InterFace AG
the face of informatics

ADF Mobile konkret – Best Practices Live erklärt

Jan Ernst

Zur Person:

- Jan Ernst
- Bachelorstudium Wirtschaftswissenschaften mit Schwerpunkt Wirtschaftsinformatik (4. Semsester) FAU Erlangen/Nürnberg
- Werkstudent bei InterFace AG

Ziel des Vortrags:

- Erfahrungen mit Oracle ADF Mobile teilen
- Daraus abgeleitete Best Practices weitergeben

Gliederung

- Einführung
- Generelles zum Design mobiler Anwendung
- Design einer Anwendung mit ADF Mobile
  - UI
    - Unterschiede Android und iOS Design
  - Business Logik
  - Data Caching
- Fazit

Einführung

- Oracle ADF Mobile:
  - Von Oracles Application Development Framework (ADF) abgeleitete Mobile Edition
  - Basiert auf Apache Cordova (PhoneGap)
  - Plattformunabhängig
  - Bietet die Möglichkeit bestehendes Wissen in Java, Oracle ADF oder generell Webentwicklung weiter zu benutzen
  - Teil der Oracle Mobile Suite, zielt ab auf Kunden, die bereits Oracle Technologien benutzen

Einführung

- Versucht folgende Schwierigkeiten bei der Entwicklung mobiler Anwendungen zu verbessern:
  - Verschiedene Plattformen (Android, iOS,…)
  - Integration in bestehende Enterprise Architektur
  - Sicherheit mobiler Daten
  - Möglichst effizient mobile Anwendungen fürs Unternehmen gestalten

Design mobiler Anwendungen

- Andere Anforderungen als z.B.: bei Webanwendungen
  - Viele unterschiedliche Devices mit unterschiedlichen Bildschirmgrößen (Smartphone Tablet)
  - Möglichkeit der Gestensteuerung
  - Kleinere Bildschirmgrößen als am Desktop

- Zwei Konzepte um eine App zu schreiben:
  - Bloße Darstellung von Informationen in Listen etc.
  - Überlegung „Was will der User?“ Anhand dieser Überlegung erstellen eines User Interfaces

→ UI hat hohen Stellenwert bei mobiler Entwicklung!

