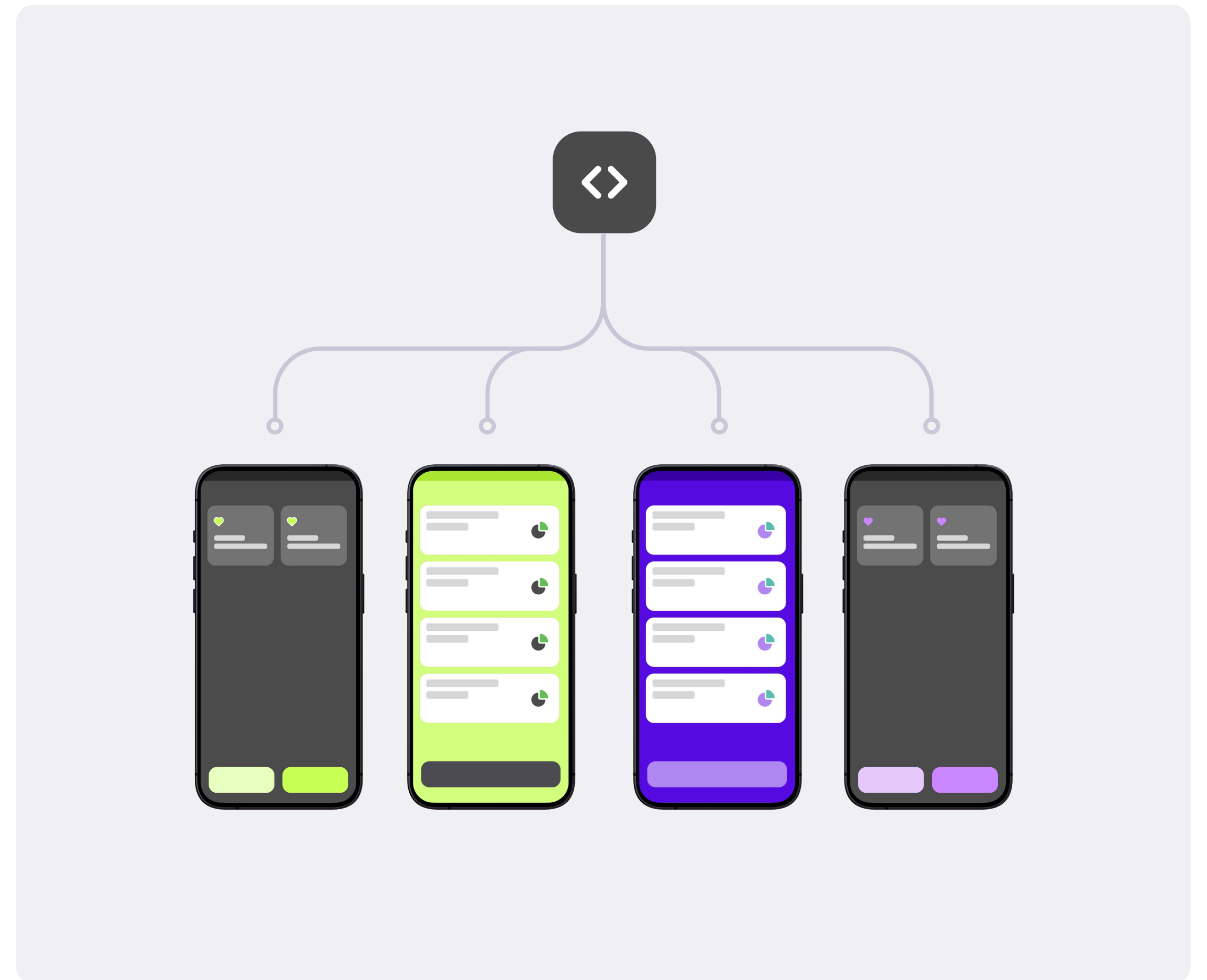


White label Android apps

Development, customisation & maintenance

What is White Label?

