Entwicklungsprozess einer mobilen App (mobile app engineering)

Zu Beginn des Entwicklungsprozesses einer mobilen App ist es wichtig, zu wissen, was die zu entwickelnde App genau leisten und in welchem Anwendungs- und Systemzusammenhang sie eingesetzt werden soll. Dabei sollte man mit dem Kunden und Auftraggeber sowie späteren Benutzern herausfinden und verstehen, über welche Funktionen und Eigenschaften die App verfügen soll.

Dieser Prozess der Analyse und Dokumentation wird in der Softwaretechnik als Anforderungsanalyse bezeichnet und ist Teil des **Requirements Engineering.**

Dabei soll in enger Kooperation und Interaktion kontinuierlich überprüft werden, ob die entwickelten Ideen und Konzepte geeignet sind, die Bedürfnisse, Wünsche und Vorstellungen des Kunden und der Benutzer umzusetzen.

A)Schritte zur Umsetzung des Projektes

1. **Projektvision und Benutzergruppen definieren**
    a. Projektvision
    enthält die grundsätzliche Idie und Zielsetzung.
    b. Ziel- und Benutzergruppen festlegen
    z.B. um welche Personen handelt es sich und welche Ausbildung haben sie? Welche Gemeinsamkeiten und Eigenschaften haben sie? Oder ist es eine völlig heterogene (uneinheitliche) Gruppe?

2. **Zusammenhang ermitteln**
    a. Anwendungskontext: Erforschung der Nutzer (Anwender) bei ihrer Tätigkeit, die mit dem App zusammenhängen.
    b. Alle Aspekte sollen überlegt werden, die eine Beziehung zur App haben, z.B. Gebäude, Personen und Backend-System. Dadurch soll nicht nur herausgefunden werden, was in das App gehört, sondern auch, was außerhalb liegt und somit nicht in das Programm einfließen wird.

3. **Personas entwickeln**
    Dabei wird ein Profil eines fiktiven Kunden mit persönlichen Daten erfunden, um sich die Anwendung besser vorstellen zu können und greifbarer zu machen.
    Beispiel: Sven Eichorn, 20 Jahre alt, Student, geboren in Wien, inkl. Passfoto usw.
    Man sollte ca. 3 – 5 Personas entwickeln.

4. **Szenarien entwerfen**
    Dabei soll mit dem Kunden ein realitätsnahes Beispiel entwickelt werden. Dabei soll man erkennen, was einem Kunden erwartet, wenn er das App startet, es als Standard durchspielt und dann sollte man beschreiben, wenn Fehler auftreten. Somit kann man z.B. die grafische Oberfläche optimieren.

5. **Storyboard erstellen**

   Damit lassen sich Anwendungsfälle grafisch darstellen und geeignete Prototypen im Rahmen des Designs entwickeln. Es kann sehr häufig händisch erfolgen und zeigt somit eine Visualisierung eines konkreten Szenarios. Vergleichbar zu einem Comic stellen sie leicht verständlich die Schritte im Prozessablauf dar. Es muss aber nicht nur die Designmöglichkeiten der App am Display zeigen, sondern kann darüber hinaus auch etwas von einer möglichen Interaktion des Nutzers mit dem Handy zeigen, wie z.B eine komplexe Umgebung, in der sich der Nutzer befindet (z.B. in einer Bibliothek, am Hochofen, in einem Museum, in der Schule).