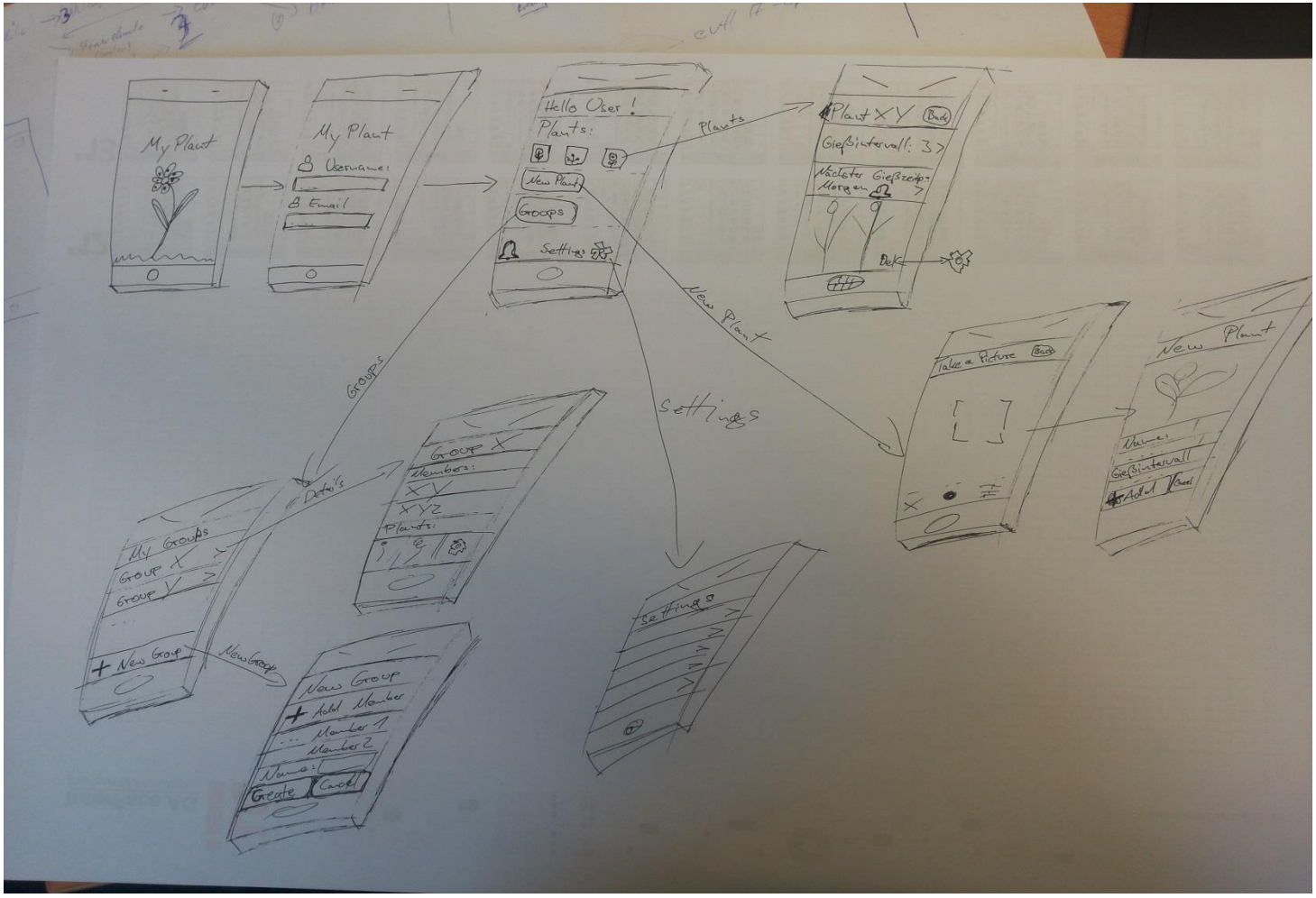
Best Practice um UI zu erstellen

1. Mockup auf Papier
2. Mockup mit Mocksoftware
3. Erstellen einer Mock Anwendung. (Minimale Business Logik, UI möglichst ausimplementieren)