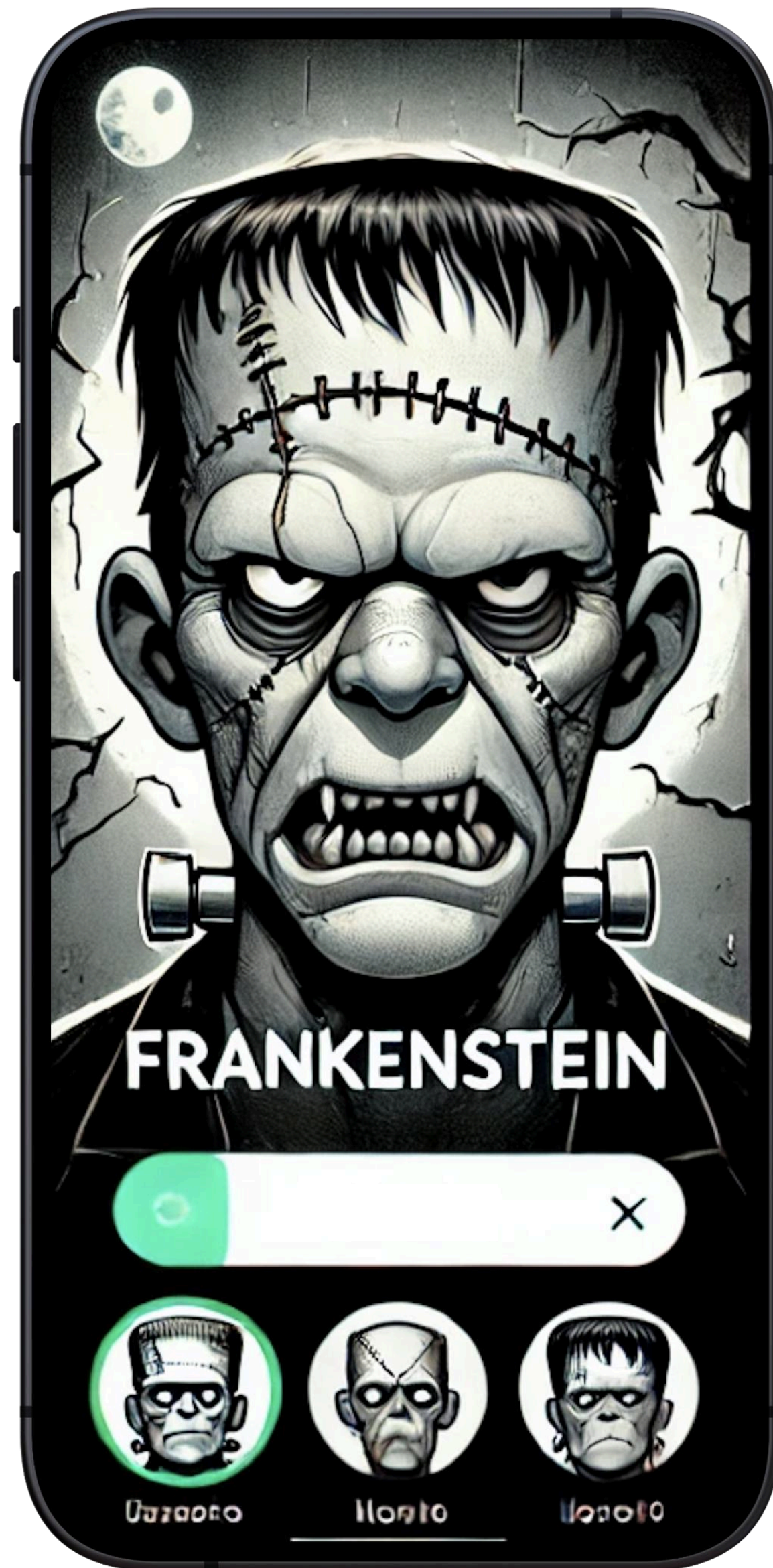


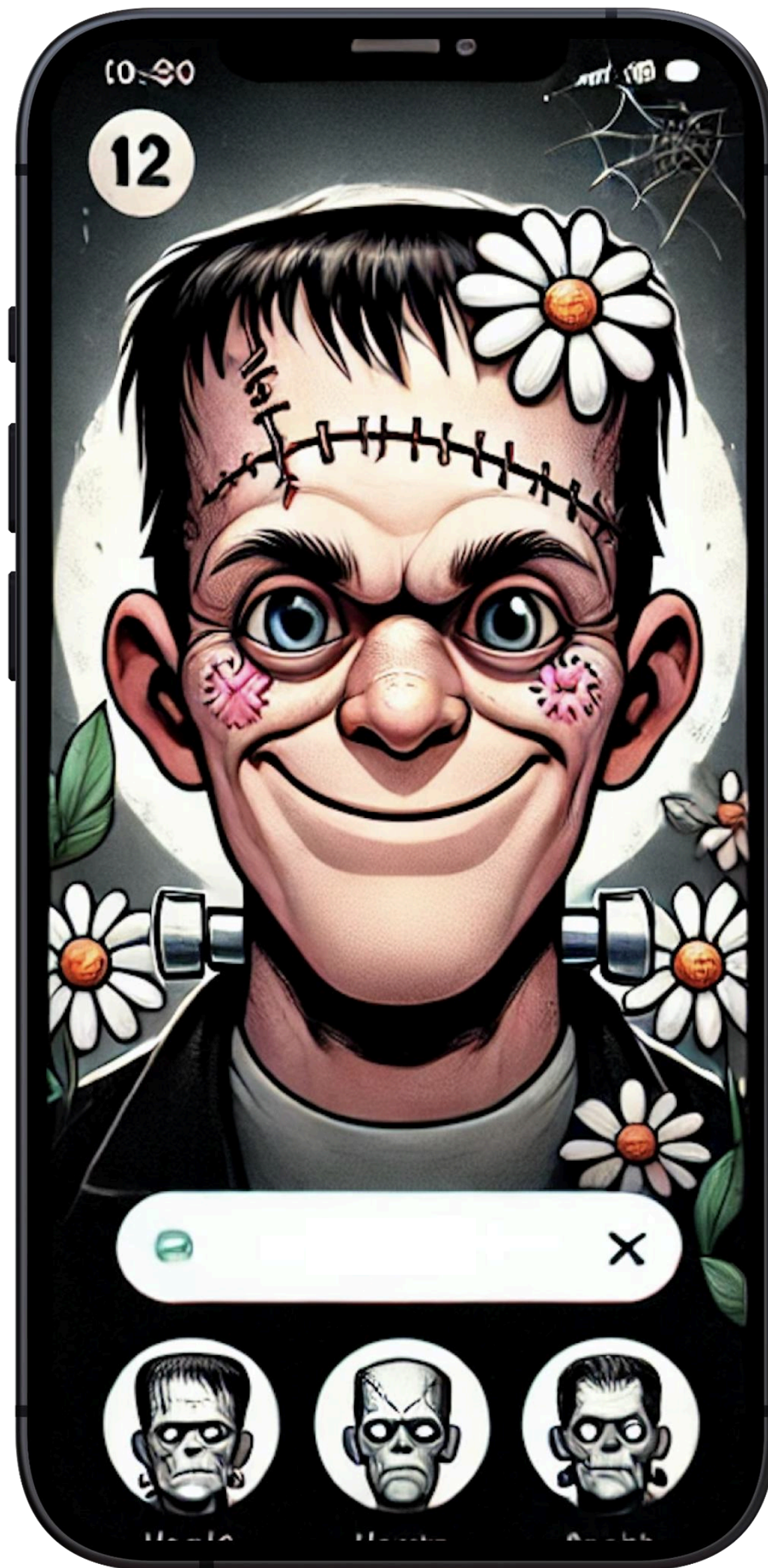
Surviving the Chaos plan:

- 1 Multiple Modularising Changes
- 2 Design System
- 3 Independent Releases
- 4 Test Automation
- 5 Publishing Automation

