ENTWICKLUNG EINES DATEITRANSFER-KLIENTEN MITTELS METHODEN DER MODELLGETRIEBENEN SOFTWARE-ENTWICKLUNG

Mario Kaulmann[1], Naoufel Frioui[1] and Carole Noutchegueme[1]

[1] *Fachbereich Informatik und Medien, Technische Hochschule Brandenburg, Magdeburger Straße 50, Brandenburg an der Havel, Deutschland*

Keywords: MDSD, modellgetriebene Software-Entwicklung, md2, Dateitransfer, REST, Android

Abstract: Das wird später geschrieben

1 EINLEITUNG

1.1 Motivation

Modellgetriebene Software-Entwicklung (MDSD) soll dazu dienen das Entwickeln von Anwendungen so zu vereinfachen, dass Menschen ohne Programmierkenntnisse, Anwendungen erstellen können, die für einen speziellen Anwendungsbereich eingesetzt werden können.
Das Entwickeln einer Smartphone-App ist eine komplexe Aufgabe. Wenn diese App mit einem Server kommunizieren soll, dann wird die Komplexität dieser Aufgabe noch erhöht.
Um trotzdem Ergebnisse erzielen zu können gibt es bereits Ansätze, mit deren Hilfe man durch Beschreibungen des Problems Code erzeugen kann, durch den eine lauffähige Anwendung entsteht.

1.2 Ziel

Das Ziel besteht darin eine Android-App mit Hilfe von Methoden der MDSD zu erstellen. Die App soll über eine REST-Schnittstelle Dateien auf einen Server laden können und Dateien von diesem Server runterladen können. Damit verschiedene Nutzer den Dateitransfer-Dienst nutzen können, soll auch eine Funktion bereitgestellt werden, die es ermöglicht, dass Nutzer sich registrieren, anmelden und abmelden können. Diese Funktionen werden von dem Server bereitgestellt und sind auch über eine REST-Schnittstelle nutzbar.
Bei der Umsetzung dieses Vorhabens wird herausgestellt, was mit den zum Zeitpunkt der Arbeit verfügbaren Mitteln möglich ist und welche Grenzen es noch gibt.

1.3 Aufgabenstellung

Zum Vergleichen des Ergebnisses, das mit Methoden der MDSD erstellt wird, wird vorher ein Prototyp erstellt, der den vollen Funktionsumfang bietet und auf klassische Weise programmiert wird.
Im ersten Schritt der Entwicklung werden Mittel zur modellgetriebenen Entwicklung eines Android-Klienten ausgesucht. Im zweiten Schritt wird versucht den Prototyp des Klienten zu entwickeln. Dabei wird der erzeugte Code anschließend in Android Studio geöffnet und kompiliert. Anschließend wird die App getestet und mit der klassisch programmierten App verglichen, um die aktuellen Grenzen aufzuzeigen und herauszufinden, ob der Prototyp die Anforderungen erfüllt.
Um den Funktionsumfang erfüllen zu können werden einige Teile bei der MDSD-App mit selbstgeschriebenen Code ergänzt.

1.4 Abgrenzung

In dieser Arbeit soll nur ein Android-Klient erstellt werden, der mit einem Server über eine REST-Schnittstelle kommuniziert. Es ist nicht Bestandteil dieser Arbeit eine Server-Anwendung zu erstellen, die die REST-Schnittstelle zur Verfügung stellt.
Das Produkt dieser Arbeit ist ein Prototyp, der durch verschiedene MDSD-Methoden erstellt wird. Der Anteil des selber geschriebenen Codes wird dabei versucht möglichst klein zu halten.

1.5 Ergebnis

2 VORGEHEN

2.1 App-Entwicklung

Parallel zu diesem Projekt befindet sich auch der Server, der verwendet werden soll in der Entwicklung. Zum Testen des Servers wurde eine Weboberfläche bereitgestellt.
Um die Umsetzung eines Zugriffs auf einen REST-Service in Android zu testen, wurde ein Prototyp auf klassische Weise programmiert. Das bedeutet, dass der Code in Android Studio ausgehend von einem leeren Projekt entwickelt wird. Dabei werden einerseits Erkenntnisse generiert, wie die Verbindung mit dem REST-Server funktioniert und allgemein, wie die App in Android umgesetzt wird, sodass man sieht, was in welche Dateien geschrieben werden muss und wie diese strukturiert sind.
Außerdem wurde die App auch mit Hilfe von MD^2 umgesetzt. Dabei sieht man die Grenzen die bei MD^2 noch bestehen. Den Code der von MD^2 generiert wurde kann man dann analysieren und mit dem auf klassische Weise erzeugten Code vergleichen. Da bei der Erzeugung des MD^2-Codes ebenfalls Muster entstehen, die im Rahmen der in Android gültigen Strukturen bestehen, kann man diese nutzen, um ein weiteres Programm zu erzeugen, dass die fehlenden Elemente automatisch erzeugt.
Die Abbildung 1 zeigt das Vorgehen bei der Entwicklung der MD^2-App.
Der Code, der von MD^2 erzeugt wird, wird in einen Ordner mit dem Namen ”src-gen” gespeichert. Auf diesem Code arbeitet der Feature Provider weiter. Das Ergebnis des Feature Providers wird in dem gleichen Ordner gespeichert. Nachdem der Code erweitert wurde kann er mit Android Studio geöffnet werden. Bei Bedarf können manuelle Veränderungen an dem Code vorgenommen werden. In Android Studio kann die App dann kompiliert und getestet werden.

