Ausgabe 03/2020

Java aktuell

Praxis. Wissen. Networking. Das Magazin für Entwickler

D: 4,90 EUR | CH: 9,80 CHF | A: 5,60 EUR | Benelux: 5,80 EUR

4 191978 304903 03

Java aktuell

iJUG Verbund
www.ijug.eu

Progressive Web Apps
PWAs und ihre Vorteile in der Praxis

Flutter
Von der Entscheidung bis zur Entwicklung

Geteilte Hologramme
Shared Augmented Reality im Überblick

DIE ZUKUNFT IST MOBILE

adesso business. people. technology.

Von Anfang an Teil des Java-Teams!

Entwickelt bereits kluge IT-Lösungen bei adesso:
Ihr neuer Kollege Kristof Hierath | Software Engineer

SOFTWARE DEVELOPMENT@adesso

Sie wollen dort einsteigen, wo Zukunft programmiert wird? Dann sind Sie mit einem Start in unserem Java-Team bei adesso genau richtig. Gemeinsam setzen wir herausfordernde Projekte für unsere Kunden um. Dafür brauchen wir Menschen, die Lust haben, ihr Wissen, ihre Talente und ihre Fähigkeiten einzubringen.

Planen und realisieren Sie in interdisziplinären Projektteams anspruchsvolle Anwendungen und Unternehmensportale auf Basis von Java/JavaScript-basierten Technologien als

→ **(Senior) Software Engineer (w/m/d) Java**
→ **Software Architekt (w/m/d) Java**
→ **(Technischer) Projektleiter Softwareentwicklung (w/m/d) Java**

CHANCENGEBER – WAS ADESSO AUSMACHT

Kontinuierlicher Austausch, Teamgeist und ein respektvoller, anerkennender Umgang sorgen für ein Arbeitsklima, das verbindet. So belegen wir nach 2016 auch 2018 den 1. Platz beim Wettbewerb „Deutschlands Beste Arbeitgeber in der ITK"!

Mehr als 650 Software Engineers Java bei adesso, über 120 Schulungen und Weiterbildungen – zum Beispiel in Angular2 oder Spring Boot – sowie ein Laptop und ein Smartphone ab dem ersten Tag warten auf Sie!

GREAT PLACE TO WORK® 2018 Beste Arbeitgeber™ ITK

IHRE BENEFITS – WIR HABEN EINE MENGE ZU BIETEN:

- Welcome Days
- Choose your own Device
- Weiterbildung
- Events: fachlich und mit Spaß
- Sportförderung
- Mitarbeiterprämien
- Auszeitprogramm

Es wird Ihnen bei uns gefallen! Mehr Informationen auf **www.karriere.adesso.de**.
Olivia Slotta aus dem Recruiting-Team freut sich auf Ihre Kontaktaufnahme:
adesso SE | Leona Demiri | T +49 231 7000-7100 | jobs@adesso.de

Es Fluttert gewaltig

René Jahn, SIB Visions GmbH

Wer eine App für mobile Geräte entwickeln möchte, steht zumindest vor einer wichtigen Frage: Mit welcher Technologie soll die App realisiert werden? Da es aktuell verschiedenste Möglichkeiten gibt, ist die Antwort nicht sofort klar. Mit diesem Problem war das Framework-Team von SIB Visions ebenfalls konfrontiert, als es darum ging, einen Client für das Open Source Application Framework JVx zu entwickeln. Wie es zu einer Entscheidung kam und vor allem was umgesetzt wurde, wird in diesem Artikel erklärt.

Das JVx Framework *[1]* ermöglicht die Erstellung von Datenbank-Applikationen für unterschiedlichste GUI-Technologien mit einer einzigen Source-Basis. Es ist sowohl eine GUI-Abstraktion als auch ein Full-Stack Application Framework. Die damit erstellten Applikationen laufen ohne Änderungen im Browser mittels JavaScript/CSS, am Desktop mit Swing oder JavaFX und auf mobilen Geräten mithilfe nativer Apps. Mit JVx können Applikationen unterschiedlichster Größe und Komplexität realisiert werden. Zwei Produkte, die mit JVx entwickelt wurden, sind beispielsweise SNOWsat Maintain *[3]* und VisionX *[4]*.