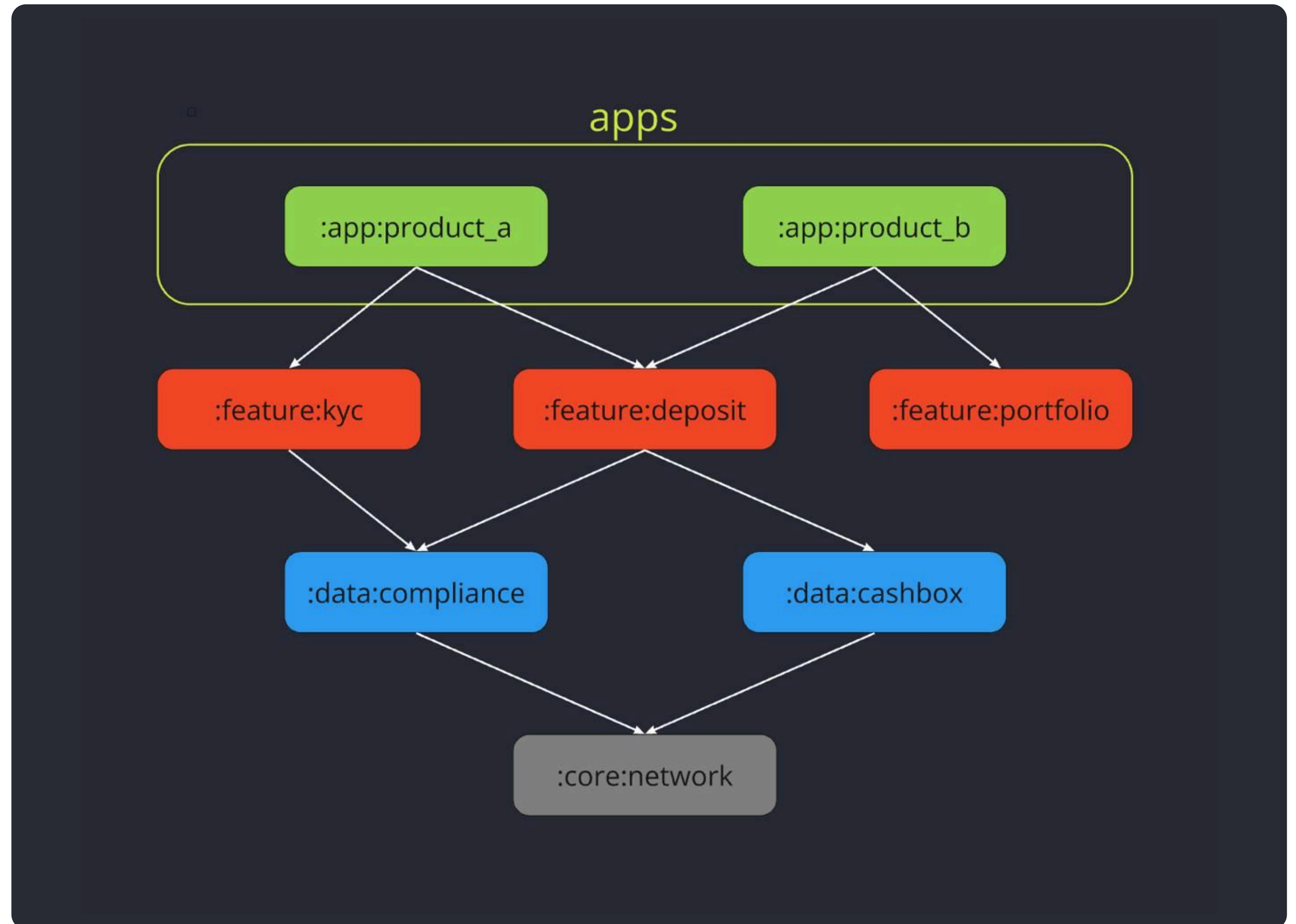
1

Multiple Modularising Changes



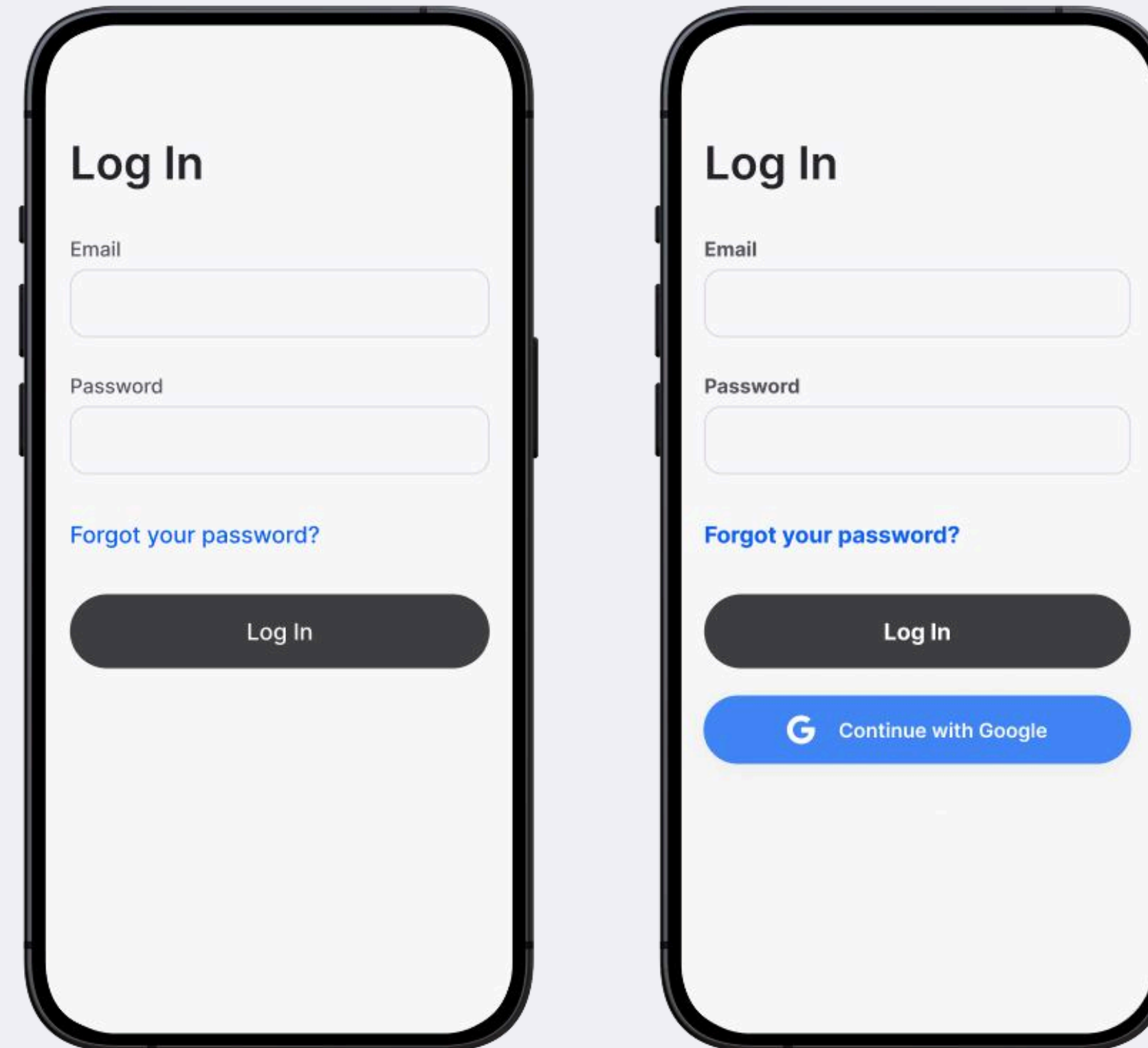


Modular- architecture



Feature toggle

1. Local
2. Remote
 - experimental (A/B)
 - operational



```
feature("google-signin").enabled
```


2

Design system

Contextual palette



```
<color name="surface_1_default">#12116</color>  
<color name="border_accent_active">#575f3b</color>  
<color name="text_secondary_disabled">#66969696</color>  
<color name="icon_warning_active">#f9bd52</color>  
<color name="surface_emphasis_inverse_default">#262925</color>
```


Component palette



```
<color name="main_calendar_button_default">  
    @color/surface_emphasis_inverse_default  
</color>
```