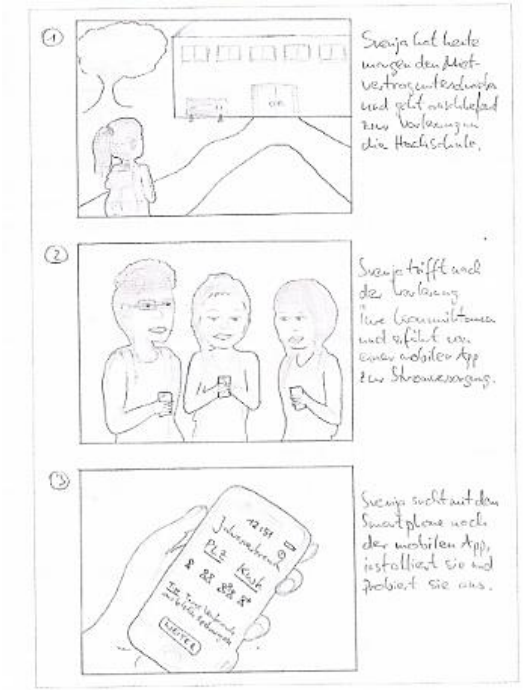


Abb. 4-6 Exemplarisches Storyboard

6. **Anforderungen ermitteln**
   a. Funktionale Anforderungen
      Welche MUSS und SOLL Funktionen gibt es? Z.B. die Anmeldung muss passwortgeschützt sein. In nachfolgenden Details gibt es dann eine Mindestlänge und welche Sonderzeichen zulässig sind.
   b. Beispiele für Kundenprozesse, auf deren Basis eigene Funktionsbereiche entwickelt werden müssen.
      i. Anmeldung (Login, E-Mail, Passwort)
      ii. Vertragsabschluss (Unterseiten z.B. Eingabe Adresse, Zahlungsart)
      iii. Vertragskündigung
      iv. Zählerstand eingeben (Stand erfassen, Verbrauchsgrafik)
      v. Kundenservice (Meinung abgeben, Kontakt, Störung melden)

B)Konzept und Design

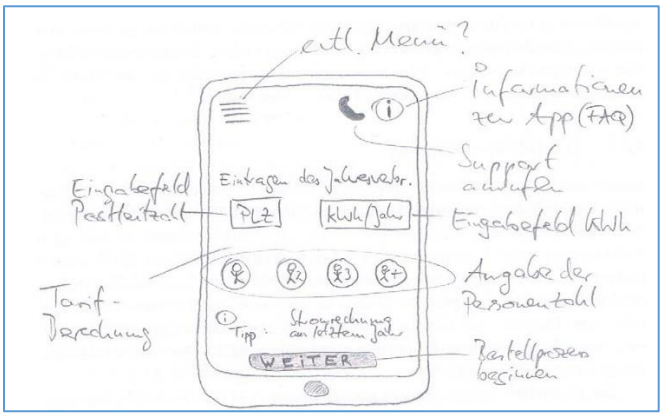
a)Visuelles Konzept entwickeln:

Dieses wird einleitend vom UI-Designer (User Interface) entwickelt, und enthält übergeordnete gestalterische Richtlinien bezüglich Farben, Schriften, Logos und Bildwelt (Styleguide). Dabei legt er ein Raster fest und macht sich Gedanken z.B. den Einsatz von grafischen Elementen wie Icons, Navigationselemente und Buttons.

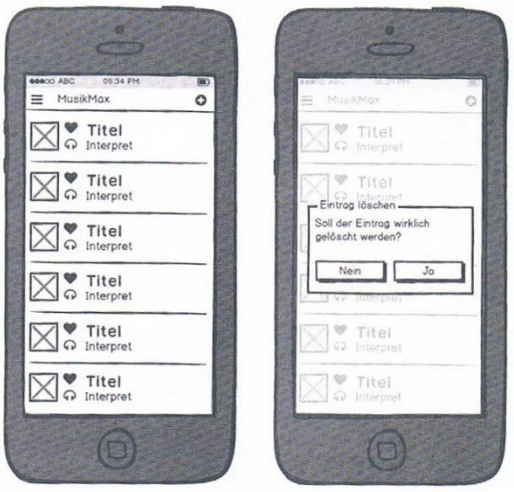
Prototyping

Dabei werden nur wesentliche Systemmerkmale implementiert, um durch Testen des Prototyps die Anforderungen zu konkretisieren. Dadurch kann das Entwicklungsrisiko deutlich reduziert werden und verbessert die Planbarkeit durch die unmittelbare Rückkopplung mit dem Auftraggeber bzw. potentiellen Benutzern.
Meistens werden die einzelnen Bildschirmseiten erst jetzt grob entworfen und oft in Skizzen mit

