Vergleich aktueller Second Screen SDKs

Florian Bauer <it221507@fhstp.ac.at>

13.02.2023

***Abstract* - Seit dem Second-Screen-Trend gegen 2010 haben sich zahlreiche Smart-TV-Lösungen und deren Möglichkeit des Verbindungsaufbaus zwischen Smart-TV und der mobilen Anwendung etabliert. Dieser Artikel soll einen komprimierten Leitfaden, in der Verwendung von SDKs, Software Development Kits, für den Einstieg in der Herstellung einer Verbindung zwischen First- und Second-Screen-Geräten liefern. Dies umfasst einen groben Überblick über die einzelnen notwendigen technischen Schritte, die jeweils bei der Sender- und Empfängeranwendung umgesetzt werden müssen. Außerdem werden die jeweiligen möglichen Funktionen jeder einzelnen Umgebung grob beschrieben.**

I. Einleitung

Der Second-Screen ist in dem letzten Jahrzehnt zur Gewohnheit geworden, was mit der Verbreitung von tragbaren Bildschirmen, vor allem Tablets und Smartphones, zusammenhängt. Second Screening ist ein Prozess, bei dem Personen, die fernsehen, ein zusätzliches elektronisches Gerät oder einen "Bildschirm" benutzen, um auf Informationen aus dem Internet oder soziale Netzwerke zuzugreifen. Diesbezüglich bieten heutzutage viele mobile Anwendungen die Möglichkeit die mobilen Inhalte auf dem großen Smart-TV (First-Screen) zu casten [1].

Um die aktuelle Situation bei der Entwicklung von Second-Screen-Anwendungen korrekt zu erfassen, wurden einige marktrelevanten Smart-TV-Plattformen analysiert. Ziel dieser Analyse ist es, einen umfassenden Überblick über die derzeit verfügbaren Lösungen zur Herstellung einer Verbindung zwischen First- und Second-Screen-Geräten zu geben und deren Eigenschaften zu erfassen. Derzeit gibt es eine Reihe von Lösungen, um eine Verbindung zwischen einem ersten und einem zweiten Bildschirm herzustellen. Diese Lösungen werden in der Regel von den Herstellern der First-Screen-Geräten bereitgestellt und sind für die am weitesten verbreiteten mobilen Plattformen verfügbar, z. B. *Android* und *iOS*.

Die meisten SDKs unterscheiden sich in Bezug auf Funktionalität und Programmierparadigmen. Ein essenzieller Punkt ist jedoch, dass alle SDKs, die von den Herstellern von First-Screen-Geräten angeboten werden, nur eine sehr begrenzte Anzahl von First-Screen-Geräten unterstützen, in der Regel also nur ihre eigene Plattform. Das bedeutet, dass für jede unterstützte First Screen Plattform ein eigenes SDK entwickelt werden muss, was den Entwicklungsaufwand für Second-Screen-Anwendungen erhöht [2].

Für eine bessere Übersicht stellt Abbildung 1 den Einsatz der unterschiedlichen SDKs zwischen den First-Screen- und den Second-Screen-Geräten dar.

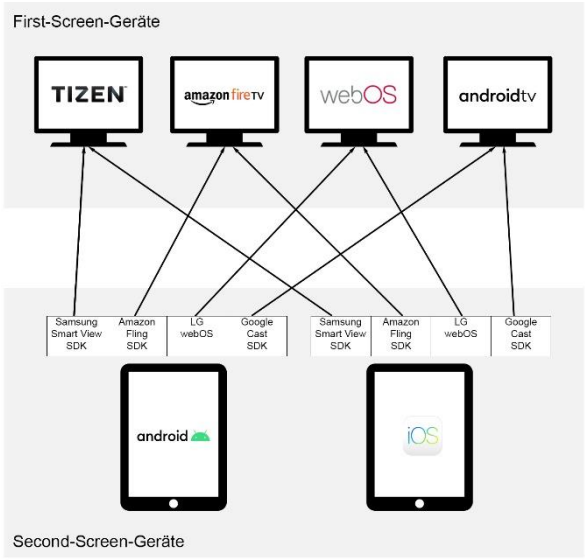

Abbildung 1: Übersicht der SDK-Integrationen in Anlehnung an [2].

