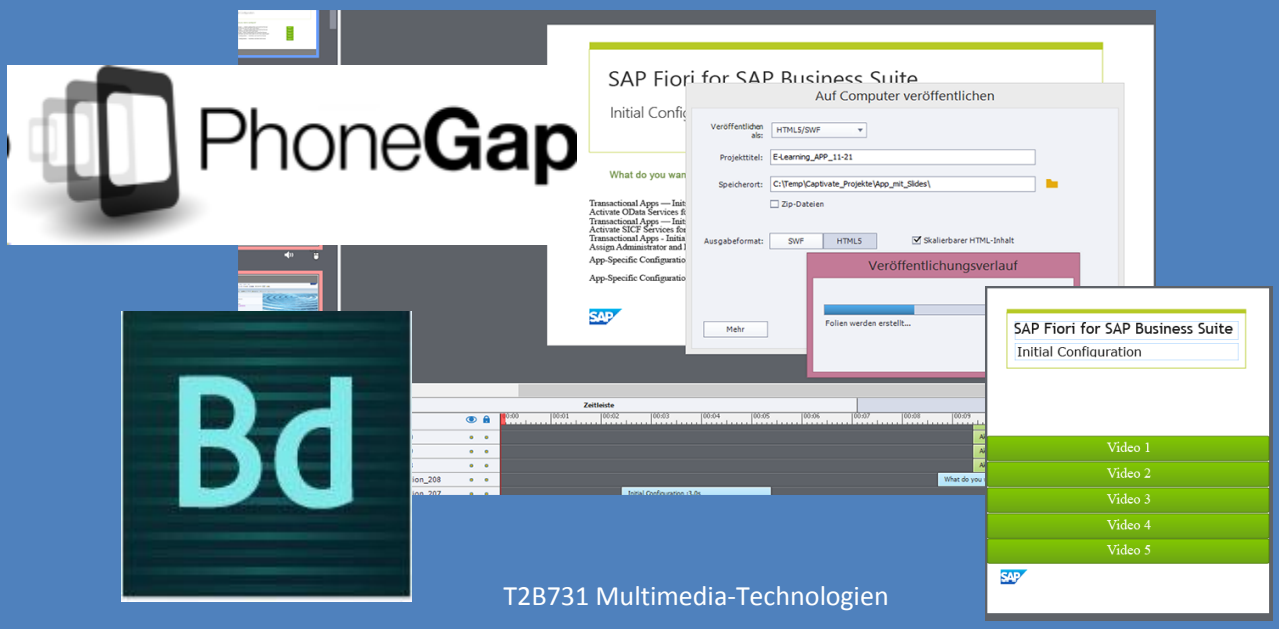
ABSCHLUSSBERICHT

ERSTELLUNG EINER E-LEARNING APP MIT ADOBE CAPTIVATE 8 UND PHONEGAP



T2B731 Multimedia-Technologien

Prof. Martin Schober

Dengra Torres, Laura

Matrikelnummer 50443

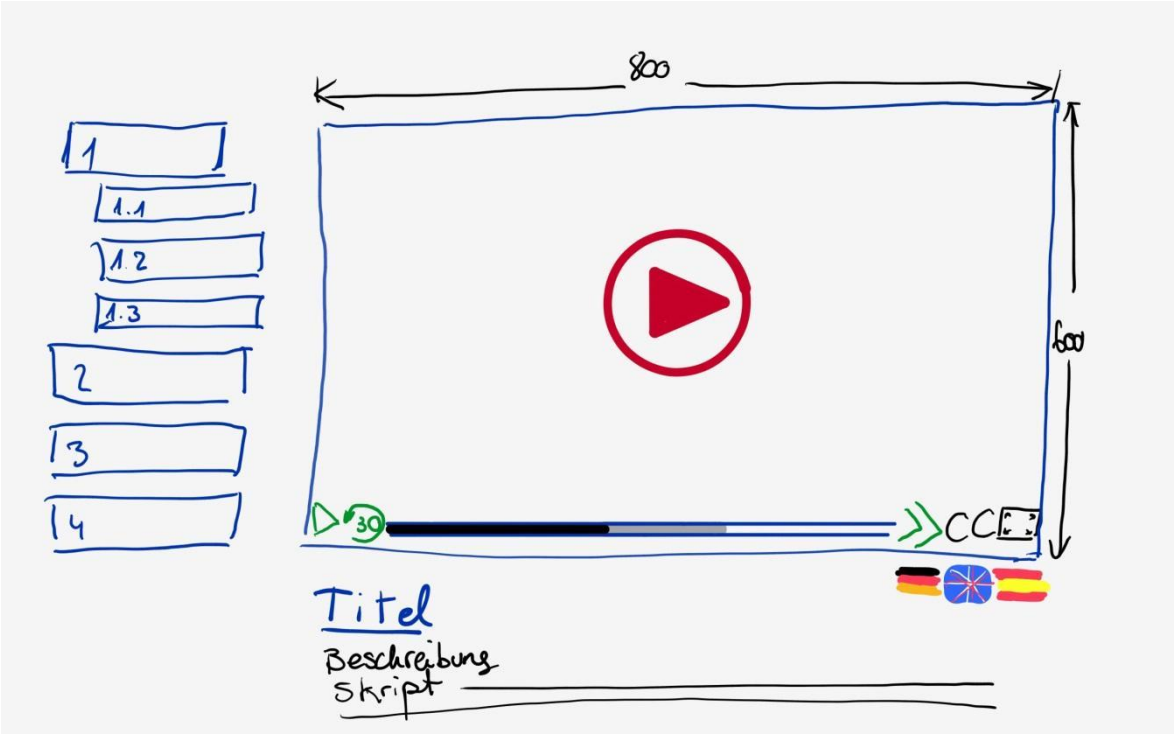
M. Sc. Kommunikation und Medienmanagement

Hochschule Karlsruhe – Technik und Wirtschaft

Inhaltsverzeichnis

Projektplan

Im Rahmen der Veranstaltung *Multimedia-Technologien* sollen die Neuerungen der Version 8 von Captivate näher recherchiert und diese dazu verwendet werden, um ein E-Learning App für die Android Plattform herzustellen. Als Vorbild für das E-Learning Design soll folgende Skizze dienen:



Es soll herausgefunden werden, welche Möglichkeiten Captivate 8 anbietet, um folgende Features in einem E-Learning-Projekt durchzuführen:

Strukturierung eines Video-Tutorials im Inhaltsverzeichnis

Wie man auf der Skizze sehen kann, soll die E-Learning-Umgebung über ein Inhaltsverzeichnis verfügen, mit dessen Hilfe man durch die verschiedenen Kapiteln und Unterkapiteln des Trainings navigieren kann.

Dazu kommt ein Video-Player und darunter die Beschreibung bzw. das Skript zum Unterkapitel.

Responsive Project Design

Wie wird ein „responsive“ Projekt hergestellt und worauf sollte man dabei beachten? Da die E-Learning Umgebung als App generiert werden soll, muss diese für Touchscreens gut bedienbar sein. Dies könnte die Unterstützung von Touch-Gesten bedeuten.

HTML5 Output to App

Wie kann ein E-Learning Projekt als HTML5 ausgegeben werden, sodass es als Android App verwendet werden kann? Die HTML5 Datei soll unter PhoneGap in App konvertiert werden. Welche Probleme können durch die Konvertierung aus Captivate auftauchen?

Projektablauf

Anfangssituation des Projekts

Zur Erstellung der E-Learning Video Serie wurden fünf How-To Videos verwendet, in denen erklärt wird, wie die SAP Anwendung Fiori aufgesetzt wird. Diese Video Serie ist auf der Video Plattform YouTube zu finden[1].

Die verwendeten Videos wurden bei SAP als *Software Simulation* mit Captivate 7 aufgezeichnet, bearbeitet und im mp4 Format veröffentlicht, weshalb mir sowohl die mp4-Dateien als auch die CPTX-Quellprojektdateien aus Captivate zur Verfügung standen. Daher schienen sie für dieses Projekt besonders gut geeignet.

