Alarm Clock: Arbeitsprozess

Jonathan Hyams
Pascal Schmalz

16. Juni 2017

Inhaltsverzeichnis

1 Zweck des Dokument **3**
1.1 Planung . . . 3
1.2 Deliverables . . . 3
1.3 Aufwandschätzung und Kontrolle . . . 3
1.4 Überarbeitung des Planes . . . 3
1.5 Guntt Chart . . . 3

2 Meilensteine **3**
2.1 GUI . . . 3
2.2 Serialisieren . . . 3
2.3 Poller . . . 3
2.4 Popups . . . 3

3 Tests **3**

4 Stolpersteine **3**
4.1 Gitignore . . . 3
4.2 JavaFX serialisieren . . . 4
4.3 PLatform.runlater . . . 4
4.4 PopUps . . . 5
4.5 Maven . . . 5

5 Versionskontrolle **5**

1 Zweck des Dokument

1.1 Planung

1.2 Deliverables

1.3 Aufwandschätzung und Kontrolle

1.4 Überarbeitung des Planes

1.5 Guntt Chart

2 Meilensteine

2.1 GUI

Zuerst wollten wir ein brauchbares GUI als Interface haben, da es oftmals einfacher ist Ideen zu diskutieren, wenn man eine Bildilche Diskusionsgrundlage h at. Unser Ziel war es das GUI als optischen Prototypen zu benutzen zu können.

2.2 Serialisieren

Der nächste Meilenstein war die Serialisierbarkeit. Wir wollten, dass wir die Reminders speichern könen. Damit kann man sie später wieder von der Festplatte laden.

2.3 Poller

Mit dem Poller wollten wir sicherstellen, dass die Notifications auch abgesendet werden, wenn das Programm “geschlossen” wurde.

2.4 Popups

Pascal

3 Tests

4 Stolpersteine

4.1 Gitignore

LaTeXgeneriert beim compilieren recht viele Dateien neben dem gewollten PDF. Diese Dateien wollten wir nicht über git synchronisieren. Es wäre sonnst möglich,dass die Dateien nicht zusammenpassen und beim erneuten compilieren zu einem Fehler führen. Wir wollten die Dateein also ins gitignore file reinnehmen, in welchem man definieren kann, welche Dateien nicht mit git verwlatet werden. Der gitignore syntax ist aber etwas speziell man kann nicht einfach ein Verzeichnis ignorieren und dann mittels einer whitelist bestimmte Dateien wieder in die Versionierung mit einbeziehen. Will man ein

solches Vehalten erreichen, darf man lediglich den Inhalt der Verzeichnsse ausklammern, und dann mittels Whitelisting wieder in die Versionskontrolle mit einbeziehen. Diesen Sachverhalt genau zu eruieren hat viel Zeit Verschlungen. Als er einmal erkannt wurde, mussten wir trotzdem noch genau darauf achten, dass wir nur genau die Dateien synchronisieren, wleche wir auch synchronisieren wollten.

4.2 JavaFX serialisieren

JavaFX Komponenten lassen sich nicht Serialisieren. Bis wir das herausgefunden haben, hat auch einige Stunden gedauert. Um dieses Problem zu umgehen haben wir herausgefunden, dass wenn man JavaFX Komponenten nicht direkt in der Klasse abspeichert, sonder sie nur zurückgibt, dann funktioniert es. Ein kurzes Beispiel: Die Table im GUI verlangt nicht Strings, sondern SimpleStringProperties. Das Problem ist aber, dass diese SimpleStringProperties nicht serialisierbar sind.

Falsch:

```
private SimpleStringProperty subject = new SimpleStringProperty
    ();

public SimpleStringProperty getSubjectProperty() {
  return subject;
}
```

Richtig:

```
private String subject = "";

public SimpleStringProperty getSubjectProperty() {
  return new SimpleStringProperty(subject);
}
```

4.3 PLatform.runlater

Ein weiteres Problem das wir hatten war das verspätete Erscheinen von Popups. Wenn man ein Popup z.B. eine Minute nach Programmstart aufruft, hat man effektiv mehrere Dutzend Exceptions bekommen. Der Grund war, dass Threads mit JavaFX sehr mühsam sind. Das Erste Problem das wir überhaupt hatten war, die richtige Exception zu finden. Die IllegalStateException hat uns dann auf den richtigen Pfad gebracht. Die Exception lautete: ÏllegalStateException: Not an FX application thread; currentThread = Thread-4 [...]". Auf StackoverFlow haben wir dann gesehen, dass man um das neu Erstellte GUI ein

```
Platform.runlater(() -> {...});
```

wirft. Das neue GUI erstellt man dann in den geschweiften Klammern. Dieses Problem zu beheben hat ca eine Woche gedauert, da wir praktisch jede Exception genau analysiert haben und selten wirklich etwas brauchbares darauslesen konnten.

4.4 PopUps

Ein sehr merkwürdiges Problem war, dass ab und zu einige Popups einfach nicht erschienen sind. Zum debuggen haben wir ConsoleNotifications gebraucht, die gleichzeitig gelaunched wurden wie die Popups. Das Komische war, dass die ConsoleNotifications immer erschienen sind, jedoch die JavaFX Popups manchmal nicht. Nach langem Suchen hat sich herausgestellt, dass das erstellte GUI, welches im Platform.runlater(() -¿ ...); aufgerufen wird, nie ausgeführt wurden. Das heisst, jedes Stück Code darin wurde ignoriert, egal ob ein normales Print Statement oder das Erstellen eines GUI. Der Grund dazu war, dass sobald alle Stages, also alle JavaFX Fenster, geschlossen sind, wird vom Compiler automatisch Platform.exit() aufgerufen. Dies führt dazu, dass alles was im Platform.runlater(() -¿ ...) ist, einfach ignoriert wird. Um dies zu umgehen, braucht man nur eine Zeile Code:

```
1   Platform.setImplicitExit(false);
```

Auch das herauszufinden hat sehr lange gebraucht, da und nie in den Sinn gekommen ist, dass so etwas überhaupt notwendig wäre. Wir haben mehr an unserem Code gezweifelt als an eine Implizit aufgeworfene Methode von JavaFX.

4.5 Maven

Ein kleineres Problem, aber dennoch ein Problem für und war Maven. Unser Dozent wollte eine Möglichkeit die das Launchen des Programms sehr einfach macht. Da wir bereits im Software Engineering and Design mit Maven gearbeitet haben, haben wir uns dafür entschieden. Es aufzusetzen gab bei uns aber immer wieder Probleme. Wir hatten zuerst einfach nur den Java Code im unserem Programm Folder, ohne Maven. Um das Maven hinzuzufügen, mussten wir das Ganze auseinander nehmen, Maven aufsetzen, und den Code wieder an den richtigen Ort tun. Jedoch hat dann die ganze Zeit der Code, der vorher noch erfolgreich kompiliert hat, nicht mehr funktioniert. Wir haben es dann noch zwei mal neu versucht aufzusetzen, danach hat es Funktioniert. Leider wissen wir bis jetzt nicht, was wir anders gemacht haben.

5 Versionskontrolle

Manuelle Version: 1.0.1

Automatische Versionierung: Last compiled: Fri 16 Jun 11:46:58 CEST 2017
Git HEAD Version: 201