andrena OBJECTS

Experts in agile software engineering

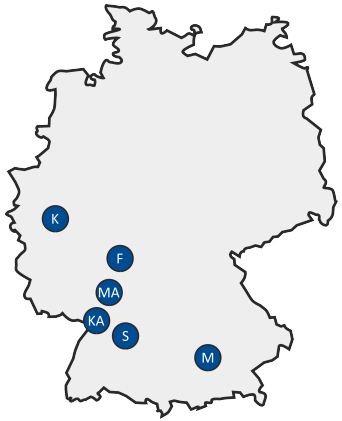
# App-Entwicklung mit React-Native, React und Redux

## - Ein Erfahrungsbericht

Rouven Röhrig und Raphael Knecht

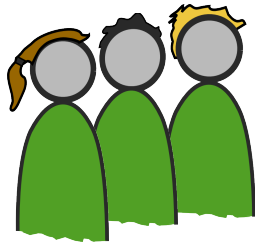
# andrena — Experts in agile software engineering

## Standorte:




Köln
Frankfurt
Mannheim
Karlsruhe
Stuttgart
München

## Mitarbeiter/innen:



mehr als 225

## Technologien/Kernthemen:

JavaScript
Big Data
Scrum
Android
React
Gradle
JPA
Clean Code
WPF
Angular
TypeScript
Agil
AWS
Rest
Java
Software Craftsmanship
Microservices
TFS
Azure
Jenkins
C#
TDD
NoSQL
WCF
iOS
Spring Boot
Docker
JavaEE
Swift
Reactive Programming
XP
Maven
Continuous Delivery
Cloud Foundry

© 2017 andrena objects ag

# Referenten

- Rouven Röhrig
- Full-Stack-Entwickler
- Java / JavaScript
- ASE / Clean Code
- rouven.roehrig@andrena.de

- Raphael Knecht
- Full-Stack-Entwickler
- Java / JavaScript
- raphael.knecht@andrena.de

© 2015 andrena objects ag

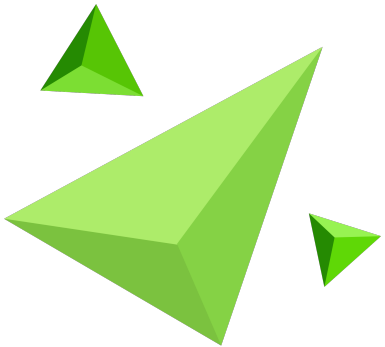
# Agenda

1. Motivation
2. Vorstellung der Frameworks
3. Unsere Erfahrungen mit der App-Entwicklung
4. Fazit

© 2017 andrena objects ag

# Motivation

## Warum App-Entwicklung mit React-Native/React/Redux?



© 2017 andrena objects ag

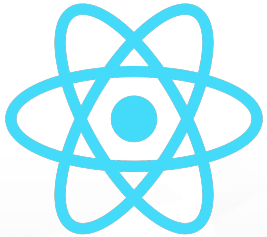
# Motivation

## Warum App-Entwicklung mit React-Native/React/Redux?

- Gemeinsame Codebasis für Android und iOS: „learn once, write anywhere“
- Entwicklung in bekannter Sprache (JavaScript) mit bekannten Konzepten (z.B. Asynchronität, „deklarative views“, „responsive design“)
- Einsatz moderner Technologien
- Große Community und stetige Weiterentwicklung der Frameworks

© 2017 andrena objects ag

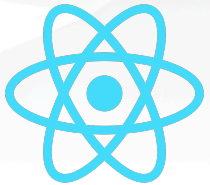
# React und React-Native



© 2017 andrena objects ag

# React

## JavaScript-Framework für (Web)-User-Interfaces

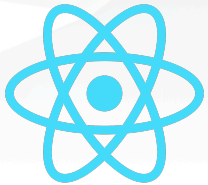


- Ermöglicht deklarative Views
- Komponentenbasiert
- Effizientes Rendering
- Deterministisches Rendering
- Ist nur die View
- Ermöglicht JSX

© 2017 andrena objects ag

# Was ist JSX

## Syntactic sugar für Markup in JavaScript



```
const style = { flex: 1 };
const MyComponent = () => (
    <View style={style}>
        <Text>Hello World!</Text>
    </View>
);
```

© 2017 andrena objects ag

# React-Native

## Framework zum Bau nativer Apps in JavaScript

- React-Komponenten (Button, Text, FlatList…)
- Glue-Code: JavaScript ⟷ Android/iOS
- Styling mit CSS und insb. Flexbox
- Buildtools
- Entwicklungstools

© 2017 andrena objects ag

# React-Native

## Komponenten und APIs

- Komponenten
  - Text, Button, InputField,…
  - Slider, Picker, FlatList, ...
- APIs
  - Vibration, Gelocation, …

© 2017 andrena objects ag

# React-Native

## Build- und Entwicklungstools

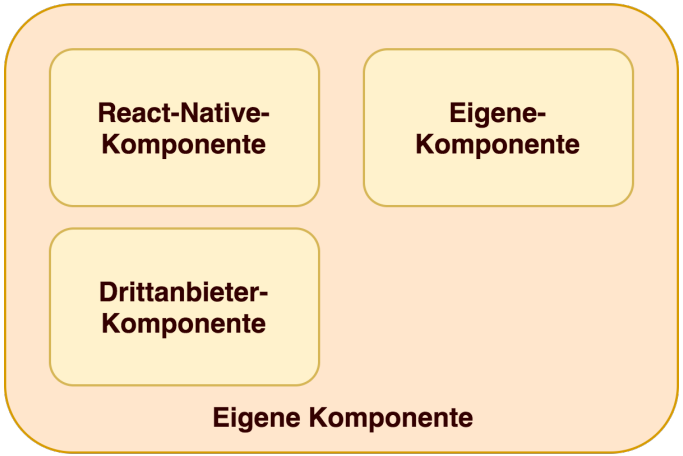
- Native App installieren
- Native App bauen

- Live Reload
- Hot Reload
- Remote JavaScript Debugging
- Inspector

© 2017 andrena objects ag

# React-Native mit React

- Beliebige Kombination von React-(Native)-Komponenten
- Ermöglicht Strukturierung und Wiederverwendung von Komponenten