2.2 Feature Provider-Entwicklung

Bei der Entwicklung des Feature Providers werden die Erkenntnisse der klassischen Programmierung ausgenutzt. Dabei werden allgemein gültige Konzepte benutzt, sowie das Wechseln einer Activity mit Hilfe eines Intents. Für die Umsetzung der Kommunikation mit einem REST-Service gibt es verschiedene Bibliotheken. Der Feature Provider benutzt ausschließlich die Bibliotheken, die bei der klassisch programmierten App benutzt wurden.
Zuerst wurde die Benutzeroberfläche entwicklet, die es ermöglicht die entsprechenden Einstellungen auszuwählen. Besonders wichtig ist, die Auswahl des

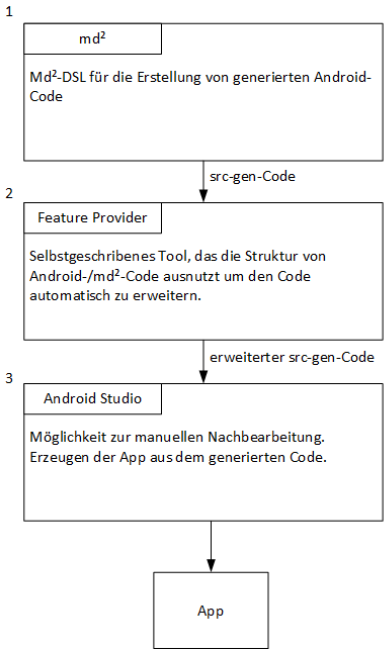

Figure 1: Es werden die einzelnen Arbeitsschritte dargestellt. Die Zahlen geben die Reihenfolge der Schritte an. An den Pfeilen steht die Ausgabe des Vorgängerschritts, die als Eingabe des nächsten Schritts dient.

”src-gen”-Ordners, da der Absolute Pfad zu diesem Ordner auf jedem Gerät anders ist.
Danach wurde die Funktionalität des Erzeugens der Features implementiert.
Zur Entwicklung der Funktionalität des Feature Providers wurde in den von MD^2 erzeugten Code geschaut an welcher Stelle zusätzlicher Code eingefügt werden muss, oder wo der Code durch anderen ersetzt werden muss. In der klassischen App wurde der Code geprüft, wie die Funktionalität umgesetzt wurde. Bei der Erstellung des Feature Providers wurden dann einige Klassen direkt übernommen mit einigen Veränderungen. Nach der Erzeugung einer neuen Funktion des Feature Providers wurden die in Abbildung 1 dargestellten Arbeitsschritte durchgeführt um zu testen, ob die App dann noch funktioniert und um zu sehen, dass das gewollte Feature hinzugefügt wurde.

3 WERKZEUGE ZUR ENTWICKLUNG

3.1 MD^2

MD^2-DSL ist eine domänenspezifische Sprache (DSL), die entwickelt wurde um datengetriebene Business-Apps in textueller Form beschreiben zu können (Heitkötter et al., 2013).
MD^2 ist ein akademischer Prototyp, der auch Anwendung in praktischen Projekten findet und dabei auch weiterentwickelt wird (Majchrzak et al., 2015).

3.2 Androidstudio

Wir benutzen diese freie Integrierte Entwiclungsumgebung für Android. Damit testen wir unsere app auf Android gerte. Hier gibt es auch die möglichkeit, virtuelle geräte zu erstellen (Eingabe wie Betriebsysteme und die Größe der Geräte einzugeben). Der MD2 erstellte Code wird dann mit Androidstudio getestet. Im Verzeichnis src-gen wird die App automatisch generiert und sie kann mit Androidstudio ausgeführt werden.

3.3 FeatureProvider

4 ENTWICKLUNG

4.1 Schnittstellen

Hier werden die Schnittstellen beschrieben, die mithilfe vom Restservice vom Server zur Verfügung stehen. Sie basieren auf Standard-HTTP-Operationen.

