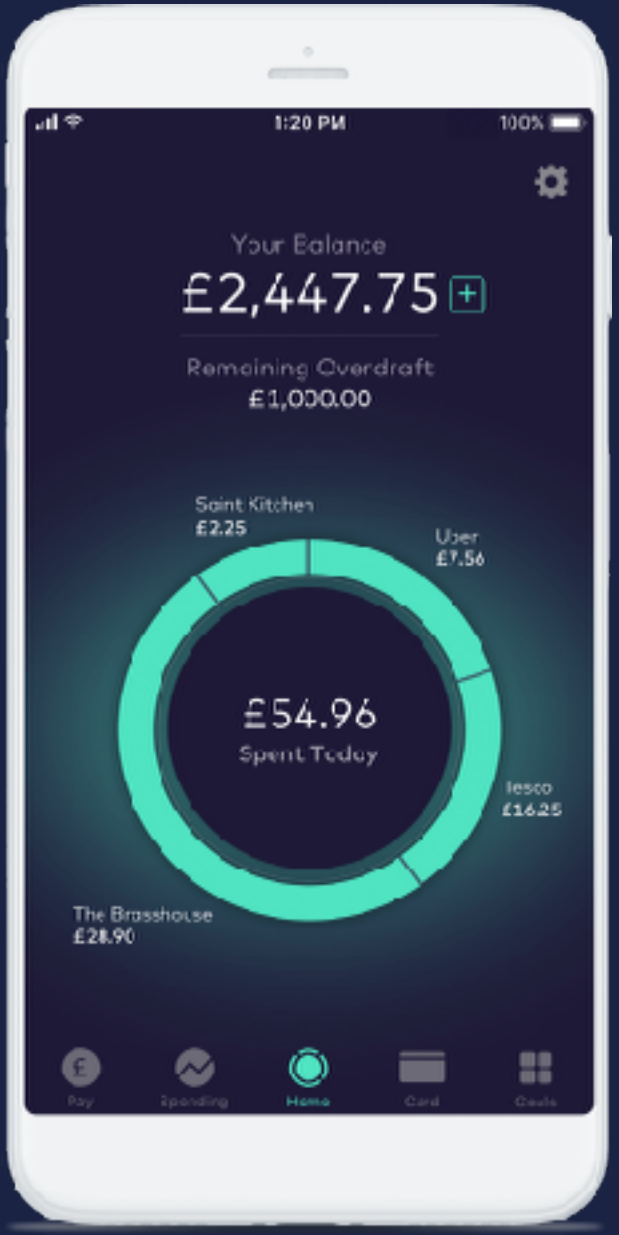
Disrupting the Banking Experience: Building a Mobile-only Bank

