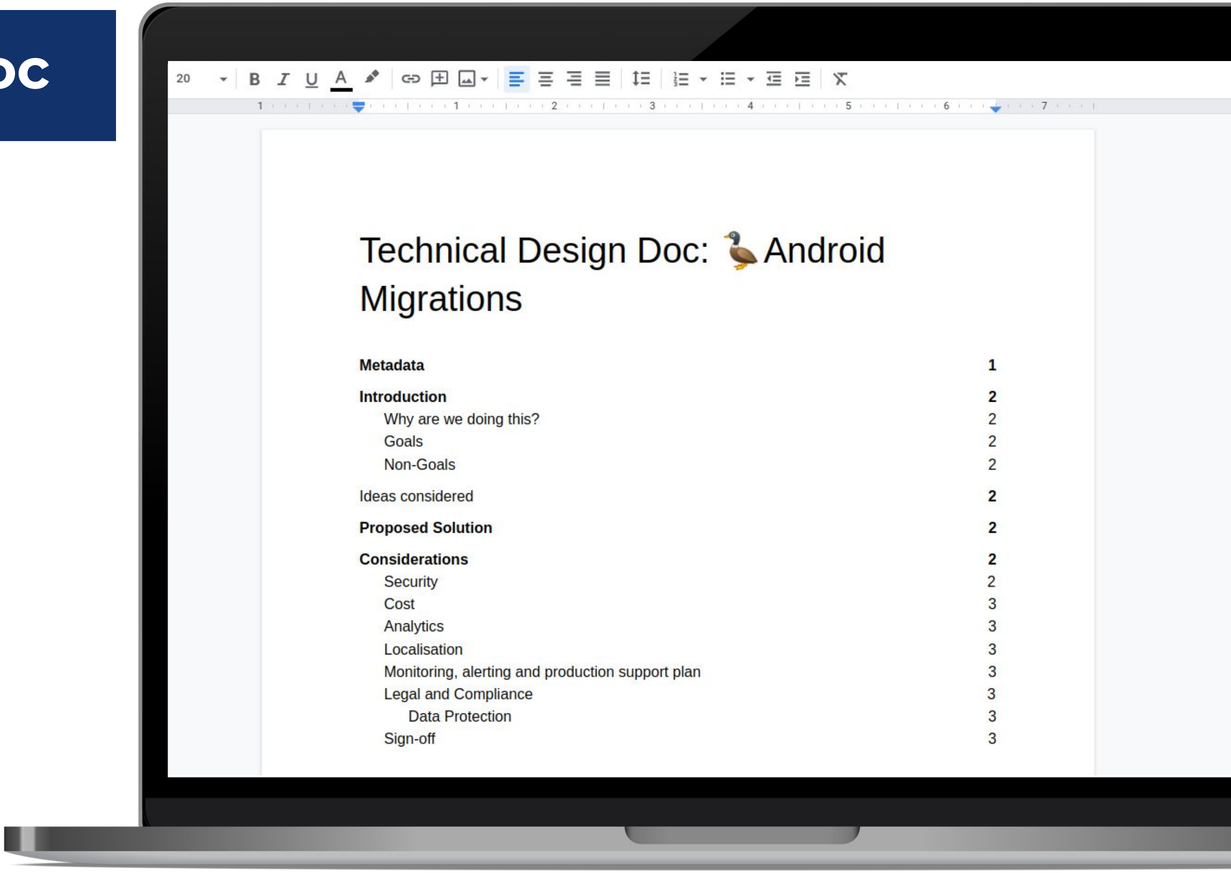
Champion is **not alone**, tasks can be created and divided in the team.

LUNO

3. Create a Design Doc

What is a design document?

- describes a **problem and solutions**
- free form or template
- **scope of work** as complete as possible
- used for **time estimate**



Technical Design Doc: 🦆 Android Migrations

4. Burden-sharing

The migration will finish faster if the whole team takes part in it. 👻

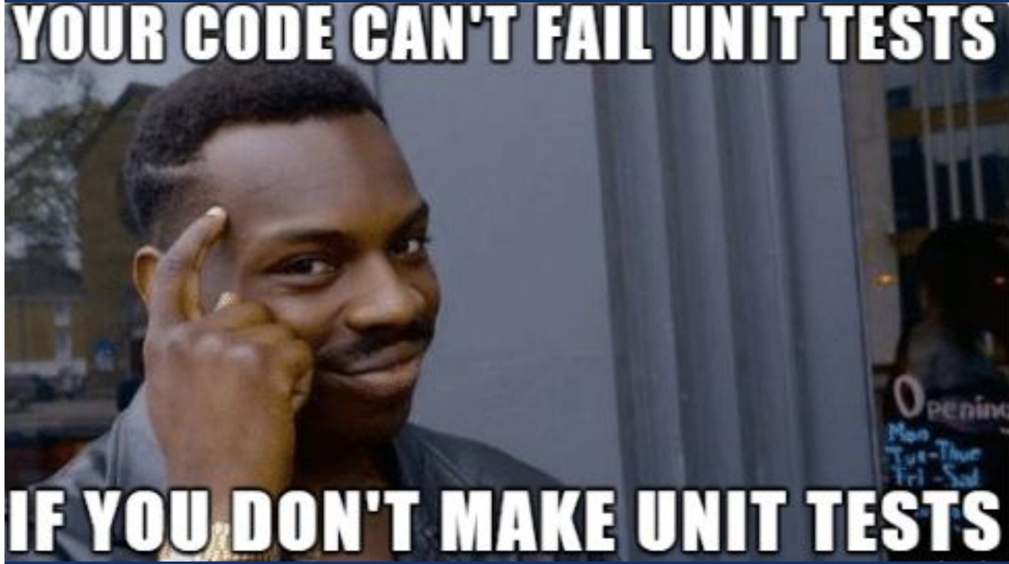
LUNO

Kotlin Conversion Tips

LUNO

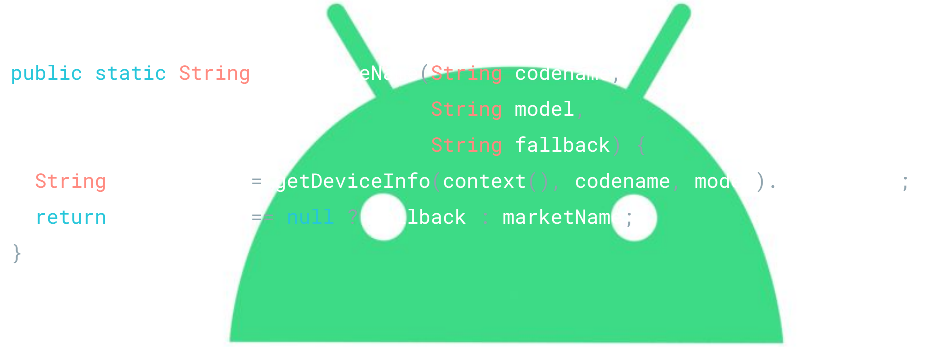
Do refactoring

Create unit tests



Don't trust converter

Always test changes



```
public static String          eN  (String codenam ,
                                   String model,
                                   String fallback) {
  String            = getDeviceInfo(context(), codename, mod ).          ;
  return            == null ?   llback : marketNam ;
}
```

