

# Smash Adressverwaltung

Function Point Schätzmethode

Höhere Fachschule für Technik Mittelland



# SMASH.

Grenchen, 14.03.2023

Gruppe Valdrin, Elia, Jan, Marc, Andreas, Yannic

# Inhaltsverzeichnis

<b>Aufgabenstellung</b> .....	<b>3</b>
Ausgangslage .....	3
Aufgabe.....	3
Beurteilungskriterien .....	4
<b>Grobkonzept</b> .....	<b>5</b>
Wireframe (Handskizze) .....	5
User Stories.....	6
Klassendiagramm.....	6
<b>Schätzung</b> .....	<b>7</b>
Anzahl Transaktionen .....	7
Schwierigkeitsgrad .....	7
Scrum Poker / Function Points .....	8
Templates zur Schätzung .....	9
<b>Realisierung</b> .....	<b>10</b>
Java Prototyp (FX) .....	10
Interaktiver Prototyp (Figma) .....	11
Funktionalität (JavaScript) .....	12
Frontend (Bootstrap + JS) .....	13
<b>Auswertung</b> .....	<b>14</b>
Funktionalität.....	14
Design .....	14
Zusammenspiel .....	14
<b>Code</b> .....	<b>15</b>

# Aufgabenstellung

## Ausgangslage

...ist ein kleiner Prototyp, den ihr mit egal welcher Technologie erstellen sollt:

- Eine Datenbanktabelle **Person** mit den Spalten Name, Vorname, Geburtsdatum und Heimatort (Interne Daten, mittel)
- Eine Datenbanktabelle **Adresse** mit den Spalten Strasse, Nummer, gültig ab, gültig bis (interne Daten, mittel)
- Eine Datenbanktabelle **Ortschaft** mit den Spalten PLZ und Ort (interne Daten, einfach)
- Die Beziehungen sind wie folgt definiert: Eine Person kann mehrere Adressen haben, nur eine ist jeweils gültig, eine Adresse hat genau eine Ortschaft
- Es muss in einem grafischen Benutzerinterface (GUI) möglich sein,
  - neue Personen zu erfassen
  - bestehende Personen zu löschen
  - bestehende Personen zu mutieren
  - einer Person eine neue Adresse hinzuzufügen
  - Ortschaften in einem Ortschaftsverzeichnis zu pflegen (neu, bearbeiten, löschen)
  - Eine Liste aller Personen mit allen Adressen am Bildschirm auszugeben

## Aufgabe

1. Erstellt kurz ein paar **Handskizzen**, wie diese Applikation aussehen soll
2. Ermittelt aus den Skizzen die **Anzahl Transaktionen** (Input, Output, Abfragen)
3. Versucht den **Schwierigkeitsgrad** abzuschätzen
4. Berechnet die **Function Points** in einem Excel Sheet
5. Programmiert die Applikation mit einer **Technologie nach Wahl** (die Cracks sollen ran, die anderen unterstützen). Der Aufwand dafür sollte 5 Stunden nicht überschreiten. Fragt, wenn ihr Fragen habt. Schön ist nicht wichtig. Nur die Funktionalität muss stimmen. Messt den Aufwand so genau wie möglich für die einzelnen Function Points.
6. Erstellt eine **Auswertung, welche FPs** wie lange in der Umsetzung brauchten
7. Gebt die **Auswertung** und die **Applikation** inkl. Source Code ab

## Beurteilungskriterien

- Vollständigkeit der Umsetzung
- Vollständigkeit und Nachvollziehbarkeit der Function Point Aufstellung
- Nachvollziehbarkeit der Aufwandsdokumentation
- Wiederverwendbarkeit der Kalkulation

Abgabetermin: 14.03.2023, 18.45h

# Grobkonzept

## Wireframe (Handskizze)

### Views zur Verwaltung

---

The left wireframe shows a list view with two main columns: 'Personen' and 'Anschriften'. Under 'Personen', there are buttons for 'Neu', 'Löschen', and 'Übersicht'. The list contains entries: 'Zalika', 'Grenden', 'Diberist', and 'Herzogbuckst'. Under 'Anschriften', there are input fields for 'ort', 'Zalika', 'PLZ', and '3052'.