Wie soeben erwähnt, werden für die Ausführung auf mobilen Geräten native Apps eingesetzt. Es gibt eine App für iOS und eine andere für Android. Erfahrene App-Entwickler werden vermuten, dass die-

se Apps unabhängig voneinander entwickelt wurden. Das ist auch genauso. Die iOS-App wurde mit Objective-C und die Android-App mit Java entwickelt. Das war im Jahr 2014 auch ein gangbarer Weg, da es nur wenige Lösungen gab, die eine Entwicklung mit nur einer Source-Basis anboten. Noch dazu waren diese nicht sehr preiswert und auch nicht immer Open Source. Dem Framework-Team bei SIB Visions waren die beiden Apps immer ein Dorn im Auge, da sie doppelte Wartung und Weiterentwicklung bedeuteten. Man verfolgte über die Jahre hinweg immer wieder Ablösegedanken und verglich diverse Technologien. Im Jahr 2019 wurde die Ablöse beschlossen. Der Startschuss fiel im Herbst und das Team begann mit der Evaluierung der möglichen Technologien.

Die infrage kommenden Lösungen waren Xamarin *[5]*, React Native *[6]* und Flutter *[7]*. Im Vorfeld schloss man Lösungen wie Qt, Ionic, Phonegap, Sencha Touch, NativeScript und Gluon Mobile bereits aus. Andere Lösungen wie Jasonette, Weex, Framework7 oder CodeOne zog man ebenfalls nicht in Betracht.

Der Entscheidungsprozess für die geeignete Technologie war relativ unkompliziert und beruhte auf messbaren Kriterien, der Erfahrung des Framework-Teams und „Gefühl". Die Anforderungen:

- Die App performt wie eine native App
- Gerätefeatures wie Anruf, Geolocation und Kamerazugriff
- Ausreichend Komponenten, die für Datenbankapplikationen benötigt werden wie Tabelle, Datumauswahl, Auswahllisten
- Die Beispiel-Applikationen stürzen nicht ab und hinterlassen einen bleibenden Eindruck
- Open-Source-basiert oder frei für Open-Source-Projekte verwendbar
- Kurze Entwicklungszyklen
- Live-Voransicht ohne längere Wartezeiten
- Gutes Toolset mit IDE-Unterstützung
- Erweiterbar durch Add-ons, Plug-ins oder Ähnliche
- Einfache Versionsaktualisierungen und wenige Abhängigkeiten zu zusätzlichen Bibliotheken

Anhand dieser Anforderungen nahm man die einzelnen Technologien unter die Lupe. Bei einigen Kandidaten verhinderten einzelne Bedenken aus dem Framework-Team den Einsatz. Die meisten JavaScript- und Browser-basierten Technologien fühlten sich nicht nativ an und wurden deswegen ausgeschlossen. Andere verfügten nicht über die vollständigen Gerätefeatures oder der Source Code wirkte zu komplex und unverständlich. Andere wiederum erfüllten die kurzen Entwicklungszyklen nicht oder boten die gewünschte Live Preview nicht wie erwartet.

Letzten Endes blieben Xamarin, React Native und Flutter übrig. Obwohl Xamarin lange Favorit war, waren die langen Wartezeiten ein entscheidender Nachteil, wodurch sich die Entwickler nicht produktiv genug fühlten. Natürlich waren dies reine Empfindungen und damit schwer messbar, aber es sollte sich als richtig herausstellen. Der Grund, warum React Native zu den Favoriten zählte, war, dass bereits Erfahrung im Team existierte und eine produktive Applikation damit umgesetzt wurde. Doch letzten Endes führten die Erfahrungen bei der App-Entwicklung auch hier zum Ausscheiden. Es gab immer wieder interne Probleme, für die Workarounds gesucht werden mussten, das Fokus-Handling war nicht wie gewünscht und der Gesamteindruck war nicht zufriedenstellend. So blieb am Ende also Flutter übrig. Doch nicht, weil es der letzte Kandidat war, sondern weil man von Anfang an den Eindruck hatte, dass es sich um die richtige Technologie handelt.

