Semsix Touch

Adaption des Streaming Musik-Players “Semsix” für das Samsung Galaxy Tab mit Adobe Flex 4.5

Projektdokumentation (v1.1)

„Semsix Touch“ ist eine Projektarbeit im
Studiengang Medieninformatik an der
Fachhochschule Stuttgart – Hochschule der Medien

Niko Andrija Rukavina

Betreuer: Dr. Ansgar Gerlicher
Projektmanagement: Ingo Schock

Veranstaltung: Projekt/Tutorium (20556), WS 2010/11

Kurzfassung

Dieses Dokument hält die Ergebnisse und Schritte meines Studienprojektes „Semsix Touch“ fest und dient zu dessen Bewertung. Der Projektarbeit ging ein Steckbrief mit den gesetzten Lernzielen und erwarteten Resultaten voraus. Diese werden rückblickend überprüft und in diesem Zuge besondere Herausforderungen und Lernerfolge dokumentiert. Es folgt eine stichwortartige Auflistung der einzelnen Arbeitsschritte.

Der Anhang umfasst eine Auswahl des entstandenen Konzeptmaterials und eine kategorisierte Auflistung aller verwendeten Online Ressourcen für dieses Projekt.

Inhaltsverzeichnis

Kurzfassung **2**

1 Projektbeschreibung und Eckdaten **4**
1.1 Beschreibung und Ausgangslage 5

2 Projektziele im Rückblick **6**
2.1 Allgemeines Projektziel 6
2.2 Ziele der Phase 1 7
2.3 Ziele der Phase 2 7
2.4 Ziele der Phase 3 10

3 Herausforderungen und Lösungsansätze **11**
3.1 Migration Flex 3.2 > Flex 4.5 11
3.2 Prototyp - nächste Schritte zum Release 12
3.3 Exemplarisches Problem und Lösungsansatz 13

4 Lessons Learned **16**

Anhang **17**

A Notizen **18**

B Wireframes **21**

C Ressourcen **24**

1 Projektbeschreibung und Eckdaten

Die wichtigsten Fakten:	
Name	„Semsix Touch“
Bearbeitungszeit	Ca. 240 Stunden
Vorgängerprojekt	„Semsix“, Martin Jakobus und Ingo Schock
Projektmanager	Ingo Schock (Computer Science and Media)
Plattform	Adobe AIR (2.5) for Android
Entwicklungsumgebung	Flash Builder 4 (Burrito) / Eclipse
Zusätzliche Software	Balsamiq Wireframing Tool, Adobe Photoshop CS3
SDK	Adobe Flex 4.5 (Hero) – pre-release
Event-Mapping	Mate Flex Framework (Version 0.9)
Endgerät	Samsung Galaxy Tab
Betriebssystem	Android 2.2 (Froyo)

1.1 Beschreibung und Ausgangslage

Ausgangspunkt der Arbeit war eine MI Projektarbeit aus dem Wintersemester 2009, namens „Semsix“. Sie wurde durchgeführt von Ingo Schock und Martin Jakobus. Es handelt sich um Musiksuchmaschine, welche frei verfügbare Musik auf gängigen online Streaming Plattformen (Bsp. YouTube) sucht. Die Ergebnisse werden mittels eines aufwändigen Bewertungsverfahrens nach zuverlässigen Treffern gefiltert und die Bezeichnungen wie Titel und Album mit einer Musikdatenbank verglichen. Dies dient zudem der korrekten Benennung von Namen und Alben nach einem einheitlichen Schema innerhalb der Applikation. Weitere Features sind die online Verwaltung von Abspiellisten, das Anzeigen von Liedtexten und eine Suche nach ähnlichen Titeln.

Die Anwendung läuft innerhalb des Adobe Flash Players als Webapplikation im Browser:

Screenshot http://www.semsix.com, Adobe Flex 3.2 Applikation

Die Adaption auf ein Tablet-System sollte mittels des noch nicht veröffentlichten **FLEX 4.5 SDK (Codename Hero)** erfolgen, welcher Entwicklern insbesondere zur Vorschau seiner Fähigkeit, Android kompatible AIR Applikationen zu erzeugen zur freien Verfügung steht. Adobe Air ist eine Laufzeitumgebung für Flashtechnologie und das Desktop-Pendant zum bekannten Flash Player für Browser-Anwendungen. Mit dem aktuellen AIR Release 2.5 ist diese Umgebung auch ab Android 2.2 für mobile Systeme nutzbar. Die mit dem Flex 4.5 erzeugten Anwendungen verhalten sich ähnlich nativen Apps des mobilen Betriebssystems.