If you are tired, just rest

Ask for review

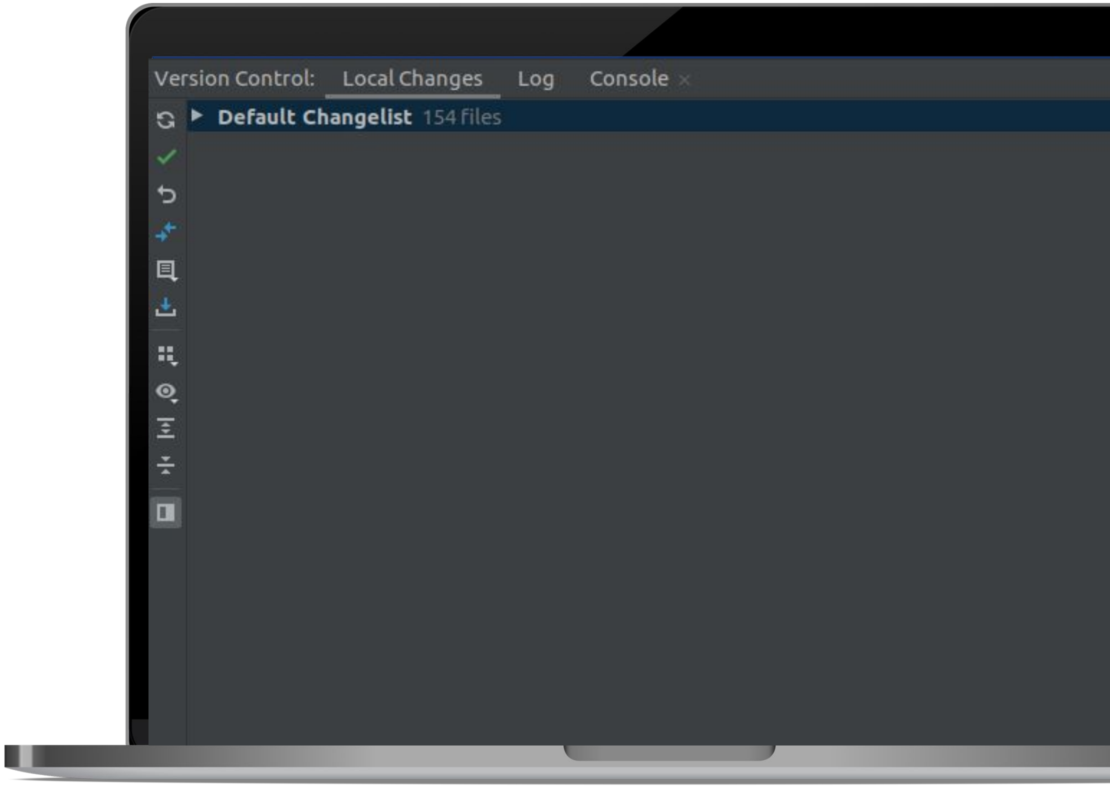
```
integer [illegible]idAlgorithm (int A, int
B){
    A=Math.abs(A);
    B=Math.abs(B);
    while (B!=0){
        if (A>B) A=A-B;
        else B=B-A;
    }
    return A;
}

integer euclidAlgorithm (int A, int
B){
    A=Math.abs(A);
    B=Math.abs(B);
    while (B!=0){
        if (A>B) A=A-B;
        else B=B-A;
    }
    return A;
}

integer euclidAlgorithm (int A, int
B){
    A=Math.abs(A);
    B=Math.abs(B);
    while (B!=0){
        if (A>B) A=A-B;
        else B=B-A;
    }
    return A;
}
```

Review process

Don't try to change the world in one pull request

☝️ Stick to the rule: **one** request solves **one** problem.

It is better to create requests **as small as possible!**



LUNO

What is a small request?

How to understand how big the diff is?

🤔 based on developer tastes.

My rules:

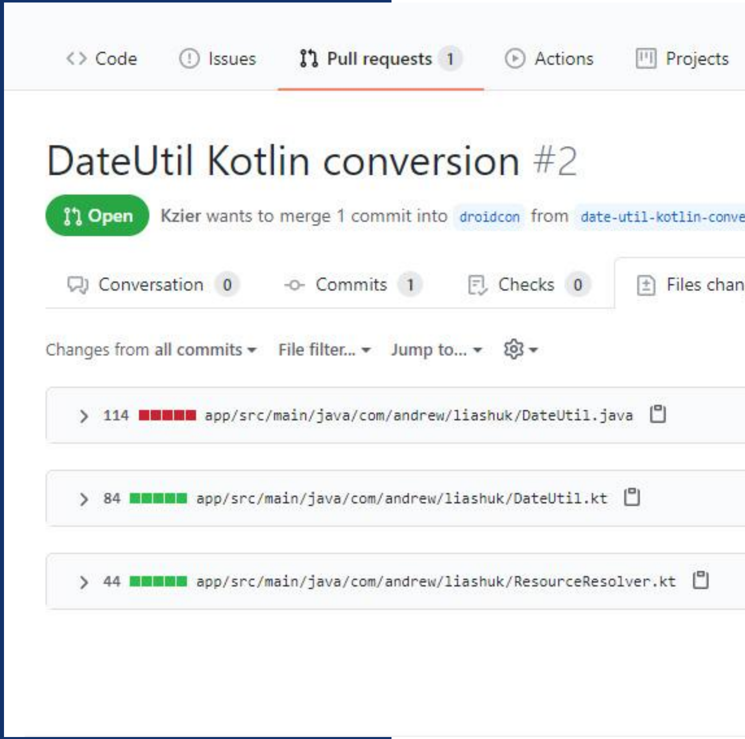
- If class size <= **100 lines** or small data class

  3-5 classes per request.

- If class contains complex logic or class size > **300 lines**

  separate request.