4. Hinzufügen von Business Logik und Backend

1. Mockup auf Papier

- Schnell und interaktiv
- Keine Technik notwendig
- Möglichkeit viel zu diskutieren, umzustellen etc.

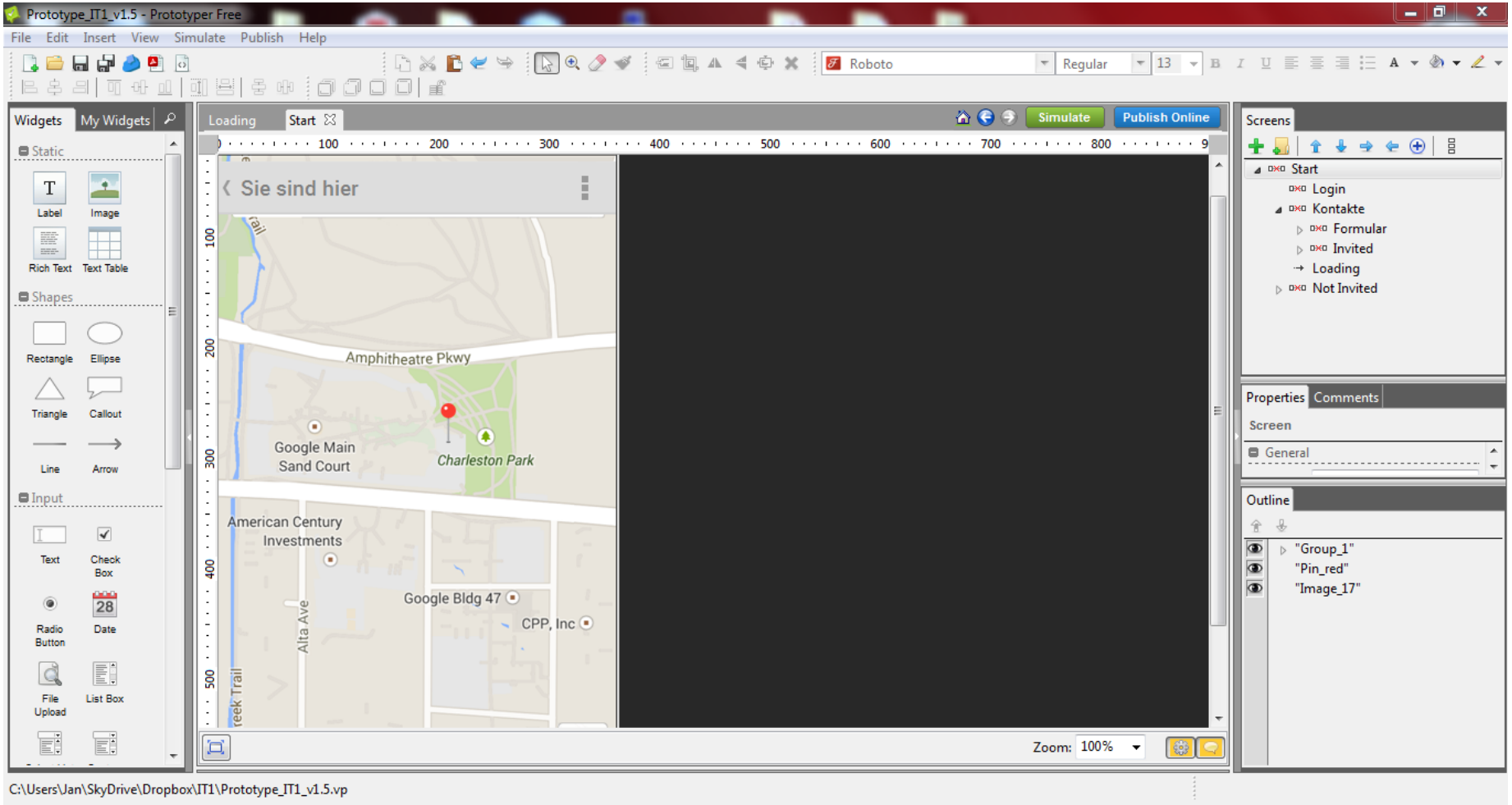
Beispiel:


My Plant
My Plant
Username:
Email
Hello User!
Plants:
New Plant
Groups
Settings
Plants
Plant XY
Gießintervall: 3>
Nächster Gießzeitp.
Morgen
Del
New Plant
Groups
Settings
Take a Picture
New Plant
Username:
Gießintervall
Add
My Groups
GROUP X
GROUP Y
+ New Group
New Group
Group X
Members:
XY
XYZ
Plants:
Details
New Group
+ Add Member
Member 1
Member 2
Name:
Create
Cancel
Settings

2. Mockup mit Mocksoftware

- Verschiedene gute Programme z.B. unter justinmind.com
- Bieten die Möglichkeit schnell ein UI zu erstellen
- Möglichkeit eine fertige App zu simulieren und z.B. Kunden vorzuführen
- Nur vorläufig!
- Gefahr, dass man sich zu sehr auf den Mockup einschießt

Beispiel:

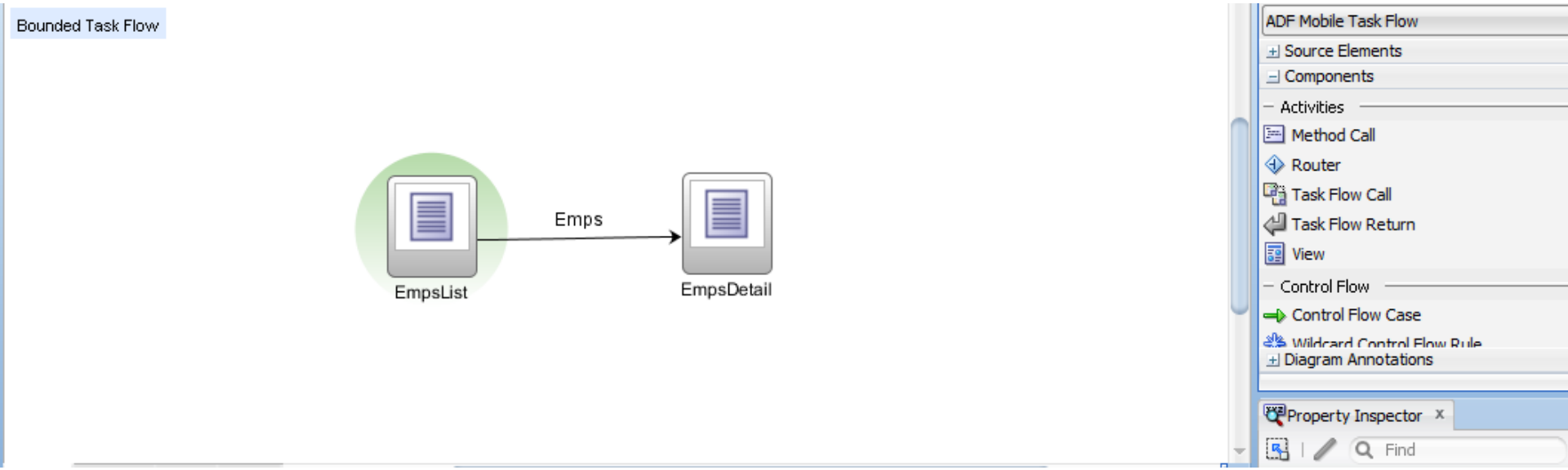

Prototype_IT1_v1.5 - Prototyper Free
File Edit Insert View Simulate Publish Help
Roboto
Regular
13
Widgets
My Widgets
Static
Label
Image
Rich Text
Text Table
Shapes
Rectangle
Ellipse
Triangle
Callout
Line
Arrow
Input
Text
Check Box
Radio Button
Date
File Upload
List Box
Loading
Start
Simulate
Publish Online
Sie sind hier
Amphitheatre Pkwy
Google Main Sand Court
Charleston Park
American Century Investments
Google Bldg 47
CPP, Inc
Alta Ave
Zoom: 100%
Screens
Start
Login
Kontakte
Formular
Invited
Loading
Not Invited
Properties
Comments
Screen
General
Outline
"Group_1"
"Pin_red"
"Image_17"
C:\Users\Jan\SkyDrive\Dropbox\IT1\Prototype_IT1_v1.5.vp

3. Erstellen einer Mock Anwendung

- Das User Interface möglichst ausimplementieren
  - Buttons
  - Textfelder
  - Popups
  - …
  - → Alles was nicht mit Daten oder Logik zu tun hat

Erstellen eines UIs in ADF Mobile

- Hierzu sind „ADF Mobile Task Flows“ gut geeignet
- Pageflow mittels Task Flows schnell erstellt



Erstellen eines UIs in ADF Mobile

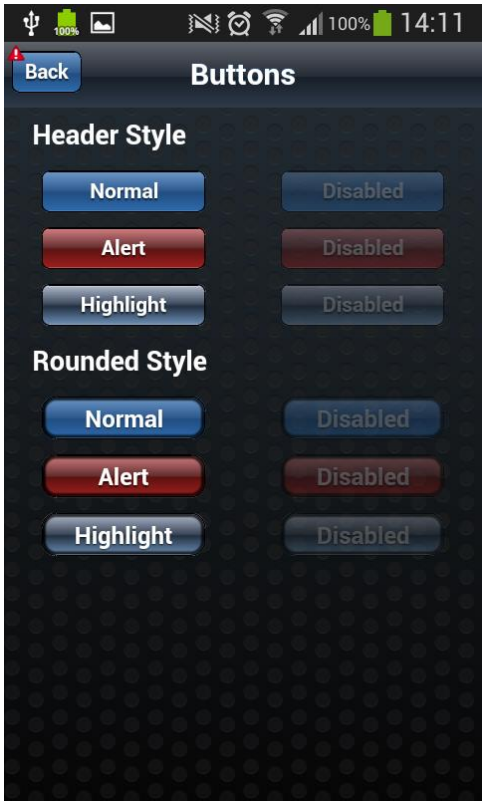
- Komponenten können per Drag & Drop hinzugefügt werden
- aber:

  ADF Mobile ist keine native App! Abänderungen des Mockups eventuell notwendig

→ Problem: Gemockte Java Objekte müssen aus allen Ebenen gelöscht werden.