Aufbau der Tutorialserie in Captivate 8

Um unsere Tutorial Reihe aufzubauen, gab es je nach ausgewählte Dateiformat zwei Möglichkeiten. Die erste bestand darin, die fünf fertigen MP4-Dateien in ein neues Captivate Projekt zu importieren, diese als Videos zu behandeln und sie in einzelne Folien hinzuzufügen. Die zweite Möglichkeit war es, die gesamten 2956 Folien mit ihren Inhalten aus den fünf jeweiligen originalen Captivate Projekten zu kopieren und sie zusammen in ein neues Captivate Projekt einzufügen.

Videobasiertes Projekt

Zuerst habe ich ein Captivate Projekt erstellt, aus dem alle Videos aufgerufen werden sollten. Dafür mussten die fünf originalen MP4-Dateien in einzelne Folien eingefügt werden. Dennoch wurden die Videos in der veröffentlichten HMTL Seite nicht richtig angezeigt. Dies lag möglicherweise daran, dass der Videopfad nach der HTML Veröffentlichung nicht mehr gestimmt hat.

Um sicherzustellen, dass der Pfad zu den Videos auch nach der Veröffentlichung stimmt, müssen sie mit einem bestimmten Prozess in das Captivate Projekt importiert werden. Damit Videodateien auf einem HTML E-Learning korrekt abgespielt werden, müssen deren Pfade relativ sein. Der beste Weg, um das zu versichern, ist es, die Videos bei der Erstellung des E-Learnings zunächst auf die Captivate Bibliothek zu importieren, und sie von dort aus in die entsprechende Folie per Drag&Drop zu ziehen. Dies stellt sicher, dass nur die Dateinamen der Videos übernommen werden, und es somit keine Probleme bei der Videowiedergabe in HTML gibt.

---

[1] SAP Fiori Set Up Part 1: https://www.youtube.com/watch?v=rUMGrg2hsNk

Folienbasiertes Projekt

Als nächstes habe ich ein folienbasiertes Projekt ausprobiert, das bei der Bearbeitung Zugang zu sämtlichen Inhalten der How-To-Videos gewährleisten sollte. Die Folien aus den anderen Projekten wurden in einem Gesamtprojekt zusammengefügt und in fünf Gruppen zugeordnet.

Das folienbasierte Projekt sollte gewährleisten, dass sämtliche Folien für den Aufbau des Inhaltsverzeichnisses auftauchen würden, unter denen der technische Redakteur die wichtigsten Inhaltsfolien für ein hierarchisch gegliedertes Inhaltsverzeichnis aussuchen könnte. Diese Gliederung sollte für die spätere Orientierung des Lesers und eine einfachere Navigation hilfreich sein.

Ein neues Projekt mit einer Folie pro Click oder Handlung zu erstellen, hat sicherlich unter Umstände einige Vorteile, jedoch ist das aus Gründen, die sich später im Ablauf des Projekts erwiesen haben, nicht optimal für die Erstellung unserer App. Nachteile der folienbasierten Version werden im Folgenden beschrieben:

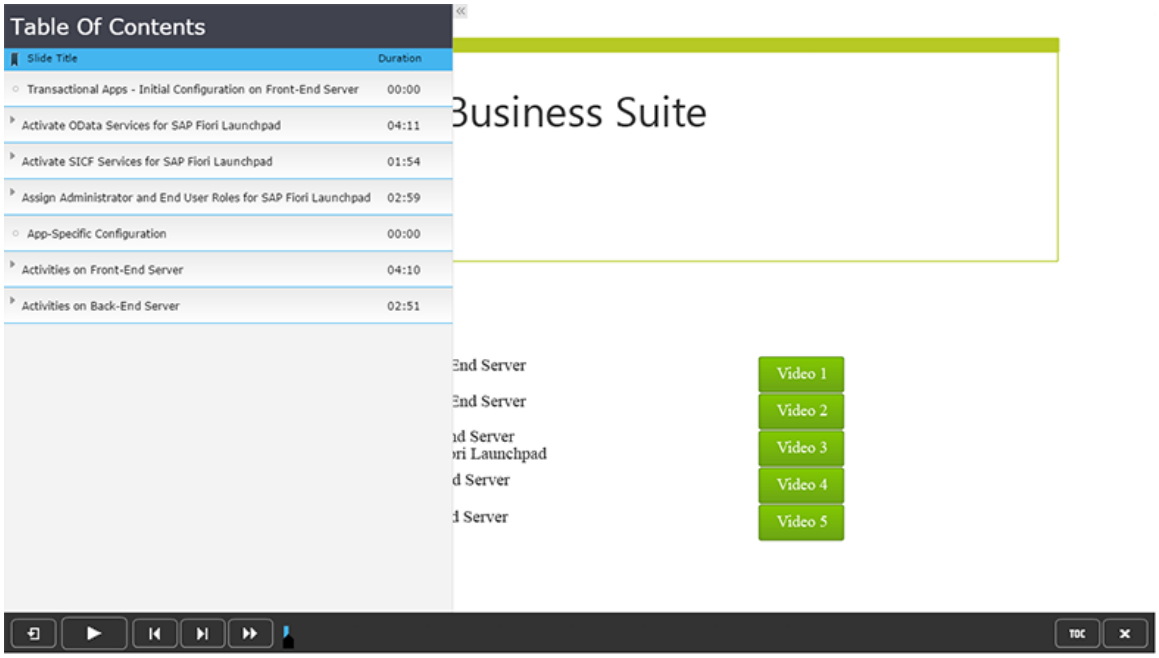
- Das zusammenfügen der Projektfolien hat ein sehr großes Projekt als Folge. Etwa 15 Minuten Tutorial Material, und insgesamt 296 Folien – eine Einführung plus in 5 Foliengruppen aufgeteilten 295 Folien, mit ihren jeweiligen Audiodateien und Bildschirmtexte. Diese riesige Ressourcenbibliothek hat in bestimmten Fällen zu viel Leistung beansprucht und zu einem Absturz des Programms geführt, weshalb von dieser Arbeitsweise eher abgeraten wird.
- Captivate Folien basieren auf Flash Animationen. Daher haben die durchgehenden Bewegungen wie Drag&Drop oder Textanimationen später in der kompilierten App nicht so gut (stockend) funktioniert. Die beste Lösung für eine fliesende Wiedergabe der Animationen scheint es, diese Folien als MP4 Dateien zu exportieren und als solche für die Veröffentlichung in HTML5 zu verwenden.
- Das folienbasierte Projekt mit sämtlichen Inhaltsfolien hat kein Audio in der App Version wiedergegeben, möglicherweise aus dem Kopieren und Einfügen der Folien in das neue Projekt. Dieses Problem konnte folgendermaßen gelöst werden: Bei späteren Erstellung einer dummy App mit in Captivate erstellten Text-to-Speech Audiodateien hat die Audiowiedergabe in der daraus kompilierten App wiederum funktioniert. Wie wir aus der Videowiedergabe gelernt haben, liegt das Problem hier wahrscheinlich auch an einem falschen Pfad, der von Captivate geschrieben wird. So wie das Problem der Videowiedergabe durch das Importieren der Dateien in die Bibliothek und nachträgliche Ziehen in die jeweilige Folie gelöst wurde, wurde das Audioproblem durch die interne Erstellung der Audiodateien

mit Text-To-Speech gelöst. Beide Methoden haben sichergestellt, dass Captivate die Pfade sowohl für die Video- als auch für die Audiodateien in das HTML Code richtig geschrieben hat.

- Nach der App Erstellung anhand eines folienbasierten Projektes wurden die Bilder kleiner gemacht, ohne die Auflösungsproportionen zu berücksichtigen. Das hatte als Folge ein verzerrtes Bild im Handy Output. Dieses Problem hat dafür gesprochen, Projekt mit automatischer Anpassung (responsive) zu erstellen.