- 2-3 classes per request



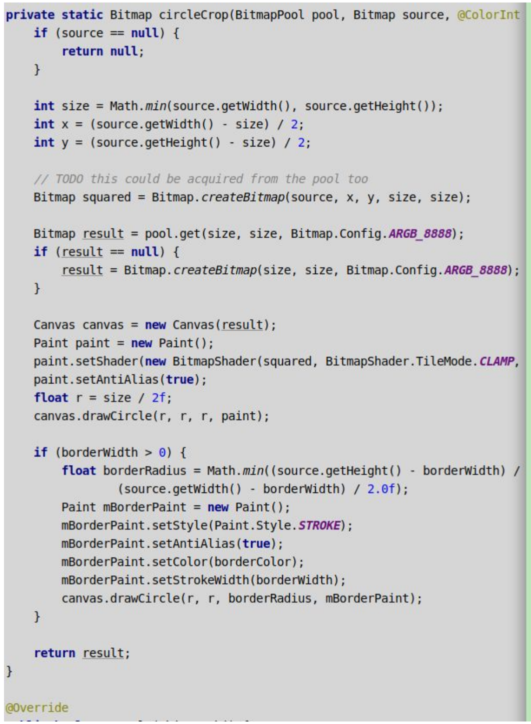
DateUtil Kotlin conversion #2

Kzier wants to merge 1 commit into droidcon from date-util-kotlin-conve

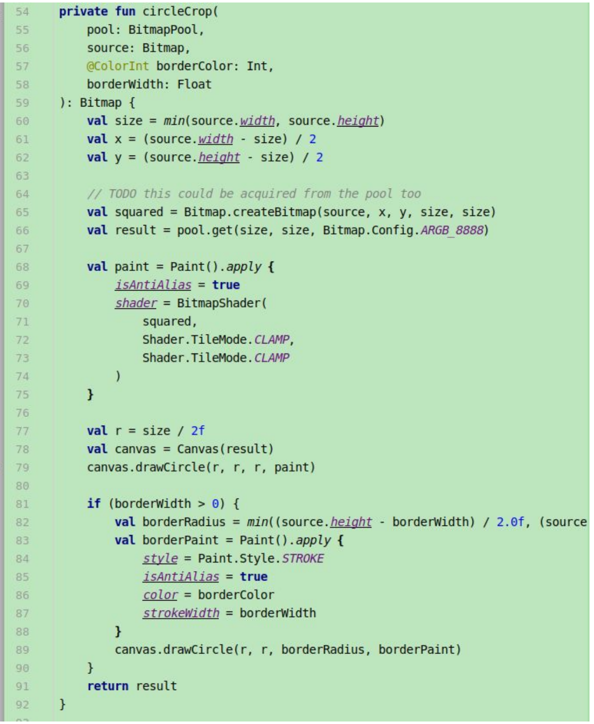
114 🟥🟥🟥🟥🟥 app/src/main/java/com/andrew/liashuk/DateUtil.java

84 🟩🟩🟩🟩🟩 app/src/main/java/com/andrew/liashuk/DateUtil.kt

44 🟩🟩🟩🟩🟩 app/src/main/java/com/andrew/liashuk/ResourceResolver.kt



```java
private static Bitmap circleCrop(BitmapPool pool, Bitmap source, @ColorInt
    if (source == null) {
        return null;
    }

    int size = Math.min(source.getWidth(), source.getHeight());
    int x = (source.getWidth() - size) / 2;
    int y = (source.getHeight() - size) / 2;

    // TODO this could be acquired from the pool too
    Bitmap squared = Bitmap.createBitmap(source, x, y, size, size);

    Bitmap result = pool.get(size, size, Bitmap.Config.ARGB_8888);
    if (result == null) {
        result = Bitmap.createBitmap(size, size, Bitmap.Config.ARGB_8888);
    }

    Canvas canvas = new Canvas(result);
    Paint paint = new Paint();
    paint.setShader(new BitmapShader(squared, BitmapShader.TileMode.CLAMP,
    paint.setAntiAlias(true);
    float r = size / 2f;
    canvas.drawCircle(r, r, r, paint);

    if (borderWidth > 0) {
        float borderRadius = Math.min((source.getHeight() - borderWidth) /
                (source.getWidth() - borderWidth) / 2.0f);
        Paint mBorderPaint = new Paint();
        mBorderPaint.setStyle(Paint.Style.STROKE);
        mBorderPaint.setAntiAlias(true);
        mBorderPaint.setColor(borderColor);
        mBorderPaint.setStrokeWidth(borderWidth);
        canvas.drawCircle(r, r, borderRadius, mBorderPaint);
    }

    return result;
}

@Override
```



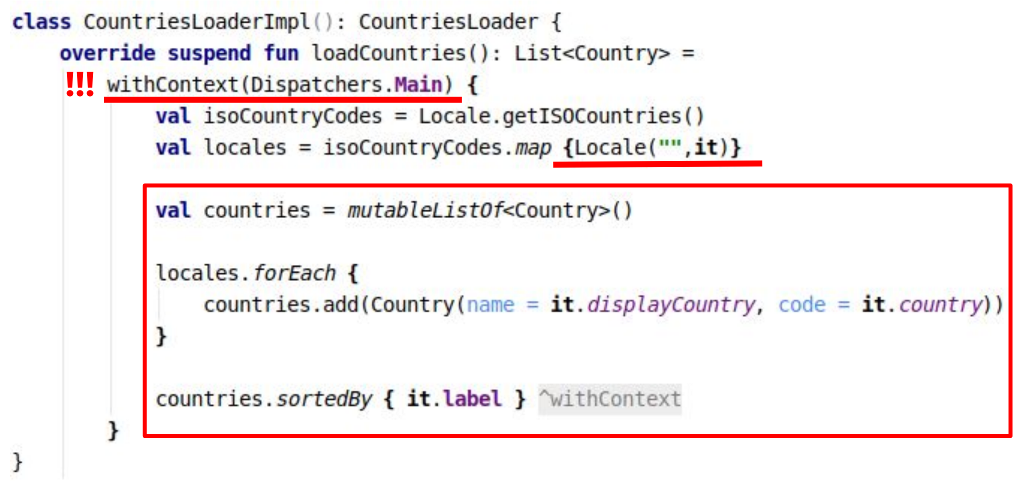
```kotlin
private fun circleCrop(
    pool: BitmapPool,
    source: Bitmap,
    @ColorInt borderColor: Int,
    borderWidth: Float
): Bitmap {
    val size = min(source.width, source.height)
    val x = (source.width - size) / 2
    val y = (source.height - size) / 2

    // TODO this could be acquired from the pool too
    val squared = Bitmap.createBitmap(source, x, y, size, size)
    val result = pool.get(size, size, Bitmap.Config.ARGB_8888)

    val paint = Paint().apply {
        isAntiAlias = true
        shader = BitmapShader(
            squared,
            Shader.TileMode.CLAMP,
            Shader.TileMode.CLAMP
        )
    }

    val r = size / 2f
    val canvas = Canvas(result)
    canvas.drawCircle(r, r, r, paint)

    if (borderWidth > 0) {
        val borderRadius = min((source.height - borderWidth) / 2.0f, (source
        val borderPaint = Paint().apply {
            style = Paint.Style.STROKE
            isAntiAlias = true
            color = borderColor
            strokeWidth = borderWidth
        }
        canvas.drawCircle(r, r, borderRadius, borderPaint)
    }
    return result
}
```