Yann Del Rey

Teresa Ng

STARLING BANK



Easy on-boarding

Zero Fees

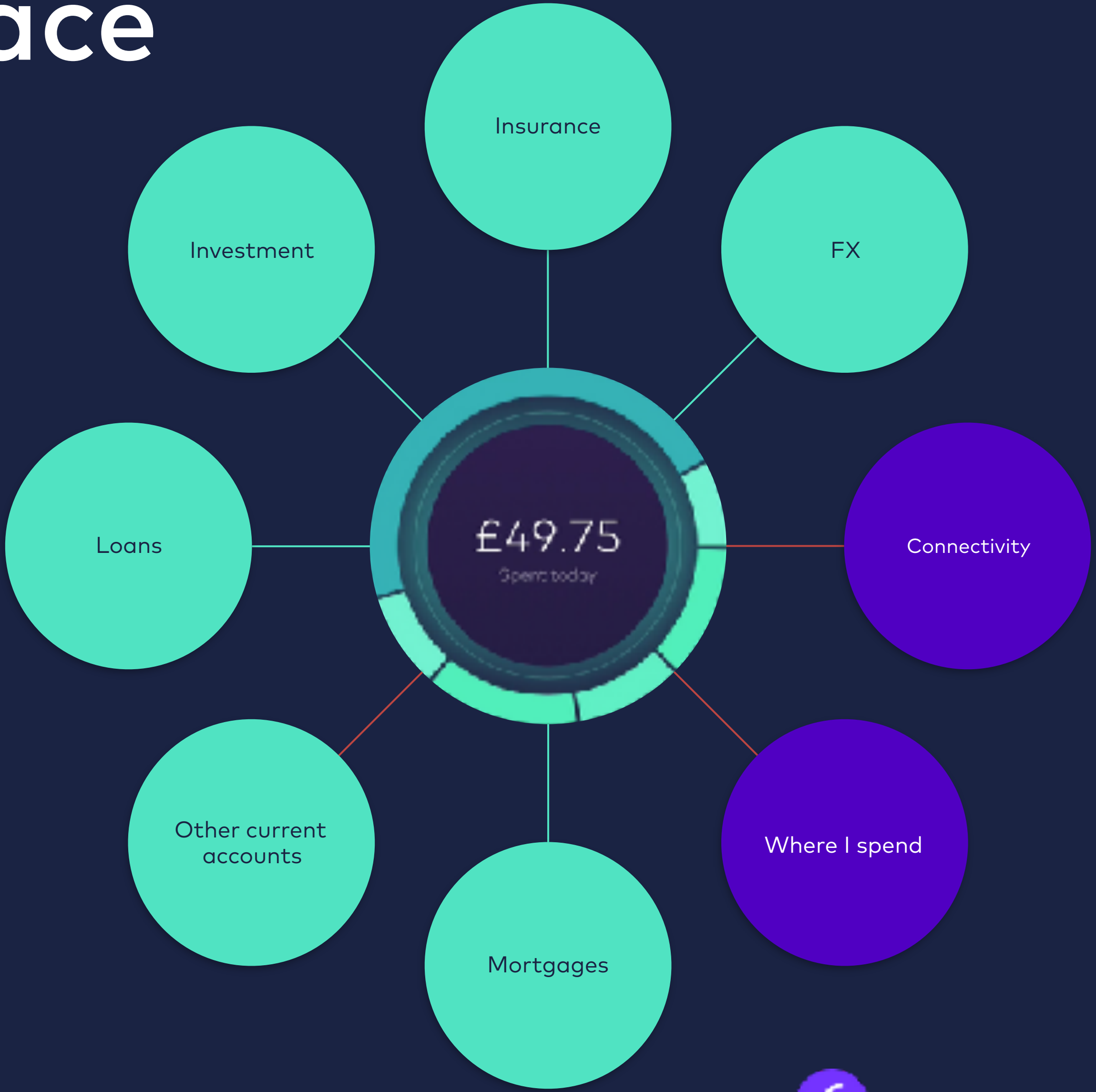
Instant Notifications

Public API - Open to all third party developers

STARLING BANK

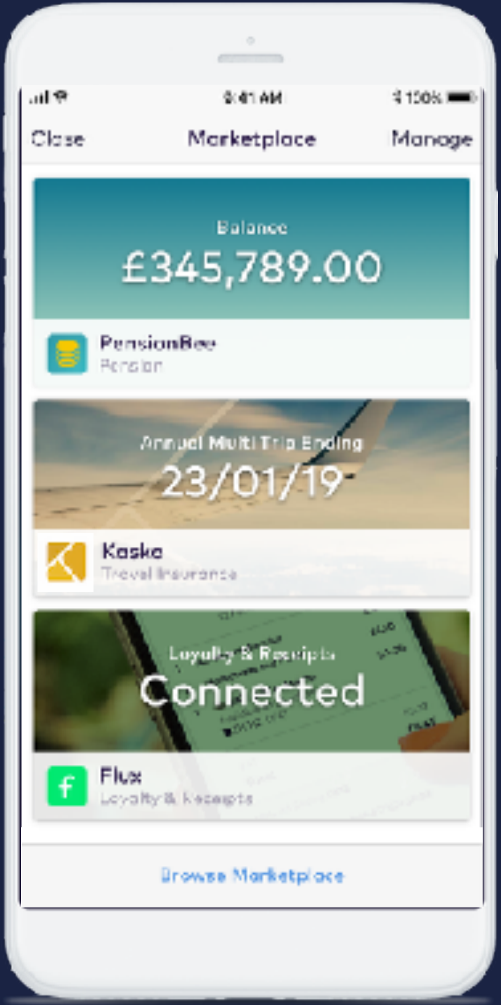
Starling Marketplace

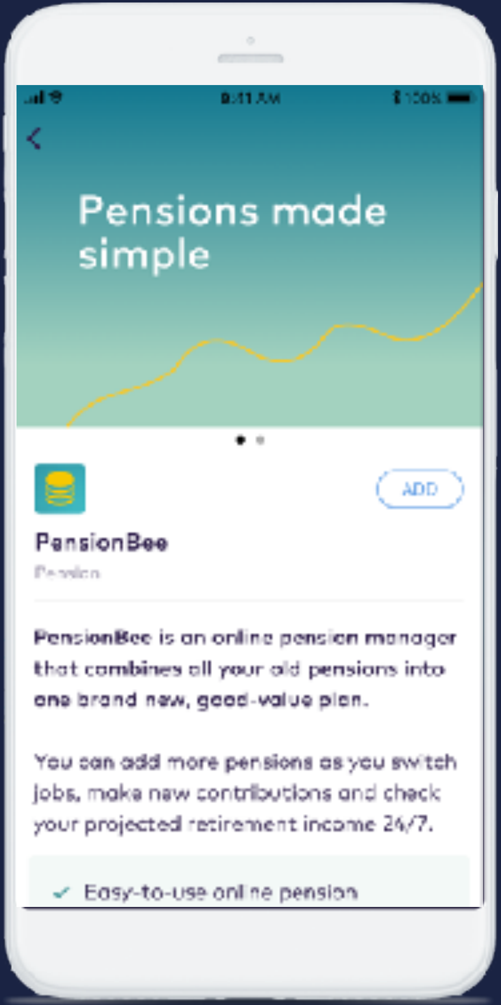
A marketplace for the best products, solving money problems



STARLING BANK

Starling Marketplace





PensionBee is an online pension manager that combines all your old pensions into one brand new, good-value plan.

You can add more pensions as you switch jobs, make new contributions and check your projected retirement income 24/7.

✓ Easy-to-use online pension

STARLING BANK

2014 Founded by Anne Boden 2014

2015 November: Early technical prototyping

2016 January: Starling raises $70M, started building the bank

July: PRA grants Starling its bank

October: Testing Mastercard debit

November: Alpha testing consumer app

December: Processing direct debits

2017 January: Starling becomes the 13th member to join Faster Payments

February: Launched Beta testing program

April: First ever Open Banking Hackathon

May: Public App Store launch

Summer: Apple & Google Pay, Spending Insights, Saving Goals

STARLING BANK

How we work

How we work

We have tried different variations:

- Feature teams
- Component teams

Team structure example

- The team:
  - All the mobile developers
  - Couple platform developers

- Experts:
  - Product managers
  - Designers

Our team structure

iOS

Android

Back end

Product managers

Designers

How do we decide who has to work on what?

Flow:

- Kanban board
- Product managers prioritise new work
- We as developers do our prioritisation
- Whoever is free and wants to work on it

How do we define the requirements?