Warum Flutter?

Das Framework-Team hatte keinerlei Erfahrungen mit Flutter. Auch die Programmiersprache Dart, mit der Flutter entwickelt wurde, war komplettes Neuland. Doch durch die Nähe zu Java stellte Dart kein Problem dar. Im Gegenteil, die Sprachfeatures machten Spaß und das Framework-Team kam gut damit zurecht. Als Entwicklungsumgebung kamen Android Studio und VSCode infrage. Egal, welche IDE eingesetzt wurde – die Produktivität war gegeben.

Einer der Wow-Effekte entstand durch die Live Preview und das Hot Reload Feature bei Änderungen am Code. Auch wenn es nicht immer problemlos funktionierte, so fühlte sich die Entwicklung rasend schnell an. Unabhängig von der Entwicklung beeindruckte die relativ junge Technologie mit einer Vielzahl von Erweiterungen und mit einem modernen, jedoch nicht neuen Ansatz der GUI-Entwicklung.

Doch Flutter erfüllte nicht nur die Anforderungen vollständig und schaffte ein gutes Gefühl, es passte auch von den internen Konzepten perfekt zu JVx. Natürlich war von Anfang an klar, dass es sich um eine relativ neue Technologie handelt und dass die Zukunftssicherheit offen ist. Doch das Entwicklungsteam hatte entschieden und schlussendlich muss es auch damit arbeiten wollen.

Der Ansatz, den Flutter verfolgt, ist ähnlich zu JVx, da es ebenfalls das GUI abstrahiert und unterschiedliche Implementierungen bietet. Davon bekommt der Entwickler jedoch im besten Falle nichts mit und wundert sich, warum das auf den unterschiedlichen Geräten so reibungslos funktioniert. Die Probleme mit den zugrunde liegenden Technologien werden vom Framework gelöst und der Entwickler kann sich auf die App-Entwicklung konzentrieren. Diese Philosophie wird auch von JVx verfolgt.

Bei Flutter handelt es sich vorranging nicht um eine Technologie, mit der grafisch anspruchsvolle Spiele entwickelt werden sollen. Auch wenn dies möglich ist und durch Low-Level APIs unterstützt wird, so liegt der Fokus dennoch auf der Entwicklung von Produktivitäts-Apps. Wer an dieser Stelle mehr über das Design von Flutter erfahren möchte, sollte sich ein Video von Ian Hickson *[8]* nicht entgehen lassen.

Es muss eine App sein

Mit Flutter sollte also eine App für JVx entwickelt werden. Diese musste am Ende des Tages eine JVx-Applikation auf mobilen Geräten darstellen. Hier könnte die Frage entstehen, warum die App nicht gleich ohne JVx entwickelt wurde. Das wäre möglich gewesen, allerdings bietet JVx die Möglichkeit, eine Applikation mit unterschiedlichen Technologien darzustellen, beispielsweise als Webanwendung, am Desktop oder auch Headless für automatisiertes Testen. Auch das sollte mit Flutter möglich sein, jedoch ist beispielsweise die Web-Implementierung noch nicht ausgereift. Daher ist die App ein guter Anfang, um in Zukunft eventuell vollständig auf Flutter zu setzen. Im Unterschied zu Flutter ist man mit JVx jedoch immer technologieunabhängig und wenn nötig, wird ganz einfach eine neue Technologie angebunden. Die damit entwickelte Applikation muss dann nicht neu entwickelt werden. Diesen entscheidenden Vorteil bietet eben nur JVx.

