DIN-Connect Projektskizze

Ideenwettbewerb 2020

Projekttitel

Y・Core

Software-Framework zur plattformübergreifenden nativen Anwendungsentwicklung.

Kontaktdaten des Ideengebers

Sascha Peilicke
Am Gutspark 4
10367 Berlin

sascha@peilicke.de
+49 174 7702469

Potentielle Projektpartner

Bestätigt

- **Sascha Peilicke** *Gründer und CEO,* LunchUp
- **Pavlo Dyban** *Tech Lead Computer Vision for Autonomous Driving, Siemens AG*
- **Christian Nietner** *Gründer und CTO, Avanetix UG*
- **Carsten Mohs** *Gründer und CEO, timum GmbH*

Angefragt

- **Prof. Dr.-Ing. Jana Dittmann** *Arbeitsgruppe Multimedia und Security, Otto-von-Guericke Universität*
- **Antonio Casero Palmero** *Lead Software Engineer, HERE Technologies GmbH*
- **Robert Manea** *Application Architect, it-economics GmbH*

Darstellung der Ausgangslage

Software wird heutzutage meist für mehrere Zielplattformen entwickelt und dort vertrieben. Ob auf dem Desktop in Form von Windows, macOS oder Linux, im Web oder auf Android oder iOS Smartphones, jede Plattform hat ihre Besonderheiten die eine einheitliche Produktentwicklung kosten- und wissensintensiv gestalten. Dazu kommt, dass jede Plattform unterschiedliche Entwicklungswerkzeuge, Programmiersprachen und Bibliotheken mit sich bringt (gerne auch mehrere konkurrierende Lösungen parallel).

Viele digitale Geschäftsmodelle finden hauptsächlich im Web statt. Jedoch ist in den letzten Jahren ein starker Zuwachs im Bereich Mobil-Anwendungen entstanden und wird insbesondere durch jüngere Nutzer nachgefragt oder auch schlicht vorausgesetzt. Um Kosten und Mehraufwände im Rahmen zu halten, suchen Firmen nach Ansätzen, die relevanten Plattformen Mobile und Web mit einer Software-Codebasis zu bedienen. Soll zum Beispiel eine erfolgreiche Web-Anwendung für Mobilplattformen angepasst werden, existieren mehrere Möglichkeiten:

1. Die (Neu-)Entwicklung von nativen Apps
2. Die Nutzung hybrider Web-Technologien (für die existierende Web-Lösung)

Die Varianten werden im Folgenden hinsichtlich ihrer Vor- und Nachteile besprochen sowie mit dem zur Normierung vorgeschlagenen Ansatz verglichen.

Native Entwicklung

Native Apps werden speziell für ein Betriebssystem, wie zum Beispiel iOS, entwickelt und stehen ausschließlich auf entsprechenden Endgeräten (z.B. iPhone oder iPad) zur Verfügung. Die Daten werden hauptsächlich auf dem Gerät gespeichert. Installation und Updates erfolgen über die Vertriebsplattform (App Store) des jeweiligen Anbieters. Die native App-Entwicklung für Android und iOS wird von Apple und Google offensiv beworben und soll sowohl die Nutzer als auch die Entwickler stärker an die jeweiligen Plattformen binden.

Vorteile

Der größte Vorteil dieses Ansatzes ist die optimale Integration in die Plattform und die Möglichkeit, alle Features auszureizen. Neueste Trends der Hersteller (Dunkle Designs, Notches, Smarte Assistenten) stehen hier als erstes zur Verfügung und werden durch Nutzer stark nachgefragt, beziehungsweise in den Bewertungen der App Stores honoriert. Die Veröffentlichung für unterschiedliche Formfaktoren (Smartwatches, Smart TV, Auto) wird ebenfalls möglich. Für viele Angebote relevant ist, dass die Plattformanbieter gelungenes Design entsprechend der Vorgaben oft mit besserer Platzierung des eigenen Angebotes „belohnen". Native Anwendung lassen sich effektiv monetarisieren. Es besteht die Möglichkeit die App im Store kostenpflichtig oder frei anzubieten sowie durch In-App-Käufe Einnahmen zu generieren. Die Sicherheit wird maßgeblich durch die Plattform garantiert, setzt jedoch regelmäßige Updates beim Nutzer voraus.