© 2017 andrena objects ag

# Redux

„Predictable state container“

© 2017 andrena objects ag

# Redux

## 3 Prinzipien und 3 Bausteine

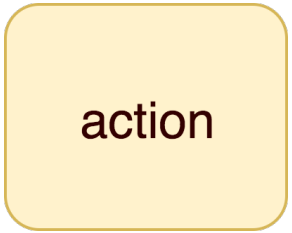
- “Single source of truth”
- “State is read-only”
- “Changes are made with pure functions”

- 3 Bausteine
  - Actions
  - State
  - Reducer

© 2017 andrena objects ag

# Redux

## Actions – „Was soll sich ändern“



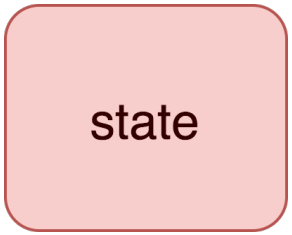
- Eindeutiger Typ
  - z.B. „ADD_TODO“
- Beliebiger Payload
  - z.B. Titel

- **Best Practice**: Action-Creator
  - Funktion, die Actions erstellt

© 2017 andrena objects ag

# Redux

## State – „Wo soll es sich ändern“



- Ein einziger Zustand
- Hierarchisch organisiert

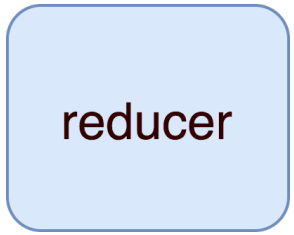
- **Store** stellt Methoden zur Verfügung

- **Best Practice**: Selector
  - Funktion für Zugriff auf State

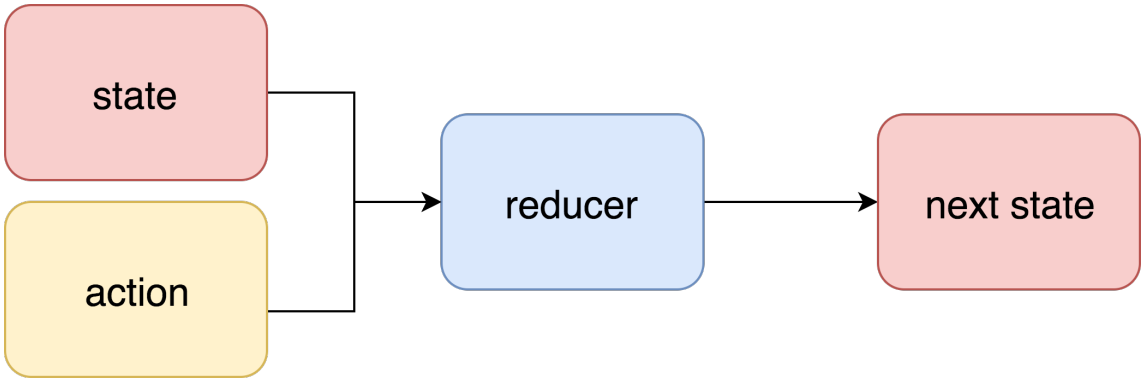
© 2017 andrena objects ag

# Redux

## Reducer – „Wie soll es sich ändern“



- Reine Funktion (keine Seiteneffekte)
- Berechnet nächsten Zustand für Teil-State
- Deterministisch




© 2017 andrena objects ag

# Redux

## Ducks Pattern: Ein Modul → Eine Datei

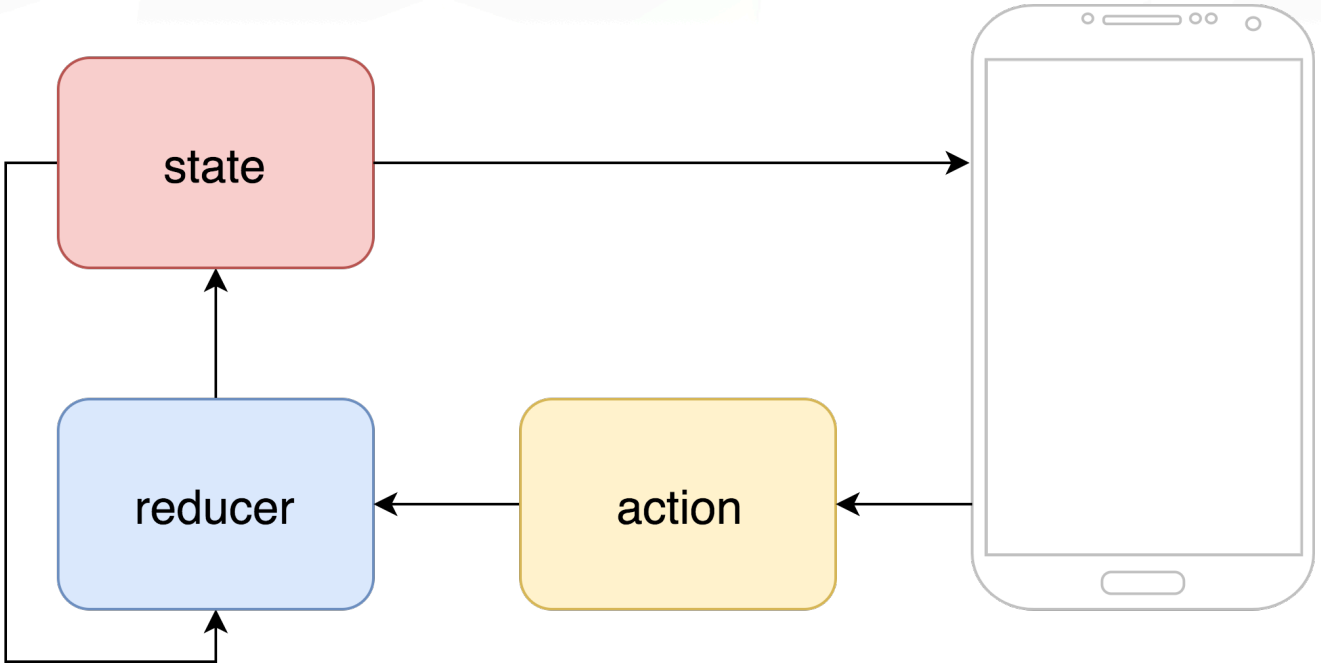
- Reducer ist **default export**
- Action Creator sind **named exports**
- Action types haben die Form **„app-name/reducer-name/ACTION_TYPE“**
- Action types dürfen exportiert werden