2 Projektziele im Rückblick

Rückblickend werden in diesem Abschnitt die Ergebnisse gegen die, Mitte Oktober 2011 im ***Projektsteckbrief*** definierten Ziele geprüft.

2.1 Allgemeines Projektziel

> Der Schwerpunkt der Arbeit liegt in einem kompletten Redesign der Benutzeroberfläche, im Rahmen eines User-zentrierten Designprozesses.

Ergebnis:

Im Laufe des Projekts hat sich der Fokus wesentlich auf die Auseinandersetzung mit technischen Herausforderungen und die Konzeptionierung der Oberfläche mittels Wireframes verschoben. Eine Neugestaltung ist dabei erzielt worden, ein formaler Test fand jedoch nicht statt.

2.2 Ziele der Phase 1

Evaluation von „Semsix“ nach Kriterien der Benutzerfreundlichkeit

- Evaluation zweier weiterer gängiger Mediaplayer, „Grooveshark“ und „Windows Media Player“.
- Die Analyse des State of the Art gestengesteuerter Benutzerinteraktion

Ergebnis:

Da sich die technischen Faktoren als wesentlich kritischer für das spätere Ziel eines lauffähigen Prototypen für die MediaNight zeigten, folgte früh eine Zurücknahme des Anspruchs auf Nutzerzentriertem Designprozess. Die Evaluation von vergleichbaren Anwendungen erfolgte daher ebenso eher explorativ, statt wie geplant, innerhalb eines formalen Verfahrens. Die Analyse des „State of the Art“ andererseits bildet dafür das Ziel der Thesis-Ausarbeitung.

2.3 Ziele der Phase 2

Redesign

- Wireframing und Early Evaluation eines neuen, touch-optimierten Interfaces für Semsix
- Grafische MockUps des Zieldesigns mit Adobe Photoshop

Ergebnis:

Eine große Zahl Konzeptzeichnungen (Scribbles) zu User Interface Layouts und einzelnen Elementen ist entstanden. Diese Arbeit war insbesondere für die Sensibilisierung und Zielfindung der Bachelor Thesis hilfreich.

Eine frühe Evaluation dieser Ergebnisse konnte im Rahmen von ad-hoc Feedback mit Kommilitonen stattfinden. Nach mehreren Generationen von Wireframes ist ein finaler Prototyp-Mockup entstanden. Das Design der Applikation auf konzeptioneller Ebene (Layout, Controls, Features) stellte die Nutzer auf der MediaNight vor wenige Schwierigkeiten und stieß auf positive Resonanz.

Jedoch, ein echter Nutzertest des Prototyps nach Usability-Kriterien wie ursprünglich geplant, gestaltete sich schwierig auf Grund der hohen Latenz zwischen Nutzereingabe und Reaktion des Systems (siehe Kapitel Herausforderungen und Lösungsansätze).

Konzept

Screen Design

Umsetzung Prototyp

2.4 Ziele der Phase 3

- *Einarbeitung in den vorhandenen Quellcode von Semsix*
- *Einarbeitung in die Flex-Technologie*
- *Anpassung der vorhandenen Codebase zur Implementierung auf dem Zielsystem mit Hilfe des Adobe Flash Builders 4.*
- *Präsentation des Ergebnisses*
- *Etablierung eines eigenen Komponentenframeworks für gestengesteurte Interaktionselemente in Flex.*

Bewertung:

Diese Hauptziele konnten erfüllt werden, mit Ausnahme einer Eigenentwicklung von Komponentenframeworks. Nach der konkreten Anpassung erwies sich der Schritt vom Emulator auf das Endgerät als relativ unkompliziert und nahtlos. Zur Überprüfung der Codeanpassungen wurde hauptsächlich der Emulator verwendet, auf Grund des niedrigeren Turnarounds - der Zeit zwischen Kompilieren, Testen und Beginn der nächsten Änderung.

Das Zieldesign konnte mit Ausnahme einiger weniger UI-Elemente funktionsfähig auf dem Endgerät umgesetzt werden. Der Prototyp zeigte sich durch eine hohe Latenz zwischen Nutzereingabe und Reaktion des Systems als eingeschränkt nutzbar, davon abgesehen aber funktionsfähig (mehr dazu siehe Kapitel: Herausforderungen und Lösungsansätze).