Split different tasks into few requests

📱 Kotlin conversion

📱 Refactor + unit tests

Always check request before requesting review



```
class CountriesLoaderImpl(): CountriesLoader {
    override suspend fun loadCountries(): List<Country> =
    !!! withContext(Dispatchers.Main) {
            val isoCountryCodes = Locale.getISOCountries()
            val locales = isoCountryCodes.map {Locale("",it)}

            val countries = mutableListOf<Country>()

            locales.forEach {
                countries.add(Country(name = it.displayCountry, code = it.country))
            }

            countries.sortedBy { it.label } ^withContext
        }
}
```

Help your reviewer:

- ✓ to focus on serious errors
- ✓ not be distracted by minor things that you could have found if you checked

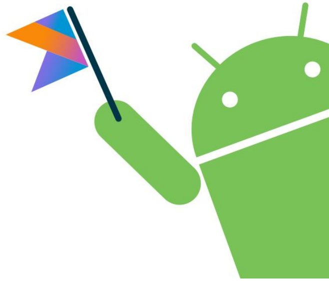
LUNO

Don't be afraid of mistakes

Do not be upset, there are no mistakes only for those who do nothing

LUNO

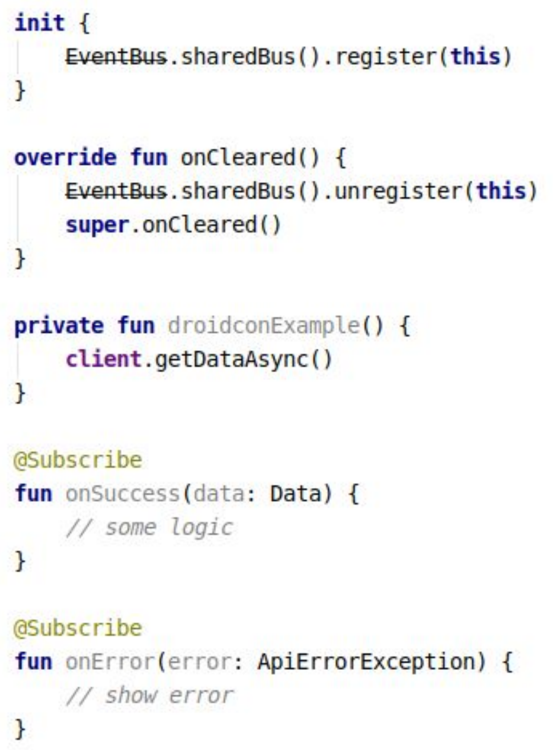
Kotlin conversion



- 🔥 The number of null pointer exception has been reduced to almost zero.
- 🔥 Latest features such as coroutines can be used now.
- 🔥 The average file size has decreased by 21%.

LUNO

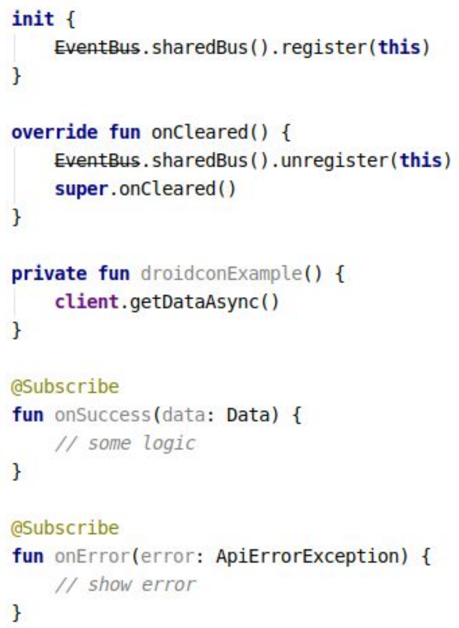
Event bus to coroutines migration



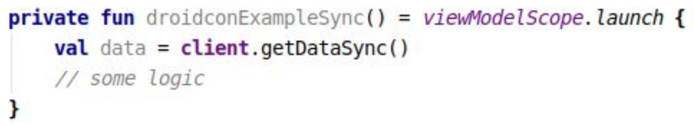
```
init {
    ~~EventBus~~.sharedBus().register(this)
}

override fun onCleared() {
    ~~EventBus~~.sharedBus().unregister(this)
    super.onCleared()
}

private fun droidconExample() {
    client.getDataAsync()
}

@Subscribe
fun onSuccess(data: Data) {
    // some logic
}

@Subscribe
fun onError(error: ApiErrorException) {
    // show error
}
```

Eventbus disadvantages:

- Very complicated debug process
- High probability of error
- Changing in one place, can break application in an unknown place
- Difficult to read code



```kotlin
init {
    ~~EventBus~~.sharedBus().register(this)
}

override fun onCleared() {
    ~~EventBus~~.sharedBus().unregister(this)
    super.onCleared()
}

private fun droidconExample() {
    client.getDataAsync()
}

@Subscribe
fun onSuccess(data: Data) {
    // some logic
}

@Subscribe
fun onError(error: ApiErrorException) {
    // show error
}
```



```kotlin
private fun droidconExampleSync() = viewModelScope.launch {
    val data = client.getDataSync()
    // some logic
}
```

Coroutines allow us to write almost synchronous code

- Easy to read
- Easy to debug
- Easy to test

New developers take less time to adapt

Protocol Buffers migration

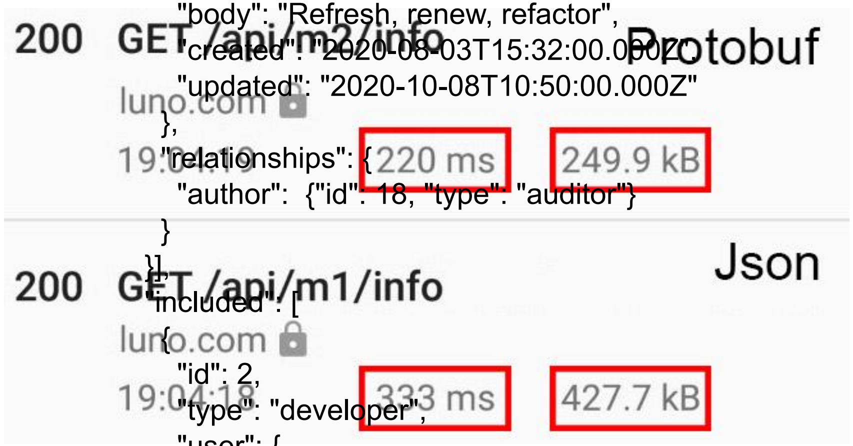
Protocol buffers are Google's **extensible** mechanism for serializing structured data.

Reduce the transferred data size and speed up requests.