- Whoever picked the feature, will help defining the feature
- Meetings are banned
- Communication is key
- starlingdevs.slack.com

Allows us to innovate

Our app architecture

STARLING BANK

Our app architecture

- Every design pattern has pros and cons
- Picking one depends on the context
- Iteration is key

MVC

- Pros: Good to quickly develop features, everyone knows it

- Cons: Doesn’t work well for a big codebase
  - Massive view controllers
  - Not maintainable
  - Hard to test
  - Hard to reuse components

STARLING BANK

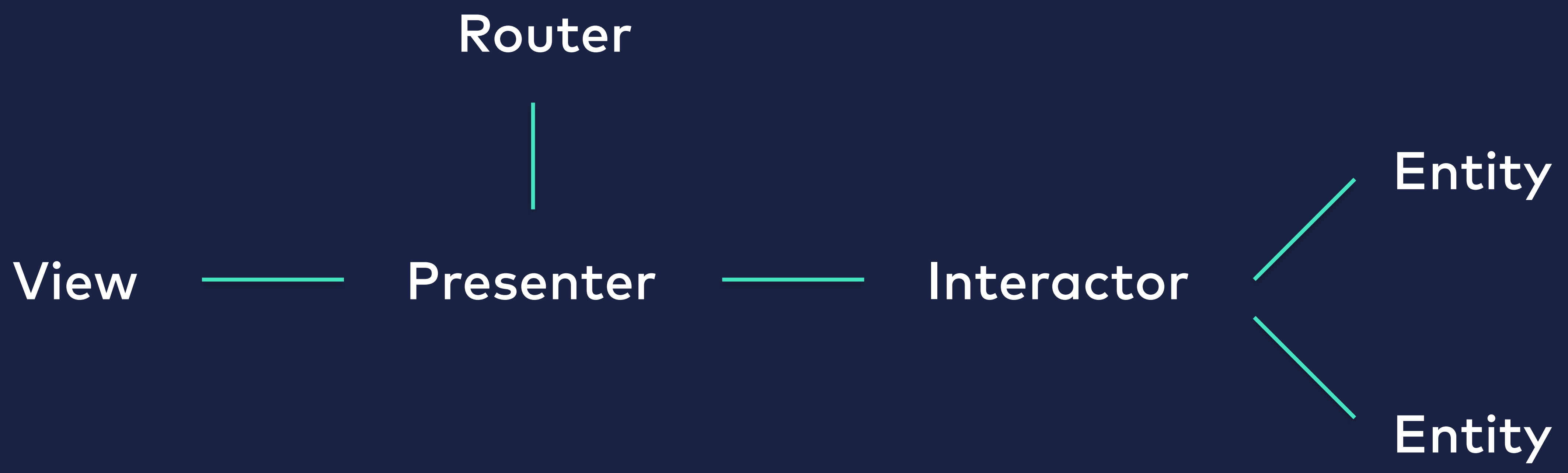
MVC with closures

Goal: Decrease view controllers size by decoupling the logic

- Pros: reduces each component responsibilities

- Cons: Still based on the MVC model
  - Doesn't solve our problem on the long term
  - Closures are difficult to track

VIPER



VIPER

Goal: Isolate each component into smaller pieces

- Pros: SOLID principles, easy to test, good for a large team
- Cons:
    - Lots of files
    - Boilerplate code
    - Protocols everywhere
    - Very difficult to iterate

VIPER with RxSwift

Goal: Reduce the number of files and boilerplate code

- What is RxSwift
- Remove protocols between interactor and presenter

Stores

Goal: Add reusability and decrease number of files

- Replace interactors by stores
- What is a Store
- Extract and centralise the network and data layers

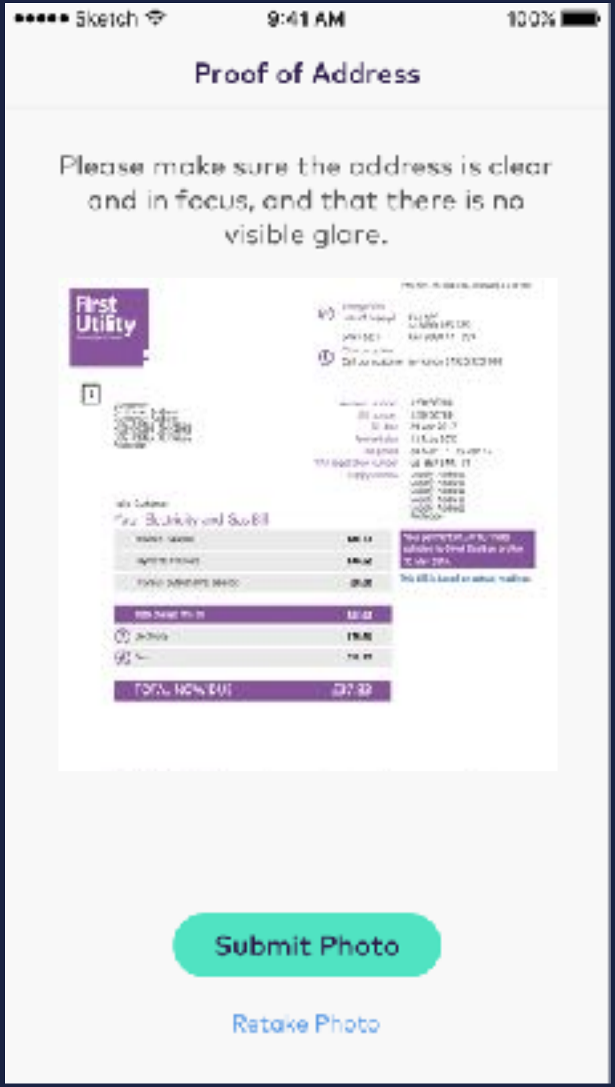
View configuration

Goal: Increase readability and decrease boilerplate code

- What is a View configuration
- Clear representation of a view
- Remove Presenter - ViewController protocols
- Easy to test and reuse

STARLING BANK

View configuration example


●●●●● Sketch 9:41 AM 100%

Proof of Address

Please make sure the address is clear and in focus, and that there is no visible glare.