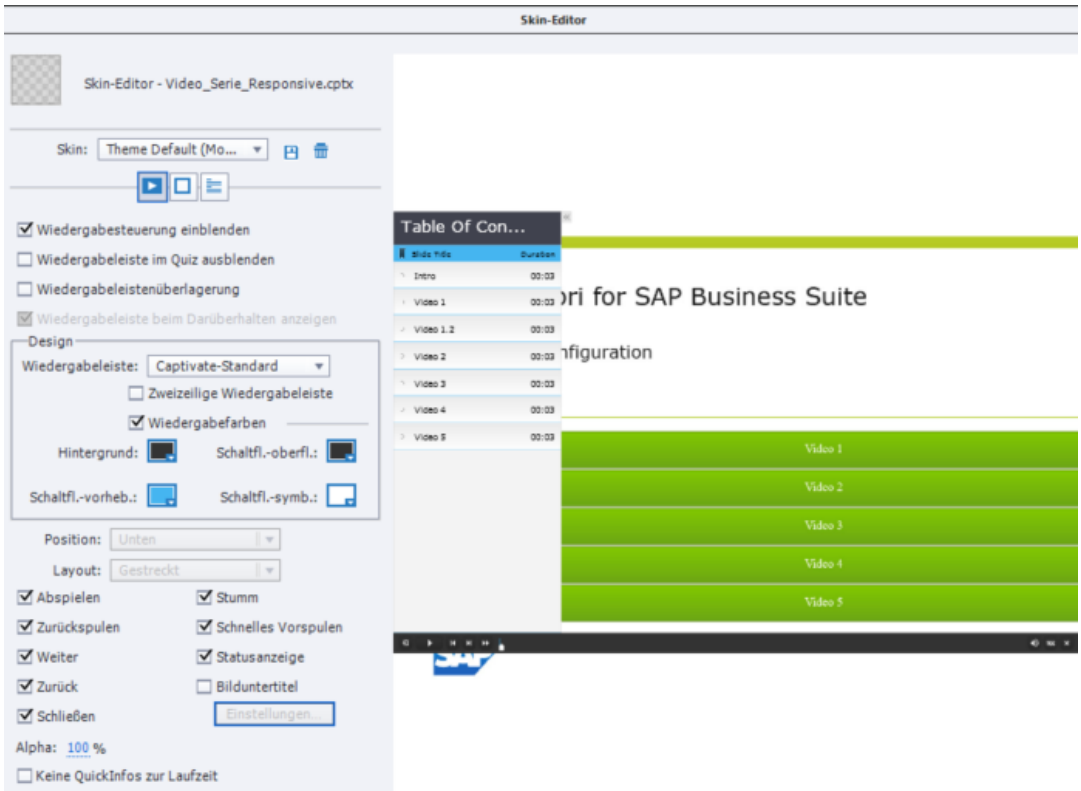
Gestaltung des Inhaltsverzeichnis und der Wiedergabeleiste

Für die E-Learning App war es wichtig, die Orientierung innerhalb eines breiten Inhalts zu vereinfachen. Deswegen wurde hier die Captivate Funktion des Inhaltsverzeichnisses verwendet. Dank dieser Funktion können die Inhalte des Lernprogramms gegliedert und hierarchisch geordnet werden.



Um einen unproblematischen Ablauf der Indexerstellung zu gewährleiten, ist es wichtig, dass die Foliennamen gepflegt werden. Die Foliennamen werden in das Inhaltsverzeichnis automatisch

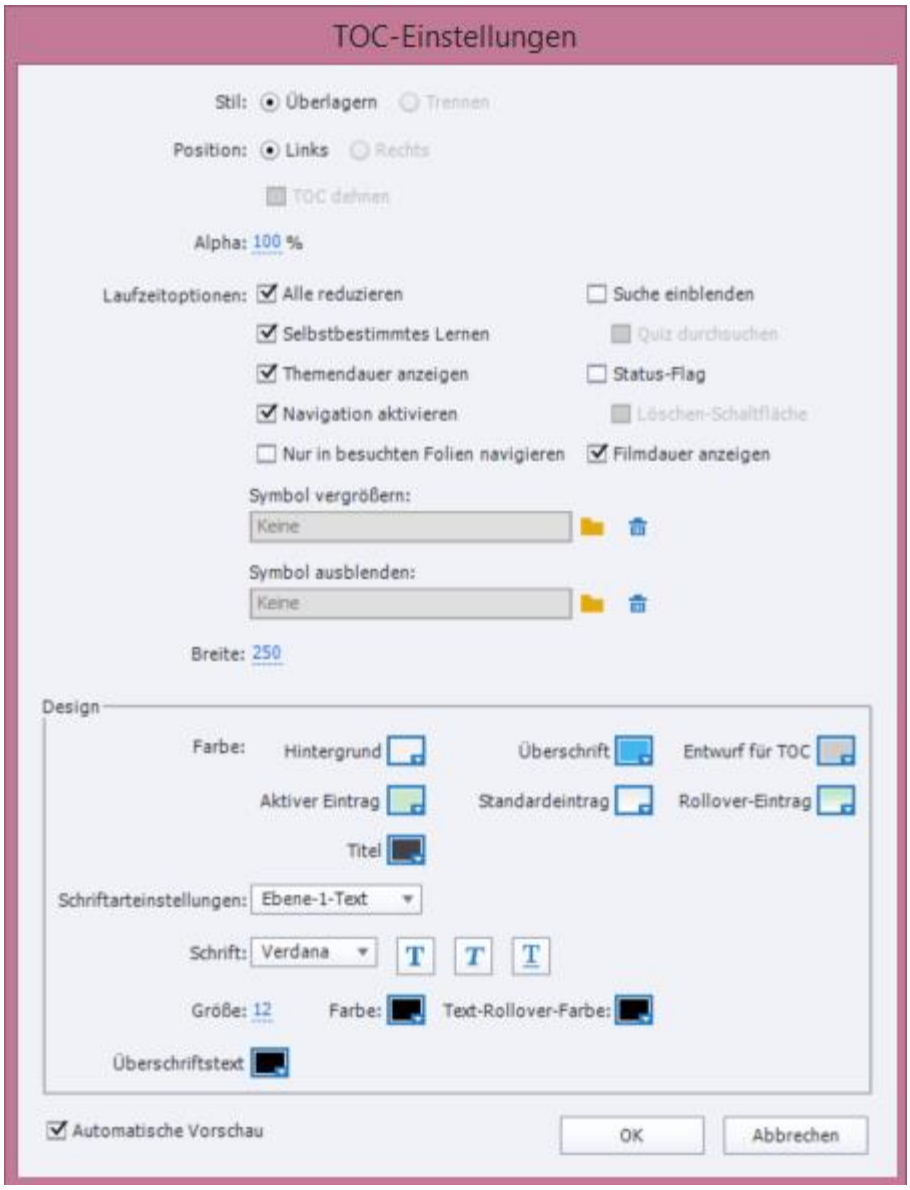
importiert, und somit ist kein manuelles Eintragen in das Inhaltsverzeichnis nötig.



Außerdem war ein einheitliches Farbenkonzept für die App notwendig, mit den Farben des Unternehmens, das die App beauftragt, in diesem Fall SAP. Aus diesem Grund wurden die SAP eigene Farben Blau und Grün für das Inhaltsverzeichnis und die Wiedergabe leiste übernommen, um ein Cloud adäquaten Look&Feel zu geben.