Shared data models, defined **once**

LUNO

```
{
  "data": [{
    "type": "article",
    "id": 3,
    "attributes": {
      "title": "Hello droidcon!",
      "body": "Refresh, renew, refactor",
      "created": "2020-08-03T15:32:00.000Z",
      "updated": "2020-10-08T10:50:00.000Z"
    },
    "relationships": {
      "author":  {"id": 18, "type": "auditor"}
    }
  }],
  "included": [
    {
      "id": 2,
      "type": "developer",
      "user": {
        "name": "Andrey",
        "age": 24,
        "gender": "male"  }
    }
  ]
}
```



```
200   GET /api/m2/info                         Protobuf
      luno.com
      19:04:19        220 ms        249.9 kB

200   GET /api/m1/info                             Json
      luno.com
      19:04:18        333 ms        427.7 kB
```

360 bytes - Json

50% faster

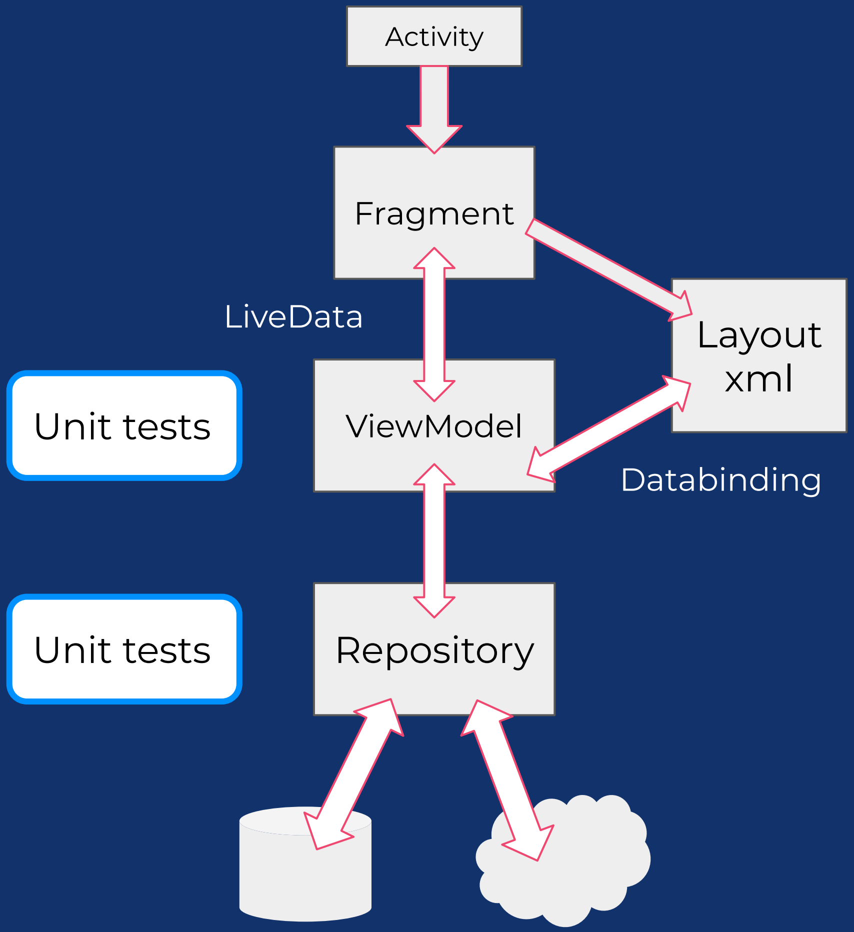
158 bytes - Protobuf

70% smaller

56% smaller

Architecture

For testing



Classic MVVM

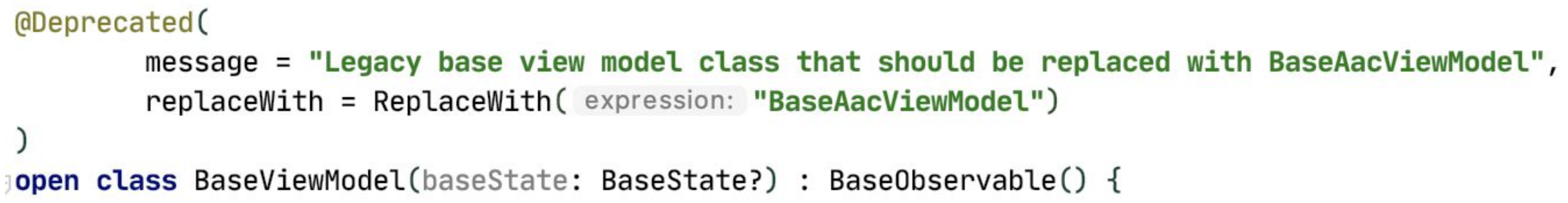
Databinding

Unit tests

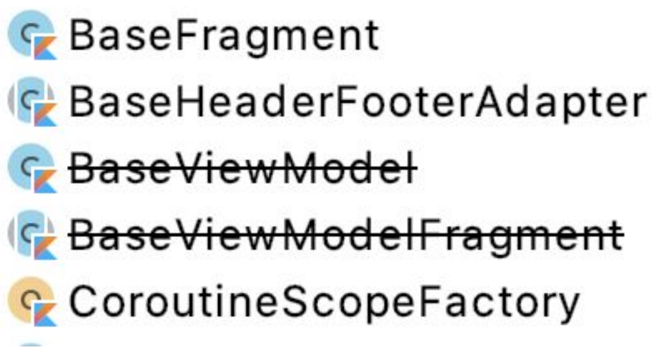
Repository

`@Deprecated`

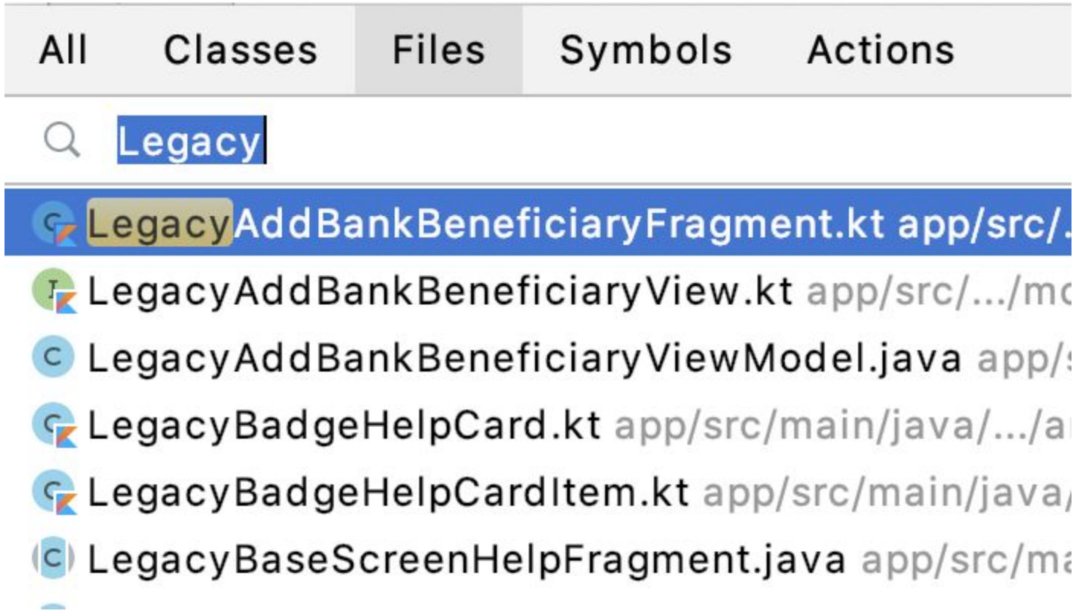
is your friend



```
@Deprecated(
        message = "Legacy base view model class that should be replaced with BaseAacViewModel",
        replaceWith = ReplaceWith( expression: "BaseAacViewModel")
)
open class BaseViewModel(baseState: BaseState?) : BaseObservable() {
```