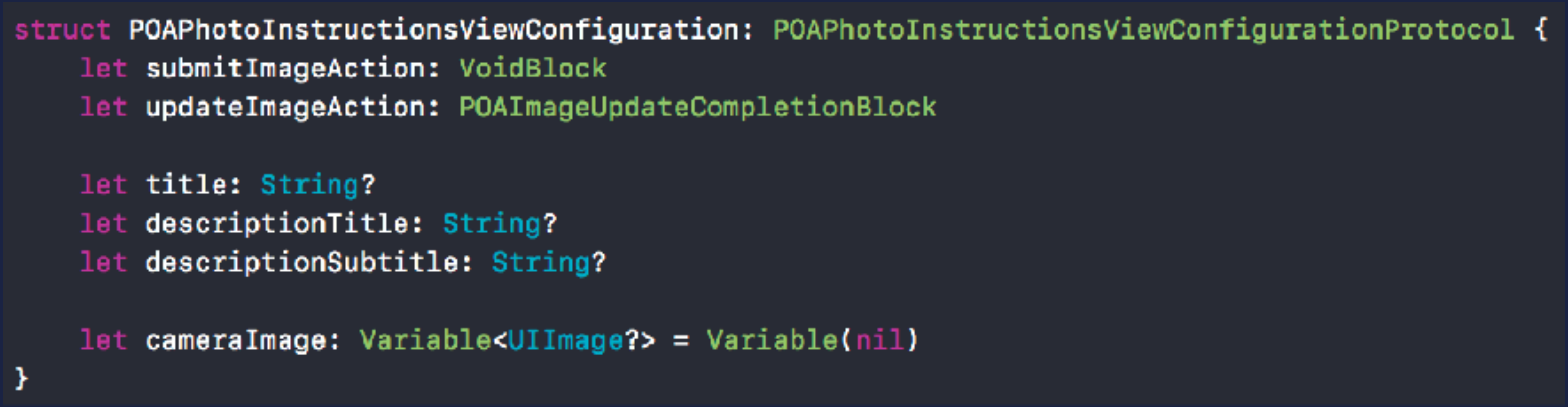
Submit Photo

Retake Photo

STARLING BANK

View configuration example

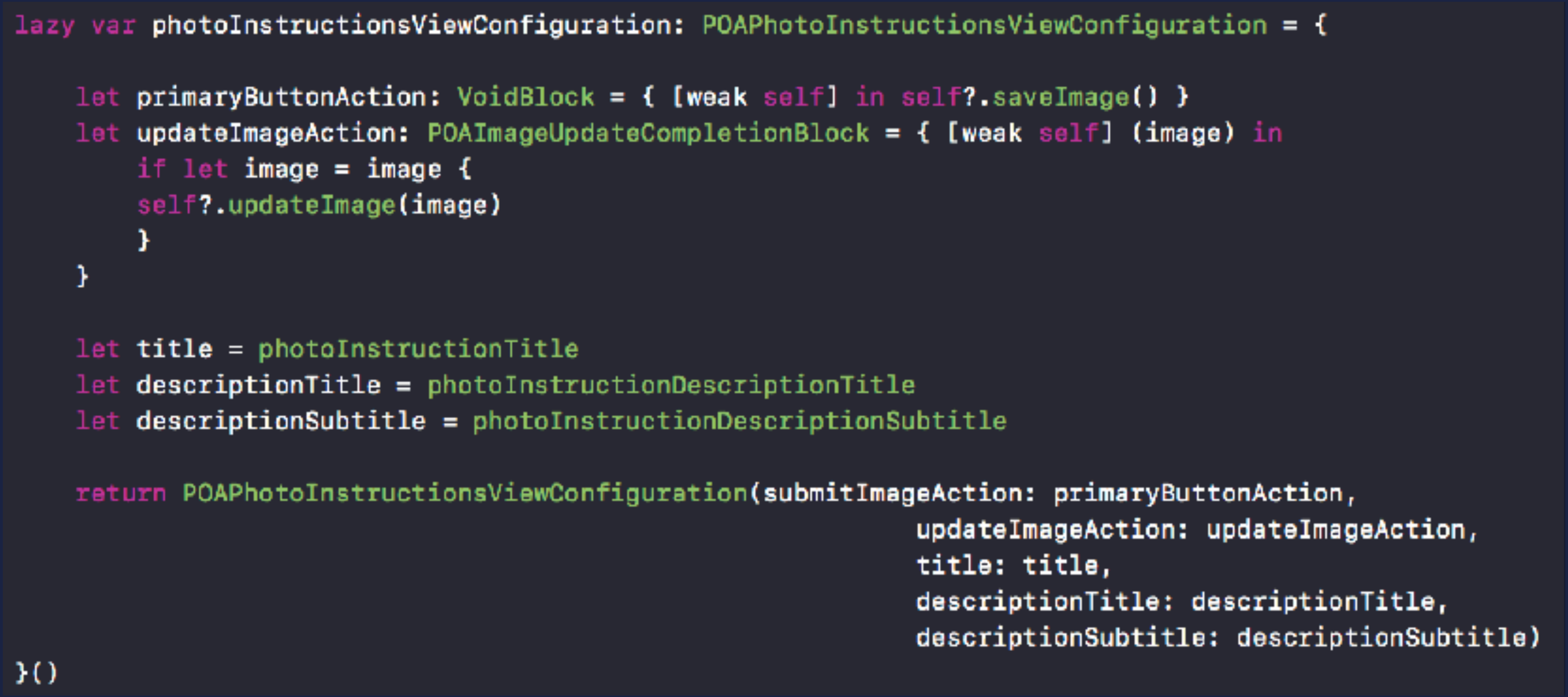
Proof of address



```swift
struct POAPhotoInstructionsViewConfiguration: POAPhotoInstructionsViewConfigurationProtocol {
    let submitImageAction: VoidBlock
    let updateImageAction: POAImageUpdateCompletionBlock

    let title: String?
    let descriptionTitle: String?
    let descriptionSubtitle: String?

    let cameraImage: Variable<UIImage?> = Variable(nil)
}
```