In diesem Kontext werden die SDKs von *Google*, *Amazon*, *LG* und *Samsung* analysiert. Außerdem wird das DIAL-Protokoll vorgestellt, welches Second-Screen Geräte grundlegend erlaubt, First- Screen-Geräte zu finden und zu starten.

II. Google Cast SDK

Das *Google Cast SDK* ermöglicht eine Erweiterung von *Android-*, *iOS-* und *Chrome-Anwendungen*, um Video- und Audioinhalte auf unterstützte First-Screen-Plattformen (Cast-Anwendungen) zu übertragen und Funktionen auf mobilen Geräten zu steuern. Es ist wichtig zu beachten, dass obwohl das SDK zwischen Sender- und Empfängeranwendungen unterscheidet, die First-Screen-Anwendung nicht ohne ihr Gegenstück, die Second-Screen-Anwendung funktionieren kann, welche als Fernsteuerung fungiert [3].

Senderanwendungen bieten drei grundlegende Funktionen: Auffinden, Starten und Verbinden von Empfängeranwendungen im lokalen Netzwerk durch die Einrichtung eines Kommunikationskanals [4].

Die Empfängeranwendungen sind in HTML, JavaScript und CSS implementiert und werden bei jedem Start von einer gehosteten URL abgerufen. Da diese Anwendungen nicht auf dem Gerät installiert sind, können keine Daten gespeichert werden, was die Funktionalität dieser Cast-

Anwendungen einschränkt. Empfänger-Apps müssen in der *Google* Cast SDK Developer Console registriert werden, wo eine App-ID für den Start mit einer Sender-App generiert wird. Jeder *Android*-Fernseher und andere unterstützte Geräte, wie z. B. *Amazon Fire TVs*, verfügen über einen integrierten Cast-Empfänger, der Cast-Anwendungen unterstützt [4].

Es ist wichtig, den Unterschied zwischen diesen Cast- und *Android TV-Anwendungen* zu beachten, die den mobilen Android-Anwendungen sehr ähnlich sind. Das *Google Cast SDK* unterstützt jedoch nur Cast-Anwendungen, weshalb es derzeit keine bestehende Lösung zur Unterstützung von *Android TV-Anwendungen* als First-Screen-Plattform gibt [3].

III. Amazon Fling SDK

Das *Amazon Fling SDK* unterstütz *Android-, iOS- und Fire OS*-Anwendungen, Medien- und Webinhalte über das lokale Netzwerk zu erkennen, zu verbinden und auf *Amazon Fire TV*-Geräte zu übertragen. Das *Fling SDK* ist auch *mit Google Cast*-Anwendungen kompatibel, da *Fire TV*-Anwendungen auf *Android TV* basieren [5].

Um die Funktionalitäten des *Fling SDKs* nutzen zu können, muss die mobile Anwendung zwei Bibliotheken als Module integrieren (Fling & Whisperplay). Dies ermöglicht der mobilen Anwendung, *Fire TVs* im lokalen Netzwerk zu erkennen und die entsprechende Anwendung für das First-Screen-Gerät zu starten bzw. wenn notwendig zu installieren. Sobald das First- und der Second-Screen-Gerät verbunden sind, kann das mobile Gerät Medieninhalte wie Video, Audio und Bilder übertragen. Darüber hinaus ist mit dem *Fling SDK* ein Austausch von Metainformationen und die Steuerung der verbundenen Inhalte möglich, jedoch besteht keine anpassbare bidirektionale Kommunikation [2].

Fire-TV-Empfängeranwendungen müssen die gleichen Bibliotheken enthalten wie ihre mobilen Gegenstücke (Fling & Whisperplay). Darüber hinaus muss die TV-Anwendung eine eindeutige Dienstkennung (SID) bereitstellen, die in einer separaten Datei (whipserplay.xml) angegeben wird, um sie aus der Ferne starten zu können [5].