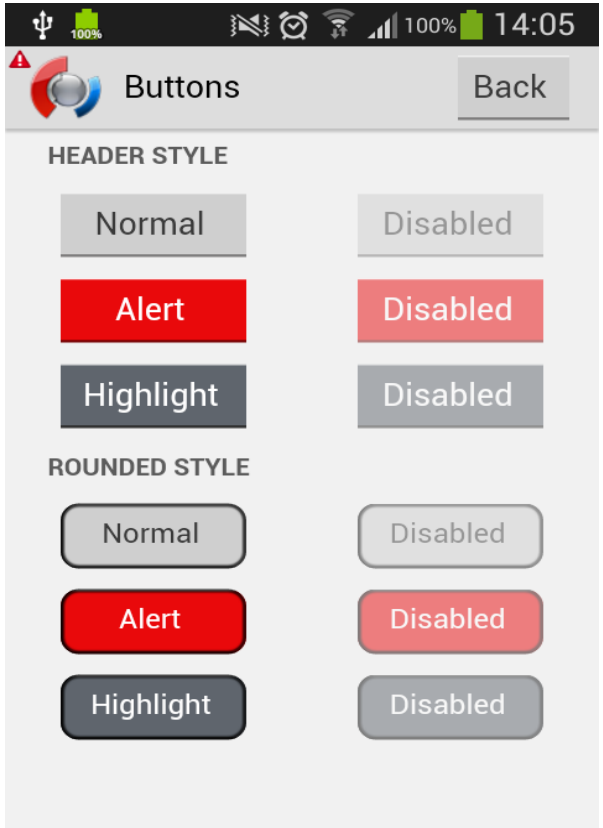
Unterschiede Android/ iOS Design

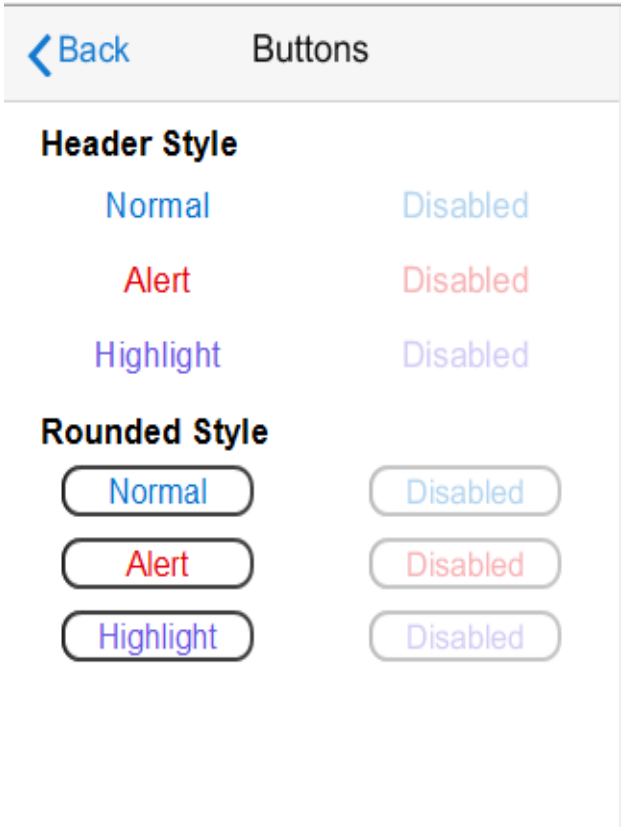
- Früher einheitliche Default Skin für ADF-Mobile Anwendungen (MobileFusionFx)
- An iOS Design orientiert
- Einheitlich sowohl für Android als auch für iOS
- Mit Möglichkeit plattformabhängiges Skinning zu integrieren



Unterschiede Android und iOS

- Ab Update 11.1.2.4.39.64**.62.** neue Skin (MobileAlta)
- An natives Design angepasste Style Familien jeweils für Android und für iOS





Unterschiede Android/ iOS Design

- Umstellen der Skin Family unter:

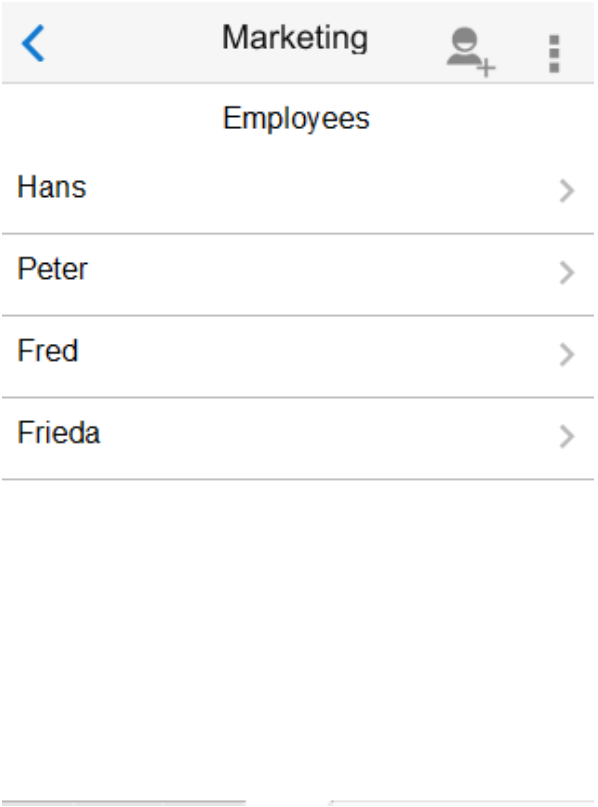
  Application Resources > Descriptors > ADF META-INF > adfmf-config.xml