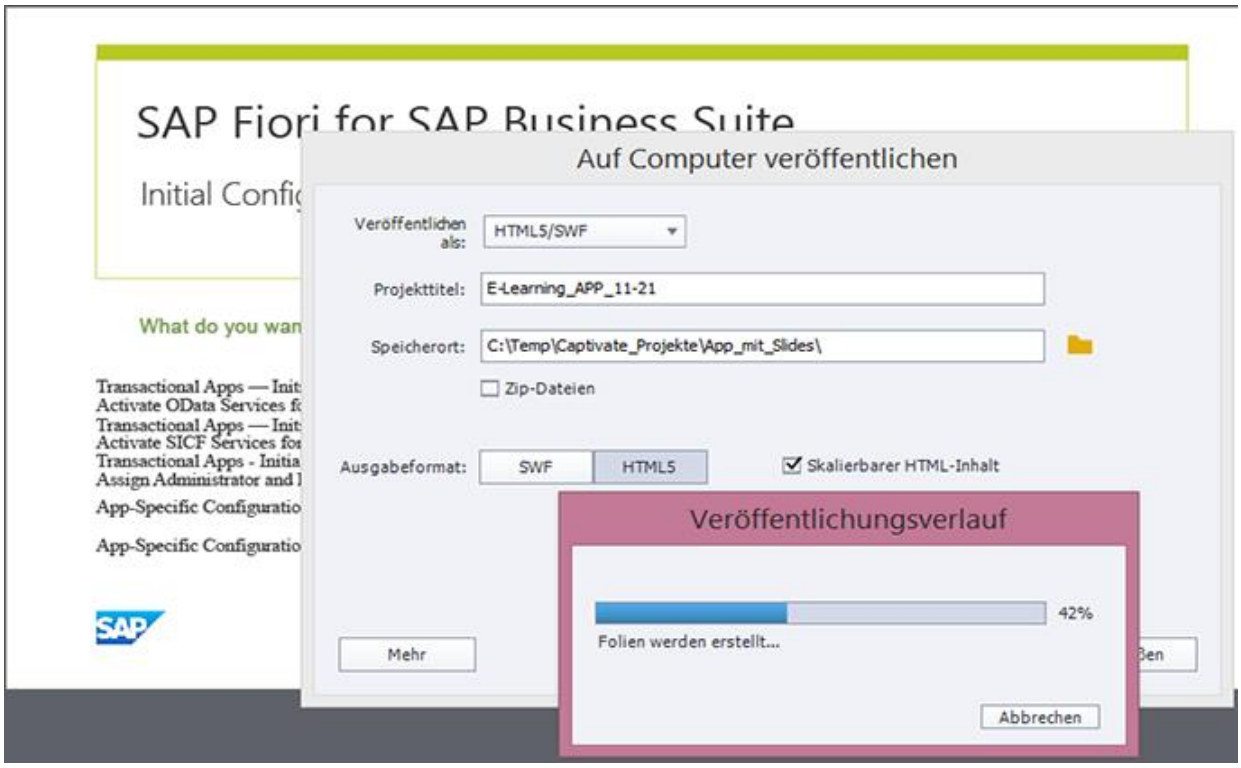
Für die Wiedergabeleiste mussten verschiedene Designs ausprobiert werden: Die Captivate vorgefertigte Wiedergabeleiste *cpPlaybarMobile* sieht zwar geeignet aus, aber sie ist nach der Veröffentlichung zu groß, um die Statusanzeige auf Smartphones darstellen zu können. Daher habe ich mich für die Captivate-Standard Wiedergabeleiste entschieden und ihre Farben dem SAP

Branding Image angepasst.



Erstellung und Test des HTML Projekts und erste App

Zur Veröffentlichung des Captivate Projekts wurde die Option „als HTML5 veröffentlichen“ ausgewählt. Dazu wurden die Auswahlkasten „Zip-Dateien“ und „Skalierbarer HTML-Inhalt“ markiert. Der erste sollte für die Veröffentlichung des Projekts über PhoneGap behilflich sein, der zweite sollte dafür sorgen, dass sich die Darstellung auf die Größe des Endgerätes anpasste.



Um das HTML Projekt online zu testen, wurde dieser mittels des FTP Programm *Total Commander* auf den Server www.technischeredaktion.de hochgeladen. Hier konnten das folienbasierte[2] Projekt und (nach späterer Behebung des Problems mit den Videopfaden) das Projekt mit intern gespeicherten Videos erfolgreich wiedergegeben werden.

Der Versuch, die Videos auf dem TR-Server zu speichern und sie von dort aus aufzurufen, hat leider nicht funktioniert. Möglicherweise muss man dafür entweder Änderungen im HTML Code vornehmen, damit die Videos aus einer externen Quelle abgespielt werden können, oder sie in einem geeigneten Server (Adobe Streaming Service) speichern.

Da das bisherige folienbasierte Projekt auf mehreren nicht responsive Projekten basiert hatte und mit einer einzigen Auflösung veröffentlicht werden konnte, wurde die daraus veröffentlichte App richtig auf dem Tablet, aber zu groß auf dem Smartphone dargestellt. Dies hat mich dazu geführt, die Inhalte des bisherigen Projekts auf einem Projekt mit „automatischer Anpassung“ zu migrieren.

[2] http://www.technischeredaktion.com/Fiori/E-Learning_APP_11-21/index.html

Die Lösung – Captivate Projekt mit automatischer Anpassung

Bei dem Design des Captivate *responsive* Projektes (dt. mit automatischer Anpassung) wurde schnell klar, dass die Anpassung nicht ganz automatisch funktioniert. Damit der Aufbau eines responsive Projektes relativ unproblematisch funktioniert, muss man die technischen Hintergründe verstehen, wie die Eigenschaften von Objekten aus einem Modus in das andere übernommen werden.

Beim Aufbauen eines responsive Projekts muss man immer im Hinterkopf haben, dass das Design für responsive Projekte in Captivate von oben nach unten aufgebaut ist[3]. Das heißt, die Änderungen, die man an der Desktop Version vornimmt, bewirken sich auf die Tablet und Mobile Ansichten. Jedoch werden die Änderungen, die in der mobilen Ansicht vorgenommen werden, nicht für die höheren Stufen der Hierarchie übernommen, immer nur für die unteren Stufen (die Änderungen in der Tablet Ansicht werden also nur für die Mobilansicht, aber nicht für Desktopansicht übernommen. Das heißt, wenn man Inhalt – z. B. Text – für alle Gerätansichten hinzufügen möchte, muss man das immer in der Desktop Ansicht vornehmen.

Wenn man Inhalte in ein responsive Projekt hinzufügt, muss man in den Positionseigenschaften der jeweiligen Objekte festlegen, inwiefern deren Position und Höhe in die anderen Modi übernommen werden sollen. Die Objektposition und Größe können z. B. anhand absoluter Werte, oder (dank der Smart-Shape Funktion) bezüglich einer Slide oder eines anderen Objektes verkleinert werden. Dafür muss man auswählen, ob die Pixelinformation oder die Prozente auf die andere Modi übernehmen werden sollen. Wie das genau funktioniert, wird anhand dieses Videos[4] verbildlicht.

Dazu sollte man berücksichtigen, dass die Auflösung eines Smartphones nicht der Auflösung der Inhalte entspricht, die auf dem Gerät wiedergegeben werden können. Als Beispiel soll die Auflösung vom Samsung Galaxy S4 dienen, der zum Testen verwendet wurde. Laut des Herstellers verfügt das Gerät über eine Auflösung von 1.080 x 1.920 Pixel. Jedoch werden App-Inhalte mit dieser Auflösung

---

[3] http://youtu.be/dDxxjfaFo9o?t=5m48s

[4] http://youtu.be/dDxxjfaFo9o?t=27m10s

zu groß dargestellt. Grund dafür ist, dass nicht die Bildschirmauflösung, sondern die „Viewport Size“ (Ansichtsbereich) für die Wiedergabe von App-Inhalten zählt. Laut der von Adobe empfohlenen[5] Quelle „View Port Size“[6] verfügt der Smartphone Galaxy S4 über einen Ansichtsbereich von 640x335 Pixels. Solche Werte sollten also auf der Projektgröße übernommen werden.

Erstellung der App mit Adobe PhoneGap Build

Für die Erstellung der Android-App wurden verschiedene Methoden ausprobiert, um einen angepassten Prozess zu finden. Die erste APK Datei wurde dank der Web-Anwendung PhoneGap erstellt. Diese Methode hat den großen Vorteil, dass man keine zusätzlichen Programme installieren muss, sondern es reicht, die Source Dateien (HTML, JS, CSS Dateien usw.) als ZIP-Datei zu komprimieren und auf der PhoneGap Webseite hochzuladen, um die gewünschte App Installationsdatei zu bekommen (im Android Fall eine APK Datei).