STARLING BANK

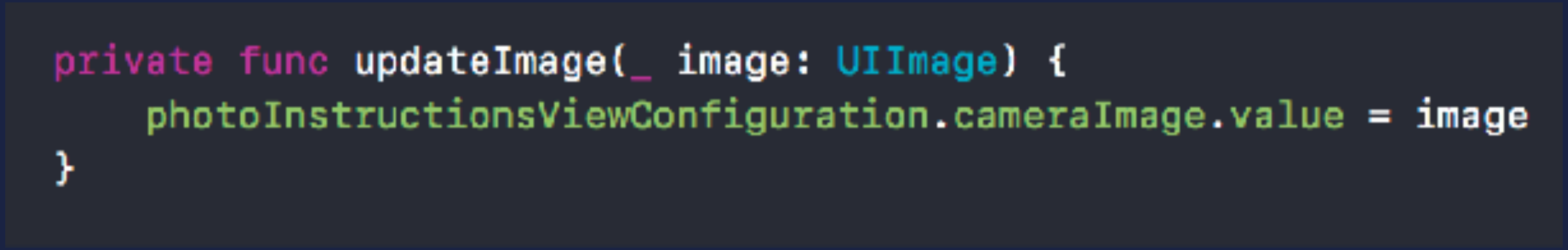
View configuration example



```swift
lazy var photoInstructionsViewConfiguration: POAPhotoInstructionsViewConfiguration = {

    let primaryButtonAction: VoidBlock = { [weak self] in self?.saveImage() }
    let updateImageAction: POAImageUpdateCompletionBlock = { [weak self] (image) in
        if let image = image {
        self?.updateImage(image)
        }
    }

    let title = photoInstructionTitle
    let descriptionTitle = photoInstructionDescriptionTitle
    let descriptionSubtitle = photoInstructionDescriptionSubtitle

    return POAPhotoInstructionsViewConfiguration(submitImageAction: primaryButtonAction,
                                                                   updateImageAction: updateImageAction,
                                                                   title: title,
                                                                   descriptionTitle: descriptionTitle,
                                                                   descriptionSubtitle: descriptionSubtitle)
}()
```

STARLING BANK

View configuration example



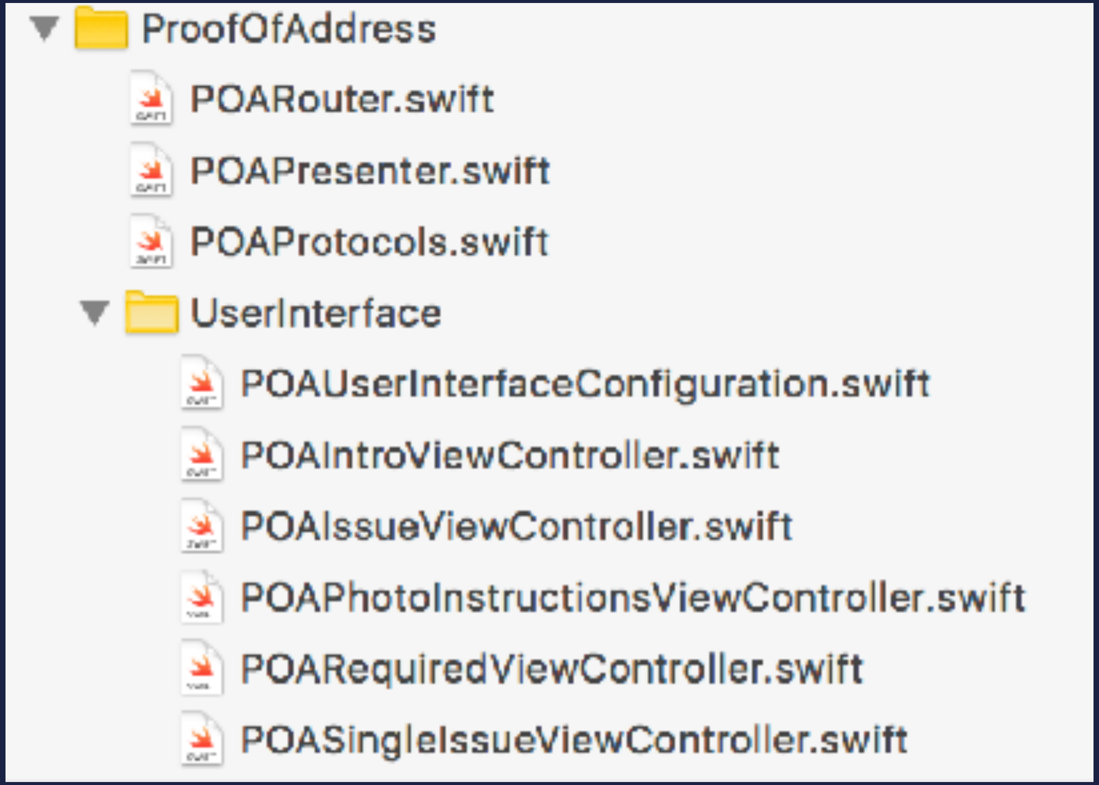
```swift
private func updateImage(_ image: UIImage) {
    photoInstructionsViewConfiguration.cameraImage.value = image
}
```

What does it look like

$$\begin{tikzcd}
\text{Router} \arrow[r, leftrightarrow, "\text{Protocols}"] & \text{Presenter} \arrow[r, dash, "\text{View Configs + RxSwift}"] \arrow[d, "\text{Protocols + RxSwift}"] & \text{ViewControllers} \\
& \text{Stores} &
\end{tikzcd}$$

STARLING BANK

What does it look like



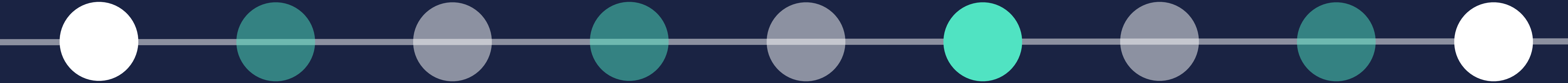
```
▼ ProofOfAddress
    POARouter.swift
    POAPresenter.swift
    POAProtocols.swift
  ▼ UserInterface
      POAUserInterfaceConfiguration.swift
      POAIntroViewController.swift
      POAIssueViewController.swift
      POAPhotoInstructionsViewController.swift
      POARequiredViewController.swift
      POASingleIssueViewController.swift
```

STARLING BANK

Testing

STARLING BANK

Oct 2016
First commit to
the Android repo



March 2016
Work begins
on the iOS app

May 2017
Public release of
both apps

What do we need to test...