IV. LG Connect SDK

Das SDK von *LG* ist ein mobiles SDK, das mehrere TV-Plattformen unterstützt und Funktionen zum Auffinden, Starten und Kommunizieren für den First-Screen und Second-Screen bietet. Zusätzlich zu diesen grundlegenden Funktionen verfügen *LG webOS*-Geräte über eine Reihe zusätzlicher Funktionen, wie das Abrufen einer Liste installierter Anwendungen, die Ferninstallation von Anwendungen, die Anzeige von Pop-up-Benachrichtigung und Tastatureingaben mit dem zweiten Bildschirm. Das *Connect SDK* ist für *iOS, Android* und *Cordova* verfügbar [2].

Die Einrichtung des *Connect SDK* für das Absendergerät (Second-Screen) kann automatisch erfolgen, indem eine Abhängigkeit im Projekt der mobilen Anwendung aktiviert wird oder durch manuelles Einbinden der Quelldateien, was mit einer gewissen Komplexität verbunden ist.

Darüber hinaus gibt es mehrere Versionen des SDK mit einer unterschiedlichen Anzahl von unterstützten TV-Plattformen, was die Komplexität des Integrationsprozess erhöht [6]. Der Start von *webOS*-Anwendungen erfordert entweder eine bestimmte *webOS*-Anwendungs-ID oder die DIAL-ID der Anwendung. *WebOS*-Empfängeranwendungen verwenden die Webtechnologien HTML, JavaScript und CSS und benötigen eine online registrierte Anwendungs-ID, um von einem mobilen Gerät aus gestartet zu werden. Empfängeranwendungen haben das *Connect SDK* über zwei JavaScript-Skripte integriert, die die Funktionalitäten der Medienwiedergabe und -steuerung sowie die bidirektionale Kommunikation mit der Senderanwendung ermöglichen [6].

Das *Connect SDK* ist ein guter Ansatz, um die Entwicklung von Second-Screen-Anwendungen zu erleichtern, dennoch hat die aktuelle Version gravierende Probleme, vor allem in Bezug auf die Aktualität. Die letzte Version zum Analysezeitpunkt wurde im September 2015 aktualisiert, was zu einer Inkompatibilität mit neueren *Android*-Versionen führt, was zur Folge hat, dass das *Connect SDK* ein ungeeignetes SDK ist [2].

V. Samsung Smart View SDK

Das *Samsung Smart View SDK* ist in der Lage, *Android-*, *iOS-* und JavaScript-Anwendungen zu erweitern, um kompatible Empfangsgeräte zu erkennen und zu starten und mit ihnen zu kommunizieren, nachdem eine Verbindung hergestellt wurde. Es unterstützt auch zusätzliche Funktionen wie die Ferninstallation der Anwendung für das First-Screen-Gerät, das Aufwecken des First-Screen-Geräts aus dem Standby mit einem mobilen Gerät (Wake on WirelessLAN) oder die Geräteerkennung über Bluetooth [2].

Der gesamte Verbindungsprozess ist in drei Phasen unterteilt, die auch die Hauptfunktionalitäten darstellen: Discover, Launch und Communicate [7].

Die Anwendungen des Senders (Second-Screen) müssen das SDK als Modulbibliothek integrieren, um auf dessen Funktionen zugreifen zu können. Um die TV-Anwendung zu starten oder zu installieren, muss die mobile Anwendung die AppID der TV-Anwendung kennen und beide Anwendungsteile müssen dieselbe channelId zur Kommunikation verwenden, die frei konfigurierbar ist. Die AppID kann während der Entwicklung frei gewählt werden, muss aber später auf eine zugewiesene ID aus dem Samsung App Store geändert werden [7].

Die Empfängeranwendungen (First-Screen) sind Webanwendungen, die aus HTML, JavaScript und CSS bestehen. Diese müssen mit der *Tizen TV IDE* entwickelt werden, da die Installation und das Debugging nur mit

dieser Entwicklungsumgebung möglich ist. Die Kommunikation erfordert, dass die Sender- und Empfängeranwendungen dieselben Kanal- und Ereignis-IDs verwenden, was eine zuverlässige bidirektionale Kommunikation auf der Grundlage von WebSockets ermöglicht [7].