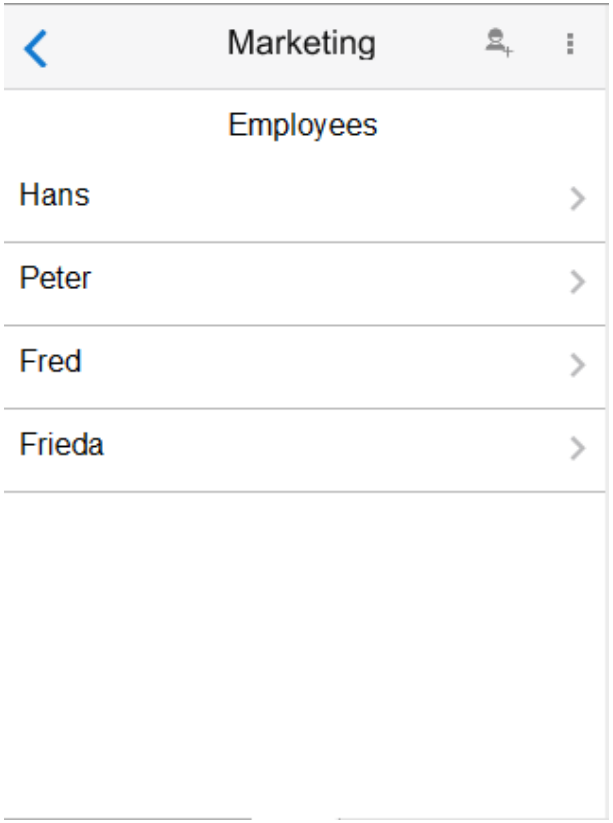
Find

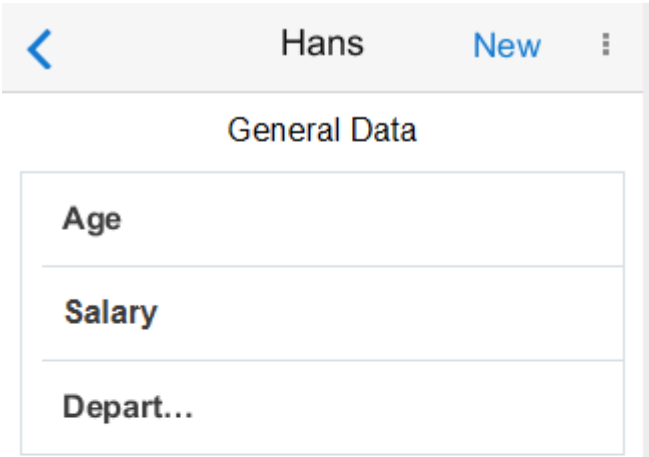
```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <adfmf-config xmlns="http://xmlns.oracle.com/adf/mf/config">
3    <skin-family>mobileAlta</skin-family>
4    <generic-type>
5      <conversion>
6        <validated>false</validated>
7      </conversion>
8    </generic-type>
9  </adfmf-config>
10
```

Unterschiede Android iOS Design

- 2 Unterschiedliche Style Familien führen zu neuen Herausforderungen:
  - Man muss ein UI schaffen, dass auf beiden Plattformen gut aussieht
  - Beispiel: Im Header oft Icons. iOS Buttons haben keinen Rand, Android Buttons schon. Sieht unter Android komisch aus
  - Lösung: Icons mit CommandLinks statt CommandButtons umgeben


Marketing
Employees
Hans
Peter
Fred
Frieda


Marketing
Employees
Hans
Peter
Fred
Frieda


Hans
New
General Data
Age
Salary
Depart...