- Empfehlung: Selectors sind **named exports**

Quelle: https://github.com/erikras/ducks-modular-redux

© 2017 andrena objects ag

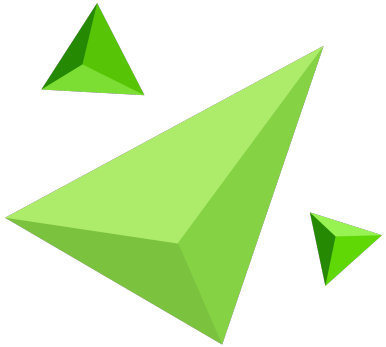
# React + React-Native + Redux




© 2017 andrena objects ag

# Erfahrungsbericht

## Unsere Erfahrungen mit der App-Entwicklung und den Frameworks



© 2017 andrena objects ag

# Handhabung von Seiteneffekten

© 2017 andrena objects ag

# Redux

## Wie und wo behandeln wir Seiteneffekte?

middleware

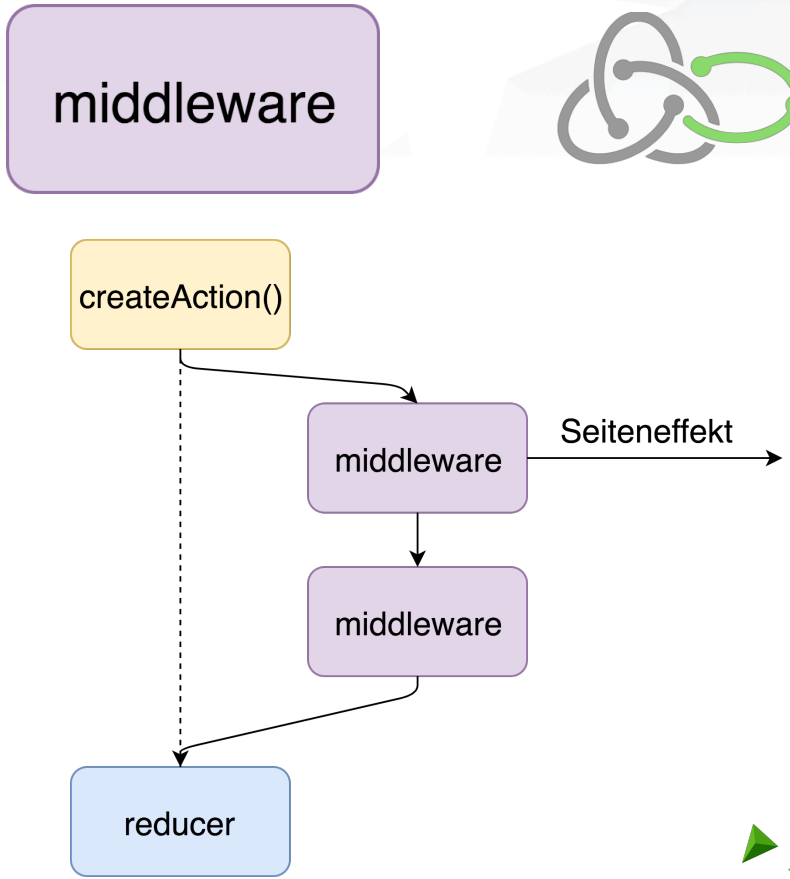
- Reducer sind reine Funktionen
  und dürfen/sollten keine Seiteneffekte beinhalten
- Eine Action sollte nur von einem Reducer behandelt werden
- Seiteneffekte werden typischerweise über eine **Middleware** abgebildet
- Zu den bekanntesten zählen Redux-Thunk und Redux-Saga

© 2017 andrena objects ag

# Redux

## Redux - Middleware

- Jede Middleware wird von jeder Action durchlaufen
- Erweiterungspunkt für verschiedene Anwendungsgebiete (z.B. Logging)
- Häufiges Anwendungsgebiet sind Seiteneffekte (z.B. Anfragen an eine API)

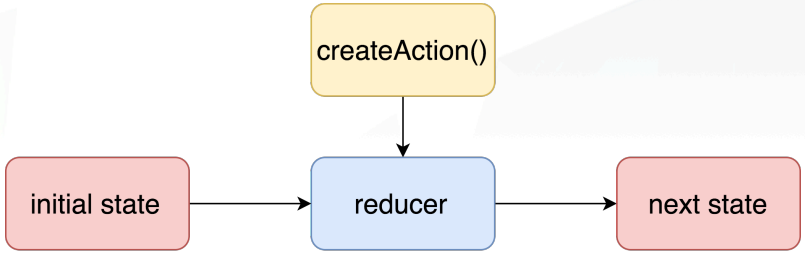


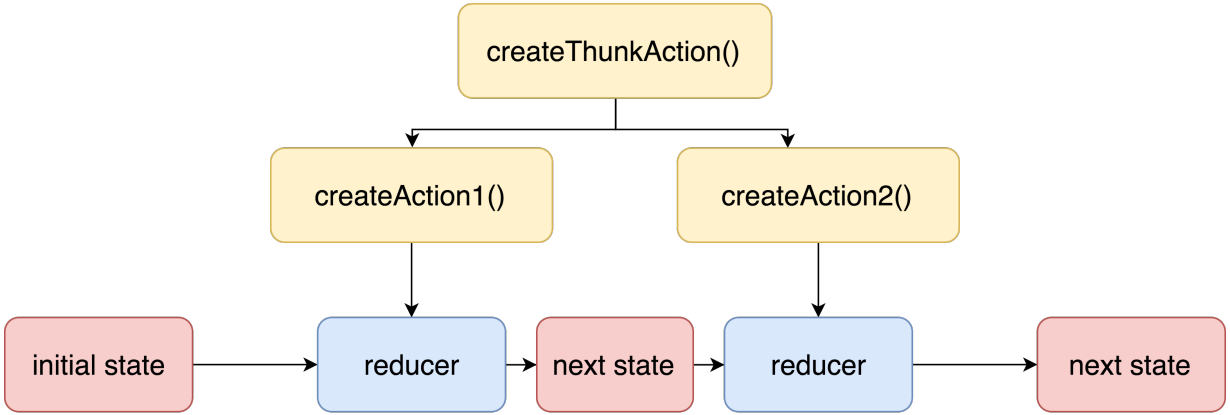

© 2017 andrena objects ag

# Redux

## Redux - Thunk

- Actions müssen nicht mehr nur „plain objects“ sein
- Actions können weitere Actions anstoßen