... with no QA team?

STARLING BANK

What do we need to test?

- What we don’t need to test:
    - Network calls are covered by our Platform tests
- Utility Methods (date formatting, currency formatting etc)
- Views, views, views
- Interaction with these views

Writing Tests

- Java - Mockito, assertJ
- Android - Espresso
- RxJava2 & Dagger2

Test All Views Are Visible

```
@Test
public void scrollAllSlides() {
  // perform
  activityRule.launchActivity(null);

  // Wait until the layout is created
  onView(withId(R.id.saving_intro_pager)).check(matches(isDisplayed()));

  SavingIntroActivity.Slide[] slides = SavingIntroActivity.Slide.values();
  for (int i = 0; i < slides.length; i++) {
    onView(allOf(withId(R.id.saving_intro_slide_image), isCompletelyDisplayed()))
        .check(matches(withImageResource(slides[i].image)));
    onView(allOf(withId(R.id.saving_intro_slide_title), isCompletelyDisplayed()))
        .check(matches(withText(slides[i].title)));
    onView(allOf(withId(R.id.saving_intro_slide_description), isCompletelyDisplayed()))
        .check(matches(withText(slides[i].description)));
    onView(withId(R.id.saving_intro_pager)).perform(swipeLeft());
  }
}
```

Testing Visibility of Views in Specific Scenarios

```
@Test
public void whenUnableToLoadMissingDataErrorIsDisplayed() throws Exception {
  doThrow(new IOException("")).when(starlingStorage).loadMissingData();

  activityTestRule.launchActivity(null);

  verify(snackbarManager).show(any(), anyInt(), anyInt(), anyInt(), any());
  verify(starlingStorage).loadMissingData();

  // Retry button tries to reload data
  onView(withText(getTargetContext().getString(R.string.button_retry)))
.perform(click());
  verify(starlingStorage, times(2)).loadMissingData();
}
```

STARLING BANK

Running the Tests

- Unit tests can be run wherever
- Different strategy is required for UI tests
- UI tests need to cover:
    - Fragmentation
    - Usability