Ein eigenes Komponentenframework war weit außerhalb der Reichweite des Projekts und zum gegenwärtigen pre-release Stadiums des Flex SDK 4.5 wenig sinnvoll. Dafür führte die intensive Auseinandersetzung mit der Funktionsweise der vorhandenen Komponentenarchitektur (notwendig für Implementierung des „Add-Buttons“) zu einem genaueren Verständnis dieses Aspektes der Flex-Architektur.

Ein enger Bezug zwischen Thesis und dem System muss verworfen werden, da die Iterationszyklen zwischen Entwurf, Implementierung und Test zu viel Zeit beanspruchen würden.

3 Herausforderungen und Lösungsansätze

Die größten technischen Herausforderungen waren:

- Aufsetzen einer produktiven Entwicklungsumgebung (Eclipse, Treiber für Galaxy Pad, Verbindung der IDE ans Endgerät zum Deployment via USB)
- Migrationsprobleme auf Grund der erheblichen Unterschiede zwischen Flex 3.2 und Flex 4.5
- Neue Sicherheits-Restriktionen der AIR Runtime im Vergleich zum Flashplayer
- Feature Unvollständigkeit des Flex SDK 4.5
- Hohe Komplexität der zugrundeliegenden Implementierung

3.1 Migration Flex 3.2 > Flex 4.5

Zur aufwändigen Migration von der Webanwendung in Flex 3.2 auf eine kompilierende mobile Applikation waren folgende wesentliche Schritte nötig:

- Anpassung aller Flex-Klassen, da sich die Vergabe von Namespaces zwischen Flex 3 und 4.5 wesentlich unterschied
- das Umschreiben bestimmter deklarativer Konstrukte (beispielsweise müssen alle nicht-sichtbaren Komponenten ab Flex 4 in einem <declerations> Tag untergebracht werden.)
- Zusätzliches Integrieren des ursprünglichen SDK 3.2, um das Nutzen von in 4.5 nicht mehr vorhandenen Komponenten zu ermöglichen, insbesondere den Akkordeon-Container und Tabellen.
- Aktualisieren des Mate Frameworks
- Neues Applikationslayout
- Anpassen der Konfigurationsdateien die eine lauffähige Version auf dem Endgerät ermöglichen.

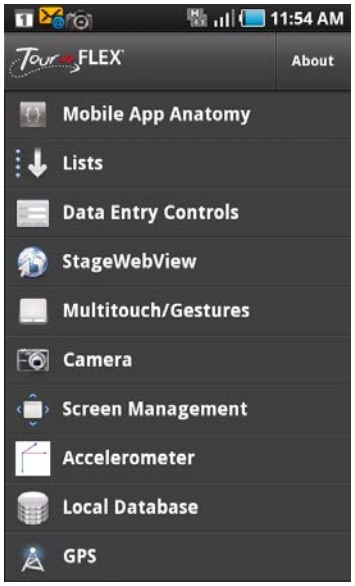
3.2 Prototyp - nächste Schritte zum Release

Unzulänglichkeiten des aktuellen Stands zum Zeitpunkt der MediaNight waren unter anderem damit begründet dass nicht alle Fehler im Build rechtzeitig behoben werden konnten und zudem eine Menge unnötiger Operationen im Hintergrund liefen welche das Gerät deutlich in die Knie zwangen. Den Löwenanteil der Einschränkung machte die Generierung der Suchergebnisse aus, welche zeitweise zum Einfrieren, manchmal auch Abstürzen der Anwendung führte. Dieser Vorgang müsste für eine Release-taugliche Version deutlich entschlackt und optimiert werden.

Diese technischen Probleme zu lösen hätte geschätzt noch weitere 2 Monate Entwicklungszeit mit häufiger Rücksprache mit den Entwicklern des Vorgängerprojekts bedurft.