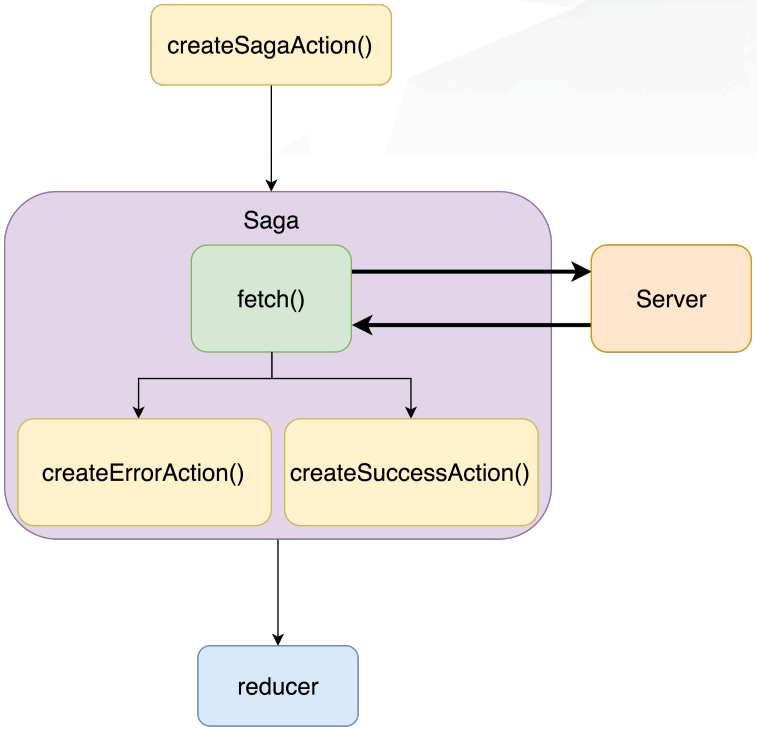




© 2017 andrena objects ag

# Redux

## Redux - Saga

- Eine Saga reagiert auf gewisse Actions und ergänzt/behandelt deren Seiteneffekte
- Verschiedene Saga-Effekte vereinfachen die Behandlung von Asynchronität und Synchronität bei Seiteneffekten




© 2017 andrena objects ag

# Warum haben wir uns für Sagas entschieden?

- Klare Trennung zwischen Redux-Actions („plain objects“) und Actions mit Seiteneffekten
- Saga-Effekte unterstützen bei der Behandlung von Asynchronität in einem synchronen Programmfluss (Stichwort: „Generatorfunction“)
- Saga-Effekte kapseln Implementierungsdetails und unterstützen dabei, übersichtlichen Code zu schreiben

© 2017 andrena objects ag

# Qualitätssicherung - Code

## Einheitliche Codeformatierung und Design Pattern

© 2017 andrena objects ag

# ESLint

- ESLint unterstützt eine einheitliche Formatierung und einheitliche Design Pattern
- Fördert „Best Practice“:
  - „comma dangle“
  - „prefer stateless function“

© 2017 andrena objects ag

# Qualitätssicherung - Unittests

## Welche Möglichkeiten bietet Jest?

© 2017 andrena objects ag

# Jest - Automatisiertes Testen

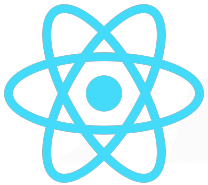
- Auf React angepasstes Testing-Framework
- Basierte ursprünglich auf Jasmine

- Wir verwenden Jest für **Unittests**
  - Schnelle Testausführung
  - Snapshottests
  - Zunehmend bessere IDE-Unterstützung

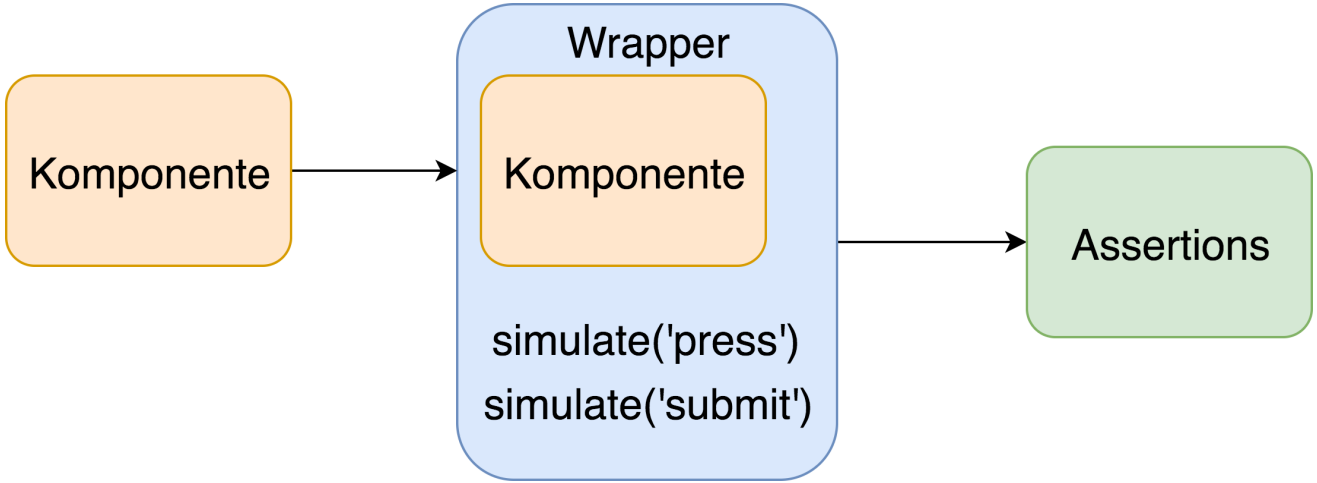
© 2017 andrena objects ag

# Jest - Automatisiertes Testen

## React-Native, React - Komponenten



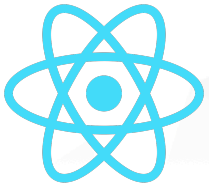
- Funktionalität (Enzyme)