- Demo

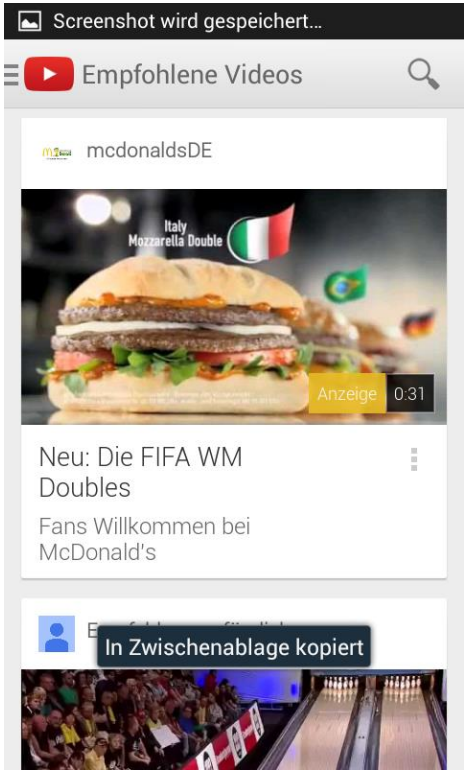
Unterschiede Android und iOS Design

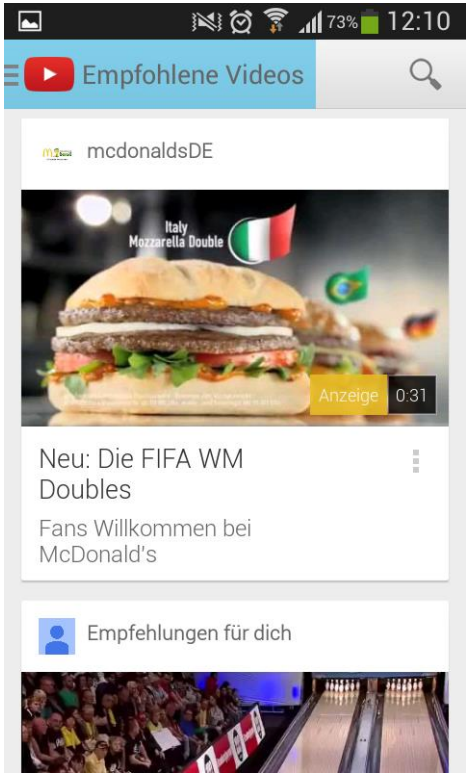
- „Back to Springboard Button“ in Android anders als in iOS
- Persönliche Meinung: Nicht Intuitiv

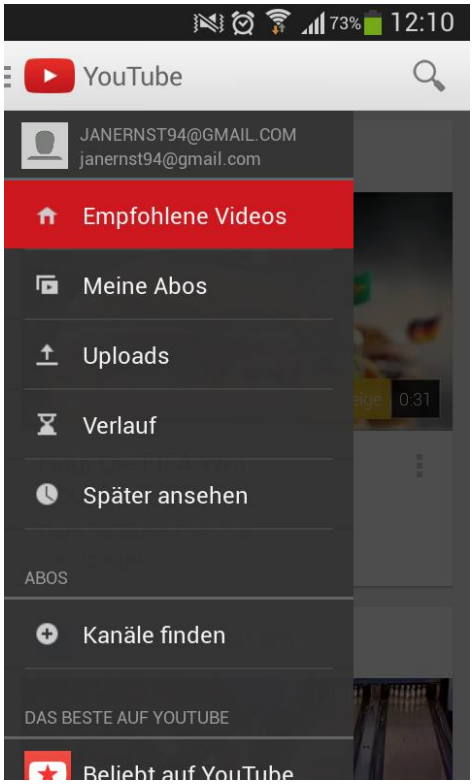
iOS:

Unterschiede Android und iOS Design

- Lösungsansätze:
  - In Android üblichen Springboard Button implementieren







Unterschiede Android und iOS Design

- Lösungsansätze:
  - Navigation Bar nutzen um zum Springboard zurück zu kommen
    - Default-Feature erstellen
    - Amx-Page mit Aufrufen auf die anderen Features erstellen

```
<amx:view xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:amx="http://xmlns.oracle.com/adf/mf/amx"
          xmlns:dvtm="http://xmlns.oracle.com/adf/mf/amx/dvt">
    <amx:panelPage id="pp1">
        <amx:facet name="header">
            <amx:outputText value="MockDemo" id="ot3"/>
        </amx:facet>
   <amx:tableLayout width="100%">
            <amx:iterator var="row" value="#{bindings.features1.collectionModel}">
                <amx:rowLayout>
                    <amx:cellFormat id="cell" halign="center" width="100%">
                        <amx:commandLink id="cl1" shortDesc="FeatureImage Link"
                                         actionListener="#{bindings.gotoFeature.execute}">
                            <amx:image id="i3" source="#{row.image}" shortDesc="Feature Image"
                                       inlineStyle="width:65px;height:65px"/>
                            <amx:setPropertyListener id="spl1" from="#{row.id}" to="#{viewScope.featureId}"/>
                        </amx:commandLink>
                    </amx:cellFormat>
                </amx:rowLayout>
                <amx:rowLayout id="rl3">
                    <amx:cellFormat id="namecell" halign="center" valign="top" height="35px"
                                    shortDesc="FeatureName Cell">
                        <amx:outputText value="#{row.name}" id="ot1"/>
                    </amx:cellFormat>
                </amx:rowLayout>
            </amx:iterator>
        </amx:tableLayout>

    </amx:panelPage>
</amx:view>
```

- Demo

Unterschiede Android iOS

- Native Navigation (Backbutton) in Android nicht implementiert
- Lösbar in Javascript/ Cordova
- Aber: jede Navigation über den Backbutton muss theoretisch einzeln implementiert werden

```
document.addEventListener("deviceReady", onDeviceReady, false);
function onDeviceReady() {
    document.addEventListener("backbutton", backKeyDown, true);
}
function backKeyDown() {
    if ($('#springBoard').length) {
        var cFirm = confirm("Are you sure you want to quit?");
        if (cFirm == true)
            navigator.app.exitApp();
    }
    else if ($('#firstPage').length) {
        adf.mf.api.gotoDefaultFeature();
    }
    else {
        adf.mf.api.amx.doNavigation("__back");
    }
}
```

Business Logik in ADF Mobile

- Wird in Java geschrieben
- Hier kann theoretisch alles gemacht werden was in Java gemacht werden kann

  Aber: ADF Mobile enthält nicht alle Java Packages

  z.B.: awt-Package fehlt, daraus abgeleitete Teile des javax-Packages.

```
1  package mobile;
2
3  import java.awt.image.BufferedImage;
4
5  public class Employee {
6      public Employee() {
7          super();
8      }
   BufferedImage image;
10 }
```

Business Logik in ADF Mobile

- Lässt sich debuggen

- Außerdem: JavaScript, Html5
- Persönliche Meinung: Ausweichtechnologien, falls etwas in JavaScript/ Html5 besser funktioniert als in Java.
- (Will man größtenteils mit JavaScript und Html5 arbeiten evtl. gleich Apache Cordova o.ä. verwenden)

Data Caching in ADF Mobile

- Zur Persistierung auf dem Device wir die Datenbank „SQLite“ benutzt.
- In globalen Kontext können Webservices benutzt werden

- „Out of the Box“ wenig vorhanden gibt aber A-Team Sample Code:
  - Wizard um Daten lokal zu persistieren
  - Wizard um mit Webservices zu kommunizieren
  - Wizard um aus Entitys/POJOs User Interface mit CRUD Funktionalität zu generieren

Schwachpunkte

- Java aktuell sehr limitiert
- Keine Integration von Cordova Plugins
- Limitierungen beim Zugriff auf gerätespezifische Funktionen
- Deployments (Apps) sehr groß (>30MB)
- Data Syncing und Persistence z.Zt. nur über Extension
- UI in Android nicht ausgereift
- JDeveloper (Installation, Bundling, Bugs, …)
  - Eclipse Plugin?

Stärken

- Standardisiertes Programmiermodell
  - Nutzung vorhandener Skills
  - Fast vollständig (keine 3d Party-Libs notwendig)
- Abstraktion von darunterliegenden Technologien(Java Script, HTML5, …)
- Cross-Platform (aktuell Android und iOS)
- Gute Integrationsmöglichkeiten in Unternehmens-IT
  - Html-Seiten, SOAP-Services, REST-Services
  - Unterstützung Security (Mobile Suite)
- Gute Dokumentation!

Fazit

- Gut geeignet um bestehende Unternehmensanwendungen auf mobilen Endgeräten zur Verfügung zu stellen
- Fokus auf Enterprise-IT

- Darüberhinausgehende Anforderungen kaum umsetzbar

InterFace AG
the face of informatics

Vielen Dank für Ihre Aufmerksamkeit!