Das *Samsung Smart View SDK* ist das am besten ausgearbeitete SDK, das diese Analyse ergab. Es bietet nicht nur eine Vielzahl zuverlässiger Funktionalitäten, sondern auch eine gute Dokumentation mit Projektbeispielen und außerdem wird dieses regelmäßig gewartet.

VI. DIAL-Protokoll

Das DIAL-Protokoll ermöglicht Second-Screen-Geräte, Anwendungen auf dem First-Screen-Gerät in einem lokalen Netzwerk zu entdecken und zu starten. Das von *Netflix* und *YouTube* entwickelte Protokoll verdankt seinen Namen dieser Funktionalität (Discover & Launch) und beinhaltet keine Lösung für die Kommunikation zwischen dem ersten und zweiten Gerät. Eine Vielzahl von First-Screen-Geräten, wie *Android TV* oder *webOS*, unterstützen dieses Protokoll. Aktuelle SDKs wie *LG Connect* basieren auf dieser Technologie. Im Gegensatz zu den SDKs der Hersteller muss die Funktionalität des DIAL-Protokolls von Entwicklern mit Hilfe von bereitgestellten Spezifikationen implementiert werden [8].

Der Prozess besteht aus zwei grundlegenden Komponenten: DIAL Service Discovery (entdecken) und DIAL Rest Service (starten). Der erste Schritt ermöglicht Second-Screen-Geräten, kompatible First-Screen-Geräte im lokalen Netzwerk mit einem M-Search SSDP-Request (Simple Service Discovery Protocol), einem auf UPnP basierenden Protokoll, zu entdecken. Die zweite Komponente wird genutzt, um gefundene Geräte über HTTP abzufragen und zu starten [9].

Der angegebene Anwendungsname der First-Screen-Anwendung ist die einzige externe Ressource, die zum Starten von Anwendungen mit DIAL benötigt wird, alles andere kann direkt implementiert werden. Beim Start der First-Screen-Anwendung ist es auch möglich, Parameter vom mobilen Gerät zu übergeben, die nicht im DIAL-Protokoll enthalten sind [9].

Empfängeranwendungen müssen einen Anwendungsnamen angeben, unter dem sie vom mobilen Teil angesprochen werden, der in ein öffentliches Register eingetragen wird und somit für alle Entwickler zugänglich ist [9].

Laut der Entwicklergemeinschaft wurden jedoch Probleme bei der Registrierung des Anwendungsnamen gemeldet. Selbst registrierte Anwendungen konnten mit dem DIAL Protokoll somit nicht gestartet werden. Die einzigen Ausnahmen von diesem Verhalten sind *Fire TV-Anwendungen*, bei denen der Anwendungsname nicht online registriert werden muss, sondern lokal angegeben wird. Die Problemursache scheint bei der fehlenden Synchronisation der Listen der bekannten Anwendungen zwischen dem DIAL Server und dem DIAL Register zu liegen [8].

VII. Fazit

Im Rahmen der Recherche zum vorliegenden Artikel konnte eine große Bandbreite bestehender Lösungen zwischen First- und Second-Screen-Anwendungen dargestellt werden, die jeweils mit Nachteilen verbunden sind, z.B. die fehlende Kommunikation zwischen den Plattformen. Da auf der Smart-TV-Seite mehrere Plattformen zur Verfügung stehen, muss diese Verbindung von der Second-Screen-Anwendung zu den verschiedenen verfügbaren First-Screen-Plattformen mehrfach hergestellt werden, was einen erhöhten Entwicklungsaufwand verursacht. Da aber auch auf der mobilen Seite eine zweite wichtige Plattform zur Verfügung steht, müssen die Verbindungen zu den verschiedenen First-Screen-Plattformen auch von dieser zweiten mobilen Plattform aus aufgebaut werden, was die Anzahl der redundanten Entwicklungsschritte verdoppelt. Zum besseren Verständnis werden die SDKs und das DIAL-Protokoll bezüglich ihrer Funktionalität in Abbildung 2 gegenüberstellt.

