MVVM mit ReactiveCocoa

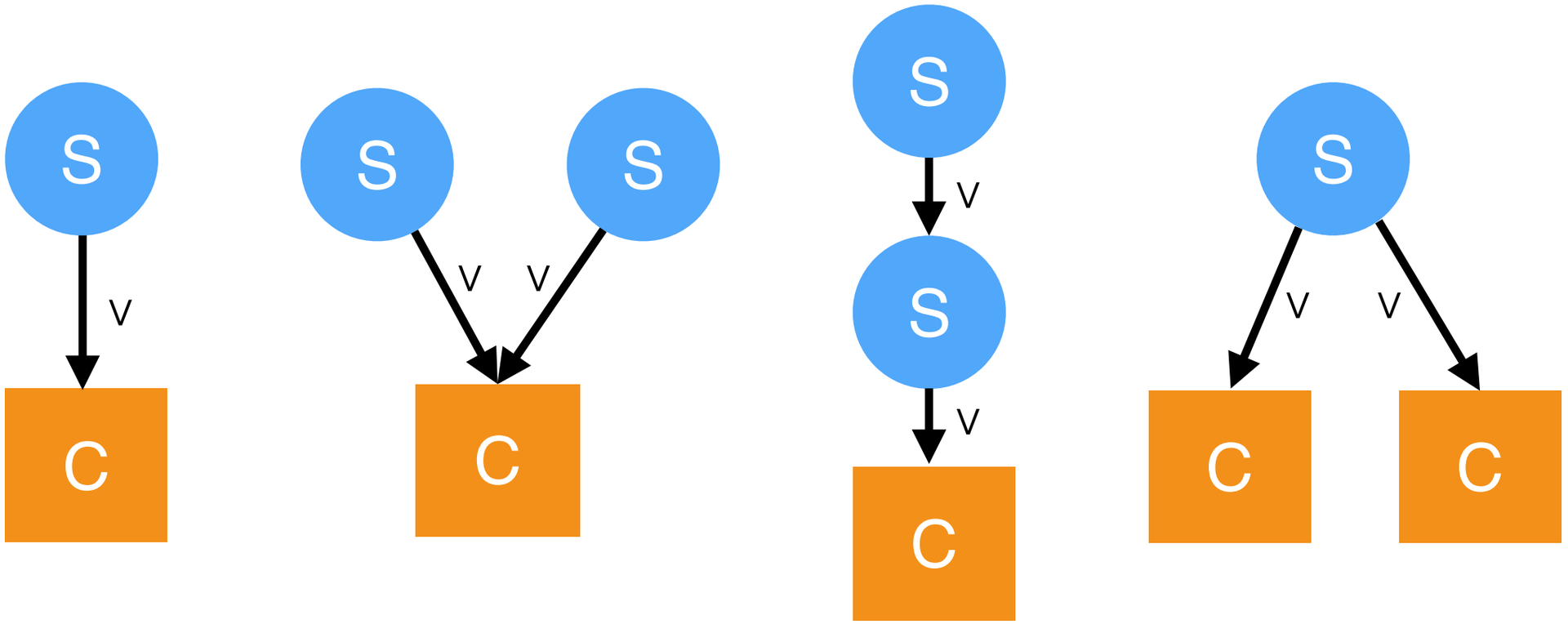
CocoaHeads Leipzig, 05. März 2014

Götz Fabian

ReactiveCocoa?

- Functional-Reactive Programming
- Auf Änderungen in Werten der Daten reagieren
- Data flow-driven
- `a := b + c`

ReactiveCocoa?



ReactiveCocoa!