Bleistift und Papier erstellt. Ziel ist es die Ideen zu visualisieren, ohne auf Vollständigkeit und Proportionen aufzupassen. Die skizzierten Prototypen werden auch als „Wireframes“ oder „Mock-ups“ bezeichnet. Anhand dieser Überlegungen werden die ersten Click-Dummys auf dem mobilen Endgerät erstellt. Diese dienen Präsentations- und Testzwecken.



Beispiel Musikverwaltung:



Eine Konzeption der **Barrierefreiheit** sollte ebenfalls berücksichtigt werden.
Dazu siehe folgende Websites:
http://www.w3.org/WAI/mobile
http://developer.android.com/guide/topics/ui/accessibility/index.html Leitfaden zur Barrierefreiheit unter Android

b)Seitenspezifikation:

Diese stellt die Basis für die schlussendliche Implementierung dar. Darin werden alle Bildschirmseiten mit alle Details wie Texten, Grafiken und Abständen beschrieben, sodass eine programmiertechnische Implementierung erfolgen kann.
Parallel dazu wird die Systemarchitektur entworfen um eine Anbindung an Backendsysteme vornehmen zu können. Dabei ist auch der Entwurf einer projektspezifischen Middleware erforderlich. Diese soll den Nachrichtenaustausch zwischen App und Backend steuern und synchronisieren.

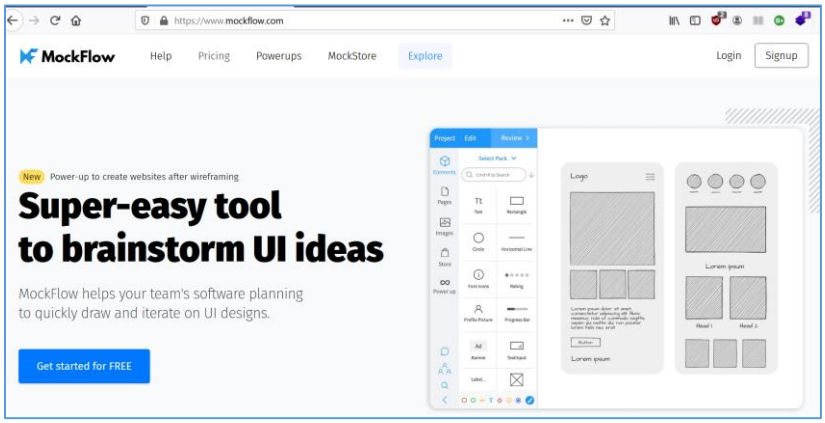
ID	VIEW-0320_contractAddTariffCalculator
Herkunft:	VIEW-0310_contractChoose / FUN-011_actionBar / FUN-015_contractAddAction
Absprungziele:	Tap auf "WEITER"-Btn: VIEW-0221 contractAddRegistration Tap auf Zurück-Btn: VIEW-0310_contractChoose

Beschreibung	Der VIEW-0320_contractAddTariffCalculator stellt in der FUN-011_actionBar neben dem Up-/Back-Btn und dem Titel "Vertrag hinzufügen" die Funktion FUN-012_infoAction und im Action Overflow FUN-013_contactAction dar.
	