3.3 Exemplarisches Problem und Lösungsansatz

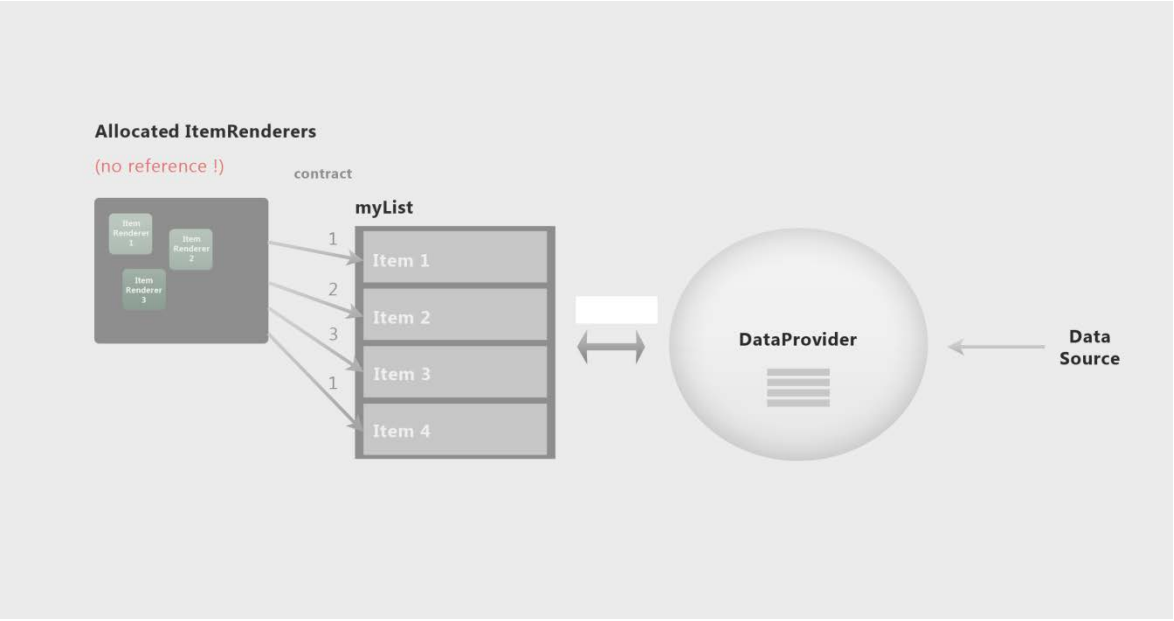
Eine technische Herausforderung war das Anbringen eines Kontextabhängigen Buttons auf den Items der Suchergebnisse.



Beispiel für mobil-optimierte Liste in Flex 4.5

Quelle blogs.adobe.com.

Für das pre-release SDK 4.5 ist nur eine mobil-optimierte Liste implementiert (siehe Bild links). Sie enthält ein optionales Bild und Icon sowie Textüberschrift und Untertitel für das Item. In der Flex-Architektur werden Elemente wie Listen oder Tabellen „**Controls**“ genannt, welche mit einem sogenannten „**Data-Provider**“, verknüpft werden. Dieser wiederum könnte eine statische XML-Struktur als Quelle haben, eine Anbindung zu einem Webservice einkapseln oder eine Vielzahl anderer Datenstrukturen beherbergen. Um die so verknüpften Datensätze nun in der Liste darzustellen, hält dieser einen sogenannten „**contract**“ mit einem **ItemRenderer**, welcher die Darstellung der Datensätze (Items) steuert. Problematisch ist, dass man gewöhnlich keine Referenz auf diesen, dem Item zugeordneten ItemRenderer bekommt und es so keine Möglichkeit gibt an die enthaltenen Elemente heranzukommen um beispielsweise auf ein „Item-gedrückt“ Event zu hören und in diesem Fall einen Button erscheinen zu lassen.



Veranschaulichung des Zusammenhangs ItemRenderer, Container, DataProvider

Es kommt das Problem hinzu dass die ItemRenderer für das Containerelement aus Gründen der Performance des Listenscrollings wiederverwendet werden. Das bedeutet dass ein Renderer-Objekt für mehrere Items zuständig ist. Selbst wenn das gedrückte Listenelement in der Lage ist, eine Anweisung an seinen zugehörigen Renderer weiterzuleiten, so wirkt sich die Änderung auf alle zusätzlich zugewiesenen Items aus.

Prototyplösung:

Nach drücken eines Items erscheint ein Button

Im gegenwärtigen SDK ist zudem kein sichtbarer Mechanismus vorgesehen, nach dem Rendering noch etwas an den Items zu ändern, was für die Implementierung dynamischer Kontrollelemente nötig ist.

Gelöst wurde das Ziel eines kontextuellen Buttons indem die dem *MobileIconItemrenderer* zu Grunde liegende Klasse *MobileItemRenderer,* an Hand eines analogen Beispiels im Internet abgeleitet wurde und statt dem vorgesehenen Icon ein zustandsbehafteter Container mit Buttons eingefügt, der sich bei Aufruf der entsprechenden Methode erweitert.