- Implementierung in Cocoa
- Signale z.B. von
  - Nutzereingaben
  - UI-Aktionen
  - Asynchrone Netzwerkanfragen
  - GPS-Koordinaten

Signale

- Signale abonnieren

```
[signal subscribeNext: ^(id newValue) {
    NSLog(@"%@", newValue)
}];
```

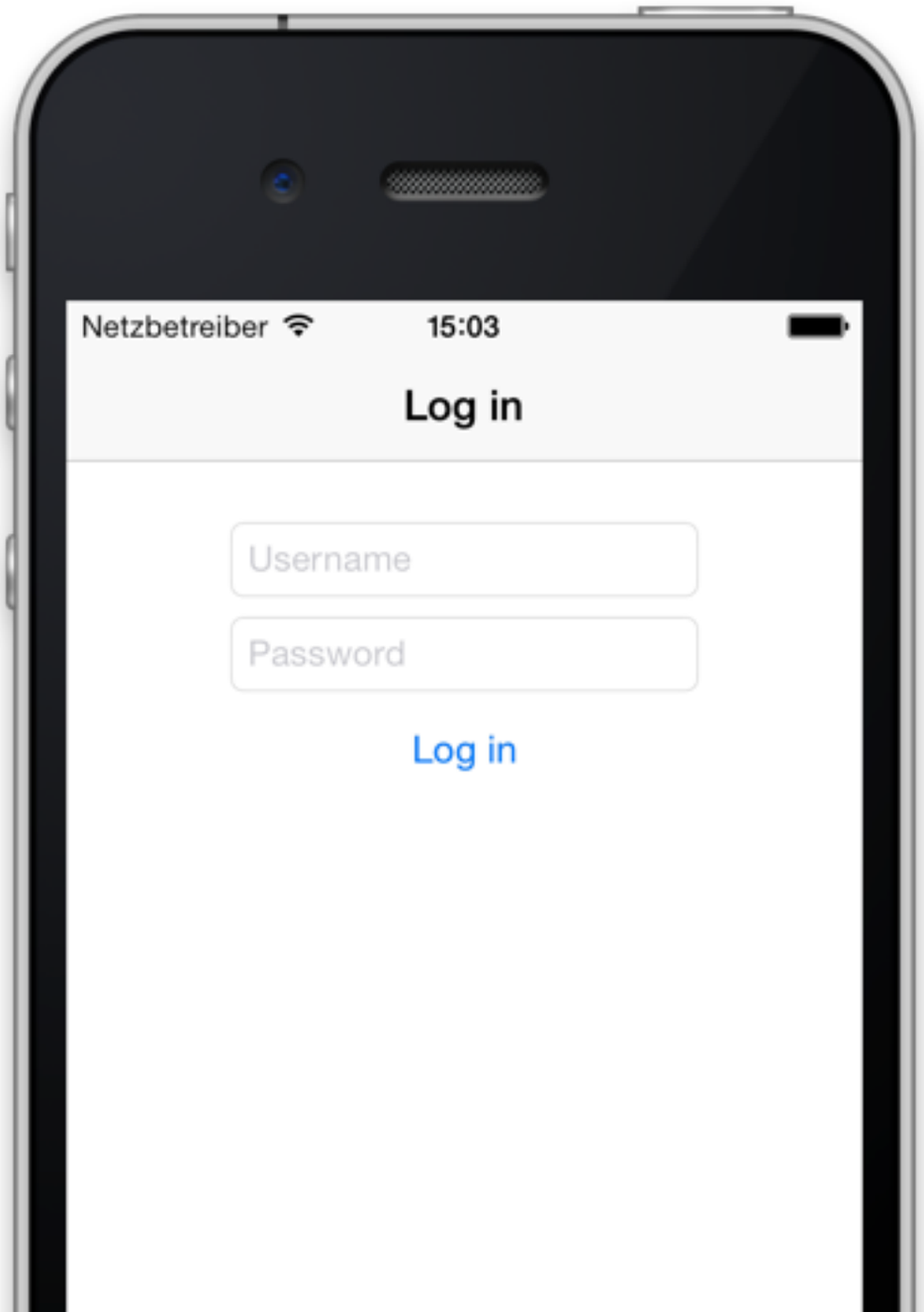
- Signale abbilden

```
[signal map: ^NSString *(NSNumber *number) {
    return [number stringValue];
 }];
```