Jedoch hat die PhoneGap Web-Anwendung gewisse Nachteile für das vorgesehene Projekt, die zu einer erheblichen Reduktion des Inhalts führen würden. Die Web Version der PhoneGap Technologie ist auf einen HTML Projekt begrenzt, der nicht mehr als 20 MB groß sein darf. Dazu kommt, dass man nur ein App gleichzeitig erstellen darf.

Die erste App Datei wurde wie folgt erstellt:

1. nach der „Veröffentlichung“ der E-Learning Video Serie mit Captivate als HTML Projekt, musste der Inhalt des Ordners, in dem sich die index.html Datei befindet, in eine ZIP-Datei hinzugefügt werden.
2. Für das Hochladen war ein Konto bei Adobe nötig, denn PhoneGap mittlerweile von Adobe angeboten wird. Diese ZIP-Datei, in unserem Beispiel „Fiori_Set_Up.zip“, wurde unter der PhoneGap Seite hochgeladen. Die Seite fragt nach den Betriebssystemen, die unterstützt werden sollen. Da die App-Erstellung für IOS einen eigenen Developer Sicherheitsschlüssel benötigt, der anhand eines eigenen iPads bzw. iPhones generiert wird, kam für unsere App nur Android in Frage.
3. Apk Datei herunterladen

[5] http://youtu.be/dDxxjfaFo9o?t=3m30s

[6] http://viewportsizes.com/mine/

4. APK auf das Device testen. Folien wurden richtig angezeigt. Problem: die Audio Dateien wurden nicht wiedergegeben.

In der genaueren Recherche nach PhoneGap Features wurde klar, dass die dass die Web Version keine Medienwiedergabe im Zusammenspiel mit Captivate unterstützt. Die Medienwiedergabe sollte laut der PhoneGap Dokumentation durch die Installation eines zusätzlichen Plug-Ins[7] erfolgen.

Da die vorgesehene E-Learning App ab dem zweiten Video die Größe von 20 MB überschreitet, und wegen der angesprochenen Probleme bei der Audiowiedergabe des HTML Projekts schien es ein Versuch wert, die Cordova Technologie lokal zu installieren, sprich das Cordova Framework auszuprobieren.

Das Cordova Framework

Installationsprozess

Da die maximal Größe, die von der Anwendung PhoneGap Build akzeptiert wird, 20 MB beträgt, reicht diese für unser Projekt nicht aus. Das spricht dafür, das Projekt entweder zu verkleinern oder besser auf das Cordova Framework umzusteigen, der keine solchen Begrenzungen hat.

Daher habe ich mich erst mit der Installation von Cordova in meinem Computer beschäftigen müssen. Dafür waren viele Schritte nötig. Da ich die verschiedenen Dokumentationsseiten der jeweiligen Tools benötigt habe, habe ich die Installationen in einer anderen Reihenfolge durchgeführt, als hier aufgelistet, was zu viele Fehlern bei der App-Erstellung und somit zu viel mehr Aufwand geführt hat, als wenn man die Programme in der folgenden Reihenfolge durchgeführt hätte. Wie die Installation optimal Schritt für Schritt laufen sollte, wird hier aufgeführt:

1. Installation von Node.js[8].
2. Installation von Cordova durch das *Node.js Command Prompt*[9]. Dafür muss man in die Command line folgendes eintippen: „`npm install -g cordova`“.
3. Installation von Java Development Kit (JDK), damit Eclipse richtig ausgeführt wird.

---

[7] https://github.com/apache/cordova-plugin-media.git

[8] Diese kann man unter http://nodejs.org herunterladen und durch eine Installationsdatei installieren.

[9] http://docs.phonegap.com/en/4.0.0/guide_cli_index.md.html#The%20Command-Line%20Interface

3.1. Wenn JDK installiert ist, muss man den „bin“ Ordner „C:\Program Files\Java\jdk1.8.0_25\bin“ in den Pfad unter Systemumgebungsvariablen einfügen[10]. Hier müssen die Pfade immer durch ein Semikolon getrennt werden.

3.2. Zu diesem Punkt sollte man überprüfen, ob JDK richtig installiert wurde. Das ist wichtig, damit die Kompilierung der Apps später funktioniert. Dafür muss man *cmd* ausführen und „`java`“ eintippen.

4. Installation von *Eclipse* und *Android SDK*[11]. Diese Installation erfolgt nicht durch eine Installationsdatei, sondern der Inhalt von der heruntergeladenen Zip-Datei – sprich die Ordner *Eclipse*, *SDK*, und die Anwendung *SDK Manager* – muss unter einem sinnvollen Ordner extrahiert werden, z. B. „C:\Development\“

4.1. Hier muss man die Systemumgebungsvariablen mit den Pfaden zu den Ordnern *platform-tools* und *tools* aktualisieren, die sich im *sdk* Ordner befinden, in meinem Fall waren sie „C:\Development\sdk\platforms“ und „C:\Development\sdk\platform-tools“.

4.2. – bis mindestens Komponente 19

5. Installation von Apache Ant. Um die Plattform kompilieren zu können. Dafür muss man die Datei apache-ant-1.9.4-bin.zip herunterladen und auf C:\ extrahieren.

5.1. Wenn man Ant auf C: extrahiert hat, dann müsste man den Pfad „C:\apache-ant-1.9.4\bin“ finden können – dieser Pfad muss man in die Umgebungsvariablen einfügen (der Pfad muss wieder durch ein Semikolon vom letzten Pfad getrennt werden).

5.2. Um zu überprüfen, ob ant richtig installiert wurde, kann man im Command prompt einfach `ant` eintippen.

---

[10] https://docs.oracle.com/javase/8/docs/technotes/guides/install/windows_jdk_install.html#BABGDJFH

[11] http://developer.android.com/sdk/installing/installing-adt.html

6. Wenn die Installation erfolgreich war, sollte man bei Eingabe von „`cordova -v`“ bzw. „`phonegap -v`“ die installierte Version als Antwort bekommen (s. Abb.).

```
C:\Users\Laura\AppData\Roaming\npm\node_modules\phonegap\node_modules\connect-p
onegap\node_modules\socket.io\node_modules\socket.io-client\node_modules\engine
 "C:\Program Files\nodejs\node_modules\npm\bin\node-gyp-bin\\..\..\node_modules
node-gyp\bin\node-gyp.js" rebuild
C:\Users\Laura\AppData\Roaming\npm\phonegap -> C:\Users\Laura\AppData\Roaming\n
m\node_modules\phonegap\bin\phonegap.js
phonegap@3.6.3-0.22.4 C:\Users\Laura\AppData\Roaming\npm\node_modules\phonegap
├── pluralize@0.0.4
├── colors@0.6.0-1
├── semver@1.1.0
├── minimist@0.1.0
├── qrcode-terminal@0.9.4
├── shelljs@0.1.4
├── prompt@0.2.11 (revalidator@0.1.8, pkginfo@0.3.0, read@1.0.5, utile@0.2.1, w
nston@0.6.2)
├── phonegap-build@0.9.0 (qrcode-terminal@0.8.0, optimist@0.3.7, shelljs@0.0.9,
phonegap-build-api@0.3.3)
├── connect-phonegap@0.13.0 (home-dir@0.1.2, connect-inject@0.3.2, ip@0.3.1, mk
ath@0.1.0, findit@2.0.0, ncp@0.6.0, shelljs@0.2.6, request-progress@0.3.1, user
gent@2.0.8, node-static@0.7.0, tar@0.1.19, gaze@0.4.3, request@2.33.0, archiver
0.10.1, localtunnel@1.3.0, socket.io@1.0.6, connect@2.12.0)
└── cordova@3.6.3-0.2.13 (q@0.9.7, underscore@1.4.4, nopt@2.2.1, cordova-lib@0.
1.13)

C:\Users\Laura>cordova -v
4.1.2

C:\Users\Laura>phonegap -v
3.6.3-0.22.4
```

7. Am Ende wurde das „GitHub Client”, das zur Installation von Plug-Ins in Cordova Projekten dienen sollte.

Erstellung der App mit Cordova

Nächster Schritt: Erstellung einer Sample App mit PhoneGap mit dem Command Line Tool. Wenn Cordova einmal installiert ist, kann man mit der App Erstellung anfangen. Dafür muss man folgende Schritte durchführen.