Um nun die fehlende Referenzierung zu umgehen kann man sich nach einem vorhandenen ListChange Event (die Auswahl der Liste hat sich verändert), eine Referenz auf das gewählte Item besorgen und anschließend den Pfad zur Instanz des ItemRenderers rekonstruieren. Auch diese Notlösug wurde nach langer Recherche einem Forenbeitrag nachempfunden. Ist der ItemRenderer ermittelt wird die dort eingetragene Methode zur Sichtbarmmachung des Buttons aufgerufen.

Zusätzlich musste das Caching der ItemRenderer deaktiviert werden, da sonst der Button allen zugewiesenen Items hinzugefügt wurde.

Diese Lösung ist rückblickend wenig empfehlenswert, da eine Untergrabung der MVC-Struktur stattfindet, durch eine Verschränkung zwischen Präsentation und Modell. Zudem geht auf Grund des abgeschalteten Cachings die Scrolling Performance der Liste deutlich nach unten.

Bessere Lösung

Ein tragfähigerer Ansatz wäre statt einer nachträglichen Referenzierung des Renderers, ein Austausch durch einen anderen Renderer mit Button gewesen. Dieses ist schon eher im Sinne der Flex Architektur und es gibt hierzu Referenzbeispiele. Das Eventhandling hätte komplett über ein Mapping realisiert werden müssen, indem sowohl Liste als auch Button die entsprechende Anbindung an das Mate Framework erhalten hätten.

Ein weiterer Lösungsansatz wäre dieser hier, von Martin Bjeld:

- **Injected ItemRenderer Callbacks**
  http://www.martinbjeld.com/tips-and-tricks/injected-itemrenderer-callbacks/

4 Lessons Learned

Für mich persönlich ein sehr erfolgreiches Projekt, leider innerhalb eines Semesters für nur eine Person viel zu groß. Das jetzige Verständnis der Flex-Technologie wurde allerdings gerade durch diese Auseinandersetzung mit architekturbedingten Schwierigkeiten und häufigen „im Dunkeln tappen“ besonders gefördert. Auch wenn der Prototyp zum Präsentationszeitpunkt nur unter Anleitung nutzbar war, konnte ich insgesamt die Sicherheit gewinnen auch komplexe Migrationsprojekte bearbeiten zu können. Einer erfolgreichen Fertigstellung des Projekts als nutzbare Applikation stünde mit der entsprechenden Zeit nichts im Wege.

Eine systematische Vorgehensweise sowie der rechtzeitige Fokus auf den technischen Aspekt, um eine lauffähige Applikation zu produzieren, sind mir geglückt. Zudem hat das Nutzen von Subversion zur Versionenverwaltung in diesem Projekt eine gewichtige Lücke in meinen bisherigen Kenntnissen der Softwareentwicklung geschlossen.

In einem vergleichbaren Projekt würde ich, zusätzlich zum Projektmanagement, zumindest einen weiteren Projektpartner voraussetzen. Leider waren zu Projektbeginn keine passenden Interessenten verfügbar. Zudem würde ich wenn möglich engeren Kontakt zu erfahrenen Entwicklern suchen, um die Einarbeitungszeit in die Technologie zu beschleunigen.

Allerdings hat mir andererseits der interdisziplinäre Ansatz, nämlich sowohl die Zuständigkeit für Konzeption der Benutzeroberfläche als auch selbstständige Implementierung viel Spaß gemacht und wertvolles Schnittstellenwissen gebildet. Diesen Anspruch an Vielseitigkeit möchte ich auch im kommenden Berufseinstieg weiter verfolgen.

Anhang

A Notizen

Projektablauf, Notizen

Der gesamte Arbeitsaufwand beläuft sich auf ca. 240 Stunden. Einen erheblichen Teil der Projektzeit verwendete ich zudem auf die Einarbeitung in Flex. Dafür nutzte ich unter anderem das Video Tutorial „Flex in a Week“ (weitere Quellen siehe Anhang). Auch das Verständnis der Technologielandschaft im Zusammenhang mit Air und Flex für Android erforderte ein bis zwei Wochen des Sichtens von Material und Artikeln. Für das Flex SDK 4.5 ist gegenwärtig noch keine Fachliteratur verfügbar.