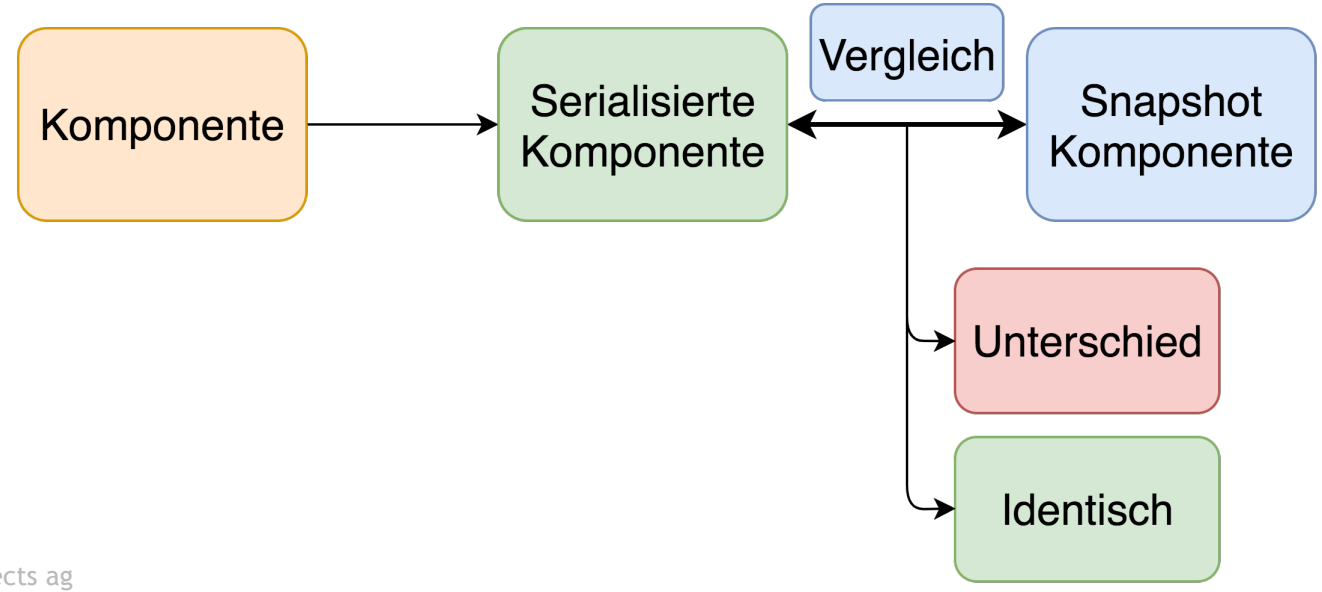

© 2017 andrena objects ag

# Jest - Automatisiertes Testen

## React-Native, React - Komponenten



- Snapshot-Tests

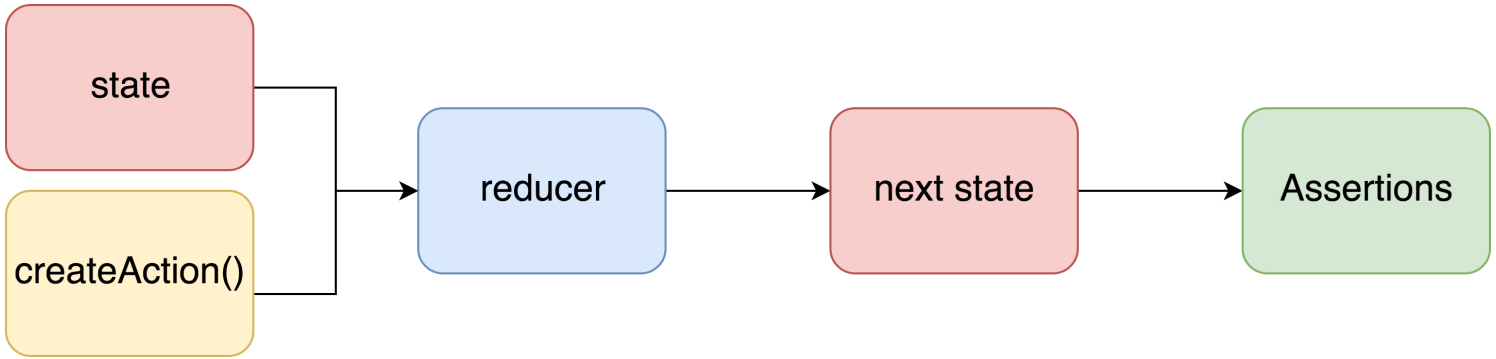



© 2017 andrena objects ag

# Jest - Automatisiertes Testen

## Redux - Reducer und Action Creator

- Reducer sind „pure functions“ und somit sehr gut zu testen
- Gemeinsame Tests mit Action Creator bieten sich an (weniger fragil)
- Tests orientieren sich an https://redux.js.org

