Mobile Applikationen mit JavaFX entwickeln

Roland Hörmann
SIB Visions GmbH
Wehlistraße 29 / Stiege 1 / 2.Stock, 1200 Wien

Schlüsselworte
JavaFX, effizient Business Applikationen erstellen, Mobile, Web & Desktop Applikationen, Open Source, JVx Framework

Einleitung
Mit JavaFX können tolle Backoffice Applikationen erstellt werden. Styling per CSS, Animationen, Transitions und andere Effekte aus der Web-Welt sind nun auch für den Desktop verfügbar.

Doch JavaFX ist nicht auf den Desktop beschränkt. Damit können auch native Apps für Android und iOS erstellt werden? Wie bitte?
Sie haben richtig gelesen, mit Java erstellen Sie eine native Anwendung für iOS und Android!

In dieser Session wird gezeigt wie Sie JavaFX auf den mobilen Device einsetzen können. Mit der Unterstützung des Open Source Framework JVx lässt sich in wenigen Minuten eine moderne Backoffice Lösung erstellen. Jedoch sind mobile Apps etwas anders zu designen als Desktop Applikationen. Wir gehen darauf ein - was aus Sicht des Desktop Entwicklers- bei der Entwicklung von Benutzerfreundlichen mobilen Applikationen zu beachten ist. Einige dieser Vorgangsweisen sollten auch bei der Entwicklung von Backoffice Anwendungen, und natürlich in der Entwicklung von Web Anwendungen berücksichtigt werden. Daher wird die Vorgangsweisen für optisch ansprechende, benutzerfreundliche, moderne Desktop, Web und Mobile Anwendungen an Hand von JavaFX und JVx veranschaulichen.

JavaFX (mobile) Geschichte
Die erste Version von JavaFX wurde Ende 2008 veröffentlicht, damals noch von Sun entwickelt und eher als Skriptsprache gedacht. Durch die spätere Übernahme von Sun durch Oracle änderte sich auch bei JavaFX einiges. Es wurde nämlich neu entwickelt in Hinblick auf reinen Java APIs anstatt von Scripting.
Im Oktober 2011 erschien die Release 2.0 von JavaFX komplett ohne Scripting und mit vielen neuen Features, allerdings ohne mobile Support. JavaFX für mobile Geräte wurde zwar von Oracle entwickelt/gestartet, aber schlussendlich nie offiziell released. Erst als Oracle entschied, den Source Code von JavaFX an die OpenSource Community zu übergeben bot sich die Möglichkeit dies zu ändern. Der Source Code wurde im Rahmen des OpenJDK Projektes übergeben und ist seither als OpenJFX zu finden. Das war auch zugleich der Startschuss für die Portierung auf mobile Geräte.

Die RoboVM war das Ergebnis, der Anstrengungen von Niklas Therning, um Java für iOS Geräte verfügbar zu machen. Im Mai 2013 konnte das erste preview Release von RoboVM verwendet werden (https://robovm.com/2013/01/). Anhand der JavaFX Beispiel Anwendung „Brick Breaker“ wurde auch gleich gezeigt das JavaFX auf mobilen Geräten sinnvoll ist.
Und nachdem das Problem mit iOS gelöst schied, dachte sich Johan Vos dass es für Android doch auch ganz einfach sein müsste, denn für Android Applikationen kann man ja auch mit Java entwickeln. Im Januar 2014 (http://www.lodgon.com/dali/blog/entry/JavaFX_and_Android), oder wohl etwas früher, fiel der Startschuss für JavaFXPorts. Das ist einerseits die Runtime für JavaFX unter Android und andererseits wurde das Entwicklen, Builden und Deployen von JavaFX

Applikationen auf iOS und Android Devices endlich mit sinnvollem Aufwand möglich. Mittlerweile sind wir im Jahr 2015 angekommen und es gibt bereits JavaFX Anwendungen in den App Stores und auch die Entwicklung von Apps ist ein Kinderspiel geworden.

JavaFX auf mobilen Geräten – Wie geht das?
Die Entwicklung und das Starten einer JavaFX Applikation am Desktop ist von der technischen Umsetzung klar. Man codiert seine JavaFX Applikation im IDE, nutzt die verschiedenen JavaFX Klassen für sein UI. Danach buildet man seine Applikation – erstellt ggf. ein jar, und lässt Sie in der JRE laufen.

Auf mobilen Devices ist dies auf Grund des App Store Models in dieser Form nicht möglich. Es wird ein vollständiges Paket von der App, im Format der Zielplattform für den App Store benötigt. Für Android ist dies ein Android Package (APK) und für iOS ist dies ein entsprechendes iOS Package (IPK).