- Signale binden

```
RAC(self, username) = textField.rac_textSignal;
```

Demo



- Login-Formular:
  - Benutzer validieren (E-Mail)
  - Passwort validieren (> 6 Zeichen)
  - „Log In“-Button aktivieren

Erweiterungen

- ReactiveCocoaIO

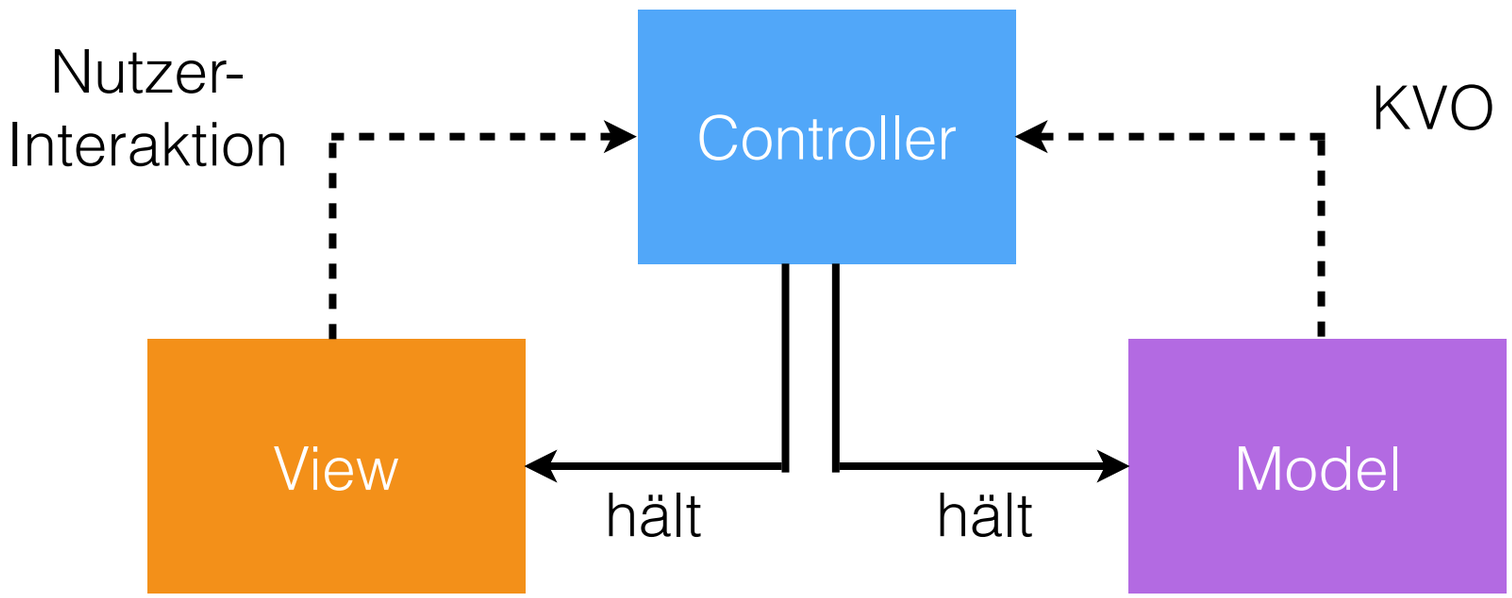
```
[[RCIOFileManager
  contentsOfDirectoryAtURL:url
  options:0]
  subscribeNext:^(RACSignal *signal) {…}]];
```

- ReactiveCocoaLayout

```
RCLAlignment(nameLabel) = @{
   rcl_rect:nameLabelRect,
   rcl_size:nameLabel.rcl_intrinsicContentSizeSignal,
   rcl_baseline:nameTextField
   };
```