Während der Entwicklung waren folgende Arten von Quellen besonders hilfreich:

- Offizielle Flex Hero Dokumentation auf Adobe Labs
- Stack Overflow coding community
- Adobe Help Forum
- Technik Blogs

Im Folgenden der Projektablauf an Hand meiner Projektnotizen und Tasks.

Basics

- Flash Builder / Eclipse eingerichtet
- Hero SDK eingerichtet
- Samsung KIES installiert (enthält passenden Treiber für Debugging)
- Subclipse Versionsverwaltung konfiguriert
- Oxygen XML Editor installiert

- Zugriff auf Codebase mit SVN erfolgreich
- Kompiliert im Kompatibilitäts-Modus erfolgreich
- Konvertierung als AIR Applikation auf dem Desktop

Proof of Concept

- Original Webanwendung unter www.semsix.com im Browser des Galaxy Tabs testen
    - Fazit: Nicht bedienbar
- Air-Applikation auf Zielauflösung erstellt
- AIR Runtime auf Samsung Galaxy Tab installiert

- Problem: zur Kompilierung als Android-fähige Version reicht der Kompatibilitätsmodus nicht aus

- Hero SDK / Burrito installieren und testen
- APK Datei erzeugen > testen

- Simples Hero Tutorial
- Einarbeitung in Flex Basics (Flex in a week)
- Komplexes Tutorial für Flex 4.5: TwitterTrends
- **Funktionaler UI Prototyp erstellt (Clickdummy)**

- **Mit Prototypen auf Semsix data-model zugreifen**

- Ursprüngliche Codebase in Endgerät-fähiges Clickdummy Projekt transplantiert
- Kompilierungsfehler abarbeiten:
    - Namespaces
    - Nicht-UI-Komponenten in Declarations-Block
    - Namespaces für Script und Style aktualisiert
    - mx.swc eingebunden
    - States umschreiben
    - Quellenpfade der lokales angepasst
    - Fehlende libraries als rsl (textlayout), war in Runtime nicht vorhanden, Linktyp angepasst

- MX.swc nachträglich integrieren
- In Codebase integrierter Clickdummy startet fehlerfrei

Semsix Touch Prototyp

- Mate DI-Framework einarbeiten
- Framework updaten

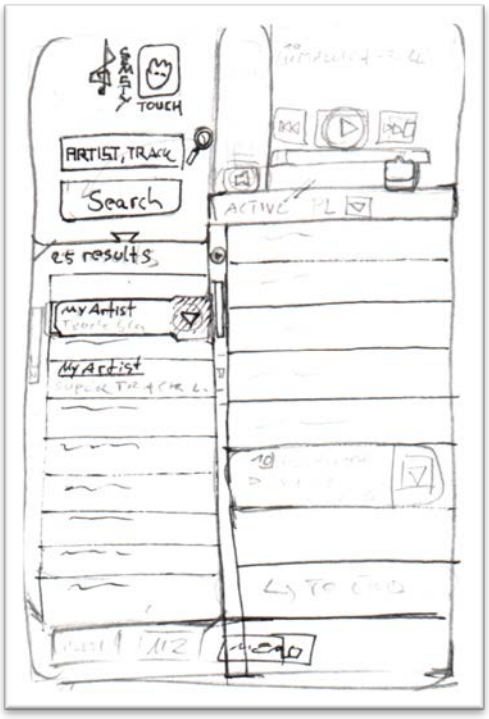
- Accordion-Container zum Laufen bringen
- List controls implementieren

- Schwelle für schärfere Suche integrieren
    - In renderer integriert (unsauber)
- Mehr Komponenten verstecken

- GUI Player Layout analysieren
- Buttons repositionieren
- Button-skinning und Symbole analysieren
- Mock-up fertigstellen
- Assets extrahieren
- Buttons in der Applikation durch zustandsbehaftete Bilder ersetzen.
- Weitere kritische Bugs bearbeiten