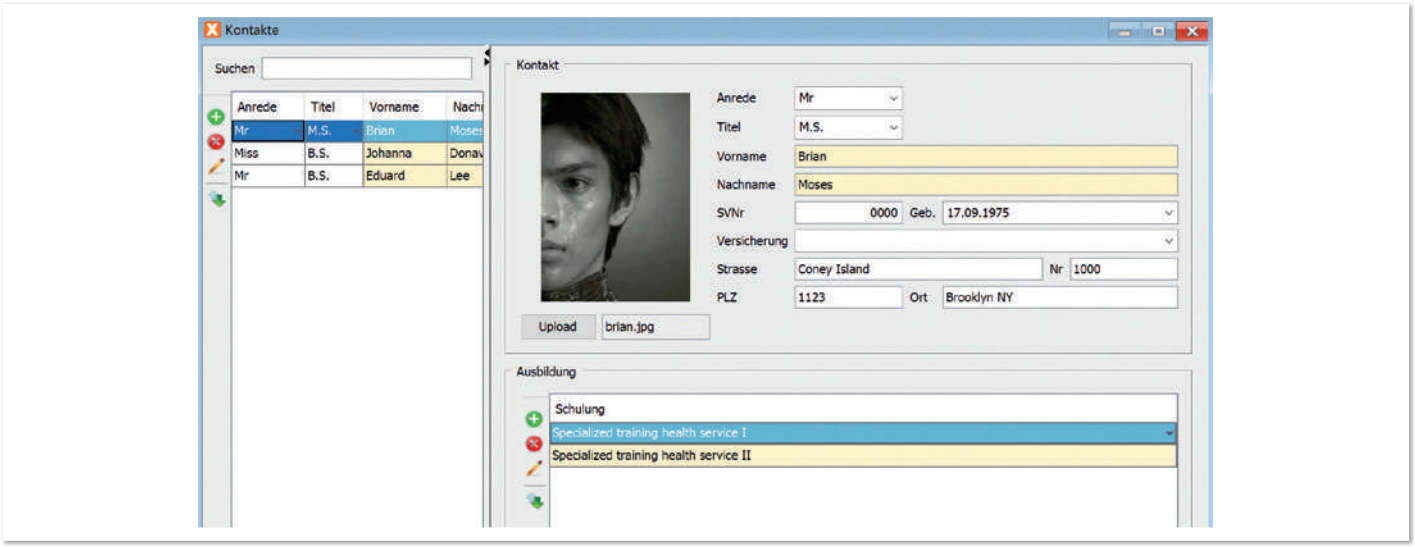

Abbildung 1: JVx-Applikation am Desktop (© René Jahn)

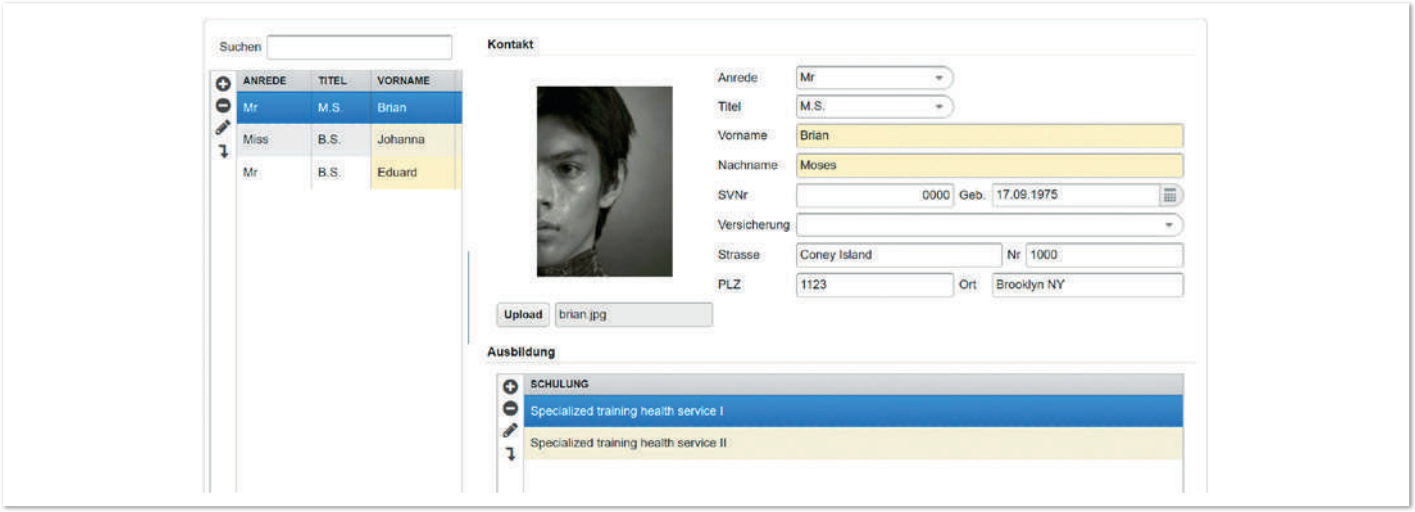

Abbildung 2: JVx-Applikation am Desktop, volle Größe (© René Jahn)