BaseFragment
BaseHeaderFooterAdapter
~~BaseViewModel~~
~~BaseViewModelFragment~~
CoroutineScopeFactory



All Classes Files Symbols Actions

Legacy

LegacyAddBankBeneficiaryFragment.kt app/src/.

LegacyAddBankBeneficiaryView.kt app/src/.../mo

LegacyAddBankBeneficiaryViewModel.java app/s

LegacyBadgeHelpCard.kt app/src/main/java/.../a

LegacyBadgeHelpCardItem.kt app/src/main/java/

LegacyBaseScreenHelpFragment.java app/src/ma

Legacy code

1. Rename
2. Feature flag
3. Copy
4. Refactor
5. Tests

Naming/folder convention

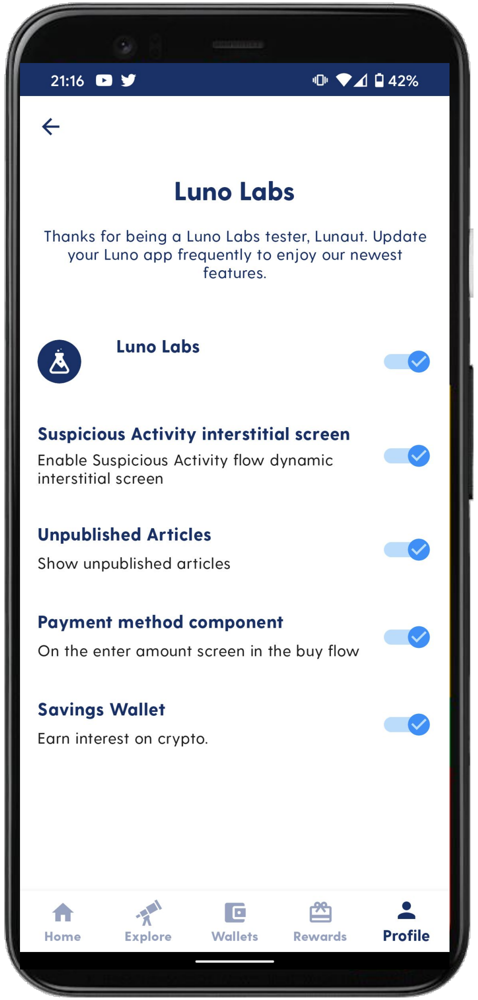
Feature flags

Backend controlled

- Staging/Production
- Lunauts
- By App version
- iOS/Android/Web

Beta User - Opt in

User - Opt in



21:16 42%

Luno Labs

Thanks for being a Luno Labs tester, Lunaut. Update your Luno app frequently to enjoy our newest features.

Luno Labs ✓

Suspicious Activity interstitial screen ✓
Enable Suspicious Activity flow dynamic interstitial screen

Unpublished Articles ✓
Show unpublished articles

Payment method component ✓
On the enter amount screen in the buy flow

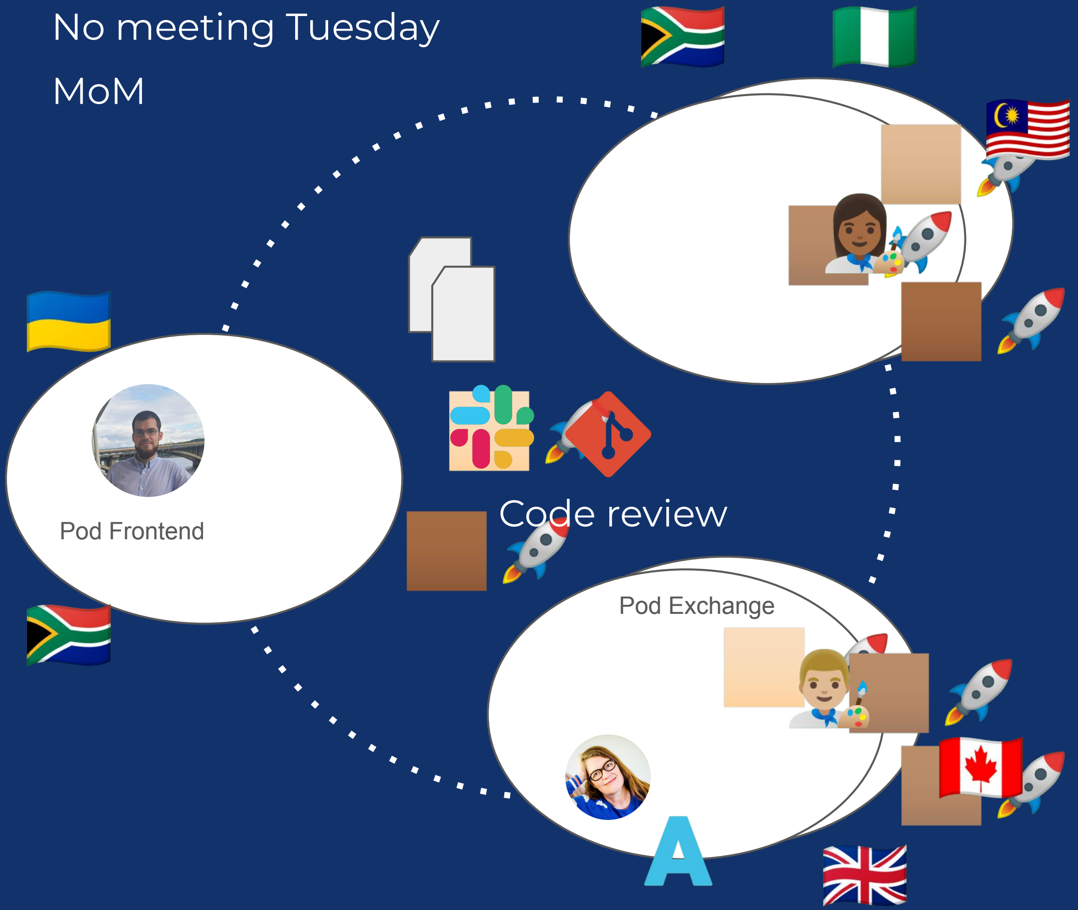
Savings Wallet ✓
Earn interest on crypto.