The right wireframe shows a detailed form view. It has a 'Personen' section with buttons for 'Neu', 'Löschen', and 'Übersicht'. The form contains fields for 'Vorname' (Max), 'Nachname' (Muster), 'Geburts-' (26.01.96), 'Heimatort' (Grenden), 'Strasse' (Lupenweg), 'PLZ' (3000), 'Nr.' (5), 'Ort' (Bern), and 'Adresse' (Bern). There are also buttons for 'Bearbeiten' and 'Speichern'.

### Excel Export zur Gesamt-Übersicht

---

name	vorname	Geburtsdatum	Heimatort	Adresse1	Adresse2
~	~	~	~	~	~

Abbildung 1 Excel Export

## User Stories

Als Smash Adressverwaltungs Nutzer möchte ich mit einem einfachen Interface meine gesmashten Personen verwalten und alles dazugehörigen Smash-Orte zu den Personen führen.

Als Smash Adressverwaltungs Nutzer möchte ich Ortschaften nur einmalig erfassen, damit ich sehe an wie vielen Ortschaften ich schon gesmasht habe.

## Klassendiagramm

Objekte:

- Person
- Adresse
- Ortschaft

EA ist tot

# Schätzung

## Anzahl Transaktionen

Anhand unseres Wireframe und Prototyp haben wir folgende Transaktionen festgestellt:

### Allgemein

- Wechsel zwischen Views (Menu)

### Ortschaften View

- Ortschaft verwalten
  - Hinzufügen
  - Löschen
  - Speichern

### Smashes View

- Person verwalten
  - Neu
  - Löschen
  - Speichern
- Adresse einer Person verwalten
  - Neu
  - Löschen
  - Speichern
  - Als Hauptadresse definieren

## Schwierigkeitsgrad

Die gesamte Applikation ist generell für einen erfahrenen DEV egal in welcher Programmiersprache sehr einfach realisierbar.

- Funktionen sind Standardverhalten, welches in jeder App anzutreffen ist.
- UI basiert auf Standardkomponenten, welche fast jede Library direkt bereitstellt.
- Die persistente Speicherung machen wir lokal auf dem Gerät, wodurch kein Hosting, Rechte-Management etc. benötigt wird.

## Scrum Poker / Function Points

Daraus ergab sich folgende Vor-Kalkulation:

Aufwandschätzung / Scrum-Poker			
Elementarprozess	Bemerkung	Typ	Scrum-Poker
<b>Datenbanken</b>			
- Design erstellen		Input	2
<b>Design GUI (SceneBuilder)</b>			
- Personen hinzufügen		Input	3
- Personen löschen		Output, Abfragen	1
- Personen Bearbeiten / Speichern		Input, Output, Abfragen	2
- Ortschafts Liste		Output, Abfragen	1
- Personen Liste		Output, Abfragen	1
- Personen und Ortschaften TAB		Output, Abfragen	1
- Personen und Ortschaften TAB		Output, Abfragen	1
<b>Übersicht Bewohner</b>			
- Gesamt Übersicht		Output, Abfragen	2
		Scrum Points	14

Function Points			
Elementarprozess	Minimaler Punktwert	Mittlerer Punktwert	Maximaler Punktwert
<b>Datenbanken</b>			
- Design erstellen (ILF)	3	4	5
<b>Design GUI (SceneBuilder)</b>			
- Personen hinzufügen (EI)	5	7	9
- Personen löschen (EO, EQ)	2	2.5	3
- Personen Bearbeiten / Speichern (EI,EO,EQ)	4	6	8
- Ortschafts Liste (EO,EQ)	2	2.5	3
- Personen Liste (EO,EQ)	2	2.5	3
- Personen und Ortschaften TAB (EO,EQ)	1	1	1
- Personen und Ortschaften TAB (EO,EQ)	1	1	1
<b>Übersicht Bewohner</b>			
- Gesamt Übersicht (EO,EQ)	3	4	5
	Gesamt min. Punktwert	Gesamt m. Punktwert	Gesamt max. Punktwert
Function Points	23	30.5	38



## Templates zur Schätzung

Wir haben Excel-Tabellen gebaut, welche die Kalkulation abnehmen und wo man nur noch die Werte eintragen kann.