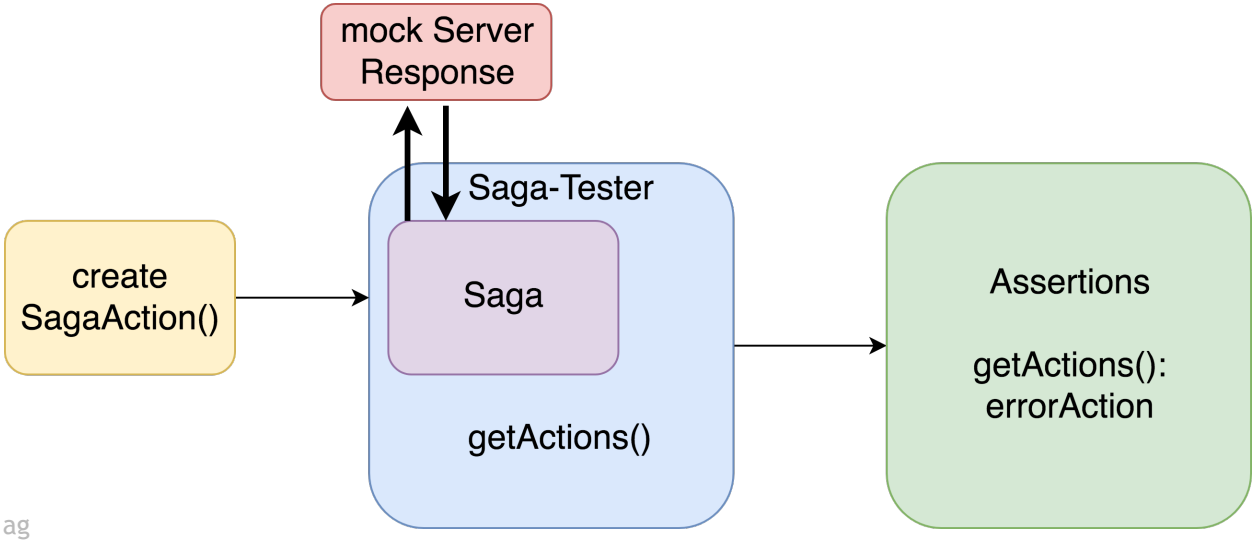



© 2017 andrena objects ag

# Jest - Automatisiertes Testen

## Redux - Saga

- Abhängigkeiten können gemockt werden
- Testen/Überprüfen der Actions

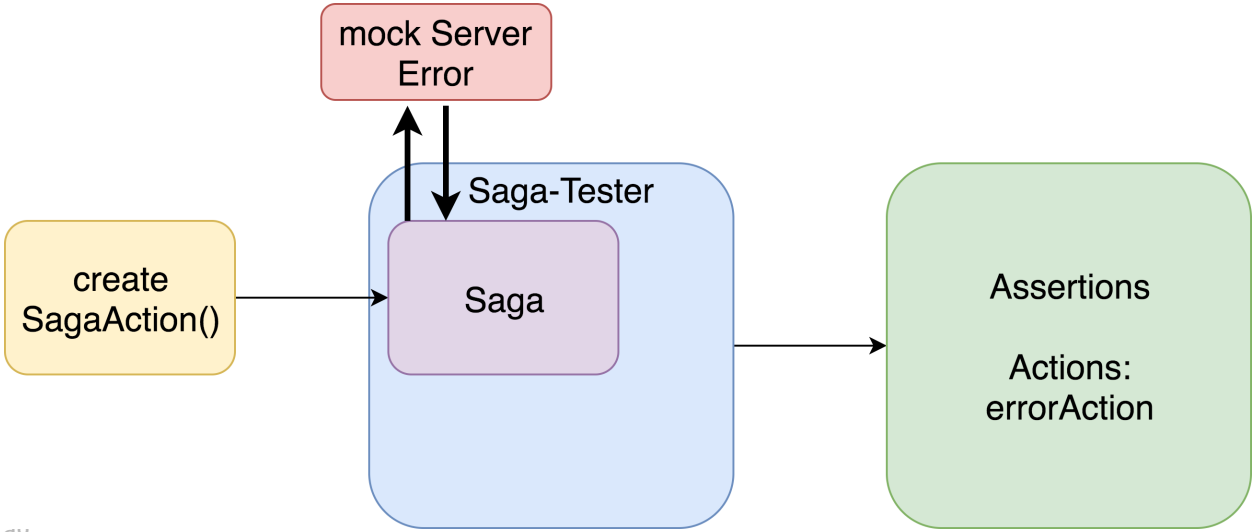



© 2017 andrena objects ag

# Jest - Automatisiertes Testen

## Redux - Saga

- Abhängigkeiten können gemockt werden
- Testen/Überprüfen der Actions




© 2017 andrena objects ag

# Projektstrukturierung

© 2017 andrena objects ag

# Projektstrukturierung

## Verzeichnisstruktur

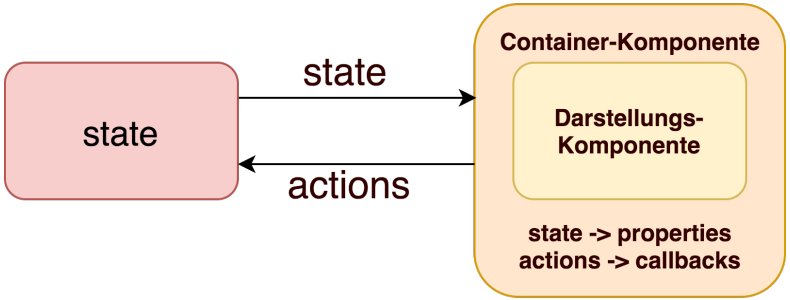
- Trennung Test- und Source-Code
- Source-Code
  - Module (Redux – Ducks-Pattern)
  - Sagas
  - Komponenten
    - Container (Kennen Redux)
    - Darstellung (Kennen Redux nicht)

© 2017 andrena objects ag

# Projektstrukturierung

## Container- und Darstellungs-Komponenten

- Container verdrahten Redux und React
  - State -> Properties
  - Actions -> Callbacks
  - (Strikt Action Creators und Selectors)




- Darstellungskomponenten können
  - Properties darstellen
  - Aktionen initiieren

© 2017 andrena objects ag

# Atomic Design

## Komponenten-Design in Anlehnung an Atomic Design

- Atome
  - TextInput, Image, Label, Button, …
- Moleküle
  - SearchPanel (TextInput + Button), ListItem, LoginButton, …
- Organismen
  - Header, Footer, Form, ImagePicker, …
- Templates
- Seiten

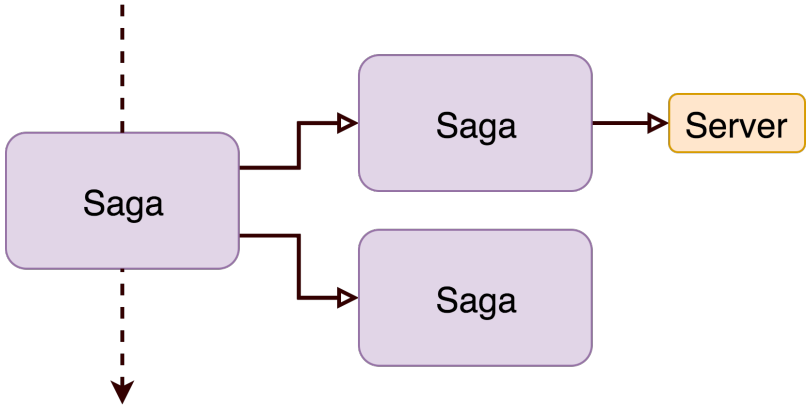
Quelle „Atomic Design“: http://atomicdesign.bradfrost.com/