LUNO

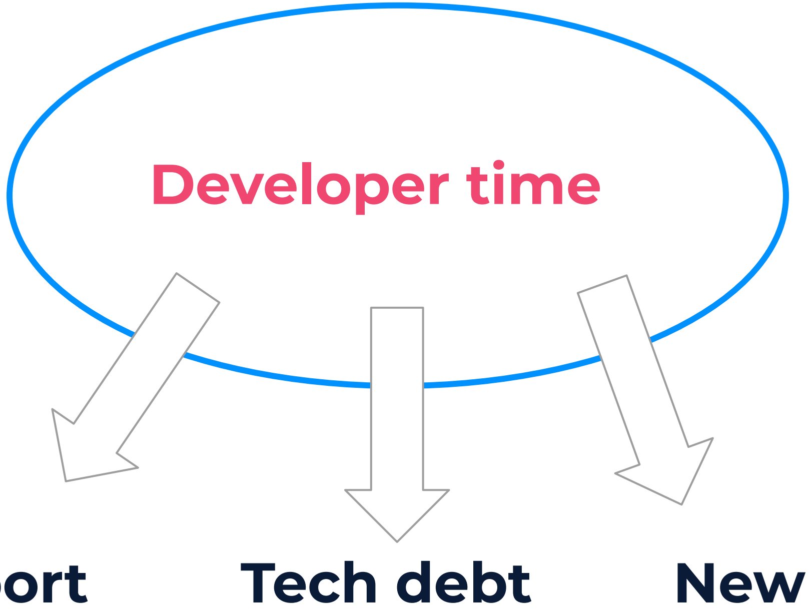
One mission, one team

One team with **champions** creates **guidelines** and **examples**

Feature teams implement in their area



Balance



20%?

“ Set an expectation that any **changes** to **existing functionality requires** a **refactor** to new architecture.

New features are implemented in the **new way**. ”

Charles Okot

“ Refactoring is like **brushing your teeth** -

A little bit every day is better than

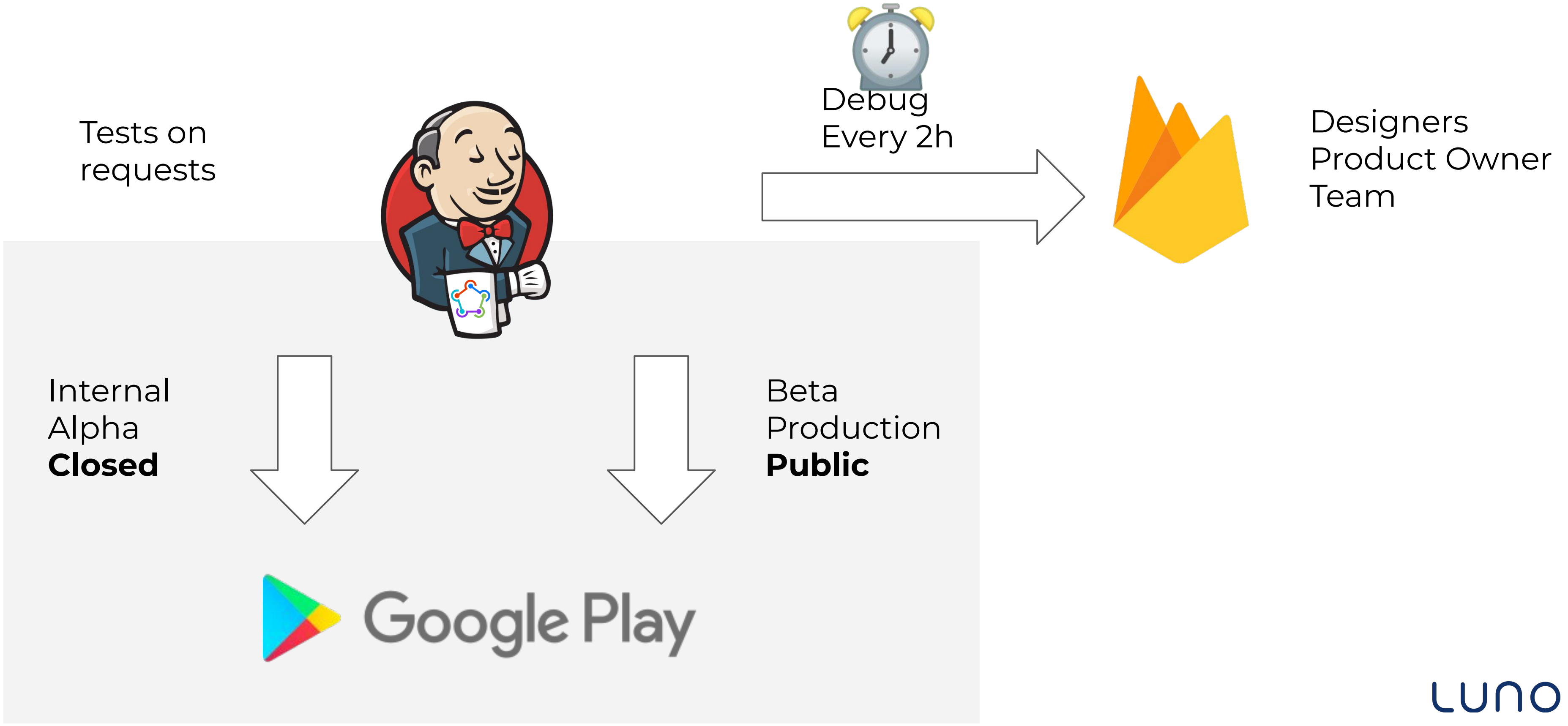
2 hours once a month. ”