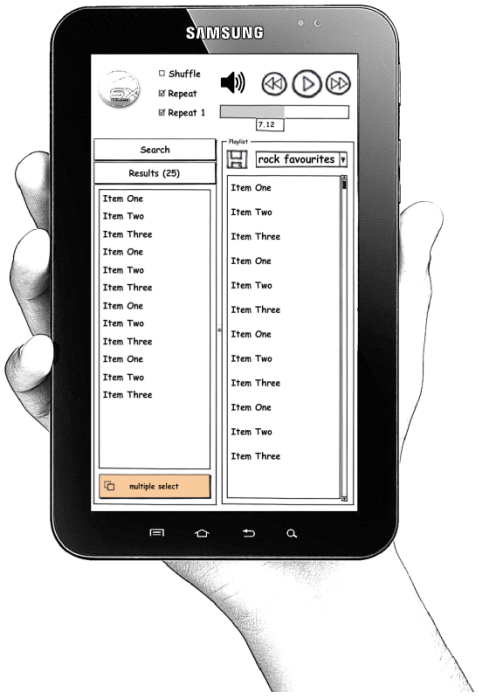
- **Demo build für Medianight fertig**

B Wireframes

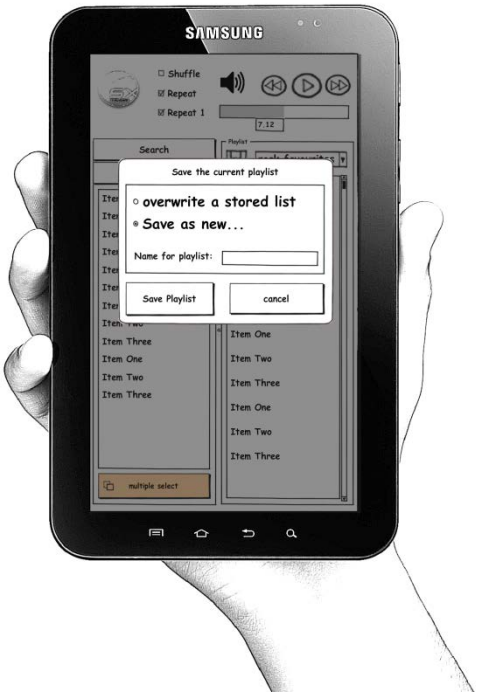


SXT A2

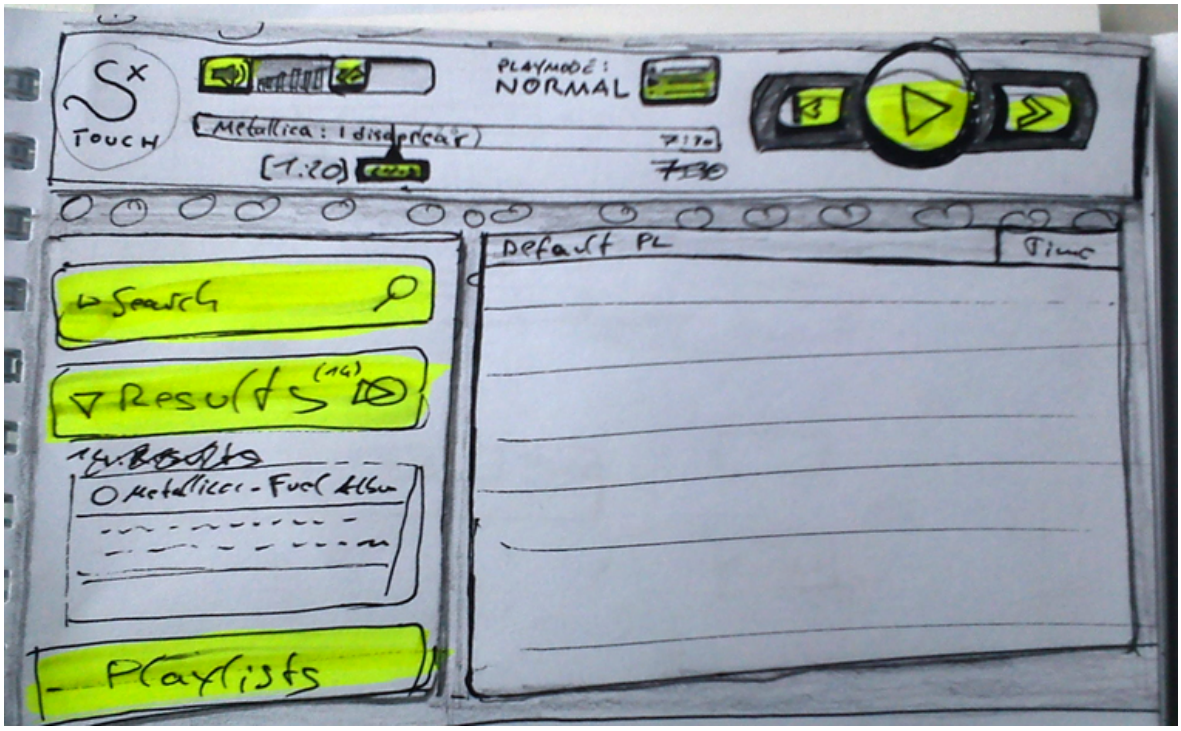
sxt B_02: Akkordeon Search/Results



sxt B_02.1.2: modal "save playlist as" dialoge 2



Wireframe SXT_C-01


TOUCH
PLAYMODE:
NORMAL
Metallica: I disappear
[1:20]
Default PL
Time
Search
Results
Playlists

C Ressourcen

Das Internet bietet eine sehr große Vielfalt offizieller Ressourcen, Technologie-Blogs und Communities zum Thema Flash und Flex. Einen großen Teil der Projektzeit war ich mit der Google-Suche nach passenden Dokumentationen und Tutorials beschäftigt um das Verständnis für das aktuelle Problem zu gewinnen.

Techblogs und Community-Foren

- Get started with AIR on Android before you get an Android device! | Greg Wilson's Ramblings
- Package Assistant Pro | Serge Jespers
- Video tutorials | Serge Jespers
- Flex 4.5 Mobile Development Post Burrito | The World In A State of Flex
- Create your 1st Flex based AIR on Android Application | InsideRIA
- Crossdomain.xml-Problem | Flexforum.de
- Developing a mobile application with Flex 4.5, AIR 2.5, Flash Builder Burrito, WCF and the Entity Framework. | Alex van Beek - blog community

- Add Swipe Support
  http://www.flexpert.be/2010/12/adding-swipe-gestures-to-you-mobile-flex-application/

- Spacer Element
  http://workflowflash.com/11616/srect-is-the-new-mxspacer.php

- Flex 3 skinning
  http://www.gotoandlearn.com/play.php?id=101

- Flex DataGrid with Combobox ItemRenderer
  http://stackoverflow.com/questions/1408729/flex-datagrid-with-combobox-itemrenderer

- MobileItemRenderer
  Creating a fancy Spark List control item renderer in Flex 4

- Avoid itemrenderer caching