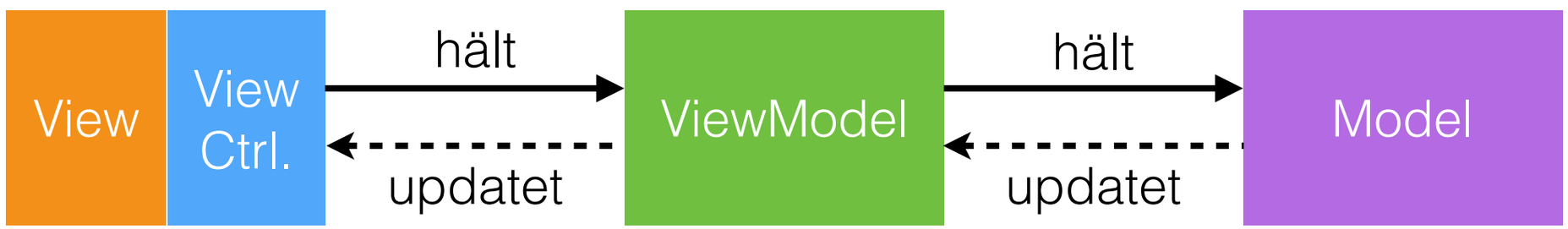
MVVM?

- Model-View-ViewModel
- Basiert auf MVC-Modell

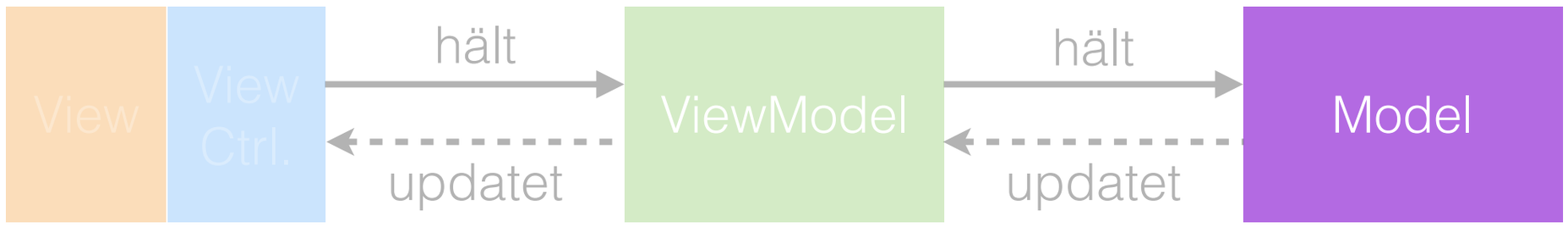


MVVM!