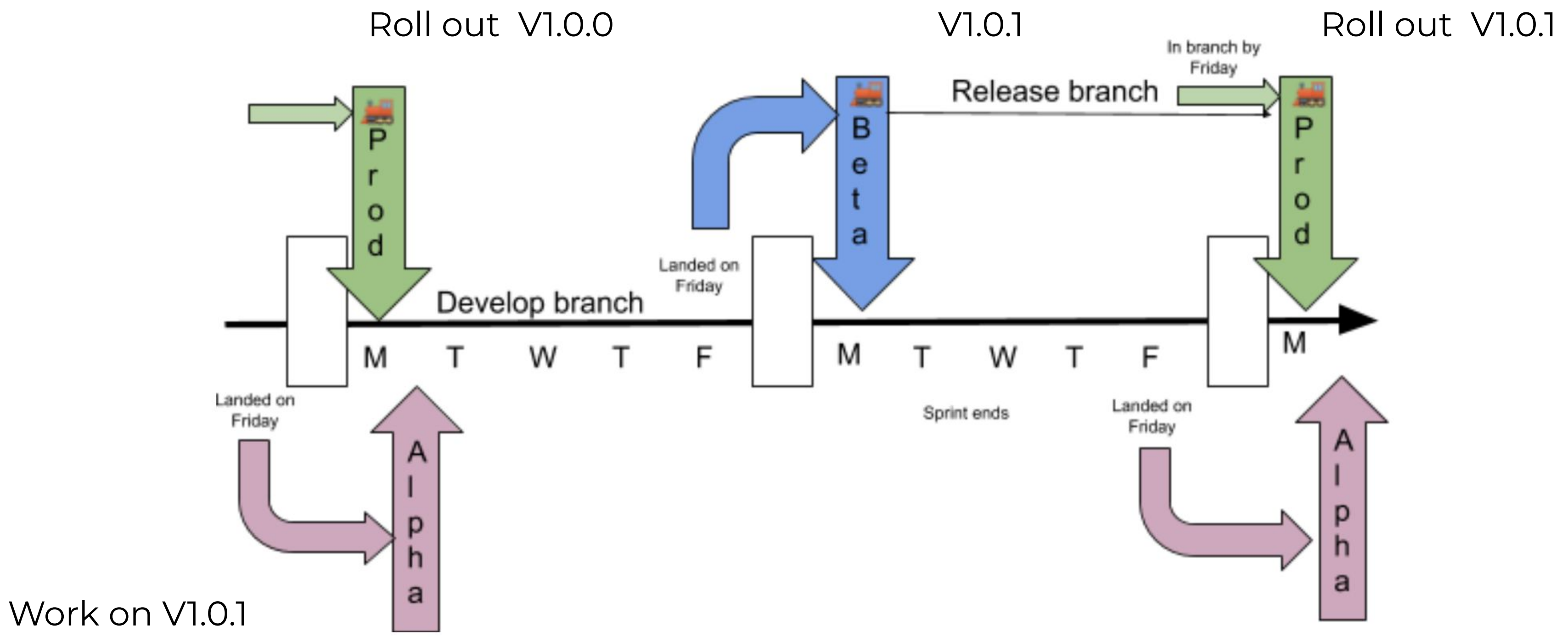
Maia Grotepass

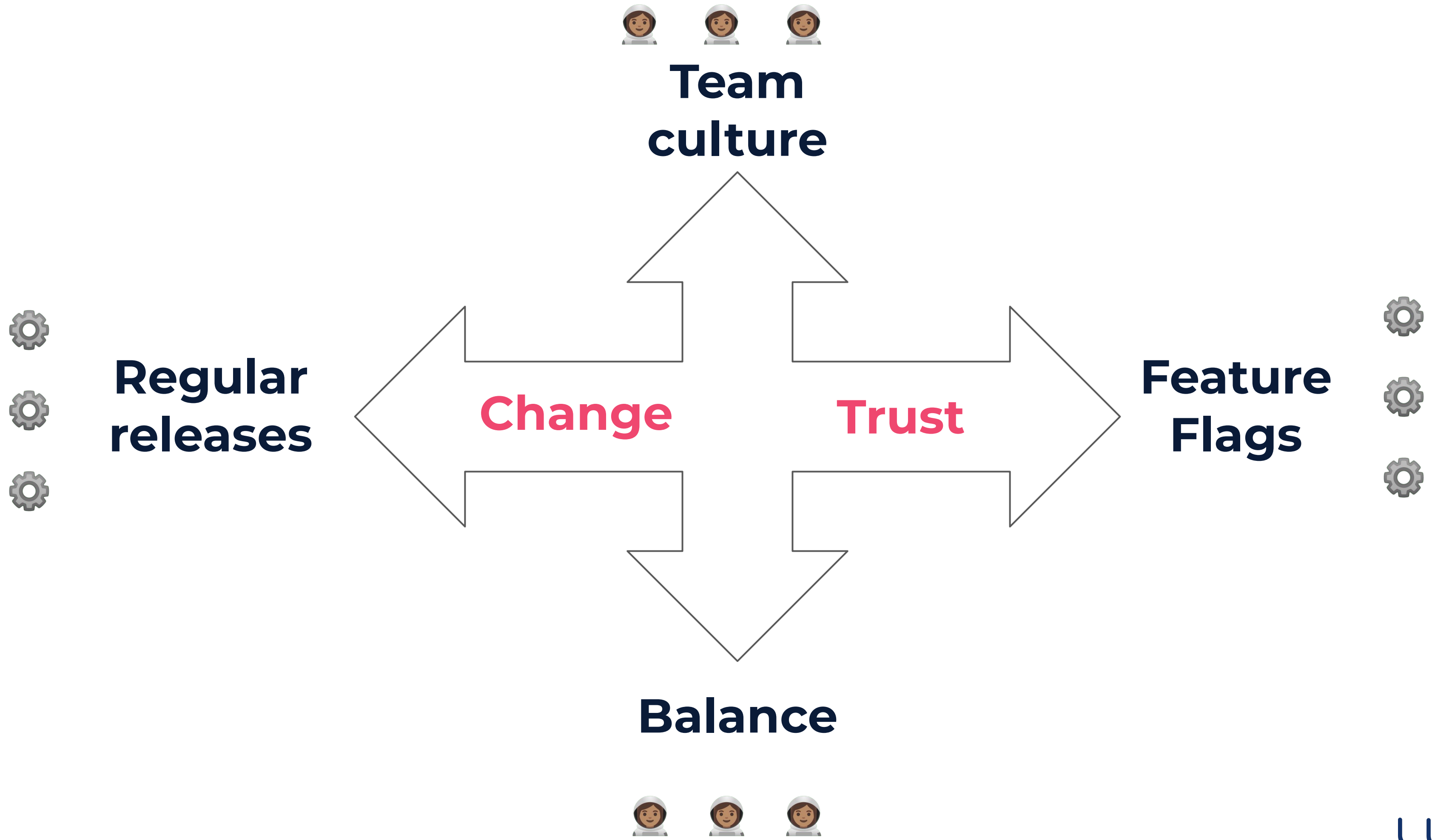
“ I 💖 Kotlin, testing and coroutines! ”

Future 🦄 Android Engineer

CI / CD


Tests on requests
Debug
Every 2h
Designers
Product Owner
Team
Internal
Alpha
Closed
Beta
Production
Public
Google Play


Roll out V1.0.0
V1.0.1
Roll out V1.0.1
In branch by Friday
Release branch
Prod
Beta
Prod
Develop branch
Landed on Friday
M
T
W
T
F
M
T
W
T
F
M
Landed on Friday
Sprint ends
Landed on Friday
Alpha
Alpha
Work on V1.0.1


Team culture
Regular releases
Change
Trust
Feature Flags
Balance

Never stop refactoring

What next

Complete migration

- Onboarding screens
- MVVM everywhere
- Protobuf everywhere

More tests

- More Unit tests
- Integration tests
- Functional tests

Other

- Startup quicker and cleaner
- Better Dependency injection?
- Navigation?
- Next shiny thing?

Questions

LUNO

Thank you