  http://stackoverflow.com/questions/3692010/avoid-itemrenders-caching-in-a-spark-list-in-flex-4

- Top hiccups when migrating from Flex 3 to Flex 4
  http://butterfliesandbugs.wordpress.com/2009/05/05/top-hiccups-when-migrating-from-flex-3-to-flex-4/

Adobe Ressourcen

Adobe Developer Connection

- Introducing Adobe Flex SDK "Hero" | Adobe Developer Connection
- Cross-domain policy file specification | Adobe Developer Connection
- Flex for mobile "Hero, Burrito on Adobe"
  http://www.adobe.com/devnet/flex/articles/mobile_development_hero_burrito.html

Adobe Cookbooks

- Adding gesture support to your Flex Hero mobile applications
- Change ItemRenderer dynamically in a list

Adobe TV

- AdobeTV
- MAX 2010 Develop - Essential Eclipse Plug-ins and Tools for Flash Builder Developers | Adobe TV
- ADC Presents - Debug Adobe AIR for Android Applications | Adobe TV
- Flex in a Week video training

Adobe Labs

- Adobe Labs - Adobe Flex SDK "Hero"

- Flex and Hero SDK FAQ
  http://labs.adobe.com/technologies/flex/mobile/faq.html

forums.adobe.com

- Adobe Forums: Flash, Android & Emulator set up

help.adobe.com

- Adobe Flex Hero * Using Adobe Flex Hero
- Adobe Flash Builder Burrito * Using Adobe Flash Builder Burrito
- Adobe Flex Hero * Connecting Google Android Devices
- Adobe AIR * Setting application properties

opensource.adobe.com

- Hero - Flex SDK - Adobe Open Source
- Mobile Application - Flex SDK - Adobe Open Source

Sonstige Ressourcen

- Mate Flex Framework — A tag-based event-driven Flex framework

- MATE Dokumentation
  http://mate.asfusion.com/page/documentation/tags/injectors

- Samsung Developer Forum
  http://innovator.samsungmobile.com/galaxyTab.do

- **BLUEPRINT** Eclipse Plugin
  http://labs.adobe.com/wiki/index.php/Blueprint:Installation_Instructions#Installing_Blueprint_-_Step_1