1. Im Command Line Tool, wählt man den entsprechenden Dateipfad aus, in dem man das App erstellen möchte. Obwohl ich den Ordner „C:\Users\Laura\my-app“ ausgewählt habe, wäre es besser gewesen, den Ordner Workspace von Eclipse auszuwählen.

```
Node.js command prompt
Your environment has been set up for using Node.js 0.10.33 (x64) and npm.

C:\Users\Laura>cd my-app
```

2. Im ausgewählten Verzeichnis, wurde folgendes eingetippt: „`cordoba create test_phonegap com.phonegap.test_phonegap PhoneGapTest`“. Dies erstellt ein

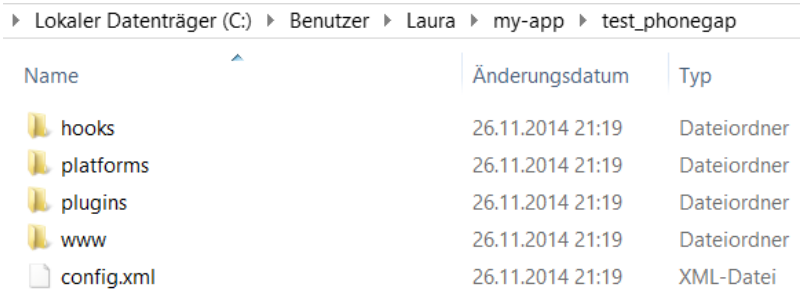
neues Verzeichnis, in diesem Fall „test_phonegap“. Darin befindet sich unter anderem der Ordner *www*, in den man später den HTML Projekt aus Captivate speichern kann.

```
C:\Users\Laura\my-app>cordova create test_phonegap com.phonegap.test_phonegap Ph
oneGapTest
Creating a new cordova project with name "PhoneGapTest" and id "com.phonegap.tes
t_phonegap" at location "C:\Users\Laura\my-app\test_phonegap"
```

3. Dann muss man in das neu erstellte Verzeichnis wechseln und die Plattform Android hinzufügen mit der Eingabe „`cordova platform add android`“.

```
C:\Users\Laura\my-app>cd test_phonegap

C:\Users\Laura\my-app\test_phonegap>cordova platform add android
npm http GET https://registry.npmjs.org/cordova-android/3.6.4
npm http 304 https://registry.npmjs.org/cordova-android/3.6.4
Creating android project...
Creating Cordova project for the Android platform:
        Path: platforms\android
        Package: com.phonegap.test_phonegap
        Name: PhoneGapTest
        Android target: android-19
Copying template files...
Project successfully created.
```

4. Dies erstellt in das test_phonegap Verzeichnis den Ordner *platforms*, wo die jeweiligen Plattformen erstellt werden. Da für unsere App nur Android in Frage kommt, wird hier nicht weiter auf die iOS App-Erstellung eingegangen.

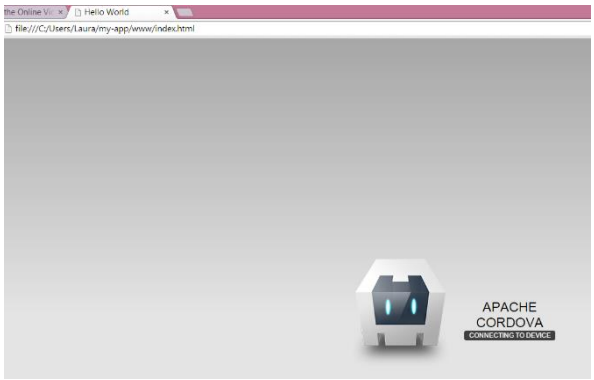


Sobald der Platforms Ordner in unserem Projekt steht, können die Apps für die jeweiligen Plattformen erstellt werden, in unserem Fall heißt das, die Android App wird mit folgender Eingabe kompiliert „`cordova build android`“

```
C:\Users\Laura\my-app\test_phonegap>cordova platform add android
npm http GET https://registry.npmjs.org/cordova-android/3.6.4
npm http 304 https://registry.npmjs.org/cordova-android/3.6.4
Creating android project...
Creating Cordova project for the Android platform:
        Path: platforms\android
        Package: com.phonegap.test_phonegap
        Name: PhoneGapTest
        Android target: android-19
Copying template files...
Project successfully created.
```

```
C:\Users\Laura\my-app\test_phonegap>cordova compile android
Running command: C:\Users\Laura\my-app\test_phonegap\platforms\android\cordova\b
uild.bat
Buildfile: C:\Users\Laura\my-app\test_phonegap\platforms\android\build.xml

-set-mode-check:

-set-debug-files:

-check-env:
 [checkenv] Android SDK Tools Revision 23.0.5
 [checkenv] Installed at C:\Development\sdk

-setup:
     [echo] Project Name: CordovaApp
```

5. Da in unserem *www* Verzeichnis lediglich eine Beispiel index.html Datei lag, wird nur eine Beispiel App daraus kompiliert. Hier ist die Sample Webseite, die cordova auf dem *www* Verzeichnis erstellt, und die später durch unser Projekt ersetzt werden soll:



6. Wenn unser HTML Projekt in das www Verzeichnis eingefügt wurde, kann die „richtige“ App kompiliert werden. Um die App inklusive unserem Inhalt zu erstellen, wird der Befehl „`cordova build android`“ eingegeben.
7. Die App kann schließlich im Eclipse Emulator geladen werden.

Problematik bei der Medienwiedergabe

Zu diesem Zeitpunkt konnte überprüft werden, dass – sowie im PhoneGap Build – die Audiowiedergabe und die Videowiedergabe bei der App aus dem Captivate HTML Projekt noch nicht funktioniert. Wie sich herausstellte, hat die Android App Erstellung mit PhoneGap Build oder mit Cordova nur ohne Audio und ohne Video funktioniert.

Es wurde versucht, mittels verschiedener Media Plug-Ins und Anpassungsmöglichkeiten das Audio in einer mit Cordova erstellten App zu wiedergeben, jedoch bisher ohne Erfolg.

Die nicht Wiedergabe von Audiodateien oder Videodateien aus Captivate Projekten innerhalb einer PhoneGap erstellten App scheint ein häufigeres Problem zu sein. [12] Momentan unterstützt Captivate

[12] Selbst wenn man das HTML selbst programmiert, wie in diesem Forum Eintrag: http://stackoverflow.com/questions/22515699/play-sound-on-phonegap-app-for-android

keine Audio/Video-Wiedergabe, wenn der HTML5 Output mit PhoneGap oder Cordova als App veröffentlicht wird. Als Test habe ich unter anderem den Media Plug-In für PhoneGap mit dem Command Prompt Tool installiert, jedoch konnte das so nicht gelöst werden. Eventuell könnte dieses Problem mit verschiedenen Plug-Ins für PhoneGap behoben werden, jedoch habe ich nach ausführlicher Recherche soweit keins gefunden, der den Audio/Video-Wiedergabefehler löst. Eventuell wäre eine Lösung durch eine tiefere Auseinandersetzung mit dem HTML Code aus Captivate möglich.

Bisher war es nicht möglich, mit den erstellten Apps auf externen Quellen zuzugreifen. Es wäre notwendig einen Plug-In zu finden, der es zulässt, eine App zu erstellen, die Quellen aus externen Domänen aufrufen kann. Möglicherweise dient der Tag `<access origin="*" >`[13] dazu, Zugang zu anderen Servern zu erlauben.