Abbildung 3: JVx-Applikation im Web, responsive Liste (© René Jahn)

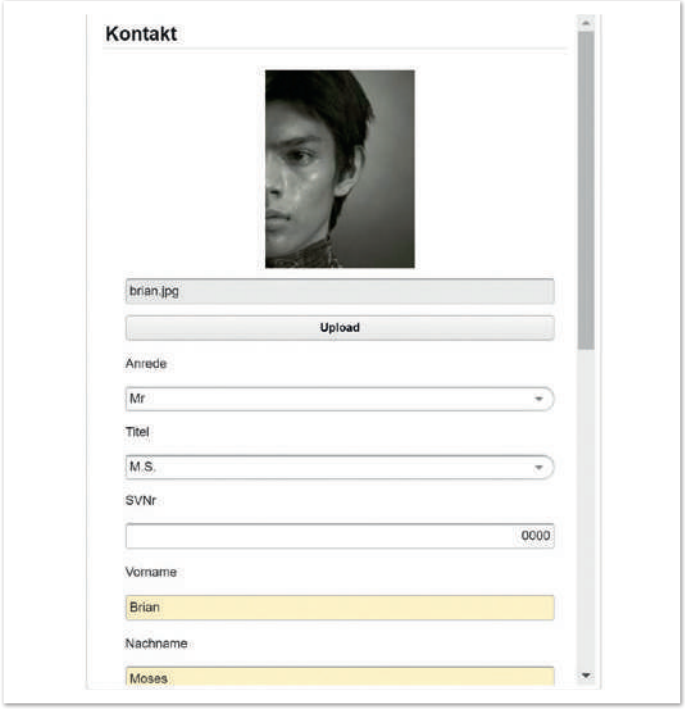

Abbildung 4: JVx-Applikation im Web, responsive Details (© René Jahn)

Um einen besseren Eindruck zu bekommen, sehen wir uns jetzt – anhand einer Beispielapplikation – eine JVx-Applikation für Desktop, Web und als Flutter-App an. Die Applikation verwaltet lediglich eine Liste von Kontakten, wie in der Desktop-Variante *(siehe Abbildung 1)* zu sehen ist.

Bei der Desktop-Variante ist im linken Bereich eine Liste von Kontakten zu sehen, im rechten die Kontaktdetails. Am Desktop ist ausreichend Platz für die gleichzeitige Darstellung aller Informationen. Die Web-Variante ist hier ein wenig ausgeklügelter, denn es gibt einerseits die volle Darstellung *(siehe Abbildung 2)*. Andererseits ist man es im Web gewohnt, dass Inhalte responsive sind. Die Kontakteverwaltung wird daher bei geringer Bildschirmgröße automatisch an die Platzverhältnisse angepasst (siehe Abbildungen 3 und 4). In *Abbildung 3* ist lediglich die Kontakteliste zu sehen. Erst, wenn ein Kontakt gewählt wird, erscheinen die Details wie in *Abbildung 4*. Die Beispielapplikation läuft natürlich auch auf mobilen Geräten im Browser ohne Probleme, allerdings bleibt es eine Web-Applikation und das fühlt sich nicht nativ an.

Wir werfen jetzt einen Blick auf die Flutter-App. In *Abbildung 5* ist ebenfalls nur die Kontakteliste zu sehen und bei Auswahl eines Kontaktes werden auch hier die Details angezeigt *(siehe Abbildung 6)*.