Nachteile

Beide Plattformen dominieren den Mobilmarkt, sind jedoch alles andere als offene Systeme. Die notwendigen Entwickler-Tools und Bibliotheken werden von den jeweiligen Herstellern bereitgestellt und fortentwickelt. Insbesondere für Apple's iOS (und macOS) sind regelmäßige, kostenintensive Hardware-Investitionen und Anpassungen an neue Betriebssystem-Versionen notwendig, um Anwendungen weiterhin veröffentlichen „zu dürfen". Obwohl Android und iOS sich in Optik und Bedienung angenähert haben, ist die technische Umsetzung sehr unterschiedlich. Das macht spezialisiertes Fachwissen und

damit hohen Personaleinsatz notwendig. In vielen Firmen führt dies dazu, das es mehrere Entwicklungs-Teams für die jeweiligen Bereiche gibt (Mobile, Web und Backend, oft noch stärker ausdifferenziert). Gute Kommunikation und regelmäßige Abstimmung sind Voraussetzung für eine effektive Produktentwicklung. Die Wartung und Pflege mehrerer Implementierung lässt sich in der Praxis jedoch nur schwer abstimmen und garantieren. Unterschiedliche Bedienungskonzepte und Designansätze der Plattformen führen zu großen Unterschieden in den Implementierungen. Jede Plattform hat ihre eigene Programmiersprache und damit einen eigenen Satz an Bibliotheken. In der Praxis ist beispielsweise iOS-Anwendungscode gar nicht oder nur unter hohem Aufwand für macOS oder Android wiederverwendbar. Diese Komplexität erhöht den notwendigen Testaufwand enorm. Zu bedenken bleibt auch die zwangsläufig entstehende starke Herstellerbindung (Vendor lock-in).

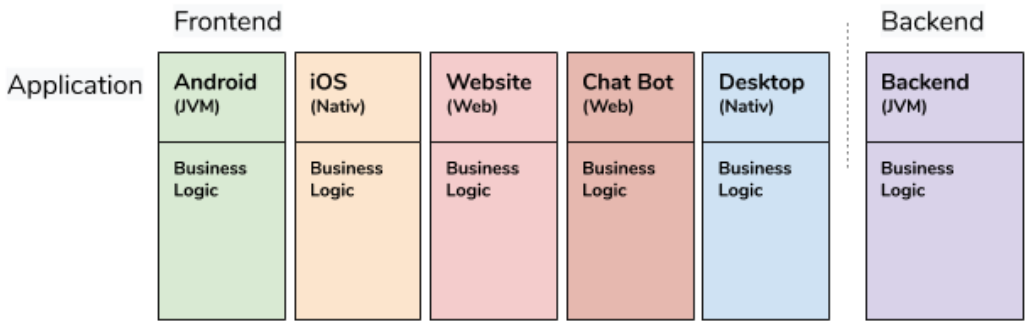

Abb. 1: Native Multiplattform-Entwicklung

Fazit

Der betrachtete Ansatz ist auf Grund der genannten Nachteile sicherlich der aufwändigste und bindet am meisten Kapital. Dies trifft umso mehr zu, je mehr Plattformen für das eigene Produkt unterstützt werden müssen (z.B. auch Windows, Linux oder macOS). Allerdings ist er durchaus der empfehlenswerteste, wenn die genannten Vorteile für das Produkt wesentlich sind und die entsprechend notwendigen Ressourcen vorhanden sind.

Hybride Web-Technologien

Ein Web-Anwendung ist im Prinzip eine speziell entwickelte HTML5-Webseite, die das Endgerät des Benutzers erkennt und ihre Inhalte dafür optimiert darstellt. Sie kann auf beliebigen Webservern gehostet werden und dort jederzeit aktualisiert werden. Dadurch liegen die Daten beim Hersteller, lokale Speicherung im Browser ist eingeschränkt möglich. Die Monetarisierung erfolgt über Werbung oder Bezahlmodelle die selbst integriert werden müssen.

