Entwicklung zweier Apps zur digitalen Kleinspende an bettelnde Personen

Zwischenvortrag BA

Paul Weber, 20.01.2022

Überblick

- Motivation
- Architektur
- Implementierung
- Status Quo
- Ausblick

Motivation

Produkt / Hintergrund

- Bargeldlosere Gesellschaft durch Corona und gesamtgesellschaftliche / technologische Entwicklung
- Gelegenheiten zur spontanen Kleinspende schwinden
- Lösungen? Andere Länder als Vorbild?

Produkt / Hintergrund

Lösungen

- Kartenlesegeräte
- -> erfordern Bankkonto
- Kryptowährungen
- -> erfordern technisches Know-How & Hardware
- Kommerzielle digitale Zahlungsanbieter
- -> Ethisch fragwürdig

Produktidee

- Spenden per Smartphone (Apple Pay, Google Pay, Kreditkarten, etc.)
- Bargeldauszahlung im Einzelhandel:

  barzahlen.de - Rewe, dm, etc.

  Solidarische Partnerunternehmen - Spätis, Cafés, etc.
- Mobile first
- QR Codes auf Bechern, Web-App mit ausdruckbaren Codes

Motivation

Technologie

- Deklarative UI Frameworks auf allen relevanten Platformen

  SwiftUI (1.0 - 2019) - iOS, macOS, tvOS, watchOS
  Compose (1.0 - 2021) - Android, Desktop (Win / Linux), Web

- Kotlin Multiplatform
- Fat client? Wie viel Code kann wirklich geteilt werden?
- Wieso native statt z.B. Flutter?

Architektur

Bestandsaufnahme

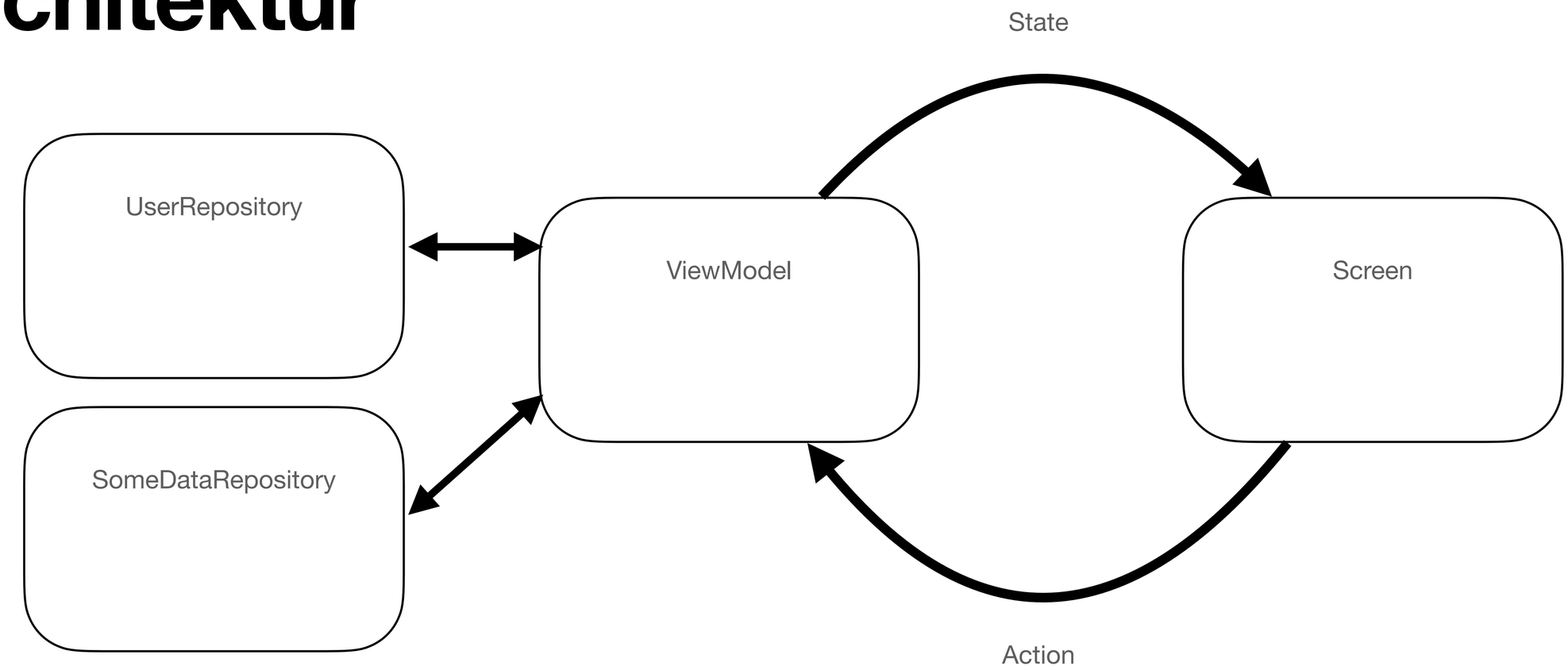
- Bisherige Standards (MVP, MVC, MVVM-C) mäßig nützlich bei deklarativen UI-Frameworks
- Aus dem Web: Redux
- Abgewandelt zu MVI (Android), The Composable Architecture (iOS)
- Nachteil: Hohe Abstraktionsebene -> Einstieg schwer
- Nachteil: Navigation ungelöst