Der optische Unterschied ist aufgrund des Material-Designs deutlich zu sehen und auch die Navigationsmöglichkeit von mobilen Geräten ist im Header vorhanden. Aber im Großen und Ganzen ist die Darstellung ähnlich wie im Browser. Das war auch der Anspruch der Entwickler. Die Applikation soll zwar mit unterschiedlichen Technologien ausgeführt werden, aber optisch und funktional weitestgehend vergleichbar sein.

Aller Anfang ist schwer

Als mit der Umsetzung der App begonnen wurde, war die Euphorie im Team riesig, da auf das richtige Pferd gesetzt wurde und Flutter frisch und modern wirkte. Das mag auch stimmen, aber leider kam es bei der Entwicklung häufig zu unerwarteten Problemen. Um nur ein paar zu nennen:

- Die Daten von Auswahllisten können nicht dynamisch nachgeladen werden. Wenn man versucht, das zu umschiffen, kommt es vor, dass die Auswahlliste geschlossen wird.
- Das Widget für die Anzeige einer Auswahlliste [10] hat keinen Event, um zu notifizieren, dass die Auswahlliste geöffnet wird.
- Von Flutter werden Tabellen-Widgets [11][12] angeboten. Dort wurde dynamisches Nachladen von Daten aber nicht vorgesehen beziehungsweise funktionierte nicht richtig.
- Es gibt Textfeld-Formatierungsmöglichkeiten [13] um gegebenenfalls nur Nummern zu erlauben. Wenn man dann das Textfeld allerdings rechtsbündig ausrichtet, kommt es zu unschönen Effekten.

Um die Einschränkungen zu umgehen, setzte man eigene Widgets um. Das war aufgrund der umfangreichen Dokumentation von Flutter auch kein Problem. Eigene Konzepte konnte man durch Flutters offene Programmierung einfach umsetzen und es gibt keine Geheimnisse, auf die nicht zugegriffen werden könnte.

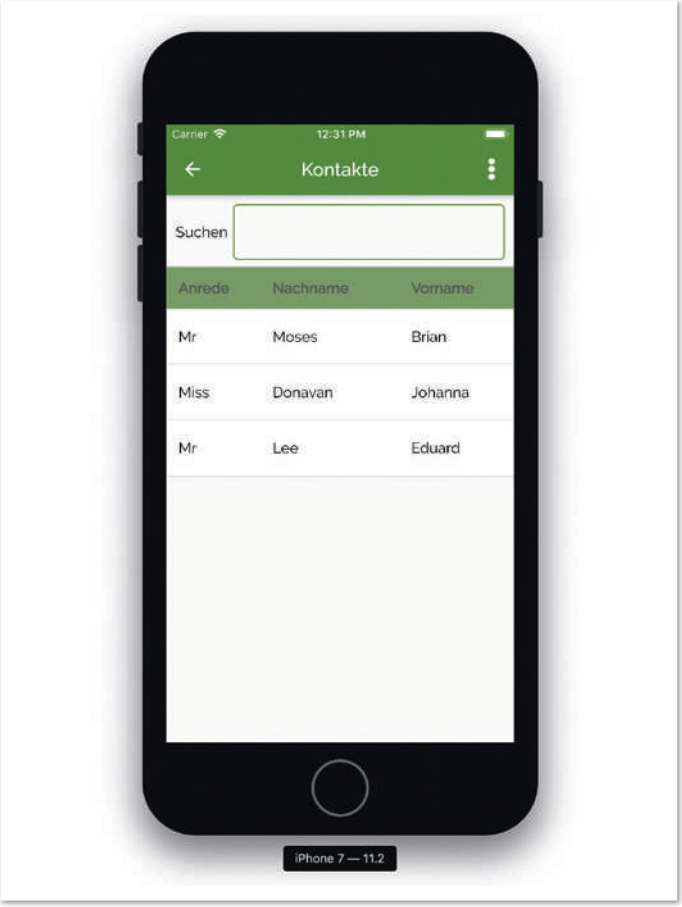

Abbildung 5: JVx-Applikation mit Flutter, Liste (© René Jahn)