→ Wiederverwendbarkeit

Aufwandschätzung / Scrum-Poker			
Elementarprozess	Bemerkung	Typ	Scrum-Poker
Datenbanken			
- Design erstellen		Input	2
Design GUI (SceneBuilder)			
- Personen hinzufügen		Input	3
- Personen löschen		Output, Abfragen	1
- Personen Bearbeiten / Speichern		Input, Output, Abfragen	2
- Ortschafts Liste		Output, Abfragen	1
- Personen Liste		Output, Abfragen	1
- Personen und Ortschaften TAB		Output, Abfragen	1
- Personen und Ortschaften TAB		Output, Abfragen	1
Übersicht Bewohner			
- Gesamt Übersicht		Output, Abfragen	2
		Scrum Points	14

Function Points			
Elementarprozess	Minimaler Punktwert	Mittlerer Punktwert	Maximaler Punktwert
Datenbanken			
- Design erstellen (ILF)	3	4	5
Design GUI (SceneBuilder)			
- Personen hinzufügen (EI)	5	4	9
- Personen löschen (EO, EQ)	2	1	3
- Personen Bearbeiten / Speichern (ELCO, EQ)	4	4	8
- Ortschafts Liste (EO, EQ)	2	1	3
- Personen Liste (EO, EQ)	2	1	3
- Personen und Ortschaften TAB (EO, EQ)	1	0	1
- Personen und Ortschaften TAB (EO, EQ)	1	0	1