Palette building

Colors

Format

hex

neutral #262929 #5E6A6B

Light #262929

Dark #5E6A6B

#262929

#5E6A6B

Format

hex

accent #A5C900 #B5D938

Light #A5C900

Dark #B5D938

#A5C900

#B5D938

Format

hex

Factors

tone_background 100 8

Light 100

Dark 8

100

8

tone_layer_gap 2 4

Light 2

Dark 4

2

4

tone_surface_tertiary 98 12

Light 98

Dark 12

98

12

tone_surface_secondary 90 20

Light 90

Dark 20

90

20

Previews

Button

OCT 25

Button

OCT 25

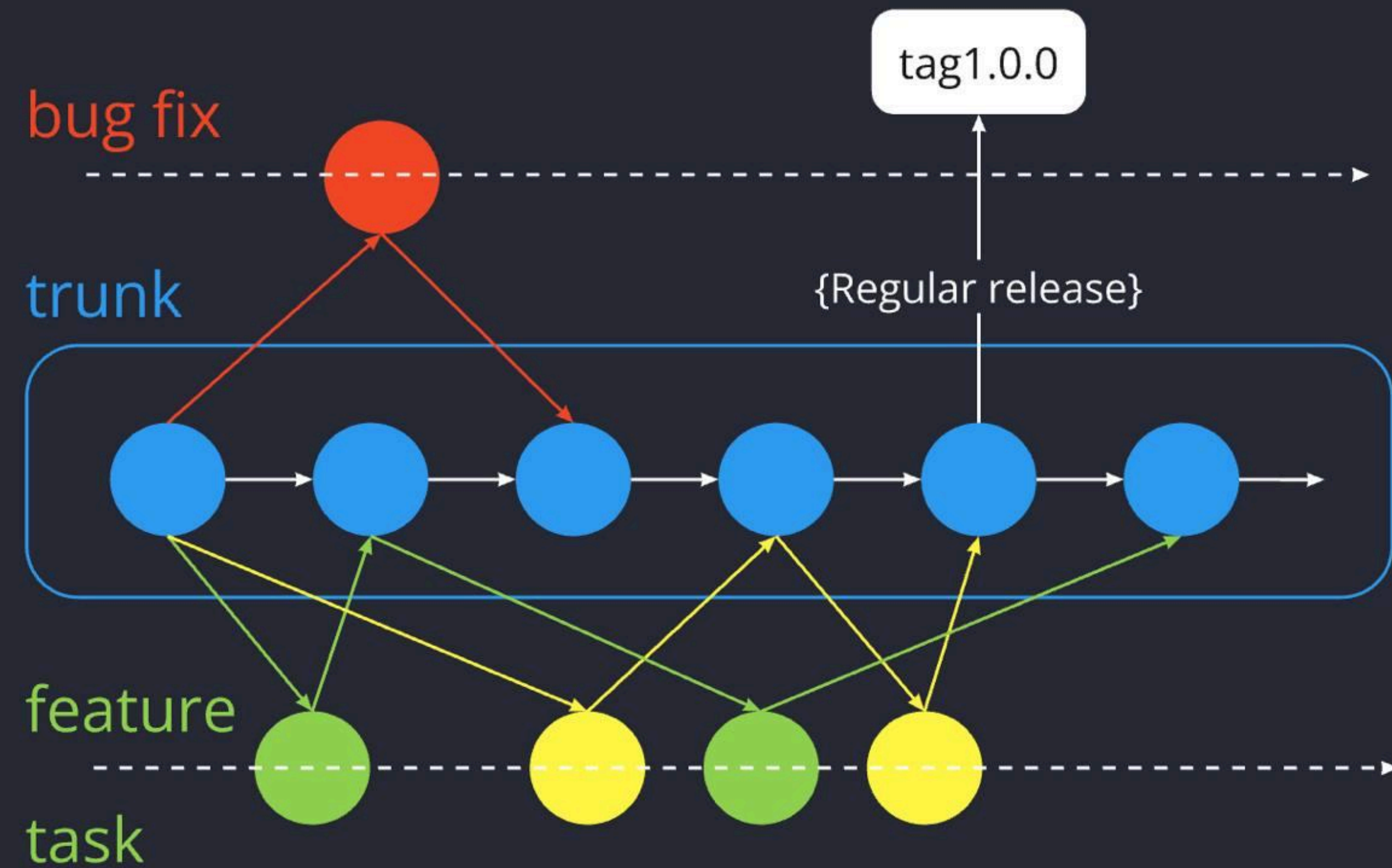
3

Independent Releases

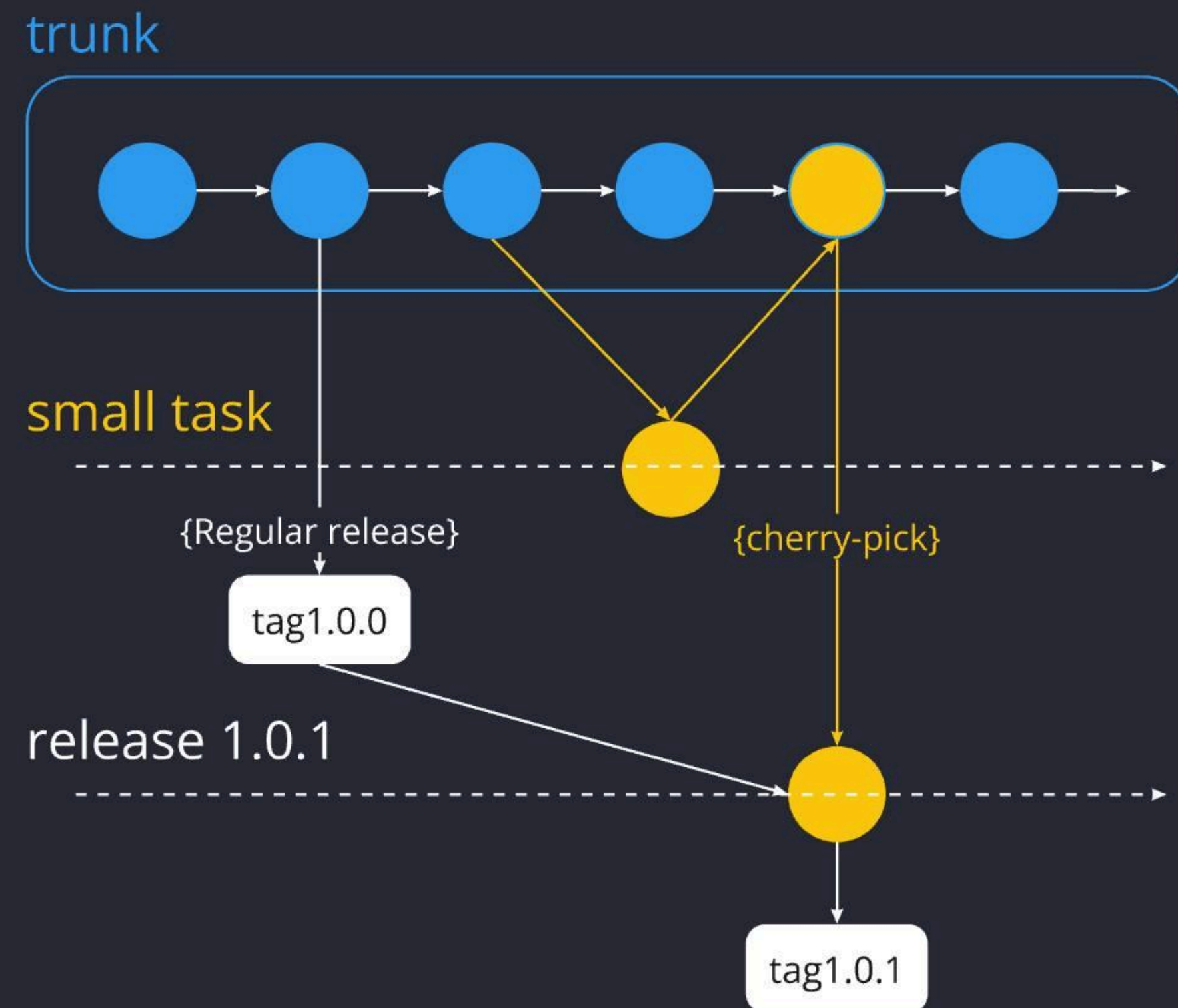
Release plan



Trunk-based development



On-demand release



4

Test Automation

Hardcoded User

```
@Test
fun tinMaskCheckTest() = run {
    user = TestUser("login@gmail.com", "qwerty")