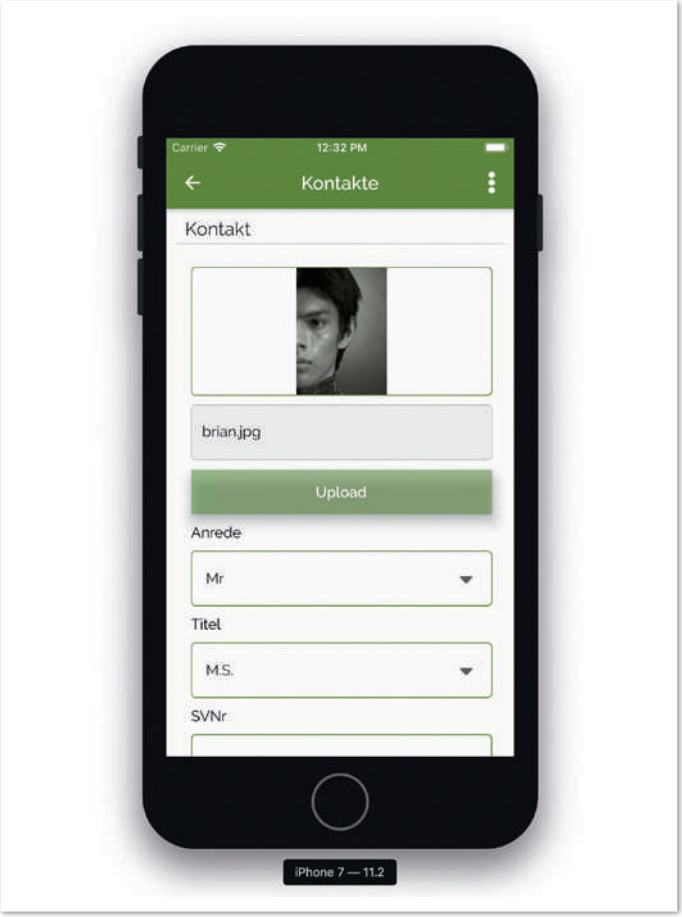

Abbildung 6: JVx-Applikation mit Flutter, Details (© René Jahn)

Dass einiges neu entwickelt werden musste, war dem Team von Anfang an klar, da Flutter diverse Layout Widgets *[14]* mitbringt, aber nicht die gewünschten. Es wäre zwar möglich, mit Schachtelungen das ein oder andere gewünschte Verhalten hinzubekommen, aber das macht die Layouts nur komplexer und erschwert die Wartung. Daher entwickelte man alle benötigten Layouts von Grund auf neu. In JVx gibt es nur wenige Layouts. Diese sind:

- BorderLayout
- FlowLayout
- FormLayout
- GridLayout

Das BorderLayout ist das einfachste von allen und entspricht dem Standard Java BorderLayout *[15]*. Das FlowLayout ist ebenfalls vergleichbar mit dem Standard Java FlowLayout *[16]* mit erweiterten Möglichkeiten zum Zeilenumbruch und Komponenten-Ausrichtung. Das FormLayout *[17]* ist ein Anker-basiertes Layout, das zeilen- und spaltenorientiert arbeitet, aber freie Platzeinteilung erlaubt. Das GridLayout ist zeilen- und spaltenorientiert, aber mit fixen Kachelgrößen.

Die Umsetzung der Layouts funktionierte wunderbar, da Flutter alles mitbringt, was für eigene Layouts benötigt wird. Das Framework-Team wunderte sich allerdings darüber, dass es nahezu keine Beispiele gab, wie eigene Layout Widgets umzusetzen sind. Im Web finden sich kaum bis keine Informationen oder sie sind einfach nur zu gut versteckt. Es blieb also nur der schwierige Weg über die manuelle Analyse des Flutter Source Codes und der darin enthaltenen Dokumentation. Im Nachhinein betrachtet, war das dann doch recht aufwendig und kostete Zeit. Wer nun wissen möchte, wie eigene Layout Widgets erstellt werden, der kann im Git Repository *[18]* stöbern. Hier wird es in Zukunft weitere Veröffentlichungen geben.