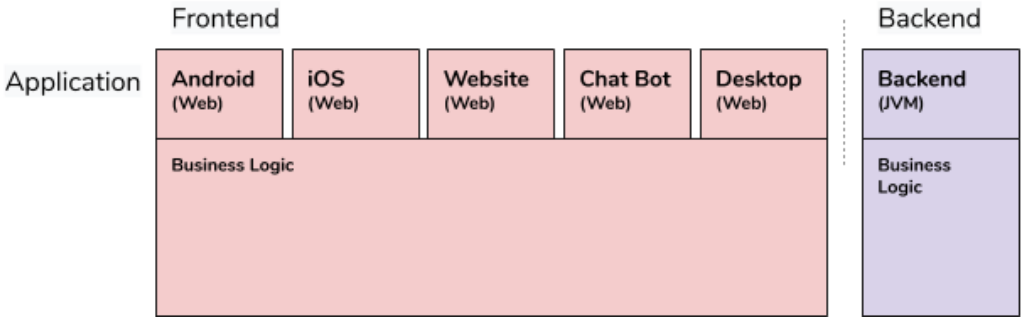

Abb. 2: Hybride Web-Entwicklung

Vorteile

Der wesentliche Vorteil ist die Nutzung einer einzigen Sprache (meist JavaScript) und deren Bibliotheken. Bereits vorhandenes Wissen zur Webentwicklung kann leicht aufgegriffen werden und bestehende Webangebote können ausgebaut werden. Spezialwissen für die Zielplattformen ist nur in geringem Umfang notwendig und beschränkt sich auf die finale Kompilierung des Quellcodes und die Veröffentlichungen in den App Stores. Es gibt sehr viele Frameworks (z.B. React Native, Flutter, NativeScript) die bereits viele Module für wiederkehrende Aufgaben mitbringen. Ein einheitliches Design lässt einfach umsetzen.

Nachteile

Eher eine Technologie für den Frontend-Bereich. Backends werden insbesondere im Firmenumfeld eher mit Java (JVM) umgesetzt. Die Anwendung läuft auf dem Endgerät zwingend in einer Browser-Umgebung. Die Anwendungen wirken daher oft träge und müssen beim Start viele Bibliotheken vom Webserver laden. Es stehen nicht alle Möglichkeiten der Plattform offen. Das führt oft zu vom Nutzer als Fremdkörper wahrgenommenen Verhalten. Meist wird eine (stabile) Internetverbindung zur Funktion vorausgesetzt. Offline-Fähigkeiten erfordern erheblichen Entwicklungsaufwand. Nutzer bewerten daher „eingerahmte Webseiten" generell schlechter (unabhängig vom jeweiligen Produkt). Der Ansatz One-size-fits-all führt zu sehr großem Umfang der Codebasis und erfordert oft spezielle Anpassungen für unterschiedliche Geräteklassen. Geräte-Sensoren oder auch der Terminkalender lassen sich nicht direkt ansprechen und erfordern Umwege über Cloud-APIs (mit den entsprechenden Implikationen bei Sicherheit und Datenschutz) . Der Testaufwand ist ebenfalls erheblich, da jede Änderung an der gemeinsamen Codebasis zwingend auf allen Geräten funktionieren muss. Da die Anwendung auf eigenen Servern gehostet wird kann dort je nach Nutzung erheblicher Rechenaufwand entstehen. Die operativen Kosten sind höher als bei nativen Anwendungen die primär auf den Endgeräten ausgeführt werden. Auch sind die Sicherheitsanforderungen erheblich anspruchsvoller umzusetzen, da die meisten Einfallstore im Bereich Web-Technologien bestehen (z.B. Code-Injection, Cross-Site-Scripting, SQL Injection, usw.).

Fazit

Hybride Webanwendungen erlauben mit reduziertem Aufwand viele Plattformen abzudecken. Oft vorhandenes Wissen kann weiter genutzt werden. Allerdings ist die

Innovationsfreude im Web-Bereich sehr stark und Lösungen kommen und gehen im Quartals-Takt. Viele Frameworks sind komplex und eine Anpassung an die eigenen Bedürfnisse bleibt aufwendig. Vorteilhaft ist die größere Kontrolle da Webanwendungen selbst gehostet werden. Demgegenüber stehen hohe Anforderungen im Bereich IT-Security. Daher grundsätzlich eher geeignet die Reichweite bestehender Webanwendungen zu erweitern.