Als Fazit für diesen Teil des Projekts, lässt sich folgendes schlussfolgern: Das Cordova Framework scheint besonders für große Lernsoftwareprojekte gut geeignet zu sein, die direkt in HTML – z. B. mit Dreamweaver – programmiert wurden. Jedoch hat Captivate noch einige zu beachtende Kompatibilitätsprobleme – wie die Audiowiedergabe aus Folien – die mit dem Cordova Framework nicht ohne weiteren Plug-Ins oder spezieller Anpassung funktionieren.

Ein Grund dafür ist, dass Adobe die Unterstützung von Mobile Devices in Captivate für iOS optimiert hat und momentan noch an der Optimierung der Unterstützung von Android Apps arbeitet. Ein guter Beweis dafür ist der neue Update, der vor kurzem für Captivate herausgegeben wurde. Um Audio in der Android App problemlos abspielen zu können, kann man bisher nur die Captivate interne PhoneGap Funktion verwenden, die Oktober 2014 mit der Captivate 8.0.1 Update zur Verfügung gestellt wurde.

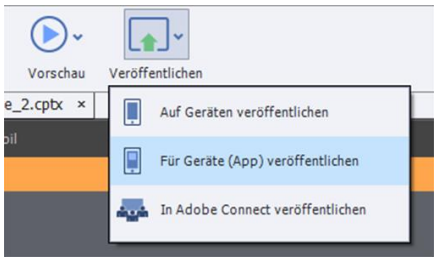
Erstellung der App mit Adobe Captivate App Packager (ab Captivate 8.0.1)

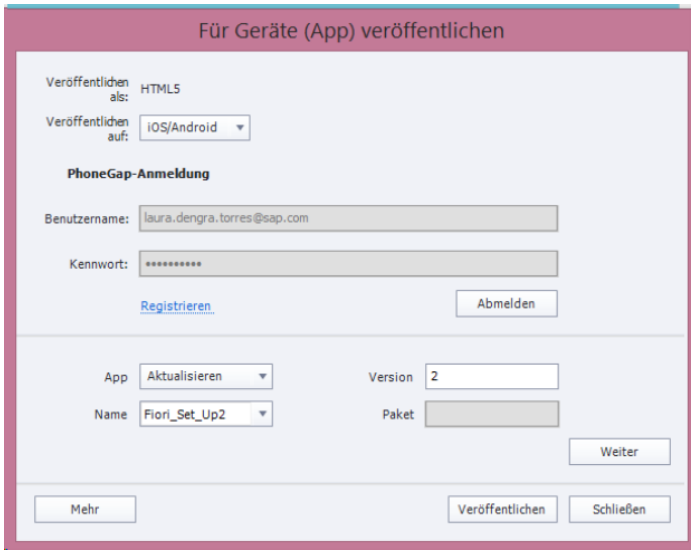
Vor kurzem hat Adobe Captivate ein neues Update herausgebracht, der eine direkte App-Veröffentlichung durch PhoneGap ermöglicht. Mit dem neuesten Captivate 8.0.1 Update kann man den Captivate App Packager verwenden, um aus einem HTML5 Projekt ein App zu erstellen. Diese Methode, die auf der PhoneGap Technologie basiert, funktioniert besser als die PhoneGap Cloud Version. Bei dem Captivate App Packager gibt es keine Probleme bei der Audio- oder

[13] http://docs.build.phonegap.com/en_US/configuring_access_elements.md.html#Access%20Elements

Videowiedergabe, die man in der PhoneGap Cloud Version sowie mit einer installierten PhoneGap Version vorfindet.

Der Unterschied zu dem bisherigen Captivate App Packager besteht darin, dass jetzt kein HTML Publizieren und kein Zippen mehr notwendig ist, sondern dass das Captivate Projekt direkt als App veröffentlicht werden kann. Hierbei sind außer Android auch iOS und Windows Mobile unterstützt. Allerdings funktioniert es mit den gleichen Beschränkungen wie PhoneGap Build. Man muss dafür ein PhoneGap Account einrichten, es kann nur ein App gleichzeitig erstellt werden und es gilt die gleiche Beschränkung von 20 MB wie bisher.





Auf dem ersten Blick und gibt es für diese App Erstellungsmethode keine Anpassungsmöglichkeiten, wie Änderungen an der Config.xml Datei, die die Einstellungen für die Erstellung der App vorgibt. Allerdings wird auf der Adobe Captivate Seite von einem Developer erwähnt, wie man an die config.xml Datei gelangt. Diese befindet sich unter dem Pfad „{installation_directory}/Templates/Publish/phonegapConfig.xml“

Hier können dann die Anpassungen für die App vorgenommen werden. In dieser Datei gibt es einige Attribute standard, die für die vorgesehene Video App verändert werden mussten. Folgende Zeilen der xml Datei sollen als Beispiel dienen:

```
<preference name="phonegap-version" value="3.3.0" />
```

Diese Zeile legt die PhoneGap Version fest, die von Captivate verwendet wird. Leider wird hier eine veraltete Version. Diese kann man mit der neuesten Version ersetzen.

```
 <preference name="fullscreen" value="false" />
```

In dieser Zeile kann man festlegen, ob die App in Vollbildmodus erstellt werden soll. Für unsere App müsste man den Wert in „true“ umschreiben, da es für das Abspielen von Videos sinnvoller ist.

```
<gap:plugin name="org.apache.cordova.device" version="0.2.8"/>
```

Außerdem werden in dieser Datei die Plug-Ins festgelegt, die bei der App-Erstellung verwendet werden. Die Versionen sind hier auch veraltet, also sollte man bei diesen Zeilen die neueste Version in die jeweiligen Plug-In-Attribute eingeben. Diese Änderungen dürften mehr Flexibilität bei der nativen Captivate PhoneGap Erstellung ermöglichen.

Dies könnte bei folgendem „Fehler“ von Captivate App Packager angewendet werden: Da keine separate HTML Veröffentlichung vor der App Erstellung stattfindet, gibt es keine Möglichkeit, das Projekt „skalierbar“ zu machen. Wenn es nicht in einem Projekt mit automatischer Anpassung gearbeitet hat, wird die App bei der gelieferten config.xml Datei zu groß für Smartphone-Bildschirme ausgegeben. Also muss man folgendes berücksichtigen: wenn man für Devices veröffentlichen möchte, sollte man entweder die config.xml Datei entsprechend mit dem Attribut „skalierbar“ anpassen oder auf jedem Fall ein Projekt mit automatischer Anpassung (responsive) verwenden.

Bei der ersten Erstellung einer App mit dem neuesten Adobe Captivate App Packager, wurden verschiedene Schwierigkeiten deutlich, die bei der nächsten Version der App beachtet wurden:

- Anhand eines extrem verkürzten *dummy* Projekts (weniger als 20 MB) konnte überprüft werden, dass es prinzipiell möglich ist, Videos in das Captivate Projekt einzubinden und sie auf der App abzuspielen. Jedoch ist hierbei die 20 MB Begrenzung schnell überschritten. Eine Lösung wäre es, aus der App auf eine externe Quelle – wie YouTube oder ein SMTB Streaming Server auf die Videos – zugreifen. Da ohne Plug-In kein Zugriff auf externen Quellen durch die App möglich war, könnte man möglicherweise diese App-Berechtigung in der config.xml Datei vergeben. Dies dürfte dann das Problem der Größeneinschränkung deutlich verbessern.

- Bei der Veröffentlichung eines Captivate Projektes als Android App muss berücksichtigt werden, dass Flash Animationen auf mobilen Geräten nicht korrekt wiedergegeben werden. Manche Animationen werden korrekt konvertiert, jedoch ruckelt das Bild bei Animationen wie Texteingabe- oder Full-Motion-Animationen. Die beste Lösung ist also, die Animationen (am besten die ganze Software Simulation) als MP4 zu veröffentlichen, sie in die Projektbibliothek zu importieren und dann als Videofolien einzubinden. Dies verbessert die Performance der App enorm.

