E-book-based interactive learning system

Adegeye Florence, 0825147

SS 2014

188.935 From Design to Software 2
E188 Institut für Softwaretechnik und Interaktive Systeme

Univ.Prof. Dr. Horst Eidenberger

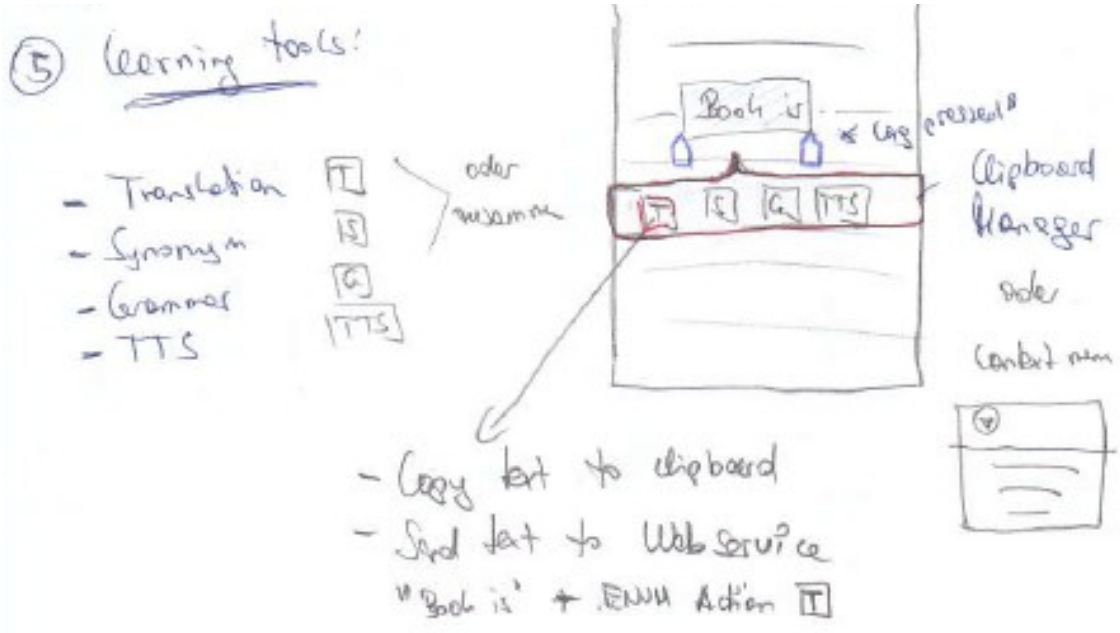
1 Kurzbeschreibung:

Das *„E-book-based interactive learning system“* ist eine Android Tablet App, mit der man nicht nur e-Books anzeigen, sondern auch mit Hilfe von integrierten Werkzeugen schnell und einfach Vokabeln lernen kann.
Mit Hilfe von Webservices (Google translate u. Glosbe.com) können folgende Informationen während dem Lesen abgerufen werden: automatische Übersetzung (Deutsch ← → Englisch), Grammatik, Synonyme, Erklärungen und Satzbeispiele.
Weiters wird zur Übung der richtigen Aussprache eine Text-To-Speech Funktionen unterstützt.

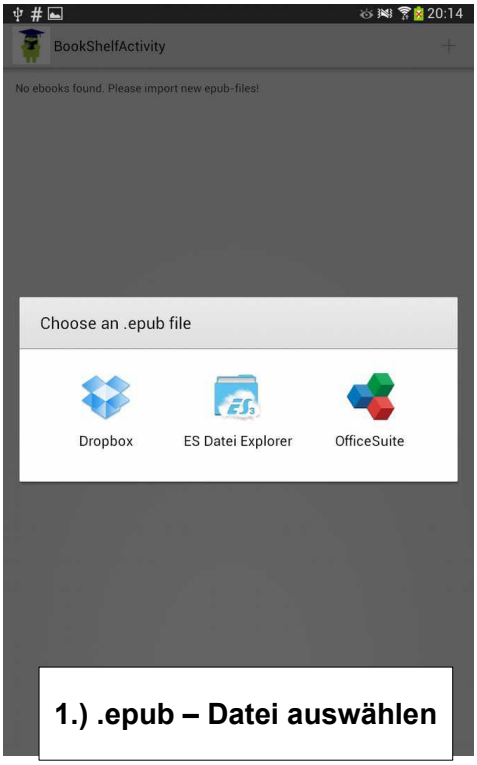
2 Ideen & Konzepte:

Alle Skizzen und Screenshots zum Projekt befinden sich im Anhang unter `\docs\...`

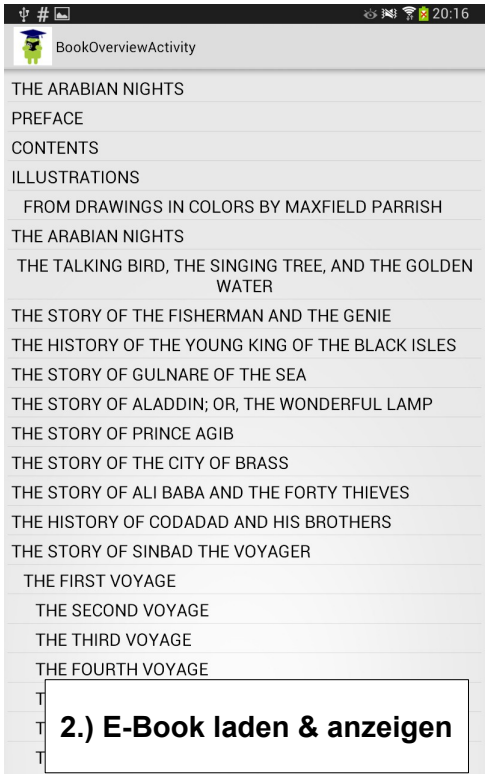
Konzept zu den Trainingstools:



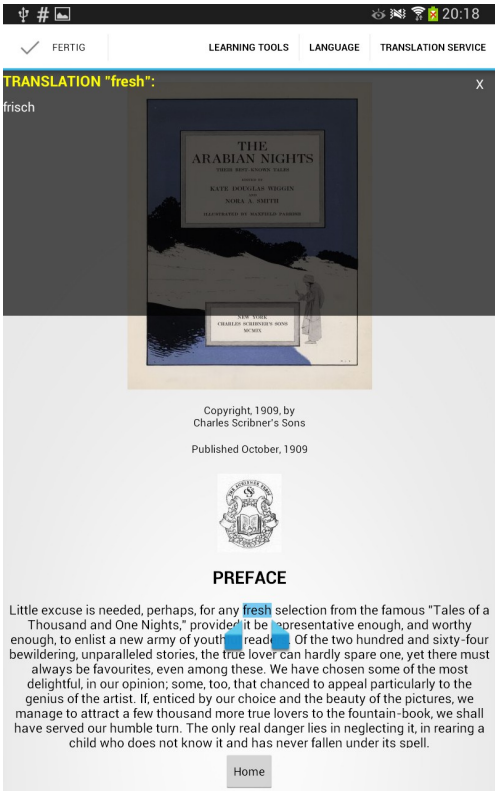
Grundabläufe - Screenshots:



1.) .epub – Datei auswählen



2.) E-Book laden & anzeigen

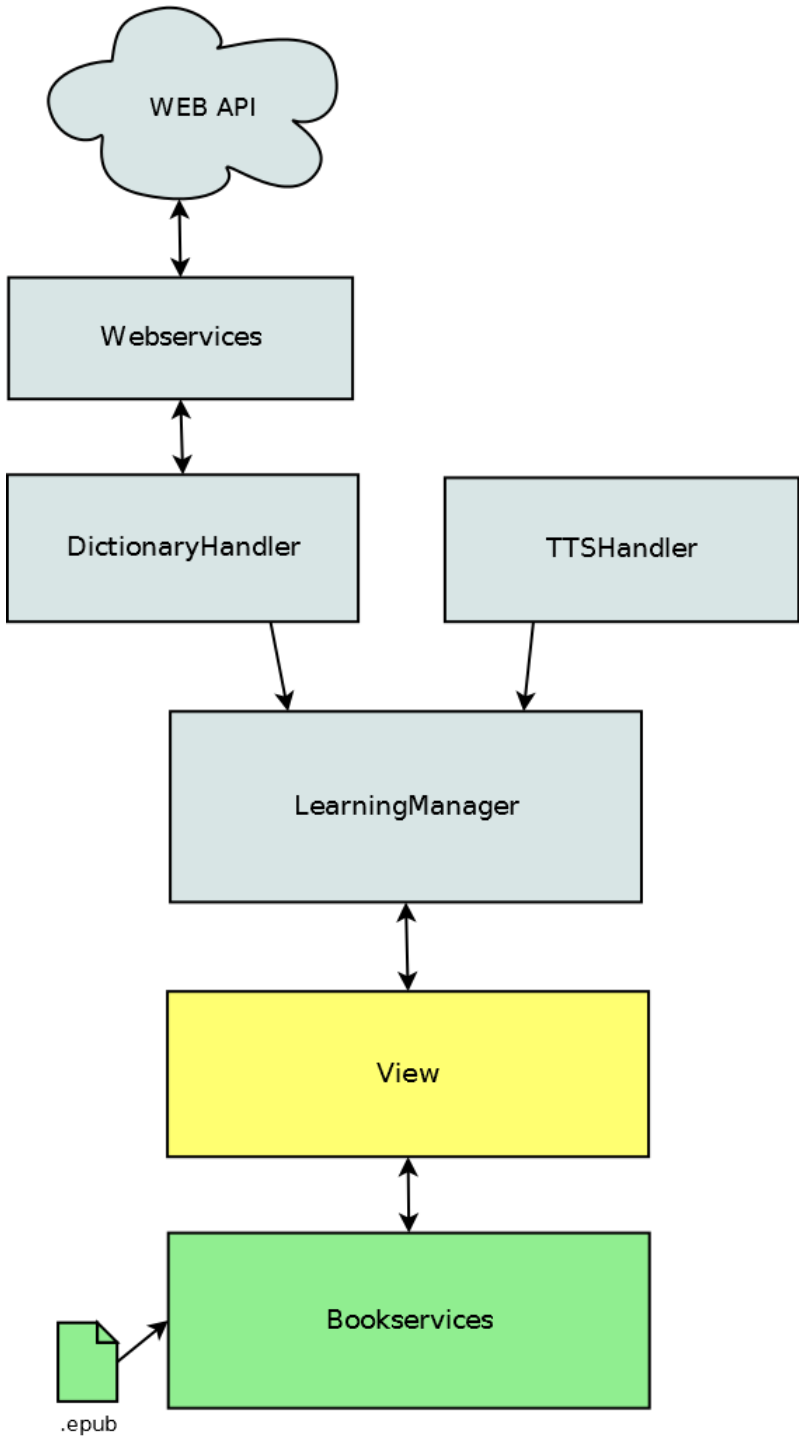


3.) Text markieren und Learning-Tool Funktion wählen:

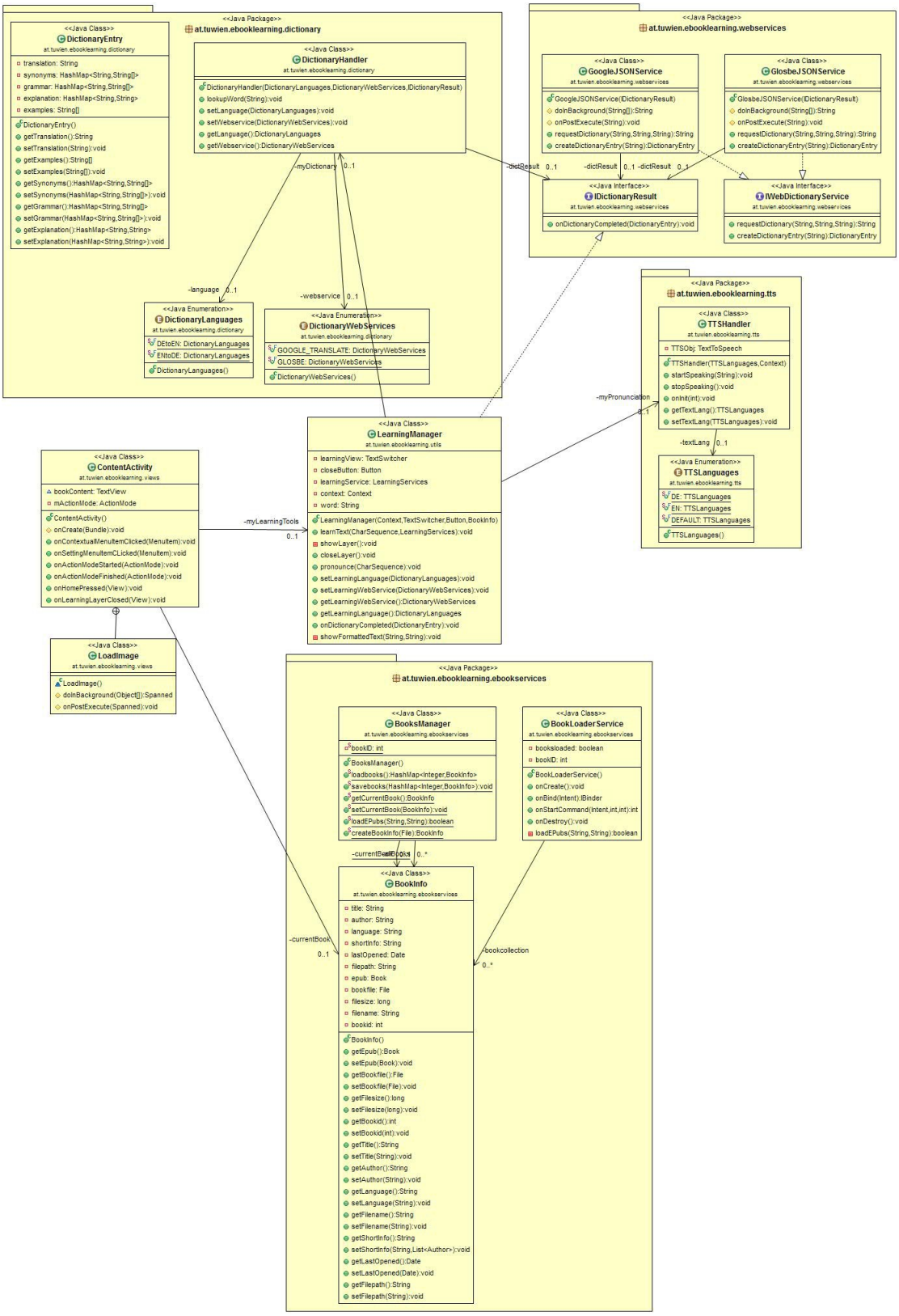
Beispiel: „fresh“ → „frisch“

3 Implementierung:

Architektur:



Klassendiagramm:



3. 1 Darstellung von eBooks:

Für die Anzeige von eBooks im .epub-Format wurde die „*siegmann epub-Library*“ [1] für Android verwendet. Diese stellt in erster Linie Funktionen und Parser für das Extrahieren von eBook-Informationen (z.B.: Inhaltsverzeichnis, Titel, Buchcover etc) zur Verfügung, enthält jedoch keine Komponenten zum Darstellen von eBooks.
Dafür sind eigene Erweiterungen in Form einer *WebView* oder *TextView* notwendig.
Hierbei wurde für die Detailanzeige eine *TextView* verwendet, da für weitere Funktionen der Text ausgewählt werden musste und *WebView* keine Methoden zum Auslesen des selektierten Textes bietet.
Der Nachteil ist, dass die HTML und Style-Informationen in der T*extView* nicht angewendet werden können, weshalb Text und Bilder wie folgt extrahiert werden muss:

```
class LoadImage extends AsyncTask<Object,Void,Spanned> {

		@Override
		protected Spanned doInBackground(Object... params) {
			// TODO Auto-generated method stub
			String displayString= (String) params[0];

			Spanned spanned = Html.fromHtml(displayString, new
			BookImageLoader(getApplicationContext(), currentBook), null);

			return spanned;
		}

		/* (non-Javadoc)
		 * @see android.os.AsyncTask#onPostExecute(java.lang.Object)
		 */
		@Override
		protected void onPostExecute(Spanned result) {
			// TODO Auto-generated method stub
			super.onPostExecute(result);
			bookContent.setText(result);
		}



	}

}
```

Alle Komponenten für das Laden und Managen der .epub-Dateien befinden sich im Package `at.tuwien.ebooklearning.ebookservices`, die dazugehörigen Viewer in `at.tuwien.ebooklearning.view.` (siehe Architektur)

1http://www.siegmann.nl/epublib/android

Vorteile:

- Durch die .epub – Library kein extra Entpacken des .epub-Files notwendig.
- Einfache und verlässliche Textauswahl und -markierung ohne *JavaScript*.
- Eigenes und buchunabhängiges Styling von eBooks möglich.

Nachteile:

- Library unterstützt nur das .epub-Format und keine DRM geschützten Dateien.
- Integrierte CSS, Links etc sind von *TextView* nicht interpretierbar, daher gehen Informationen zur Dokumentstruktur, Styling und Verweise verloren.
- Bilder müssen in einem zusätzlichem Schritt geladen und behandelt werden (BookImageLoader.java)[2]. Zu viele Bilder auf einmal verursachen eine *OutofMemoryException*.

Probleme:

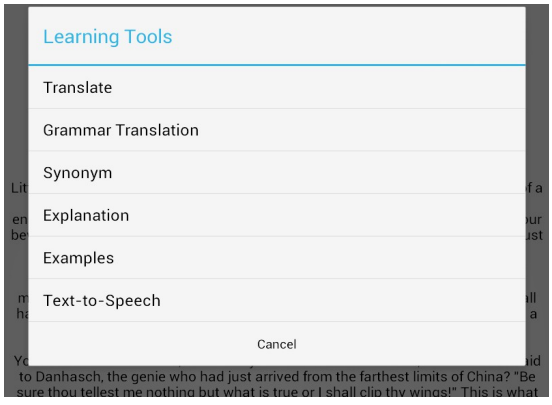
Wenn kein richtiges Inhaltsverzeichnis vorhanden ist, werden nicht nur Bilder von einem Kapitel, sondern vom ganzen Buch geladen. Dabei kann es zu einer *OutofMemoryException*, da in Android jedes Bild als Bitmap im internen Speicher geladen muss.

Ohne Stylesheet ist die Darstellung des eBooks nicht ganz korrekt und die Verweise zu anderen Seiten funktionieren nicht.

Alternativen & Verbesserungen:

- Zu große Bilder skalieren oder die Implementierung einer Speicherverwaltung.
- Dynamisches Nachladen der Bilder (bzw.Text) beim Scrollen.
- Statt direkte Verwendung der Library auf einen bestehenden Open-Source eBook Reader aufbauen, da diese bereits eigene Engines zum Rendern von eBooks enthalten. Beispiele:
  - *FBReader* [3] → Basis WebView
  - *CoolReader* [4] → Basis WebView
  - *PageTurner* [5] → Basis TextView mit eigenem HTMLSpanner

3. 2 Learning Tools - Wortanalyse:



Für die Lernfunktionalitäten werden Webservices verwendet, welche beim Markieren eines Textes im Hintergrund aufgerufen werden. Leider stehen viele solcher Web APIs für Übersetzungen oder Sprachanalyse nicht mehr gratis (z.B.: Google translate u. Bing) zur Verfügung. *Wiktionary* [6] würde zwar als offene Wörterdatenbank alle notwendigen Informationen liefern, jedoch ist die Web API trotz unterschiedlicher Formate ungeeignet und man müsste erst einen aufwendigen Parser für die Website