	In der ToolBar wird
	• der Titel des Views dargestellt ("Tarif berechnen")
	• ein Zurück-Btn zur Vertragsauswahl: VIEW-0310_contractChoose eingebunden
	• Info-Icon (per Tap öffnet sich ein Overlay mit Textinformationen)
	• ein ActionOverflow mit den Einträgen eingefügt
	  • "Kontakt" (FUN-013_contactAction, per Tap wird ein Anruf zur Hotline gestartet)
	  • "Über ENPURE" (Link zu VIEW-0760_about)
	  • "AGB" (Link zu VIEW-0770_legal)
	  • "Impressum" (Link zu VIEW-0780_imprint)
	  • "Datenschutz" (Link zu VIEW-0771_privacy)
	  • "Stromkennzeichnung" (Externer Web-Link zu http://www.enpure.de/stromkennzeichnung.pdf)
	  • Widerrufsbelehrung (Link zu VIEW-0773_revocation)
	  • Online-Streitbeilegung (Link zu VIEW-0776_onlineDisputeSettlement)
	
	Im Header darunter wird das Standard Vertragsbild, Vertragsicon sowie der Vertragsname (vorgelegte Straße, Hausnummer) (wie auf VIEW-0220_registration) dargestellt.
	
	Das Headerbild wird über den gesamten View als Hintergrundbild verwendet.
	
	Unter dem Header wird der Tarifrechner FUN-040_tariffcalculator dargestellt.
	 
	
	Am Ende des Views ist ein inaktiver "Weiter"-Btn platziert. Dieser ist immer am unteren Displayrand platziert (fixed footer).
	
	Sobald in der FUN-040_tariffcalculator ein Tarif für die (neue) PLZ berechnet wurde, wird der "Weiter"-Btn aktiv.
	
	Per Tap auf den Button wird der Nutzer zum View VIEW-0221 contractAddRegistration weitergeleitet.
	• Bereits bekannte Daten werden vorausgefüllt, können aber vom Nutzer geändert werden (FUN-050_customerInformation und FUN-080_billingInformation).
	• Im Header steht "[STRAßE], [HAUSNUMMER]" – vorbelegt von VIEW-0220_registration. Der Headertitel wird automatisch geändert, sobald der Nutzer eine andere Straße und Hausnummer in den Eingabefeldern angibt.

Abb. 5–19 *Seitenspezifikation des Tarifrechners der mobilen App ENPURE*[29]

Herunterladen der App ENPURE aus dem Play Store.

Website: https://www.mockflow.com/



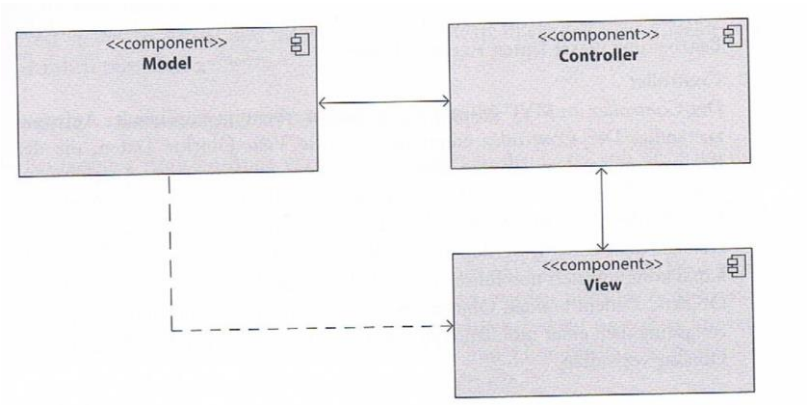
C)Softwarearchitektur

Eine Softwarearchitektur beschreibt allgemein die Strukturen eines Softwaresystems durch Architekturbausteine und ihre Beziehungen und Interaktionen untereinander sowie ihre physikalische Verteilung.

1. Model-View-Controller
2. Model-View-Presenter
3. Model-View-ViewModel

Ad 1) Beim **MVC-Muster** werden folgende Bereiche getrennt und in eigene gleichnamige Komponenten ausgelagert:

- Die Datenhaltung im sogenannten Model
- Die Logik im Controller und
- Die Präsentation in der View.



D)Implementierung

Programmierung nativer mobiler Apps

Dies erfolgt für die beiden mobilen Betriebssysteme Android und iOS mithilfe einer objektorientierten Programmiersprache (Java bzw. Objective-C oder Swift) sowie mithilfe einer mächtigen, komfortablen und kostenlosen integrierten Entwicklungsumgebung (Android Studio bzw. Xcode).
Xcode ist aber nur auf einem Apple-Computer unter dem Betriebssystem macOS lauffähig. Das Android Studio von Google gibt es in beiden Betriebssystemen.