Übersicht Bewohner			
- Gesamt Übersicht (EO, EQ)	3	2	5
	Gesamt min. Punktwert	Gesamt m. Punktwert	Gesamt max. Punktwert
Function Points	23	17	38

[https://hftm-my.sharepoint.com/:x:/g/personal/andreas\\_moser\\_hftm\\_ch/EWl6ax\\_eQIjIpgrzAzE9FMsBDbjLXPBN2jccVR020zcxXA?e=nc2Zra](https://hftm-my.sharepoint.com/:x:/g/personal/andreas_moser_hftm_ch/EWl6ax_eQIjIpgrzAzE9FMsBDbjLXPBN2jccVR020zcxXA?e=nc2Zra)

# Realisierung

## Java Prototyp (FX)

Wir haben gestartet mit einer Java FX View. Jedoch machen wir es lieber mit JavaScript, ist einfacher und schneller realisiert mit unseren Skills im Team 😊

Yannic: 1h für FXML-View:

File Help

Personen Ortschaften

Vorname Max

Nachname Mustermann

Geburtsdag 07.03.2023

Heimatort MenuButton

+ Löschen Speichern

Adresse 1

Strasse Zürichstrasse

Nr 2a

PLZ 3000

Ort Bern

Standard Adresse

File Help

Personen Ortschaften

Ortschaft Grenchen

PLZ 2540

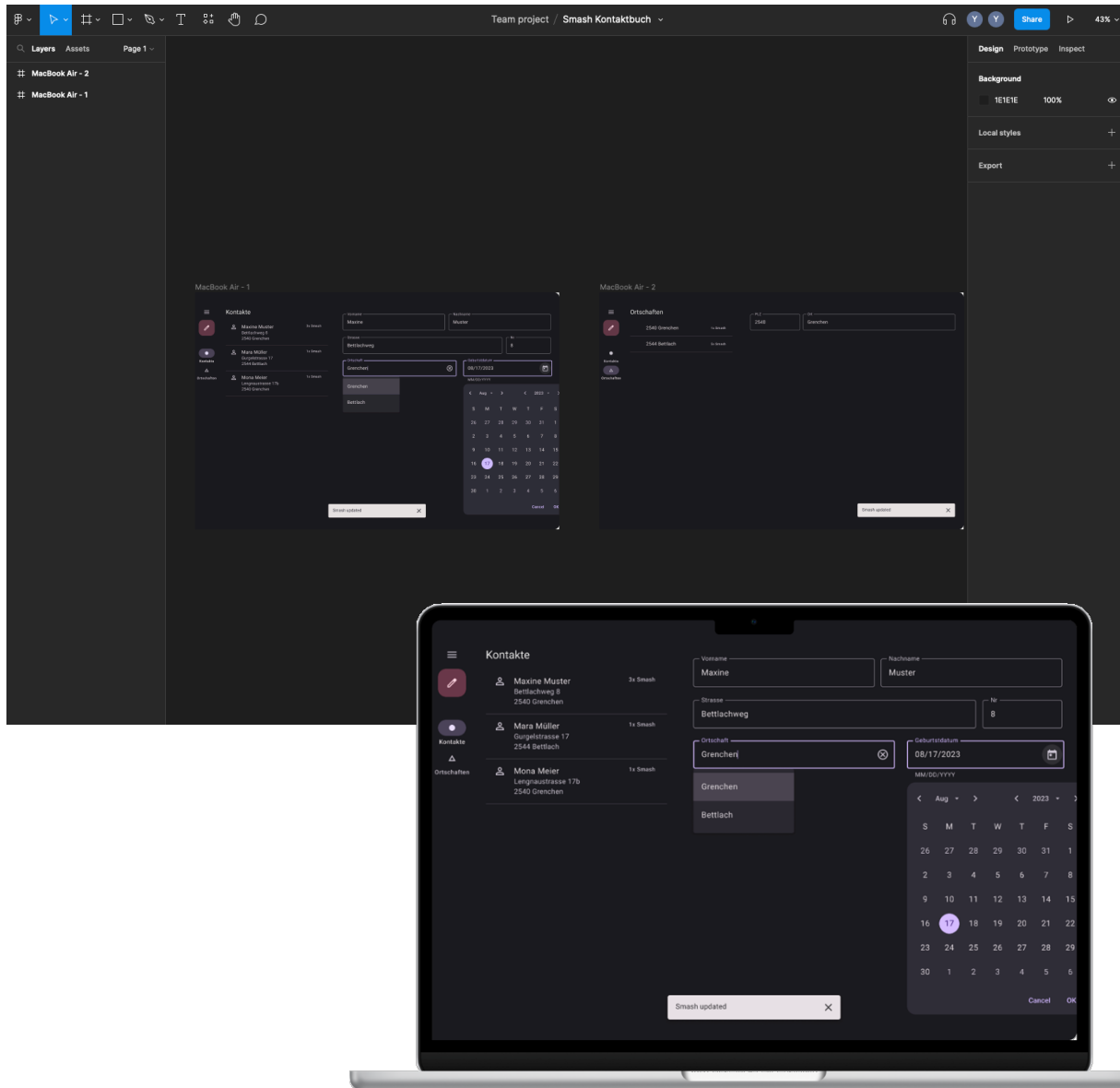
Optimierungen der Views aufgrund des oben dargestellten konkreten Prototyps:

- In der Adresse wird die Ortschaft auch als Dropdown integriert, um noch modularer und vernetzter zu sein.
- Der Boolean «Standard Adresse» wird im Personen Objekt abgebildet, bleibt aber aus usability Gründen in der View am dargestellten Platz.

# Interaktiver Prototyp (Figma)

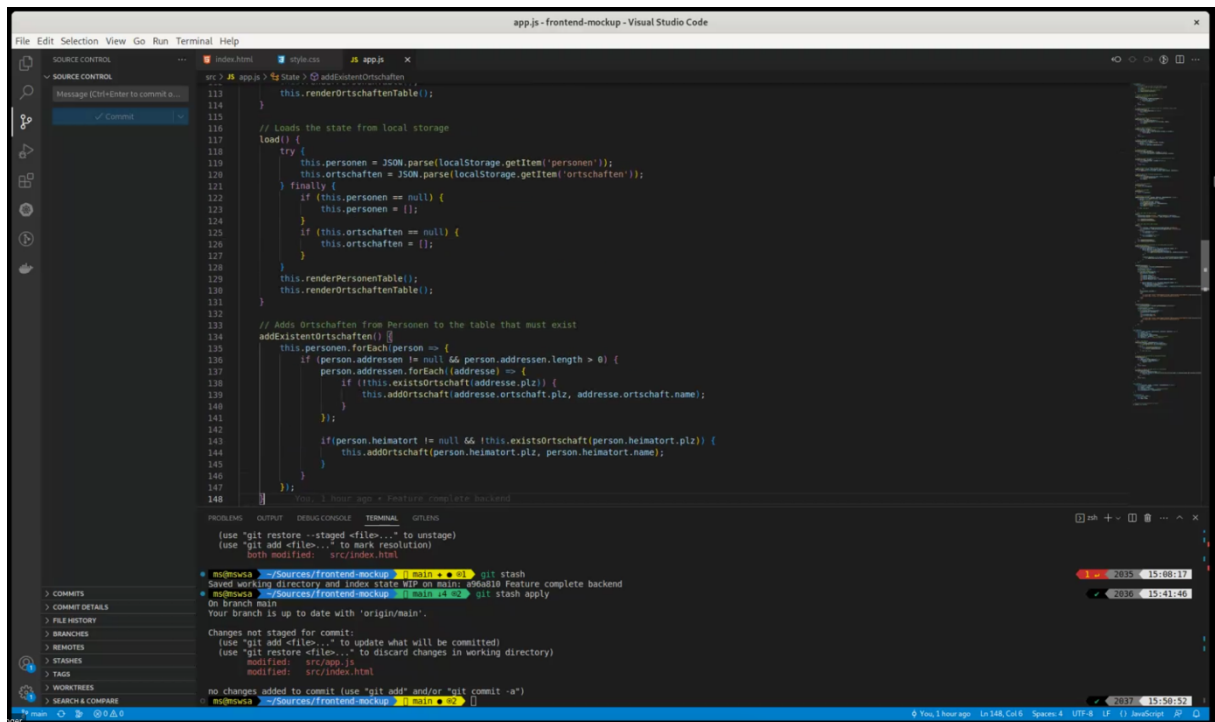
Yannic: 2h Erstellung Prototyp

<https://www.figma.com/file/WUW5kQB85tTIUfAAxu0WDW/Smash-Kontaktbuch?node-id=0%3A1&t=0urCgsepQJqgePv3-1>



# Funktionalität (JavaScript)

Marc: 2h Realisierung Backend Logik



```
app.js - frontend-mockup - Visual Studio Code
File Edit Selection View Go Run Terminal Help

SOURCE CONTROL
Message [Ctrl+Enter] to commit...
✓ Commit

src > app.js > State > addExistentOrtschaften
113     this.renderOrtschaftenTable();
114   }
115
116   // Loads the state from local storage
117   load() {
118     try {
119       this.personen = JSON.parse(localStorage.getItem('personen'));
120       this.ortschaften = JSON.parse(localStorage.getItem('ortschaften'));
121     } finally {
122       if (this.personen == null) {
123         this.personen = [];
124       }
125       if (this.ortschaften == null) {
126         this.ortschaften = [];
127       }
128     }
129     this.renderPersonenTable();
130     this.renderOrtschaftenTable();
131   }
132
133   // Adds Ortschaften from Personen to the table that must exist
134   addExistentOrtschaften() {
135     this.personen.forEach(person => {
136       if (person.addressen != null && person.addressen.length > 0) {
137         person.addressen.forEach(adresse => {
138           if (!this.existsOrtschaft(adresse.plz)) {
139             this.addOrtschaft(adresse.ortschaft.plz, adresse.ortschaft.name);
140           }
141         });
142       }
143       if (person.heimatort != null && !this.existsOrtschaft(person.heimatort.plz)) {
144         this.addOrtschaft(person.heimatort.plz, person.heimatort.name);
145       }
146     });
147   }
148 }
// End of User app - Feature complete backend

PROBLEMS OUTPUT DEBUGCONSOLE TERMINAL GIT LENS
(use "git restore --staged <file>..." to unstage)
(use "git add <file>..." to mark resolution)
both modified: src/index.html

ms@mswaa ~/Sources/frontend-mockup [main • • 0] git stash
Saved working directory and index state WIP on main: unable to Feature complete backend
ms@mswaa ~/Sources/frontend-mockup [main:4 • 2] git stash apply
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   src/app.js
    modified:   src/index.html

no changes added to commit (use "git add" and/or "git commit -a")
ms@mswaa ~/Sources/frontend-mockup [main • 42]
git
```

→ Code siehe Repo

## Frontend (Bootstrap + JS)

So sieht das Resultat aus.

Marc: 2h

### Personen Details

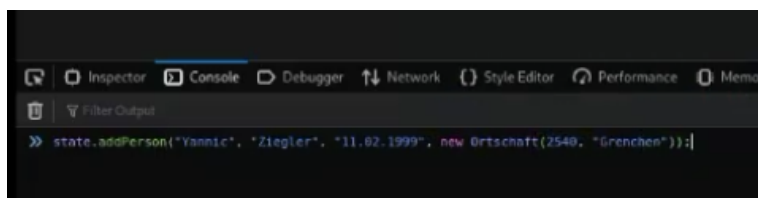
Neue Person hinzufügen

Name	Vorname	Geburtsdatum	Heimatort	Adressen	Anpassen
Marc	Singer	12.04.1994	3000 - Bern		