2 https://github.com/psiegman/epublib/issues/13
3 https://github.com/geometer/FBReader
4 https://github.com/amahule/CoolReader
5 https://github.com/NightWhistler/PageTurner

implementieren.
Daher habe ich mich für folgende Varianten für Deutsch ←- → Englisch entschieden:

- **Google translate [7] (unofficial):**

  Dabei wurde nicht die offizielle, kostenpflichtige WebAPI verwendet, sondern durch Aufruf der *translate.google.com* – URL und entsprechenden Parametern, wird ein *JSON* zurückgeliefert, welches alle relevanten Informationen zu einem Wort oder zu Sätzen liefert. Die genaue URL zur der jeweiligen Sprache kann im Browser mit z.B.: Firebug ermittelt werden:

```
public final static String
GOOGLE1_FROM="https://translate.google.com/translate_a/single?client=t&sl=";
//+ language to translate from
public final static String GOOGLE2_DEST="&tl="; //+ language to translate to
public final static String
GOOGLE3_PHRASE="&hl=de&dt=bd&dt=ex&dt=ld&dt=md&dt=qc&dt=rw&dt=rm&dt=
ss&dt=t&dt=at&ie=UTF-8&oe=UTF-8&prev=bh&ssel=0&tsel=0&sc=1&q=";
// + add word to lookup
```

  Vorteile:

  - Übersetzt auch ganze Sätze u. hat mehr Informationen als Glosbe.com

  Nachteile:

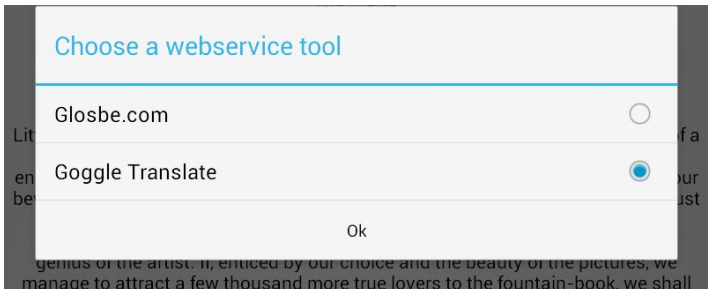
  - URL, Parameter und das JSON – Format kann jederzeit geändert werden, da der Zugriff nicht offiziell ist, die Wortanalyse funktioniert dann nicht mehr.

- **Glosbe.com [8]:**

  Diese Website stellt eine API zur Verfügung, mit der man auf den Suchservice zugreifen und ein JSON aufrufen kann. Glosbe.com enthält mehrere Quellen als Datenbasis unter anderem auch *Wiktionary* und *Google translate.*

  Vorteile:

  - Offizielle WebAPI u. URL mit Dokumentation
  - Enthält mehr Beispielsätze



  Nachteile:

  - Liefert aufgrund der vielen, unterschiedlichen Datensätze nicht immer korrekte Ergebnisse. z.B: „fresh“ wird auch als „frech“ übersetzt. (vgl. Screenshots)
  - Kein Vorhandensein von Grammatik und Synonymen.

6 http://de.wiktionary.org/wiki/API
7 https://rupeshpatel.wordpress.com/2012/06/23/usage-of-google-translator-api-for-free/
8 https://de.glosbe.com/a-api

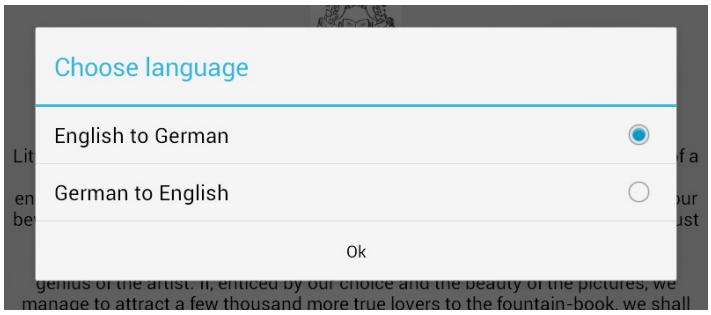
- Kann nur einzelne Wörter übersetzen.