    scenario(LoginScenarioOrientations(user))
}
```


Created User

```
@Test
fun tinMaskCheckTest() = run {
    user = qaLibSteps {
        val userType = UserType(NETHERLANDS)
        registerUser(userType, PartiallyVerified(TIN))
    }
    scenario(LoginScenarioOrientations(user))
}
```


Brand parameters

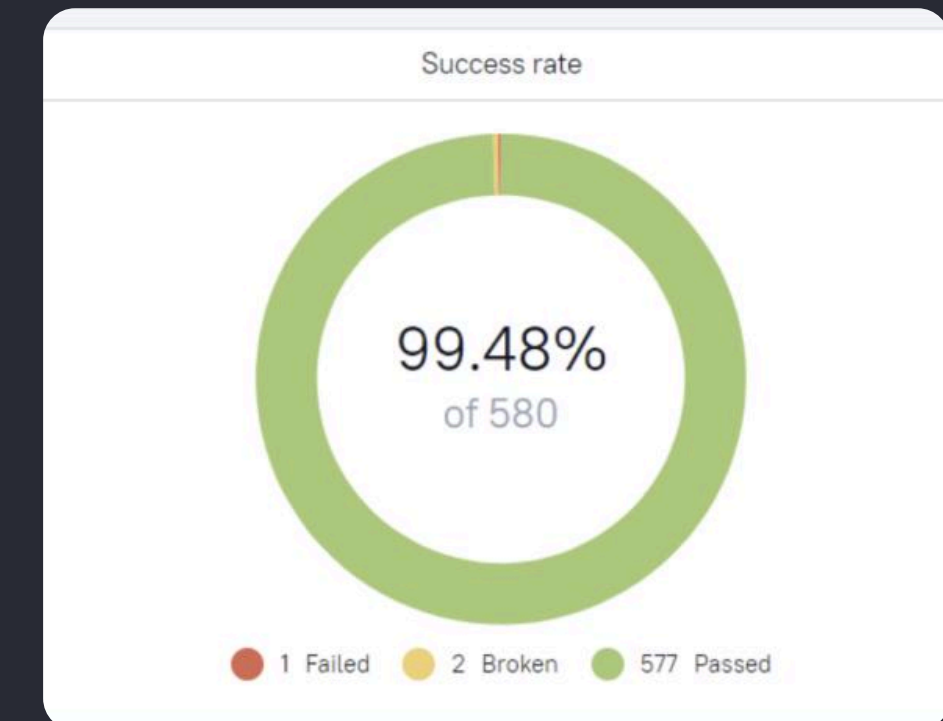
```
@Test
fun tinMaskCheckTest() = run {
    user = qaLibSteps {
        val userType = brandUserType()
        registerUser(userType, PartiallyVerified(TIN))
    }
    scenario(LoginScenarioOrientations(user))
}
```

```
TestBrand(
    packageId = "com.testbrand",
    appName = "App name",
    country = NETHERLANDS,
    saasLogoType = QUADCODE,
    imageSamples = testBrandSamples,
),
```