Erstellte Projekte

Im Rahmen dieses Projektes sind folgende Lernprogramme erstellt worden.

E-Learning Video Serie:

HTML Projekt:

http://www.technischeredaktion.com/Fiori/E-Learning_APP_11-21/index.html

HTML Responsive Projekt:

http://www.technischeredaktion.com/Fiori/Video_Serie_Responsive_index.html

Video Tutorial: Erstellung einer Android App mit PhoneGap:

https://www.youtube.com/watch?v=vS67Gcz-Cs8

Android App:

Mit dem nativen Captivate Packager erstellt. Dies stellt eine verkürzte Version dar, da ein Importieren sämtlicher Videos aus Größeneinschränkungsgründen nicht möglich war.

QR Code:

Alternativ kann folgende Webseite aufgerufen werden:

https://build.phonegap.com/apps/1167588/builds

Präsentation, als Webseite aufrufbar:

http://www.technischeredaktion.com/Fiori/Praesentation_App_Erstellung/index.html

Fazit und weitere Schritte

Bei diesem Projekt wurde festgestellt, dass man mit Captivate nicht nur Software-Simulationen als Video erstellen kann, sondern auch ausführliche Lernprogramme mit verschiedenen Videos und anderen Inhalten für verschiedene Gerättypen erstellen und sie als Webseite auf eine Server veröffentlichen kann, ohne Programmieren zu können.

Es wurde festgestellt, welche die Vor- und Nachteile bei den verschiedenen Projektgestaltungsformen „Software Simulation“ „Videofolien“ und „Projekt mit automatischer Anpassung“ zu berücksichtigen sind und welche Auswirkungen die App-Erstellung auf der Grundlage der verschiedenen Projekttypen haben.

Dazu konnten die Möglichkeiten und Einschränkungen bei der Inhaltsverzeichniserstellung und Gestaltung in der Veröffentlichung für verschiedene Bildschirmgrößen herausgefunden werden.

Es wurden Schwierigkeiten beim Importieren des Inhalts in die Captivate Bibliothek festgestellt, die beim Projektaufbau unbedingt zu berücksichtigen sind, damit die Inhalte nach der Veröffentlichung richtig wiedergegeben werden.

Was die App-Erstellung aus Captivate-Projekten angeht, lässt sich schlussfolgern, dass sie für kleinen Projekten mittlerweile gut funktioniert, solange man den neuen PhoneGap-basierte Captivate App-Packager verwendet. Dennoch funktioniert diese Methode nicht, wenn man ein großes Projekt mit Audio/Video Inhalten erstellen möchte. Für größere Projekte ist es eher Cordova empfehlenswert, jedoch ist es hierfür eine sehr aufwändige Einarbeitung nötig, wobei man sich auch grundlegend mit den Cordova Plug-Ins befassen müsste.

Nach diesem Projekt wird die Frage aufgeworfen, inwiefern sich die mit PhoneGap/Cordova erstellten Apps durch den Einsatz der Apache Cordova Plug-Ins anpassen lassen und ob beispielsweise ein Zugriff auf extern gespeicherten Videos ermöglicht werden kann.

Dazu wäre es nötig herauszufinden, wie mit Cordova/Eclipse eine App erstellt werden kann, die auf der Android Plattform (auf iOS funktioniert das schon) Video und Audio wiedergeben kann.

Verwendete Quellen

Adobe Systems Inc. (2014). *PhoneGap | Home*. Von http://phonegap.com/ abgerufen

Adobe Systems Software Ireland Ltd. (13. Oktober 2014). *Adobe Captivate-Hilfe | Neue Benutzerfreundlichkeit*. Von http://helpx.adobe.com/de/captivate/using/new-user-experience.html#webobj abgerufen

Adobe, & Jaisingh, P. (13. Oktober 2014). *Rapid eLearning | Adobe Captivate Blog*. Von A free update to Adobe Captivate 8 is now available!: http://blogs.adobe.com/captivate/2014/10/a-free-update-to-adobe-captivate-8-is-now-available.html abgerufen

AdobeELearning. (13. Oktober 2014). *YouTube*. Von Adobe Captivate 8.0.1: Native App Publisher : https://www.youtube.com/watch?v=3bl_O_gP92Q&list=UULpsk3UF7lh5l9fVkLIqu6g&index=8 abgerufen

AdobeELearning. (25. November 2014). *YouTube*. Von Create Your First Responsive Course: http://youtu.be/dDxxjfaFo9o?list=UULpsk3UF7lh5l9fVkLIqu6g abgerufen

Apache Cordova. (2014). *PhoneGap Documentation:*. Von http://docs.phonegap.com/en/3.5.0/index.html abgerufen

Gardner, A. (8. Oktober 2013). *stackoverflow*. Von PhoneGap "git" command line tool is not installed: http://stackoverflow.com/questions/19256096/phonegap-git-command-line-tool-is-not-installed abgerufen

GitHub. (27. November 2014). *cordova-plugin-media index.md at master • apache cordova-plugin-media • GitHub.html*. Von https://github.com/apache/cordova-plugin-media/blob/master/doc/index.md abgerufen

goldengekko, & Jern, M. (04. April 2014). *PhoneGap, The only mobile app cross-platform solution that works*. Von http://www.goldengekko.com/?blog_post=mobile-app-cross-platform-solution-works abgerufen

Google Inc. (3. Dezember 2014). *Android SDK |Android Developers*. Von http://developer.android.com/sdk/index.html abgerufen

lynda.com, & Griffith, C. (23. Oktober 2014). *lynda.com*. Von Building Mobile Apps with the PhoneGap Command-Line Interface: http://www.lynda.com/Nodejs-tutorials/Building-Mobile-Apps-PhoneGap-Command-Line-Interface/170057-2.html abgerufen

PhoneGap. (31. Juli 2012). *YouTube*. Von Introduction to PhoneGap - An Open Source Framework: https://www.youtube.com/watch?v=wOH4aGows40 abgerufen

Samsung Electronics GmbH. (28. Mai 2013). *Samsung Galaxy S4 - LTE Smartphone - GT-I9505.html* . Von http://www.samsung.com/de/consumer/mobile-device/smartphone/smartphone/GT-I9505ZKADBT abgerufen

SAP. (25. Juni 2014). *YouTube*. Von SAP Fiori Set Up Part 5 | App-Specific Configuration - Activities in Back-End Server: https://www.youtube.com/watch?v=fyUSXd2_eHw abgerufen

SAP. (27. Mai 2014). *YouTube*. Von SAP Fiori Set Up Part 2 | Transactional Apps - Initial Configuration on Front-End Server - Activate SICF Services for SAP Fiori Launchpad: https://www.youtube.com/watch?v=2kyXje4v50o abgerufen

SAP. (27. Mai 2014). *YouTube*. (SAP SE) Von Fiori Set Up Part 1 | Transactional Apps - Initial Configuration on Front-End Server - Activate OData Services for SAP Fiori Launchpad: https://www.youtube.com/watch?v=rUMGrg2hsNk abgerufen

SAP. (11. Juni 2014). *YouTube*. Von SAP Fiori Set Up Part 3 | Transactional Apps - Initial Configuration on Front-End Server - Assign Administrator and End User Roles for SAP Fiori Launchpad: https://www.youtube.com/watch?v=OjSBvr13MrE abgerufen

SAP. (25. Juni 2014). *YouTube*. Von SAP Fiori Set Up Part 4 | App Specific Configuration - Activities on Front-End Server: https://www.youtube.com/watch?v=dJloQd5d2EQ abgerufen

Schäfer, H. (2014). *Open UI5 Developer | The Swiss Knife for UI5 Developers*. Von PhoneGap: http://openui5.blogspot.com/p/phonegap.html abgerufen

Stoecker, D. (2013). *eLearning – Konzept und Drehbuch. Handbuch für Medienautoren und Projektleiter* (2. Ausg.).

Wikipedia. (2. Dezember 2013). Von Viewport: http://de.wikipedia.org/wiki/Viewport abgerufen