- Weiterentwicklung des Presentation Model
- Microsoft-Entwicklung



Model

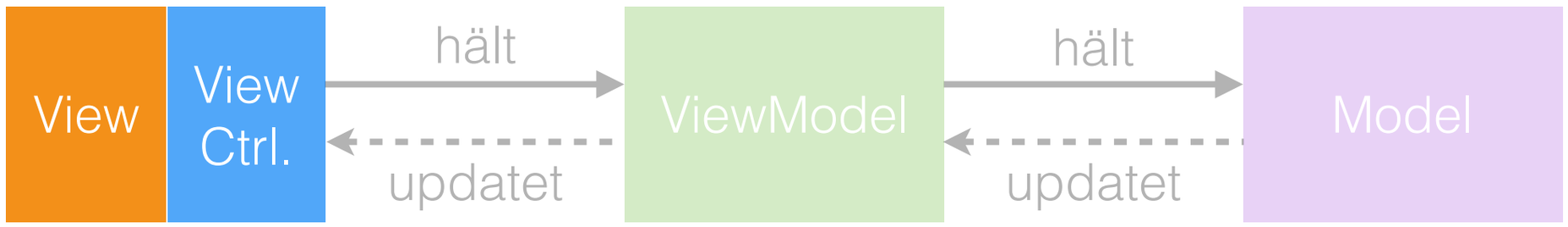


- Domänenmodell
- Datenzugriff

ViewModel

- Stellt Daten bereit für Views
- Hält gesamte Logik / Aktionen

View/ViewController

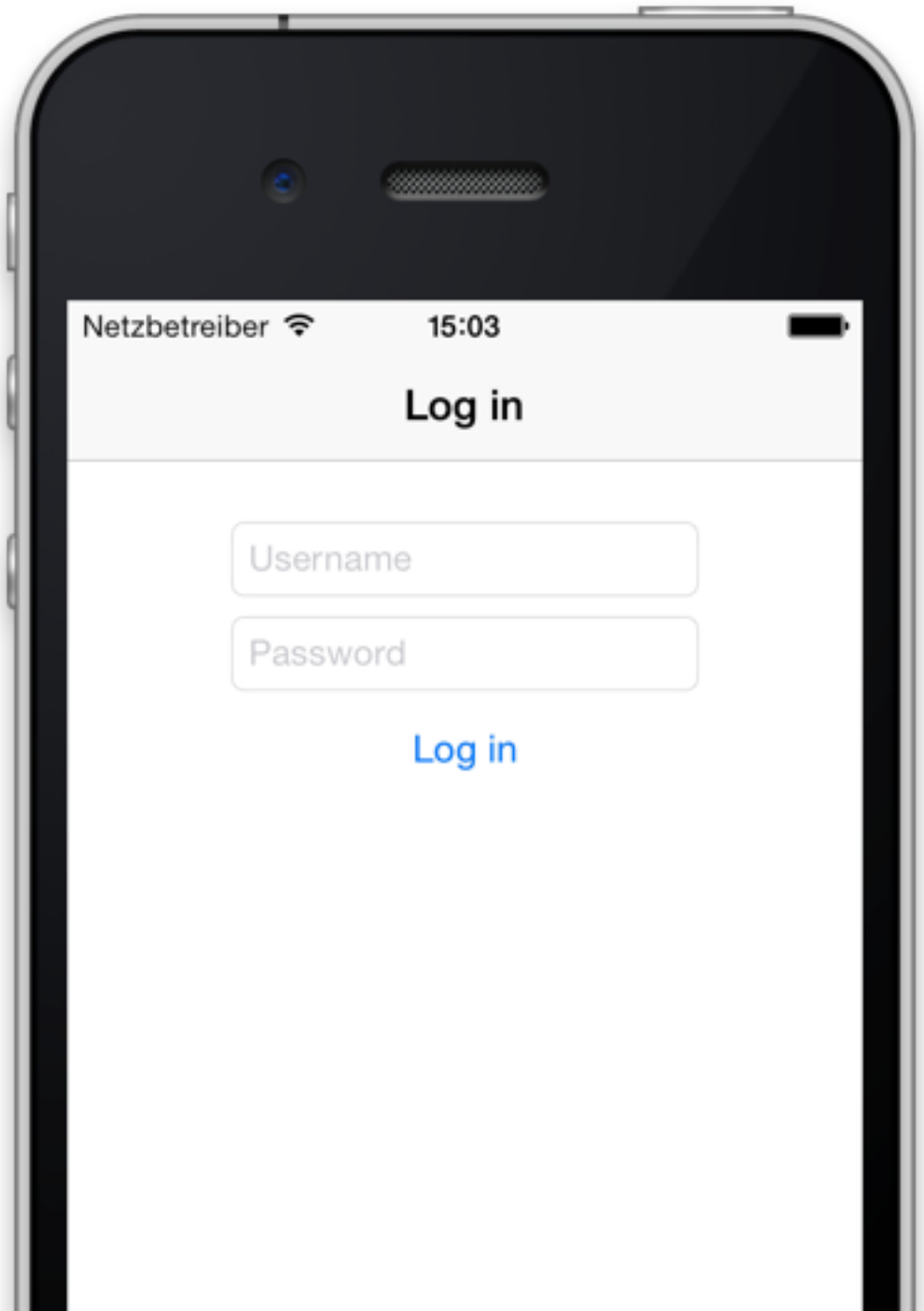


- UI-Elemente
- Stellt Daten vom ViewModel dar

Vorteile

- ViewModel ist von View entkoppelt
- Kleiner ViewController
- Logik ist im ViewModel
- UI einfacher zu ändern
- Testing:
  - UI muss nicht getestet werden
  - ViewModel kann getestet werden

Demo



- Umstellung auf ViewModel
- Testen des ViewModels