Nachdem man die meisten Widgets implementiert und die bekannten Einschränkungen gelöst hatte, konnte mit Volldampf an der Umsetzung der App gearbeitet werden. Das Geniale war, dass die Euphorie im Team anhielt und nicht gedämpft wurde. Das lag vor allem daran, dass Fortschritte schnell erzielt wurden und nicht mit hohem Aufwand verbunden waren. Die Entscheidung für Flutter wurde zu keinem Zeitpunkt angezweifelt.

Die Endausbaustufe

Wie zu Beginn des Artikels erwähnt, wurde VisionX mit JVx entwickelt. Dabei handelt es sich um eine Low-Code-Development-Plattform, um JVx-Applikationen zu erstellen. Die Applikationen basieren auf Open Source und können auch ohne VisionX weiterentwickelt werden. Dadurch, dass die Applikationen JVx verwenden, kann auch für dieses Produkt in Zukunft der Flutter Client eingesetzt werden. Die Vorteile sind gewaltig, da man mit VisionX eine Applikation erstellt, die dann im Browser, am Desktop und auf mobilen Geräten auf die gleiche Art und Weise funktioniert. Die Applikation kann jederzeit und mit jeder beliebigen Java-IDE (Eclipse, IntelliJ, NetBeans etc.) bearbeitet werden. Bei Bedarf ist es möglich, jede Technologie an die eigenen Bedürfnisse anzupassen. Sollte etwa der Funktionsumfang des Flutter Client nicht ausreichen, so kann man diesen ganz einfach an die eigenen Wünsche anpassen.

Fazit

Bei Flutter handelt es sich um eine neue Technologie, die rasend schnell die Entwicklergemeinde erobert. Die Möglichkeiten sind enorm und es fühlt sich gut an, damit zu entwickeln. Es ist eine Technologie, die Entwickler gerne einsetzen. Natürlich hat Flutter noch Luft nach oben und die Visionen sind groß. Auf die Web-Variante sollte besonders geachtet werden. Die Entscheidung für Flutter war, auch nach mehreren Monaten Bedenkzeit, die absolut richtige. Die Anforderungen konnten vollständig umgesetzt werden und es blieben keine Wünsche offen. Wir sind gespannt, was in Zukunft auf uns zufluttert.

Quellen

[1] *https://de.wikipedia.org/wiki/JVx_(Framework)*
[2] *https://www.informatik-aktuell.de/entwicklung/programmiersprachen/ein-gui-fuer-alle-faelle.html*
[3] *https://www.snowsat.com/aut/de/produkte/snowsat-maintain.html*
[4] *https://visionx.sibvisions.com/*
[5] *https://dotnet.microsoft.com/apps/xamarin*
[6] *http://www.reactnative.com/*
[7] *https://flutter.dev/*
[8] *https://www.youtube.com/watch?v=dkyY9WCGMi0*
[10] *https://api.flutter.dev/flutter/material/DropdownButton-class.html*
[11] *https://api.flutter.dev/flutter/material/DataTable-class.html*
[12] *https://api.flutter.dev/flutter/widgets/Table-class.html*
[13] *https://api.flutter.dev/flutter/services/TextInputFormatter-class.html*
[14] *https://flutter.dev/docs/development/ui/widgets/layout*
[15] *https://docs.oracle.com/javase/tutorial/uiswing/layout/border.html*
[16] *https://docs.oracle.com/javase/tutorial/uiswing/layout/flow.html*
[17] *http://doc.sibvisions.com/jvx/reference#formlayout*
[18] *https://github.com/sibvisions/flutterclient/tree/master/lib/ui/layout*

René Jahn
SIB Visions GmbH
rene.jahn@sibvisions.com

René Jahn ist Mitbegründer der SIB Visions GmbH und Head of Research & Development. Er verfügt über langjährige Erfahrung im Bereich der Framework- und API-Entwicklung. Neben der Open-Source-Sparte bringt er seine Expertise auch in der Produktentwicklung ein. Seine Leidenschaft ist die Evaluierung neuer Technologien und Integration in klassische Business-Applikationen.

Java aktuell

Mehr Informationen zum Magazin und Abo unter:

https://www.ijug.eu/de/java-aktuell

iJUG Verbund
www.ijug.eu

Eventpartner: AOUG SOUG swiss oracle user group iJUG Verbund ORACLE