# Detailplanung Iteration 2

Zeitraum: 23. April bis 3. Juni

### Festgelegte Termine:

* 4. Mai: Abgabe Reflexionsberichte (siehe T4 im SWProjektkalender) / Endgültiges Ergebnis der Analyse steht fest / Abgabe der Systemanalyse-Ergebnisse
* 13. Mai: Konzept für Systemarchitektur ist konsolidiert / Review mit Betreuer (siehe T5 im SWProjektkalender)
* 3. Juni: Ende von Iteration 2, Dokumentation der SW-Architektur abzugeben

### Team-Meetings:

* *Freitag, 29. April 11:00*
* *Mittwoch, 4. Mai, 10:00 Uhr*
* *Freitag, 13.Mai, 9:00 Uhr (in 210)*
* **Freitag, 20.Mai, 9:00 Uhr (in 210)**

### Sonstiges

* **ReSharper installieren:** <https://www.jetbrains.com/shop/eform/students/> **=> 20. Mai**
  + Bei Fragen an die Teamleiter (Tamara, Theresa) oder Stefan wenden!
* **Klassen und Methoden ansehen, Funktionsweise überlegen! => 20. Mai**

### Einteilung Arbeitspakete:

* Überarbeitung / Fertigstellung der Analyse Ergebnisse
  + Allgemein alle bzw. dafür eingeteilte Gruppe, je nachdem was genau anfällt
* Schnittstellenübersicht
  + Im Teammeeting machbar / automatisch generierbar => Stefan
* Systemkomponenten (Klassendiagramm pro Paket für wichtigste Klassen -> auf Schnittstellen achten!)
  + Größere Gruppe, untereinander aufzuteilen, evtl. auch ein Teil im Teammeeting;   
    Tamara, Jonas, Stefan, Jeremy
  + Bis 4. Mai
  + View (eingeteilt in Forms, Controls, Component Representation)
  + Controller
  + Logger
  + Model (eingeteilt in Network Components und Business Logic)
* Auswahl Architekturrelevanter Use Cases, Systemoperationen => Sequenzdiagramme / Kommunikationsdiagramme dazu
  + Auswahl der Use Cases und Finden der Systemoperationen im Teammeeting: 22.04.16
  + Sequenz- / **Kommunikationsdigramme in kleinerer Teilgruppe: Mirjam, Theresa**
* Oberflächen-Prototyp erstellen (ohne größere Funktionalität)
  + Zum Teil schon vorhanden (Skizzen); Dimitrij, Max, Stefan
  + Skizzen zu Oberflächen bis 29. April
  + ~~Oberfläche erstellen~~ siehe Grundgerüst der Klassen erstellen
* Grundgerüst der Klassen und Interfaces erstellen => 4. Mai, was bis dahin möglich ist
  + Vermutlich kleinere Gruppe: Dimitrij, Jeremy, Tamara, Max, Alex, Jonas
  + Im Klassendiagramm gefundene Methoden leer anlegen
    - *Controller: Jonas*
    - *Model – Network Components: Jeremy & Tamara*
    - *Model – Business Logic: Dimitrij*
  + Einfache Methoden können bereits programmiert werden
  + Oberfläche: Max, Stefan, Alex
* Use-Case-basierte Testfälle finden / erstellen
  + Kleinere Gruppe; Mirjam, Denis, Oleg, Alex bis ca. 13. Mai
* Erste Funktionalität ausprogrammieren
  + **View: Stefan, Max**
* Erste Tests durchführen
  + Sofern Funktionalität vorhanden, die testbar ist
* Regelmäßiger Build (mind. 1x (?) die Woche) sobald nennenswerte Code-Menge vorhanden
  + Build-Admin! (Stefan Pölloth)
* **Syntax überlegen für Skriptsprache für Testszenario (Weiterarbeiten)**
  + **Denis, Oleg => Bitte bisherige Ergebnisse auf Github stellen 20. Mai**
* Genauere Überlegungen zum Simulationsablauf
  + Dimitrij, Jeremy, Denis, Alex, Mirjam => 13. Mai
* Speicherformat Projekt (Struktur) Was und Wie?
  + => später, wer noch wenig zu tun hatte
* **Layer für Layerstack (welche Layer werden gebraucht, was machen sie?)**
  + **Denis, Oleg => 20. Mai**

### Arbeitspaket:

* Überarbeitung / Fertigstellung der Analyse Ergebnisse
* ~~(Konzept für) Systemarchitektur erstellen~~
  + ~~Paketdiagramm~~  => Wurde bereits erledigt
* Schnittstellenübersicht
* Systemkomponenten (Klassendiagramm pro Paket für wichtigste Klassen -> auf Schnittstellen achten!)
  + View (eingeteilt in Forms, Controls, Component Representation)
  + Controller
  + Model (eingeteilt in Network Components und Business Logic)
  + Logger
* Auswahl Architekturrelevanter Use Cases, Systemoperationen => Sequenzdiagramme / Kommunikationsdiagramme dazu
* Oberflächen-Prototyp erstellen (ohne größere Funktionalität)
  + Skizzen zu Oberflächen
  + Oberfläche erstellen
* Grundgerüst der Klassen und Interfaces erstellen
  + Im Klassendiagramm gefundene Methoden leer anlegen
  + Einfache Methoden können bereits programmiert werden
* Use-Case-basierte Testfälle finden / erstellen
* Erste Funktionalität ausprogrammieren
* Erste Tests durchführen
* Regelmäßiger Build (mind. 1x (?) die Woche) sobald nennenswerte Code-Menge vorhanden