1. Nutzeroperationen
   - Erstellung von einem Nutzer(Post)
     Hier ist die Erstellung von einem Nutzer möglich. Wenn ein Nutzer unsere App. benutzen will, muss er erstmal sich registrieren lassen. Dafür braucht man:
     – Ein Username
     – Ein Passwort

     Antwort 200: Die Operation war erfolgreich
     Antwort 400: Schlechte Anfrage, falls der Nutzer ungültige oder schon vorhandene Parameter einträgt.
   - Nutzer im System einloggen (Post)
     Hier ist das Einloggen von einem Nutzer im System möglich. Diese Funktionnalität ist nur möglich, wenn der Nutzer schon registriert ist. Dafür braucht man:
     – Ein Username
     – Ein Passwort

     Antwort 200: Die Operation war erfolgreich
     Antwort 400: Schlechte Anfrage, falls der Nutzer ungültige oder schon vorhandene Parameter einträgt.
   - Delete (Meldet die aktuelle angemeldete Benutzersitzung ab) Hier ist die Abmeldung von der Sitzung möglich. Dafür braucht man den Token vom Nutzer
     Antwort 200: Die Operation war erfolgreich
     Antwort 400: Schlechte Anfrage, falls der Nutzer ungültige oder schon vorhandene Parameter einträgt.
2. Dateioperationen
   - Hochladen von Dateien(Post) Hier ist das Hochladen von einer Datei möglich. Dafür braucht man:
     – Der Token
     – Der FolderId
     – Die Datei

     Antwort 200: Die Operation war erfolgreich
     Antwort 400: Schlechte Anfrage, Im Fall die Größe der Datei größer als 30Mb ist.
     Antwort 403: Verboten
     Antwort 404: Nicht gefunden
   - Herunterladen von Dateien (Get) Hier ist das herunterladen von einer Datei möglich. Dafür braucht man:
     – Der Token
     – Der FolderId (Id vom übergeordneten Ordner)
     – Datei (Id von der Datei)

     Antwort 200: Die Operation war erfolgreich
     Antwort 403: Verboten
     Antwort 404: Nicht gefunden
   - Bearbeitung vom Dateiname (Put) Hier ist die Bearbeitung von einer Datei möglich. Dafür braucht man:
     – Der Token
     – Der FolderId (Id vom übergeordneten Ordner)
     – Die Datei (Id von der Datei)
     – Der Name von der Datei

     Antwort 200: Die Operation war erfolgreich
     Antwort 400: Schlechte Anfrage
     Antwort 403: Verboten
     Antwort 404: Nicht gefunden
   - Lschen von einer Datei (Delete) Hier ist das Löschen von einer Datei möglich. Dafür braucht man:
     – Der Token
     – Der FolderId (Id vom übergeordneten Ordner)
     – Die DateiID (Id von der Datei)

     Antwort 200: Die Operation war erfolgreich
     Antwort 403: Verboten
     Antwort 404: Nicht gefunden
3. Verzeichnisoperationen
   - Herunterladen von einem Ordner (Get) Hier ist das herunterladen von einem Ordner möglich. Dafür braucht man:
     – Der Token
     – Der FolderId (Id vom Ordner)

     Antwort 200: Die Operation war erfolgreich
     Antwort 403: Verboten
     Antwort 404: Nicht gefunden
   - Hochladen von einem Ordner (Post)
     Hier ist das Hochladen von einem Ordner möglich. Dafür braucht man:
     – Der Token
     – Der FolderId
     – Der Ordner

     Antwort 200: Die Operation war erfolgreich
     Antwort 400: Schlechte Anfrage, Im Fall die Größe der Datei größer als 30Mb ist.
     Antwort 403: Verboten
     Antwort 404: Nicht gefunden
   - Bearbeitung von einem Ordner (Put)
     Hier ist die Bearbeitung von einem Ordner möglich. Dafür braucht man:

- Der Token
- Der FolderId (Id vom übergeordneten Ordner)
- Den Folder (z.B ein neur Name)

  Antwort 200: Die Operation war erfolgreich
  Antwort 400: Schlechte Anfrage
  Antwort 403: Verboten
  Antwort 404: Nicht gefunden

- Delete
  Hier ist das Löschen von einem Ordner möglich. Dafür braucht man:
- Der Token
- Der FolderId (Id vom übergeordneten Ordner)

  Antwort 200: Die Operation war erfolgreich
  Antwort 403: Verboten
  Antwort 404: Nicht gefunden

5 ZUSAMMENFASSUNG

QUELLEN

Heitkötter, H., Majchrzak, T. A., and Kuchen, H. (2013). Md2-dsl eine domnenspezifische sprache zur beschreibung und generierung mobiler anwendungen. *Proceedings der 6. Arbeitstagung Programmiersprachen (ATPS), Software Engineering*.

Majchrzak, T. A., Ernsting, J., and Kuchen, H. (2015). Model-driven cross-platform apps: Towards business practicability. *Proceedings of the CAiSE 2015 Forum at the 27th International Conference on Advanced Information Systems Engineering (CAiSE 2015)*.