Existierende Standards

Für den Bereich Software Engineering relevante Normenausschüsse:

- DIN NA 043-01-07 AA Software und System-Engineering
- IEEE Software & Systems Engineering Standards Committee (S2ESC)
- ISO im Bereich Information Technology der JTC1 / SC7 (Systems and Software Engineering)

Nutzen und Ziele des Y · Core

Aus den vorhergehenden Betrachtungen ist es also sinnvoll, möglichst viele Teile der Geschäftslogik eines Software-Produktes wiederzuverwenden, jedoch auf native Benutzeroberflächen zu setzen. Die Geschäftslogik wird in einer Bibliothek ausgelagert und damit nur einmal implementiert. Dies reduziert den Testaufwand erheblich und verbessert die Wiederverwendbarkeit. Die Benutzeroberflächen werden spezifisch für die Zielplattform umgesetzt, bleiben individuell und sind dadurch nicht Teil der Norm. Die Geschäftslogik kann In-House entwickelt werden, während die Erschließung neuer Plattformen über externe Partner möglich wird, ohne diesen den kompletten Quellcode zugänglich zu machen. Die Unabhängigkeit von den Plattformanbietern bleibt ebenfalls gewahrt. Ein Software-Development-Kit (SDK) gibt die Struktur vor und bringt grundlegende Funktionalität mit (Anforderungen als Teil des Standards).

Aus der praktischen Erfahrung ergibt sich, dass sich der Grad der Gemeinsamkeiten zwischen den relevanten Plattformen unterscheidet. Ein typisches Backend wird auf mehreren Knoten, also Servern, in der Cloud oder einem Rechenzentrum ausgeführt. Es existiert keine direkte Benutzeroberfläche, sondern maschinenlesbare Schnittstellen (APIs), die viele Klienten gleichzeitig bedienen können. Frontends hingegen, also Applikationen, mit denen Benutzer interagieren, unterscheiden sich davon dadurch, dass die Ausführung nur lokal auf einem Gerät stattfindet, nur ein Benutzer interagiert gleichzeitig und eine grafische Benutzeroberfläche existiert. Daraus ergibt sich eine relativ kleine Schnittmenge. Zwischen den verschiedenen Arten von Frontends hingegen existieren weit mehr Ähnlichkeiten (siehe Abb. 3).

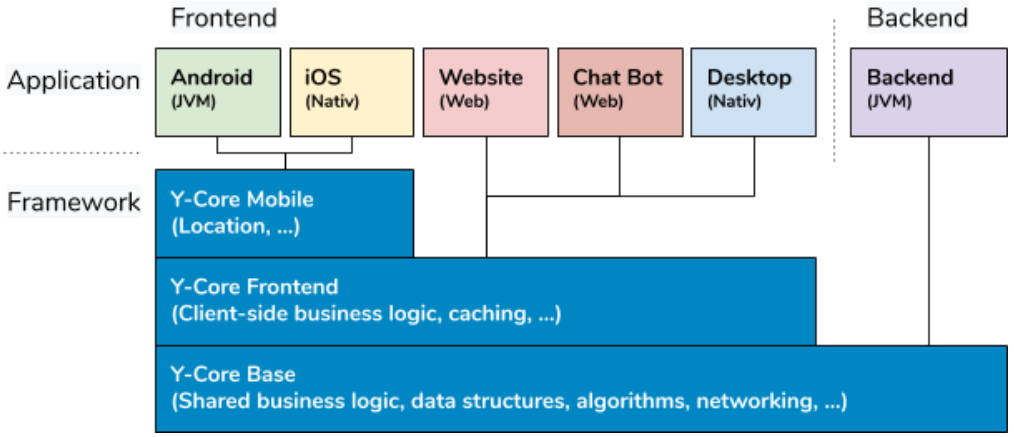

Abb. 3: Y・Core - Mehrfachverwendung von Modulen über Zielplattformen hinweg

Mobilplattformen unterscheiden sich aus Feature-Sicht fast gar nicht. Sowohl Android als auch iOS bieten Ortungsdienste, Integration in Kalender und Kontakte als auch viele weitere Schnittstellen. Browser-Umgebungen bieten weniger Möglichkeiten, stellen aber ebenfalls eine grafische Benutzeroberfläche dar.