© 2017 andrena objects ag

# Projektstrukturierung

## Redux und Redux-Sagas

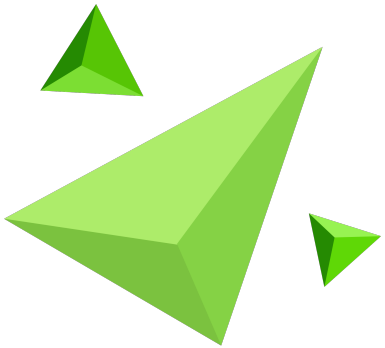
- Sagas strukturieren in Anlehnung an Ducks Pattern
- Keine Sagas, die alles können
- Eine Verantwortlichkeit pro Saga / Redux-Modul




© 2017 andrena objects ag

# Build-Automatisierung

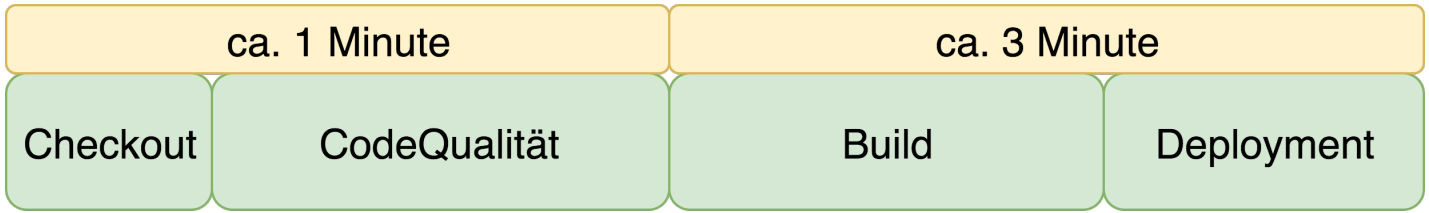
## Sind manuelle Schritte notwendig?



© 2017 andrena objects ag

# Build-Automatisierung

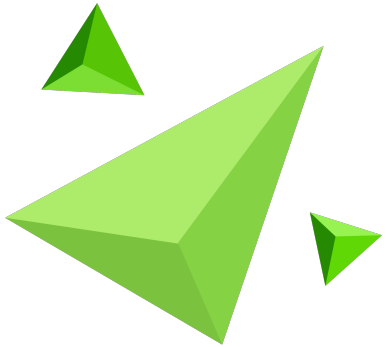
- Wir verwenden Jenkins und das Pipeline-Plugin
- Automatisiert:
  - Messung der Code-Qualität,
  - Build der Installationsdateien
  - Deployment




© 2017 andrena objects ag

# Herausforderungen

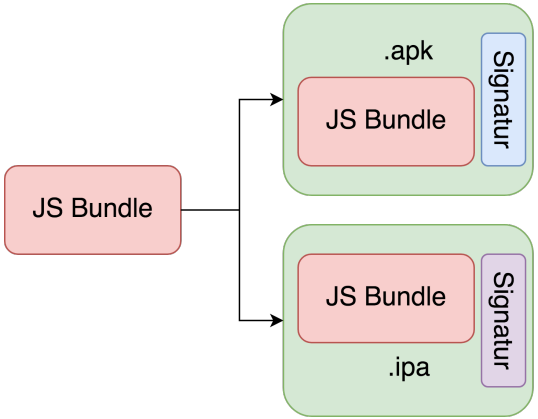
## Keine Erfahrung in der App-Entwicklung - unsere Herausforderungen



© 2017 andrena objects ag

# Herausforderungen

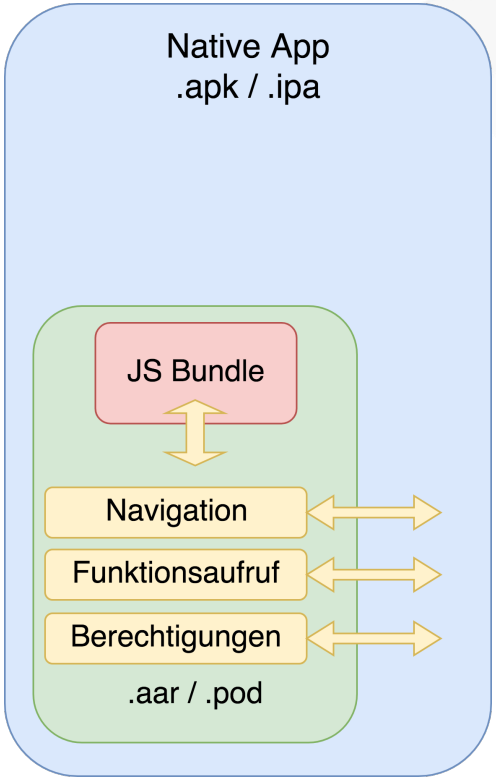
- Deployment in AppStore und PlayStore
  - Code-Signatur und automatisiertes Deployment




© 2017 andrena objects ag

# Herausforderungen

- Integration in native Android- und iOS-App
  - Nativer Glue-Code
  - Kommunikation zwischen nativer App und JavaScript
  - Automatisierte Tests




© 2017 andrena objects ag

# Herausforderungen

- Natives Verhalten bzw. plattformspezifisches UX/UI
- JavaScript-Package-Abhängigkeiten
  - React-Native v. 0.50.0
  - Alpha- und Beta-Versionen
  - Breaking Changes
  - Kompatibilität mit Testingframeworks