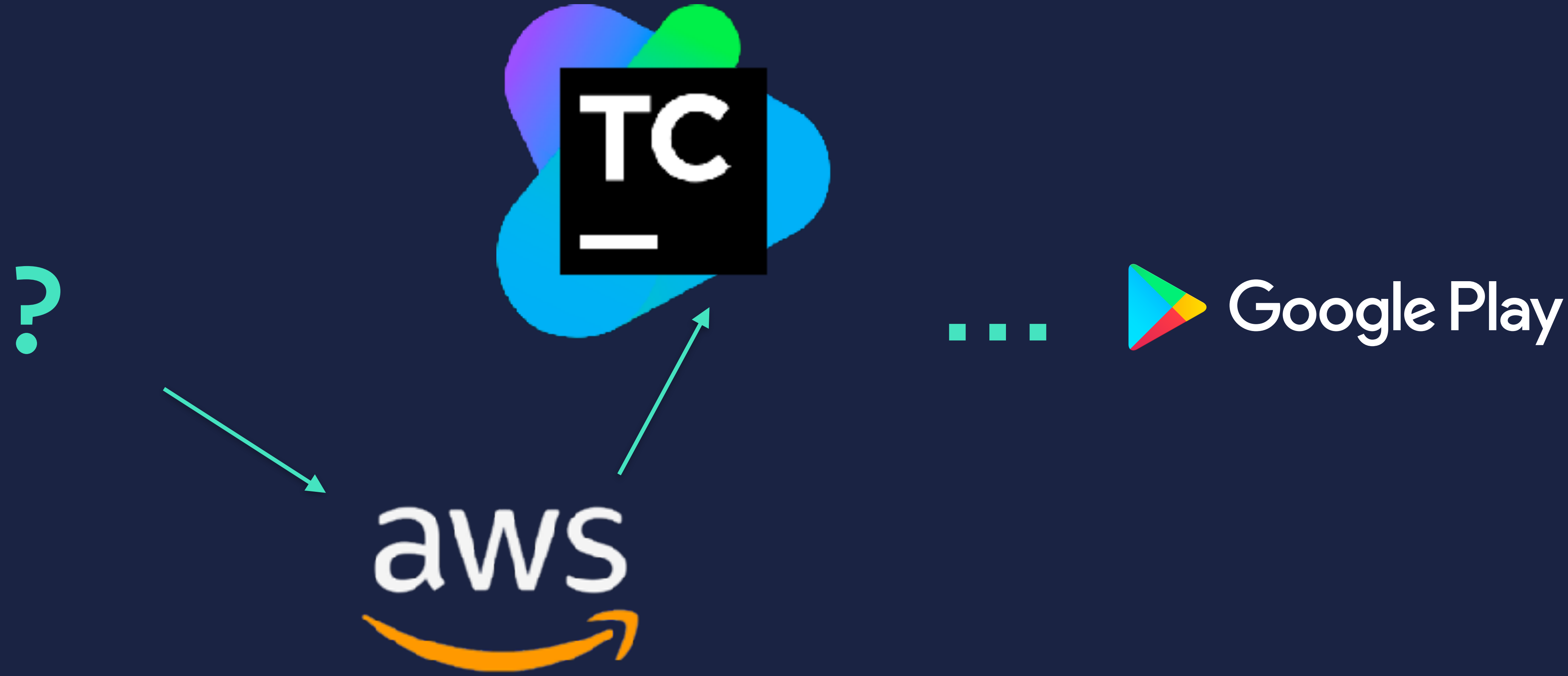
Exploring Options

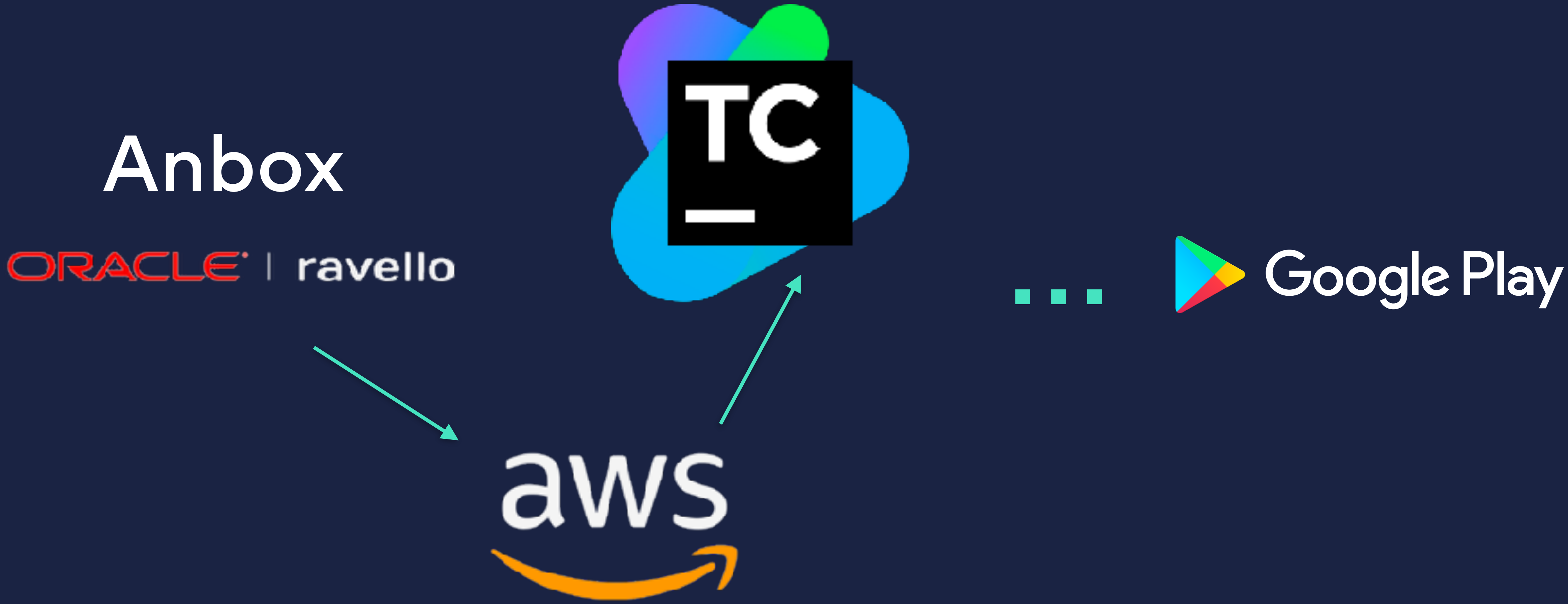
STARLING BANK

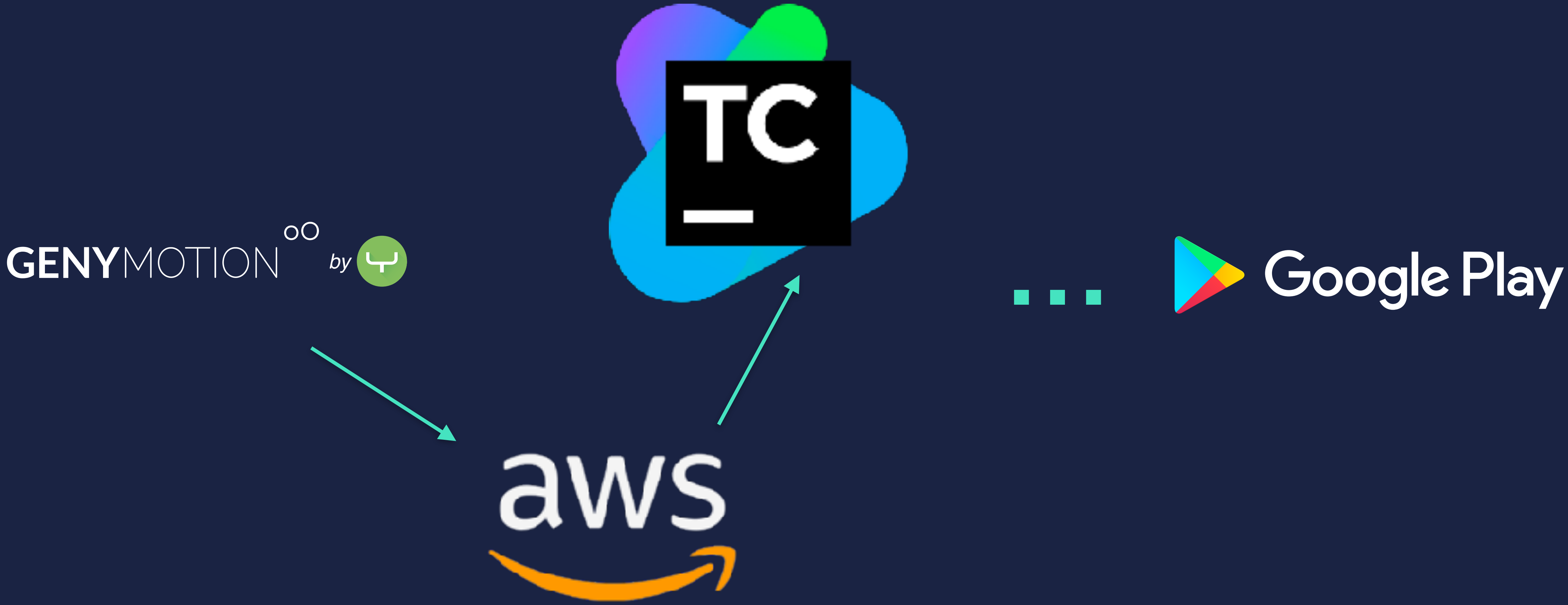
Firebase

SAUCE LABS

STARLING BANK


?
aws
TC
...
Google Play


Anbox
ORACLE | ravello
TC
aws
...
Google Play

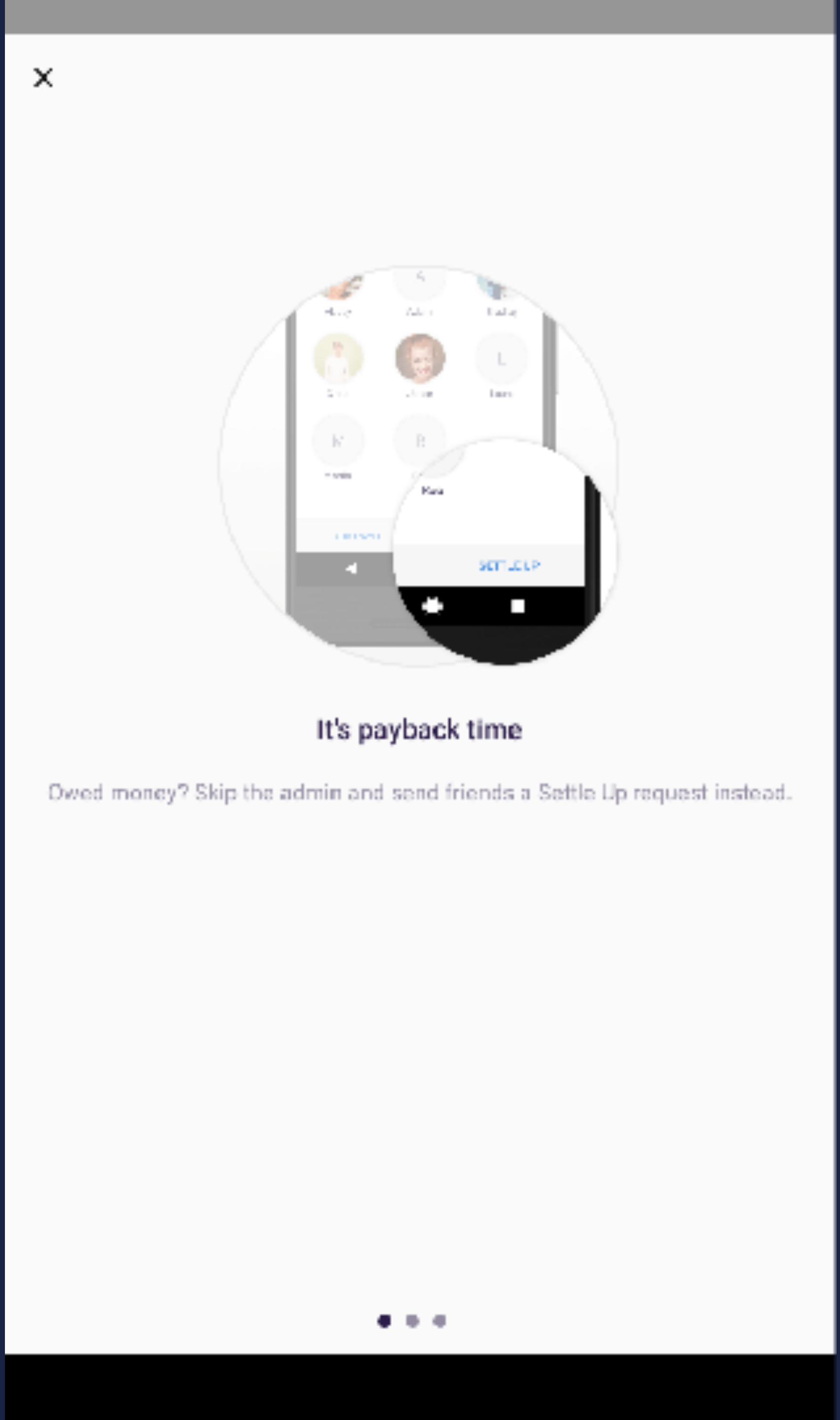

GENYMOTION by
TC
aws
Google Play

Test Reporting

- Log the reason for failures
- Record all the UI tests
- Take screenshots of failures

Example of Reporting in Action

```
android.support.test.espresso.NoMatchingViewException: No views in
hierarchy found matching: with string from resource id:
<2131756434>[payments_add_payee] value: Add payee

View Hierarchy:
+>DecorView{id=-1, visibility=VISIBLE, width=600, height=1024, has-
focus=false, has-focusable=true, has-window-focus=true, is-
clickable=false, is-enabled=true, is-focused=false, is-focusable=false,
is-layout-requested=false, is-selected=false, layout-
params=WM.LayoutParams{(0,0)(fillxfill) ty=1 fl=#85810100 pfl=0x20000
wanim=0x1030465 needsMenuKey=2}, tag=null, root-is-layout-
requested=false, has-input-connection=false, x=0.0, y=0.0, child-
count=3}
|
......
```


×
It's payback time
Owed money? Skip the admin and send friends a Settle Up request instead.

```
@Test
public void testAddPayeeActivityLaunched() {
  // given
  when(payeeEntity.observeAll()).thenReturn(Flowable.empty());
  Intents
      .intending(activityOf(PayeeLookupActivity.class))
      .respondWith(new Instrumentation.ActivityResult(RESULT_OK, null));

  // perform
  startActivity();
  onView(withText(R.string.payments_add_payee)).perform(click());

  // verify
  Intents.intended(activityOf(PayeeLookupActivity.class));
}
```

Failed here

Resolved by adding this to the test set-up:

```
when(preferences.hasStarlingPayRequestIntroBeenShown())
.thenReturn(true);
```

STARLING BANK

How we can take this further

- Slackbot integration
- Concurrency
- Appium for application upgrade tests

To conclude...

- Stability of the app does not need to be sacrificed
- This is just a start

Q & A

@StarlingDev
@StarlingBank
@DaProd_ (Yann)
@NovemberGave (Teresa)

STARLING BANK