Alle Webservices wurden als *Interfaces* implementiert, daher kann man durch eine einfache Erweiterung weitere Parser für andere Webservices integrieren. Diese können dann im Menü ausgewählt werden, standardmäßig ist *Google translate* eingestellt.

Weiters wurden in der Softwarearchitektur die Komponenten für die LearningTools von der View getrennt, um diese bei Bedarf zu erweitern.
Mit Hilfe des „*LearningManagers“* werden Funktionalitäten des „*DictionaryHandlers*“ und des *„TTSHandlers“* an die jeweilige View gehängt.

3. 3 Learning Tools – Text-to-Speech:

Für die Aussprache eines Wortes wurde die integrierte TTS – Funktion von Android [9]verwendet. Abhängig vom Gerät können herstellerspezifische Sprachbibliotheken (z.B.: Samsung Text-to-Speech) oder Google als Standardbibliothek verwendet werden. Diverse Sprecheinstellungen wie z.B.: unterstützte Sprache, Geschwindigkeit, Sprechertyp (männlich /weiblich), Slang (amerikanisch / britisch) etc sind vom System abhängig und muss im Vorhinein in den Einstellungen des Gerätes vorgenommen werden.



Die Sprache wird jedoch automatisch aus den Metadaten des gewählten eBooks erkannt und beim Start des Buches voreingestellt. Weiters wird auch der gesamte Text bzw. User Interface und das Menü in der App an die jeweilige Sprache angepasst.
Eine Änderung ist jederzeit im Menü möglich, falls z.B.: Lehnwörter vorhanden sind.

Sollten keine Metadaten vorhanden sein, wird defaultmäßig die Standardsprache des Gerätes verwendet.

Verbesserungen:

- Statt der eBook Metadaten soll die Sprache direkt aus dem Wort erkannt werden. Dies wäre möglich, wenn z.B.: ein eBook folgende, standardisierte HTML- Anweisung bei Lehnwörter verwenden würde [10]: `<span lang="en">WorldWideWeb</span>`

9 http://developer.android.com/reference/android/speech/tts/TextToSpeech.html
10 http://wiki.selfhtml.org/wiki/Webstandards/Internationalisierung#Sprachen_maschinenlesbar_deklarieren

4 Resümee:

Das Hauptproblem war nicht die Integration der Webservices, sondern die Darstellung von epub-Dateien, da das Rendering in Android mit *TextView* nicht optimal lösbar ist. Das Problem könnte zwar mit *WebView* gelöst werden, allerdings wird der selektierte Text aus der View für die LearningTools benötigt, welche aus der *WebView* leider nicht ermittelt werden kann. Ein Workaround mittels *JavaScript* (`window.getSelection()`) wäre zwar möglich, wird jedoch nicht in jeder Webkit-Version unterstützt.[11] [12]
Eine Lösung wäre daher bestehende Open-Source eBook Reader zu verwenden, die sich bereits optimal um das Rendering kümmern und diese dann mit den Webservices zu erweitern.

5 Verwendete Ressourcen:

- Eclipse, Android SDK, Android 4.1.2
- epub-Library, (mehr in „readme.txt“)

6 Anhang:

- Eclipse-Projekt: gesamter Sourcecode inkl.Binary .apk
- Test .epub- Dateien
- Dokumente mit Konzepten/Skizzen in /docs/
- Screenshots in /docs/
- Stundenliste mit Recherche-Quellen
- Readme.txt mit Systemanforderungen

7 Quellen:

Diverse Tutorials und Informationen siehe Stundenliste.

11 http://code.google.com/p/android/issues/detail?id=24842
12 http://code.google.com/p/android/issues/detail?id=14540