- **Android**
  der Quellcode mobiler Android-Apps wird mit der objektorientierten Programmiersprache Java erstellt. Dazu wird die kostenlose integrierte Entwicklungsumgebung (IDE) Android Studio eingesetzt.
- **iOS**
  Die Basis des mobilen Betriebssystems iOS ist das Framework Cocoa Touch. Es ist dabei die Schnittstelle zwischen dem Betriebssystem iOS du den mobilen Apps, die unter iOS ausgeführt werden.
  Das Software Development Kit von iOS enthält alle Werkzeuge, die man zur Entwicklung

benötigt. Dabei arbeitet man mit der integrierten Entwicklungsumgebung Xcode von Apple. Xcode ist kostenlos.

- **Cross-Plattform-Entwicklung**
  Dabei wird mit EINER Quellcodebasis Software für unterschiedliche Betriebssysteme entwickelt. Das jeweilige Betriebssystem muss nur eine Java-Laufzeitumgebung (Java Runtime Environment JRE) mit der Java Virtual Machine (JVM) bereitstellen, damit Java-Programme ausgeführt werden können.
  Die Herausforderung ist, die nativen APIs ansprechen zu können, damit das Look and Feeling einer nativen App entsteht.
  - **Frameworks und Technologien:**
    - **Xamarin**
      ist eine Softwareanwendung, die in C# programmiert ist. Mithilfe der DIE vom Xamarin Studio, die auch in Microsoft Visual Studio integriert ist, können native mobile Apps entwickelt werden.
    - **React Native**
      wurde 2015 von Facebook veröffentlicht und nutzt das JavaScript-Framework React zur Entwicklung. React konzentriert sich dabei auf die GUI und den Datenfluss von sogenannten Singel-Page-Applications.
    - **Apache Cordova mit Ionic**
      Apache Cordova: (wird ab 2020 eher nicht mehr so oft verwendet)
      ist ein Open-Source-Framework, mit dem sich mobile Apps auf Basis von HTML5, CSS und JavaScript erstellen lassen. Mit PhoneGap gibt es auch eine kommerzielle Distribution dazu.
      Eine mobile Cordova-App wird dabei in einer bildschirmfüllenden WebView ausgeführt. Das HTML-Grundgerüst mit CSS und JavaScript wird dabei nicht umgewandelt. Der Zugriff auf bestimmte Hardwarekomponenten wie z.B. die Kamera erfolgt mithilfe von Plug-Ins.
      Aufgrund der Kombination unterschiedlicher Technologien werden Cordova-Apps auch als hybride Apps bezeichnet. **Sie sind keine nativen mobilen Apps**.
      Ionic:
      ist ein Open-Source-Framework, mit dessen Hilfe grafische Benutzeroberflächen unter dem Einsatz von HTML5, CSS3 und JavaScript entwickelt werden. Es basiert auf AngularJS von Google, einem JavaScript-Framework. Mobile Ionic-Apps sind auf mobilen Endgeräten und mobilen Betriebssystemen mit Webbrowser bzw. WebView lauffähig.

E)Veröffentlichung

Es gibt die Möglichkeit, ein öffentliches Rollout über die Standard-App-Stores der Betriebssystemhersteller wie **Google Play** bzw. den **App Store** von Apple vorzunehmen.
Dabei sind die spezifischen Qualitätsanforderungen zu erfüllen, um den Veröffentlichungsprozess erfolgreich durchlaufen zu können.
Neben diesen Stores gibt es noch weitere Apps Stores.

- Stores von Amazon
- Yandex (http://store.yandex.com)

- Aptoide (http://www.aptoide.com)
- HockeyApp
- RestFlight

Quelle:
Vollmer, Guy in: Mobile App Engineering, dpunkt-Verlag, 2017, Heidelberg