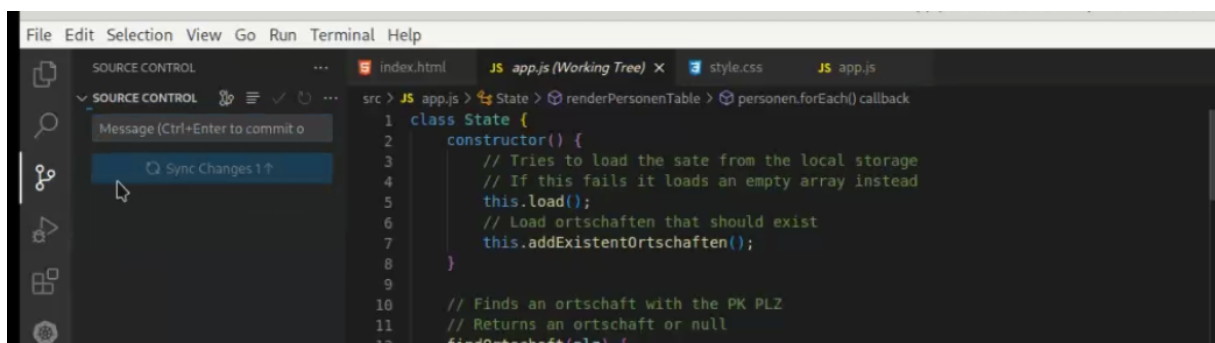
### Ortschaften Details

Neue Ortschaft hinzufügen

PLZ	Ort
-----	-----



```
>> state.addPerson("Yannic", "Ziegler", "11.02.1999", new Ortschaft(2540, "Grenchen"));
```



```
1 class State {
2   constructor() {
3     // Tries to load the state from the local storage
4     // If this fails it loads an empty array instead
5     this.load();
6     // Load ortschaften that should exist
7     this.addExistentOrtschaften();
8   }
9
10  // Finds an ortschaft with the PK PLZ
11  // Returns an ortschaft or null
12  findOrtschaft(plz) {
```

→ Code siehe Repo

# Auswertung

Wir haben den Aufwand vor der Realisierung in unterschiedliche Bereiche geteilt. Das war nicht so genial leider ... :/

Function Points			
Elementarprozess	Minimaler Punktwert	Mittlerer Punktwert	Maximaler Punktwert
<b>Datenbanken</b>			
- Design erstellen (ILF)	3	4	5
<b>Design GUI (SceneBuilder)</b>			
- Personen hinzufügen (EI)	5	7	9
- Personen löschen (EO, EQ)	2	2.5	3
- Personen Bearbeiten / Speichern (EI,EO,EQ)	4	6	8
- Ortschafts Liste (EO,EQ)	2	2.5	3
- Personen Liste (EO,EQ)	2	2.5	3
- Personen und Ortschaften TAB (EO,EQ)	1	1	1
- Personen und Ortschaften TAB (EO,EQ)	1	1	1
<b>Übersicht Bewohner</b>			
- Gesamt Übersicht (EO,EQ)	3	4	5
	Gesamt min. Punktwert	Gesamt m. Punktwert	Gesamt max. Punktwert
Function Points	23	30.5	38

## Funktionalität

Viele von uns haben nur mit Java FX programmiert bisher und war es schwer ein gemeinsames Verständnis zum Aufwand zu erhalten.

- Vieles was in java FX mit Scene Builder etc. sehr aufwendig ist, ist in der Realität total simpel, wenn man ein intelligentes Tool nutzt 😊
- Die Prototypen waren leider sehr FX lastig und könnten mit Kenntnis der schlussendlich genutzten Technologie einfacher und simpler gestaltet werden. Mehr auf Standards setzen.
- Die Arbeitsteilung durch grosses Unwissen im Team war auch nicht optimal, da so die Tasks nicht gut aufgeteilt werden konnten.

## Design

In unserer Liste fehlte zudem der Prozess zur Erstellung des UX / UI mit Wireframes und Prototypen.

- Nächstes Mal würden wir diese als User Story auflisten und sagen, dass man als User gerne das UI komplett sehen möchte.

## Zusammenspiel

Effektiv erstellten wir in der Realisierung zuerst das Backend mit der Funktionalität der gesamten Applikation und danach machten wir uns an das GUI.

- Eine Arbeitsteilung ist nach Skills so üblich in der Realität. Das Slicing würden wir anders machen nächstes Mal.
- Das Team sollte besser zusammengesetzt sein und mehr Skills abdecken, welche wirklich DEV Erfahrung mitbringen ^^

## Code



Der Code dieses Projektes ist als öffentliches Repository auf GitHub verfügbar:

<https://github.com/SirZesso/smash-verwaltung>