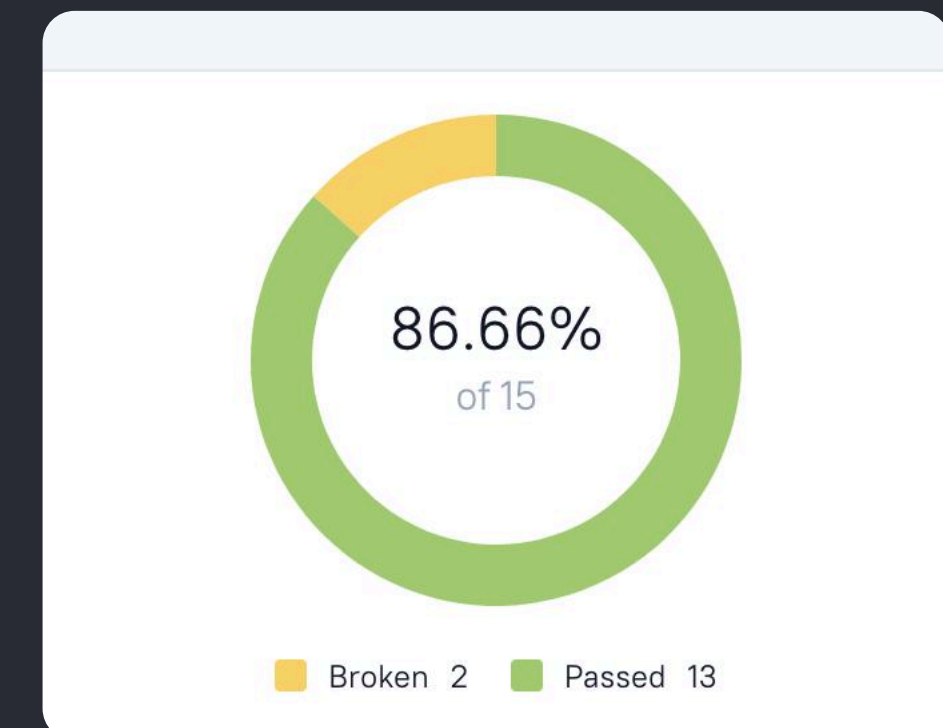

Smoke brand scope

- Mandatory e2e smoke scenarios
- App identity checks (including screenshot testing)
- Social authentication
- Push notifications
- Terms links
- Other brand related checks

Full regression



Smoke brand



5

Publishing Automation

CI plugins

Jenkins



GitlabCI



Jenkins example

```
androidApkUpload
  googleCredentialsId: "qwerty",
  apkFilesPattern: "release.aab",
  trackName: "production",
  releaseName: "1.0.1",
  rolloutPercentage: "20",
  deobfuscationFilesPattern: "proguard.map",
  recentChangeList: [[
    language: 'en-GB',
    text: "We've made the app as stable as a cow on a unicycle.
          It's not going anywhere now!"
  ],
]
```


That's It!



No more tangled mess of unrelated features



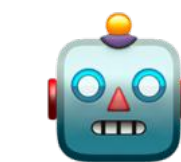
No more 'this button feels off' - now it's officially off!



Deploy without tears: customers don't have to wait forever



"Let's just check it manually" – No, let's not



`adb push release.apk /store` - well, almost

Thank you <3