Die vom Framework bereitgestellte Funktionalität gliedert sich somit je nach Plattform auf. Daher ergab sich der Name Y・Core und das Logo.

Um dies zu ermöglichen, werden Anforderungen an die interne Software-Architektur definiert. Es bietet sich an, die verschiedenen Komponenten zum einen nach Funktion in unabhängige Module zu gruppieren. Eine Anwendung interagiert mit der Bibliothek konzeptionell über „Use cases". Das heißt, in der öffentlichen Schnittstelle werden Geschäftsfälle wie „Nutzer einloggen" oder „Account erstellen" definiert. Die zugrundeliegende Umsetzung (Implementierung) bleibt verborgen und kann damit leichter verändert werden, ohne dass Anpassungen am Frontend notwendig werden.

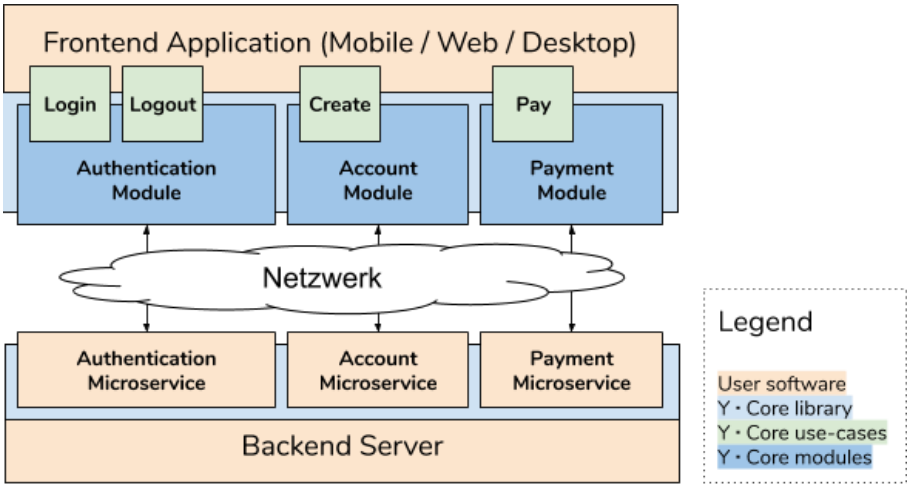

Abb. 5: Y・Core - Interaktion zwischen dem Frontend und dem Backend

Die genutzte Programmiersprache Kotlin (auf Java basierend) ist für Backends und Android identisch und erlaubt es, existierendes Wissen weiter zu nutzen, da Backends basierend auf dem Java Stack (JVM) weit verbreitet sind. Die Erstellung der Bibliotheken für verschiedene Plattformen ist durchaus komplex und wird über mitgelieferte Build-Umgebung als Teil der Norm realisiert:

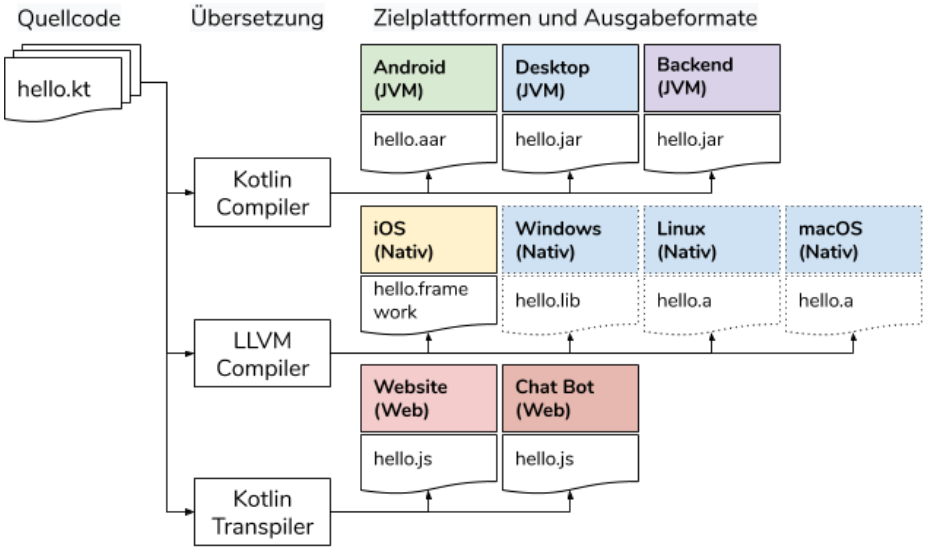

Abb. 4: Y・Core - Quellcode-Übersetzung und Ausgabeformate

Die Reduzierung auf eine Quellsprache bedeutet weniger externe Abhängigkeiten und bessere Wartbarkeit. Eine Sicherheitslücke oder ein Fehler in der Geschäftslogik muss nur einmal behoben werden und kann als Bibliotheks-Update schnell in die Frontends integriert werden.

Wer profitiert von der Innovation

Durch seine Skaleneffekte steht der Y・Core auch kleineren Herstellern preisgünstig zur Verfügung.

Wie werden die Ergebnisse nach Projektabschluss verwertet

Ziel ist die Bereitstellung eines offenen Standards mit einem SDK für die relevanten Plattformen sowie der Pilotanwendung LunchUp als Demonstrationsobjekt.

LunchUp ist eine App, die Teilnehmer über das Mittagessen vernetzt und die gesamte Organisation, wie Treffpunkt, Restaurantvorschlag und Tischbuchung übernimmt. Gerade im Firmenumfeld ist die Verbesserung der Mitarbeiterkommunikation (unter dem Schlagwort „New Work") sehr gefragt und kann auch im Eventbereich eingesetzt werden. In diesem Rahmen konnte der Ideengeber den Y・Core Ansatz erfolgreich einsetzen, um trotz begrenzter Ressourcen einen marktfähigen MVP (Minimum Viable Product) für seine Produktvision umzusetzen (siehe Webauftritt).

Bestandteile des SDKs sind neben den Architekturvorgaben, der Laufzeitumgebung, auch Standardmodule (z.B. Nutzerverwaltung) und eine erweiterbare Bibliotheksstruktur. Die notwendige Build-Umgebung zur Übersetzung in die Zielplattformen wird ebenfalls bereitgestellt. Die individuell gestalten Benutzeroberflächen unterliegen weiterhin dem Urheberrecht des Nutzers und sind nicht Teil des Standards. Ziel der Öffentlichkeitsarbeit ist es, Entwicklern das SDK zur Nutzung des Standards zu erläutern und bekannt zu machen.

Markt- und gesellschaftliche Relevanz

Es ist Realität, dass die Marktplätze im Web und auf Mobilplattformen durch nordamerikanisch geprägte Großkonzerne dominiert oder gar kontrolliert werden. Gerade für kleine und mittlerer Unternehmen und Startups eröffnen sich neue Möglichkeiten, mit den Marktführern in Wettbewerb zu treten, ohne die Kontrolle über ihr geistiges Eigentum abzugeben.

Möglichkeiten zur Einreichung bei Normungsorganisationen

Eine EN- und ISO-Standardisierung wird wegen der grundsätzlichen Bedeutung angestrebt.

Kompetenzen des Bewerbers und seiner Partner

Sascha Peilicke *Gründer und CEO,* LunchUp
Der Ideengeber besitzt langjährige Erfahrungen im Bereich der Informationstechnologie, der Personalführung, sowie im Vertrieb. Er beschäftigte sich bereits vor dem Studium der Informatik mit der Software-Entwicklung und ist insbesondere im Open-Source-Umfeld sehr aktiv. Er führte bereits mehrere multinationale Teams in verschiedenen Positionen, zuletzt als CTO eines etablierten Berliner Startups. Aktuell befindet er sich im Aufbau seiner eigenen Firma, welches eine Lösung zur Mitarbeiter-Vernetzung und -Kommunikation für den modernen Arbeitsmarkt entwickelt.