	Samsung Smart View SDK	Amazon Fling SDK	LG Connect SDK	Google Cast SDK	DIAL Protocol
Discover	Ja	Ja	Ja	Ja	Ja
Launch	Ja	Ja	Ja	Ja	Ja
Communicate	Ja	Nein	Ja	Ja	Nein
Fully Functional	Ja	Nein	Ja	Ja	Nein
Single Target Platform	Ja	Ja	Ja	Ja	Nein
Install	Ja	Ja	Ja	Nein	Nein
WoW	Ja	Nein	Nein	Nein	Nein
List installed Apps	Ja	Nein	Ja	Nein	Nein
Mouse Controls	Nein	Nein	Ja	Nein	Nein
Power off device	Nein	Nein	Ja	Nein	Nein

Abbildung 2: SDK-Funktionsübersicht in Anlehnung an [2].

Zukünftige Artikel könnten sich mit der Konzeption von einer möglichen universellen SDK befassen, welche von allen marktrelevanten First- und Second-Screen-Plattformen genutzt wird. Beim Verfassen dieses Artikels gab es noch keine gemeinsame Lösung am Markt. Hierfür muss jedoch jede einzelne Technologie und SDK in die Tiefe getestet und analysiert werden, im vorliegenden Artikel wurde nur ein grober technischer Überblick geschaffen.

VIII. LITERATUR

[1] H. Gil de Zúñiga, V. Garcia-Perdomo, und S. C. McGregor, „What Is Second Screening? Exploring Motivations of Second Screen Use and Its Effect on Online Political Participation“, Journal of Communication, Bd. 65, Nr. 5, S. 793–815, Okt. 2015. Zugegriffen: 9. November 2022. [Online]. Verfügbar unter: doi: 10.1111/jcom.12174.

[2] V. Lohmüller, „Second Screen Applications: A Multi-Platform Software Development Kit and Optimization of Human-Computer Interaction in Distributed Systems“, phd, 2019. Zugegriffen: 9. November 2022. [Online]. Verfügbar unter: https://epub.uni-regensburg.de/41136/.

[3] N. Lavrell, Integrating the Google Cast Technology in a Second-screen Solution. 2014, Zugegriffen: 7. November 2022. [Online]. Verfügbar unter: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-107933.

[4] Google Developer Dokumentation, „Overview | Cast | Google Developers“. Zugegriffen: 7. November 2022. [Online]. Verfügbar unter: https://developers.google.com/cast/docs/overview.

[5] Amazon Developer Dokumentation, „Understanding the Amazon Fling Service | Fling SDK“. Zugegriffen 7. November 2022. [Online]. Verfügbar unter: https://developer.amazon.com/de/docs/fling/understanding-the-amazon-fling-service.html.

[6] LG Developer Dokumentation, „Connect SDK Overview — connectSDK documentation“. Zugegriffen: 7. November 2022. [Online]. Verfügbar unter: https://connectsdk.com/en/latest/discover/overview.html.

[7] Samsung Developer Dokumentation, „Smart TV“, Samsung Developers. Zugegriffen: 7. November 2022. [Online]. Verfügbar unter: https://developer.samsung.com/SmartTV/design/design/smart-view-sdk.html.

[8] E. Bergwik, Using the DIAL Protocol for Zero Configuration Connectivity in Cross-Platform Messaging. 2014. Zugegriffen: 7. November 2022. [Online]. Verfügbar unter: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-107625.

[9] DIAL-Protokoll Spezifikation, „DIAL-2ndScreenProtocol-2.2.1.pdf“. Zugegriffen 6. November 2022. [Online]. Verfügbar unter: https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGlhbC1tdWx0aXNjcmVlbi5vcmd8ZGlhbHxneDo1MWVmNzNhZDUyYTI0YTkz.