© 2017 andrena objects ag

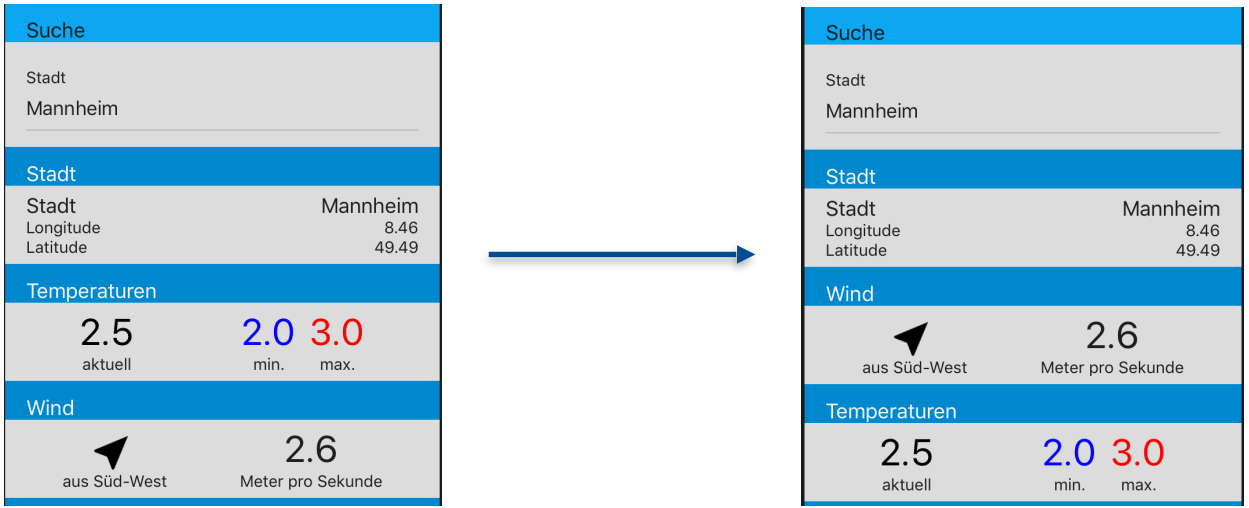
# Argumente für React-Native

## Welche Features überzeugen uns?

© 2017 andrena objects ag

# Hot Reloading

- Hot Reloading ermöglicht es Änderungen direkt anzuzeigen, dabei bleibt der App- bzw. Redux-Zustand erhalten




© 2017 andrena objects ag

# Debugging

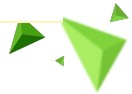
```
action sagas/wetter-saga/WETTER_SUCHE @ 12:58:45.935
  prev state ▶ {temperaturen: {…}, stadt: {…}, stadtSuche: "", wetter: Array(0)}
  action     ▶ {type: "sagas/wetter-saga/WETTER_SUCHE", stadt: "Berlin"}
  next state ▶ {temperaturen: {…}, stadt: {…}, stadtSuche: "", wetter: Array(0)}
action WetterApp/stadtSuche/SET_STADT_SUCHE @ 12:58:45.936
  prev state ▶ {temperaturen: {…}, stadt: {…}, stadtSuche: "", wetter: Array(0)}
  action     ▶ {type: "WetterApp/stadtSuche/SET_STADT_SUCHE", stadt: "Berlin", @@redux-saga/SAGA_ACTION: true}
  next state ▶ {temperaturen: {…}, stadt: {…}, stadtSuche: "Berlin", wetter: Array(0)}
action wetterapp/wetter/SET_TEMPERATUREN @ 12:58:46.228
  prev state ▶ {temperaturen: {…}, stadt: {…}, stadtSuche: "Berlin", wetter: Array(0)}
  action     ▶ {type: "wetterapp/wetter/SET_TEMPERATUREN", temperaturen: {…}, @@redux-saga/SAGA_ACTION: true}
  next state ▶ {temperaturen: {…}, stadt: {…}, stadtSuche: "Berlin", wetter: Array(0)}
```

© 2017 andrena objects ag

# Debugging

```
action WetterApp/stadtSuche/SET_STADT_SUCHE @ 12:58:45.936
  prev state ▼ {temperaturen: {…}, stadt: {…}, stadtSuche: "", wetter: Array(0)}
               ▶ stadt: {}
                 stadtSuche: ""
               ▶ temperaturen: {}
               ▶ wetter: []
               ▶ __proto__: Object
  action     ▼ {type: "WetterApp/stadtSuche/SET_STADT_SUCHE", stadt: "Berlin", @@redux-saga/SAGA_ACTION: true}
                 stadt: "Berlin"
                 type: "WetterApp/stadtSuche/SET_STADT_SUCHE"
                 @@redux-saga/SAGA_ACTION: true
               ▶ __proto__: Object
  next state ▼ {temperaturen: {…}, stadt: {…}, stadtSuche: "Berlin", wetter: Array(0)}
               ▶ stadt: {}
                 stadtSuche: "Berlin"
               ▶ temperaturen: {}
               ▶ wetter: []
               ▶ __proto__: Object
```

© 2017 andrena objects ag



# Schnelle Testausführung

- Direkte Testausführung ohne vorherigen Build
- Dauer im Sekundenbereich
- Ausführung über Command Line oder IDE

```
Test Suites: 209 passed, 209 total
Tests:       801 passed, 801 total
Snapshots:   161 passed, 161 total
Time:        12.387s, estimated 15s
Ran all test suites.
✨  Done in 13.00s.
```

© 2017 andrena objects ag

# Snapshottests

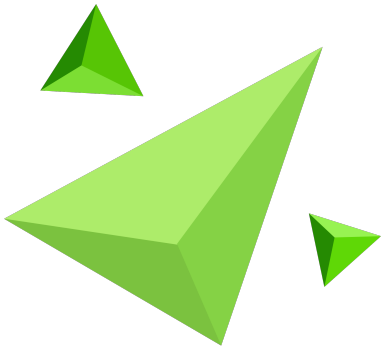
- Snapshottests zeigen Änderungen bei der Darstellung direkt an

```
Received value does not match stored snapshot 1.

- Snapshot
+ Received

@@ -14,11 +14,11 @@
        Array [
          Object {
            "color": "#212121",
          },
          Object {
-           "color": "red",
+           "color": "orange",
            "fontSize": 32,
          },
```

© 2017 andrena objects ag

# Fazit

## App-Entwicklung mit React-Native



© 2017 andrena objects ag

# Fazit

- Einstieg in React-Native ohne JavaScript-Erfahrung durchaus anspruchsvoll

- Probleme durch wenig Erfahrung in der App-Entwicklung
  - „ohne nativen Code geht es nicht“
  - Signatur und Deployment bei iOS
  - Plattformspezifisches UX/UI

- React-Native Versionen
  - Monatliche Updates mit teilweise „Breaking Changes“

© 2017 andrena objects ag

# Fazit

- JavaScript-Frameworks: „learn once, write anywhere“
  - Synergieeffekte z.B. Webentwicklung mit React/Redux
- Strikte Trennung von Darstellung und Zustand (bzw. Logik)
- Redux als “Single source of truth” -> deterministisches Verhalten
- Trennung von Seiteneffekten und „normalen“ Redux-Actions
- JavaScript für bekannte Konzepte (z.B. Asynchronität, Styling mit Flexbox )

© 2017 andrena objects ag

# Fazit

- Codeänderungen werden sofort angezeigt
- Debugging umfangreich und einfach
- Unittests
  - schnelle Ausführung
  - Darstellungskomponenten können getestet werden
  - hohe Testabdeckung >90% möglich

© 2017 andrena objects ag