Pavlo Dyban *Tech Lead Computer Vision for Autonomous Driving, Siemens AG*
Experte für Bildverarbeitung und statistische Datenanalyse sowie der Softwareentwicklung in C++ und Python.

Prof. Dr.-Ing. Jana Dittmann *Arbeitsgruppe Multimedia und Security, Otto-von-Guericke Universität Magdeburg*
Risikobetrachtung und Untersuchung der Sicherheit durch die Forschungsgruppe „Security Evaluation Group" im Instituts für Technische und Betriebliche Informationssysteme.

Christian Nietner *Gründer und CTO, Avanetix UG*
Als Serial Entrepreneur und Physiker optimiert er Geschäftsprozesse zwischen klassischem und Quantum Computing. Beteiligt als potentieller Nutzer des Y・Core in diesem Bereich.

Carsten Mohs *Gründer und CEO, timum GmbH*
Betrachtung des Konzepts hinsichtlich Integrationsfähigkeit in proprietäre Schnittstellen von Drittanbietern im Bereich Immobilien- und Gebäudemanagement.

Robert Manea *Application Architect, it-economics GmbH*
Für die Software-Agentur mit Fokus Apps für Kunden ist das Konzept höchst interessant, um die umrissenen Vorteile bei zukünftigen Projekten nutzen zu können.

Antonio Casero Palmero *Lead Software Engineer, HERE Technologies GmbH*
Als Experte für iOS-Entwicklung steht er mit Fachwissen und Markterfahrung zur Verfügung, um bei der technischen Umsetzung behilflich zu sein.

Standardisierungs-Scope

Der geplante Standard definiert Anforderungen an Merkmale von Software-Produkten für die Veröffentlichung auf mehreren Zielplattformen, insbesondere für kleine und mittlere Unternehmen.

Der Standard beinhaltet ein Software-Framework zur plattformübergreifenden Anwendungsentwicklung und legt Anforderungen an die Softwarearchitektur und den Entwicklungsprozess fest, um Aufwände zu reduzieren, die Wiederverwendbarkeit, die Time-To-Market zu verkürzen und die Anwendungssicherheit gegenüber existierenden Verfahren wesentlich zu verbessern. Die Norm erlaubt es gerade kleinen und mittleren Unternehmen sowie Privatanwendern, die Umsetzung ihrer Software-Produkte effizienter und kostengünstiger zu gestalten, um damit effektiver mit globalen Playern im Markt konkurrieren zu können.

Projektplan

Arbeitspakete

- **AP 1:** Entwicklung des Prototyps zur Marktreife
  - Entwicklung eines Demonstrationsobjektes (i.e. die App LunchUp) abschließen
  - Beispiel-SDK mit Dokumentation
  - Bereitstellung generischer Komponenten (Authentifizierung, Nutzer-Verwaltung, etc.)
- **AP 2:** Öffentlichkeitsarbeit
  - Broschüre zum Standard
  - Dokumentation des Entwicklungsprozesses und der Architektur
  - Bewerbung des Angebots (z.B. YouTube Erklärvideo)
  - Dedizierter Webauftritt zur Vermarktung mit Beispielcode auf GitHub
- **AP 3:** Qualitätssicherung
  - Externe Validierung (Software-Testing und Anwendungs-Sicherheit)
- **AP 4:** Standardisierung
  - Beratung

Möglicher Ablauf

	2020	

	03	04	05	06	07	08	09	10	11	12	Kosten
AP 1	X	X	X	X	X	X	X	X			13000 €
AP 2			X	X	X	X	X	X	X	X	8000 €
AP 3			X				X	X	X		10000 €
AP 4			X	X	X	X	X	X	X		4000 €
Berichte / Sitz.	K	M		M	Z		M	Z	M	A	
								Fördersumme			**35000 €**

Erläuterungen

- AP 1: Neben Hardware-Investitionen (Testgeräte) auch Fremdleistungen im Bereich der Software-Entwicklung angedacht.
- AP 2: Für die Öffentlichkeitsarbeit entstehen Kosten unter anderem für Werbematerial (Print und Video), sowie für die Teilnahme und Präsentation auf Fachmessen
- AP 3: Fremdleistungen für das Projekt bzw. Kosten zur Qualitätssicherung / Sicherheitszertifizierung