Architektur

Ideal

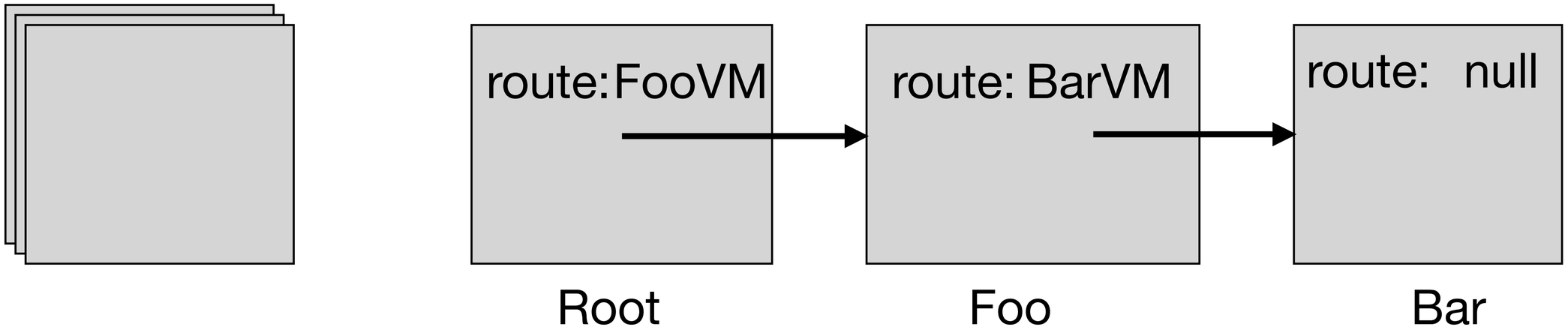
- KISS
- ViewModel hält UI State
- ViewModel hält Navigation State
- ViewModel bietet Actions an

Architektur



Architektur

(Deklarative) Navigation

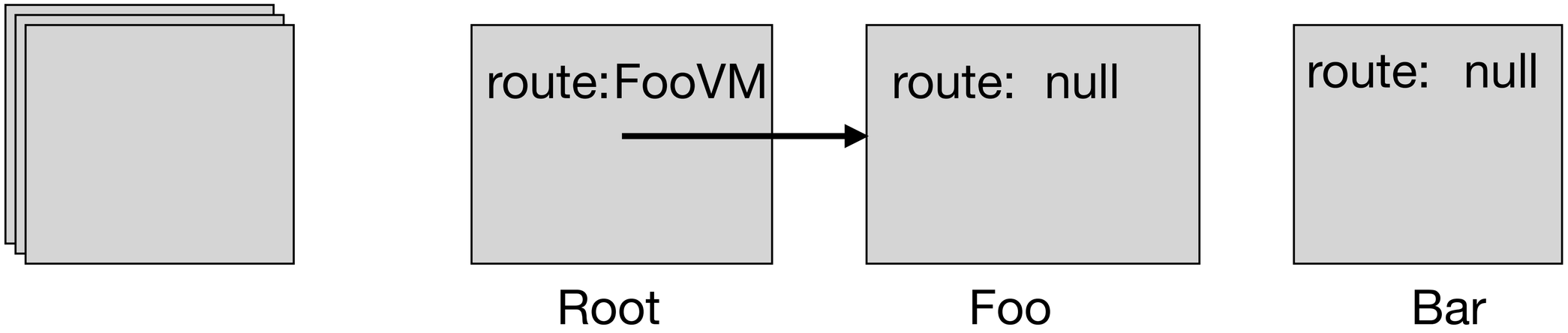


ViewModel:

var route: Observable<ViewModel?>

Architektur

(Deklarative) Navigation

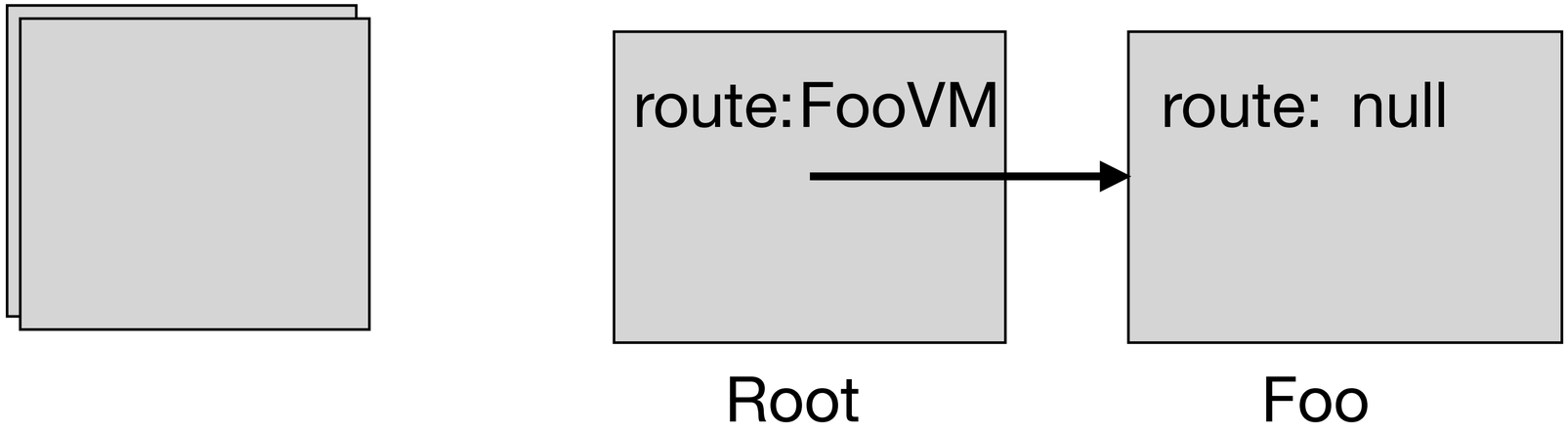


ViewModel:

var route: Observable<ViewModel?>

Architektur

(Deklarative) Navigation

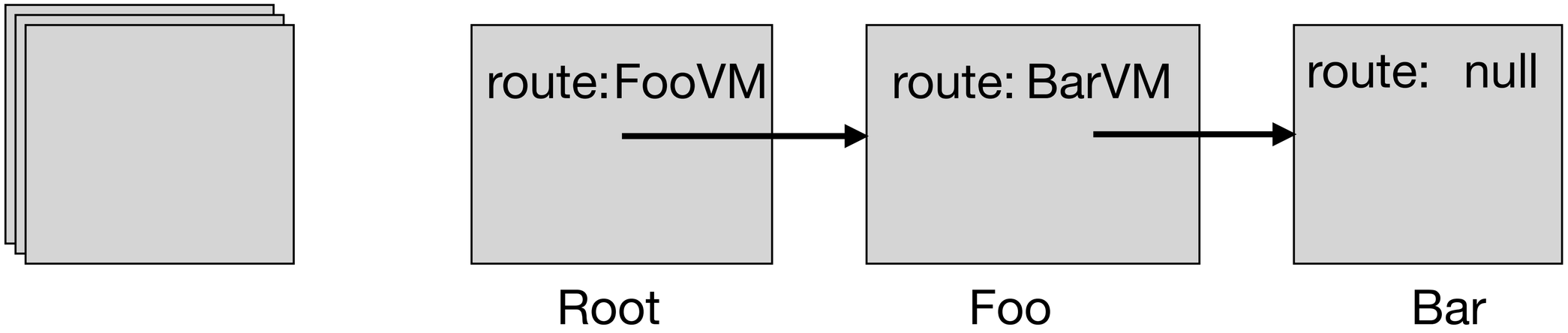


ViewModel:

var route: Observable<ViewModel?>

Architektur

(Deklarative) Navigation

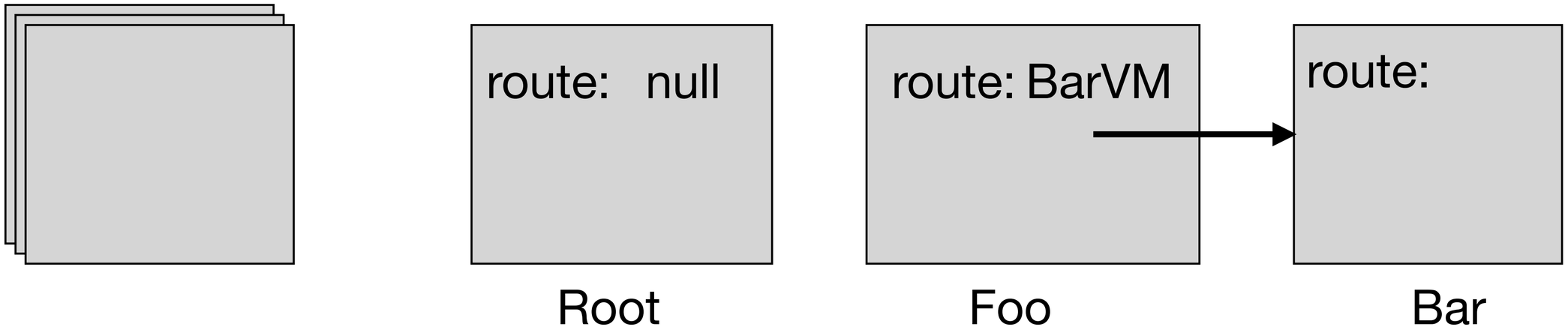


ViewModel:

var route: Observable<ViewModel?>

Architektur

(Deklarative) Navigation

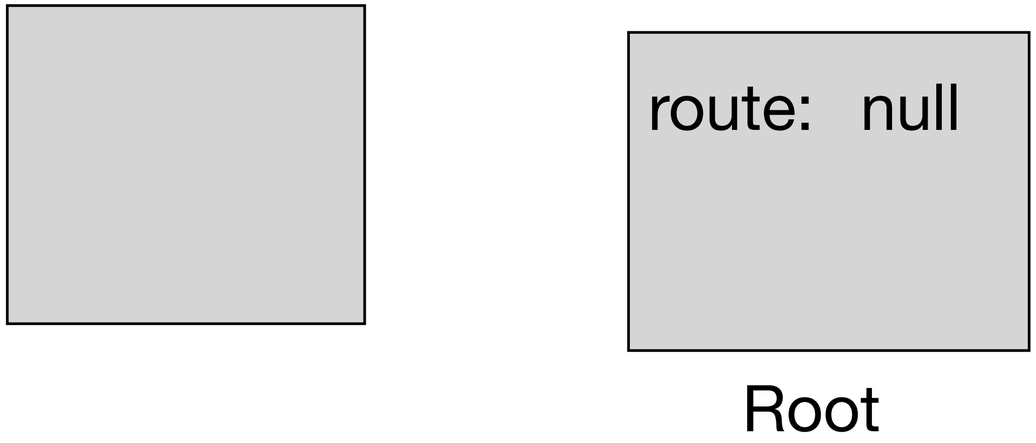


ViewModel:

var route: Observable<ViewModel?>

Architektur

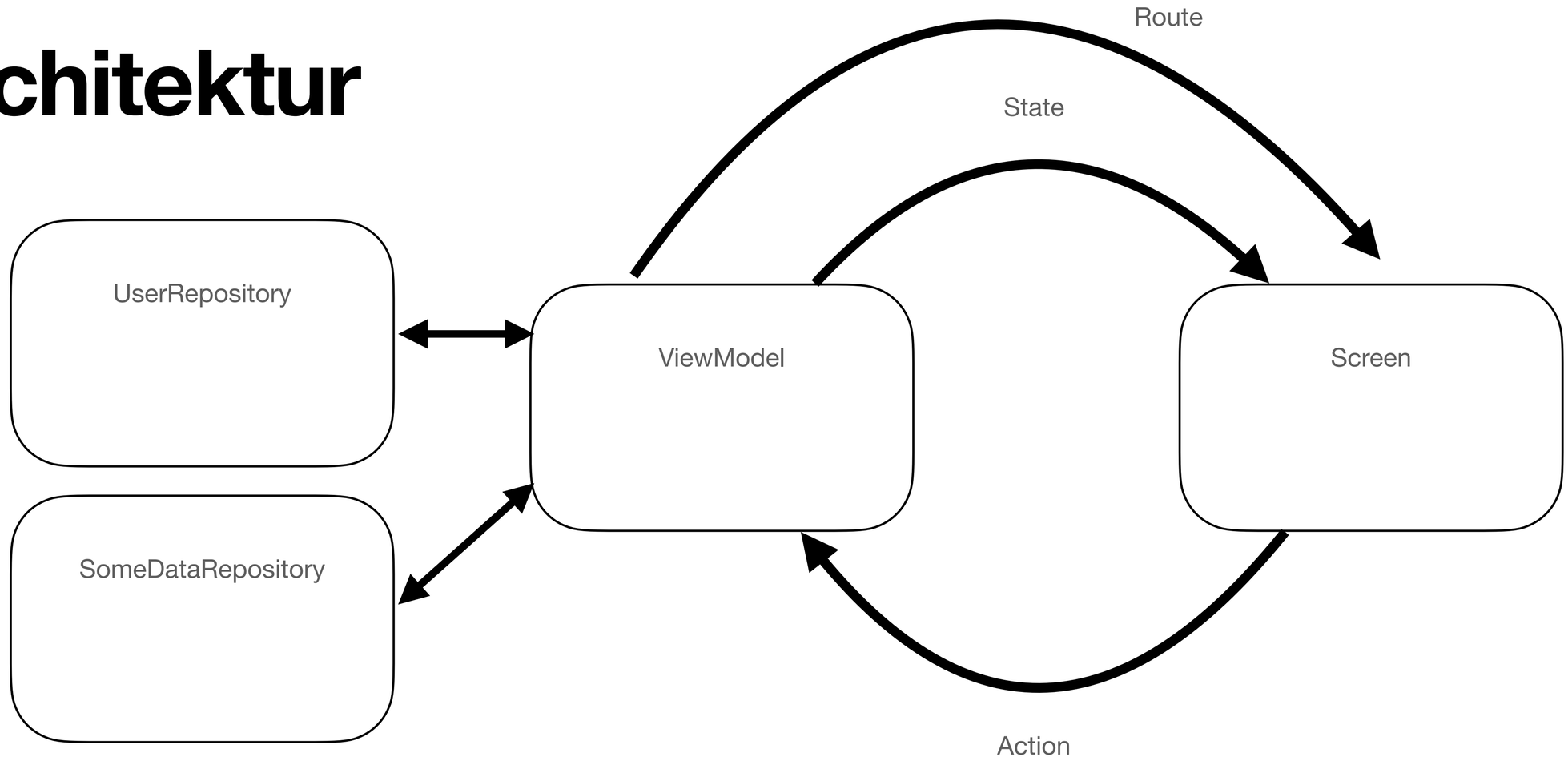
(Deklarative) Navigation



ViewModel:

var route: Observable<ViewModel?>

Architektur



Implementierung

Plan

- iOS zuerst da größte Reibung der Technologien erwartet
- Backend nebenbei
- Android nachziehen (hoffentlich ohne große Probleme und super schnell)

Implementierung

Backend

- ktor - Kotlin, simpel, extrem gut dokumentiert
- Keine Datenbank, Daten werden nur im Speicher gehalten (PoC mit Fokus auf Frontend)
- Funktionalität reduzieren auf das Nötigste

Implementierung
Änderungen des Scopes

- OAuth (Sign in with Apple, Google OAuth) gestrichen
- Usermanagement gestrichen
- Zahlungsanbindung für Spenden reingenommen

Implementierung
iOS

- Furchtbar lange Kompilierungszeiten: 2-5 Minuten (Debug), >8 Minuten (Production)
- EXC_BAD_ACCESS
- Drei-Sprachen-Problem: Kotlin -> Objective C -> Swift
- Integration von (manchen) Bibliotheken schwer bis unmöglich

Implementierung
Android

- Alles wunderbar, außer:
- Navigation!?
- Nicht deklarativ, nur imperativ - schade!

Status Quo

- Zwei Apps
- Spenden ist möglich mit Test-Kreditkarten
- Spenden empfangen ist möglich
- Geld ausgezahlt bekommen ist “möglich”
- App Clips / Instant Apps in Arbeit

Ausblick

UX Analyse

- UX ist sehr holprig an manchen Stellen
- Angedacht war ein Vertestung der Benutzererfahrung mit echten Benutzern, möglichst aus der Zielgruppe
- Covid-Inzidenz: 2280
- Zielgruppe ist besonders schutzbedürftig
- ???

Ausblick

Technologie

- Bibliothek des Zahlungsanbieters in den geteilten Code “runterziehen”
- Navigation auf Android “deklarativieren”
- Fokus der Architektur mehr auf Navigation
- Spielereien wie WebSockets

Danke für’s Zuhören