Um nun so ein Paket zu erstellen brauchen wir die Plattform spezifische Java Runtime, die Plattform spezifischen JavaFX Libraries, sowie die Klassen und Libraries der entwickelten Applikation.

Unter Android ist die Java Runtime kein Problem, weil bereits am Device vorhanden. Um ein Android Package (APK) zu erstellen, brauchen wir die Libraries der Android SDK, die Android spezifischen JavaFX Libraries, sowie die Applikationsspezifischen Ressourcen, Klassen und Libraries.

Unter iOS ist dies ein bisschen schwieriger, weil wir hier die Java Runtime in unser Package inkludieren müssen. Dabei handelt es sich nicht um eine VM im klassischen Sinn, sondern, es wird mittels RoboVM ein fertiges IPK mit allen Binärdateien erstellt. Die Java Applikation wird sozusagen Cross compiled. Analog zu Android brauchen wir auch hier die iOS spezifischen JavaFX Libraries, sowie die Applikationsspezifischen Ressourcen, Klassen und Libraries dazu.

Um das Builden und Deployen zu vereinfachen, setzen wir JavaFXPorts ein. Das Open Source Projekt von Gluon (Johann Vos und Kollegen) ermöglicht mit dem jfxmobile Gradle Plugin für NetBeans bzw. Eclipse oder IntelliJ, das Erstellen von APKs und IPKs mit wenig Aufwand. Weiters kann unsere Applikation auch gleich am entsprechenden Device bzw. Simulator ausgeführt werden.

Weiters verwenden wir das Open Source Framework JVx, um den Codieraufwand für unsere JavaFX App sehr gering zu halten. JVx bietet sehr viel Vereinfachung in Bezug auf Datenbindung und Persistence. Es bietet bereits fertige Applikationsrahmen für die unterschiedlichen UI Plattformen wie Desktop, Web und Mobile an. Unter Applikationsrahmen ist die Menüführung, die Navigation, der Login usw. also der Rahmen um die verschiedenen Screens bzw. Views gemeint. Die genannten Vorteile bringen gerade bei Business Applikationen einen großen Zeitgewinn sowie Reduktion des Komplexitätsgrades.

Als Entwicklungsumgebung verwenden wir JDK 8, Eclipse Mars und natürlich das mobile plugin (https://bitbucket.org/javafxports/javafxmobile-plugin) aus dem JavaFXPorts Projekt (http://docs.gluonhq.com/javafxports/). Für die iOS App brauchen wir einen Mac Rechner mit MacOS X 10.9 oder höher und die Xcode 6.x Entwicklungsumgebung. Für Android benötigen wir die Android SDK. Zusätzlich verwenden wir noch den aktuellen Build der JVx Libraries (http://www.sibvisions.com/de/jvxmdownload).

Um nun unsere Beispielapplikation mit dem javafx mobile plugin zu builden und am Device zu testen, verwenden wir folgendes buildfile:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'org.javafxports:jfxmobile-plugin:1.0.3'
    }
}

apply plugin: 'org.javafxports.jfxmobile'

repositories {
    jcenter()
}

mainClassName = 'com.gluonapplication.GluonApplication'

jfxmobile {
    android {
        androidSdk = 'C:/Tools/android-sdk_r24.4-windows/android-sdk-
windows'
        manifest = 'src/android/AndroidManifest.xml'
    }
    ios {
        infoPList = file('src/ios/Default-Info.plist')
    }
}
```

Unter „mainClassName“ haben wir unsere JavaFX main Klassen eingetragen. Für iOS werden unter „forceLinkClasses“, die einzubindenden Klassen unserer Applikation hinterlegt. Unter „dependencies“ haben wir das javafx mobile Plugin als Abhängigkeit eingetragen.

Interessant ist, dass die aktuelle Android VM leider nicht die Java 8 Sprachfeatures unterstützt, z.B. Lambda expressions, Stream API, neues DateTime API. Für Lambdas kann das Retrolambda Plugin verwendet werden um Java 6 kompatiblen Bytecode zu erstellen. Auf „Streams“ und „Optional“ muss derzeit verzichtet werden. Für Streams kann auch noch ein Backport (http://sourceforge.net/projects/streamsupport/) eingesetzt werden, womit viele APIs auch mit Java6 eingesetzt werden können. Die Einschränkung ist in der Praxis aber kein gravierendes Problem. Natürlich kann es zu Problemen führen, wenn man Bibliotheken einsetzt, die Java8 Sprachfeatures einsetzen und nicht Open Source sind.

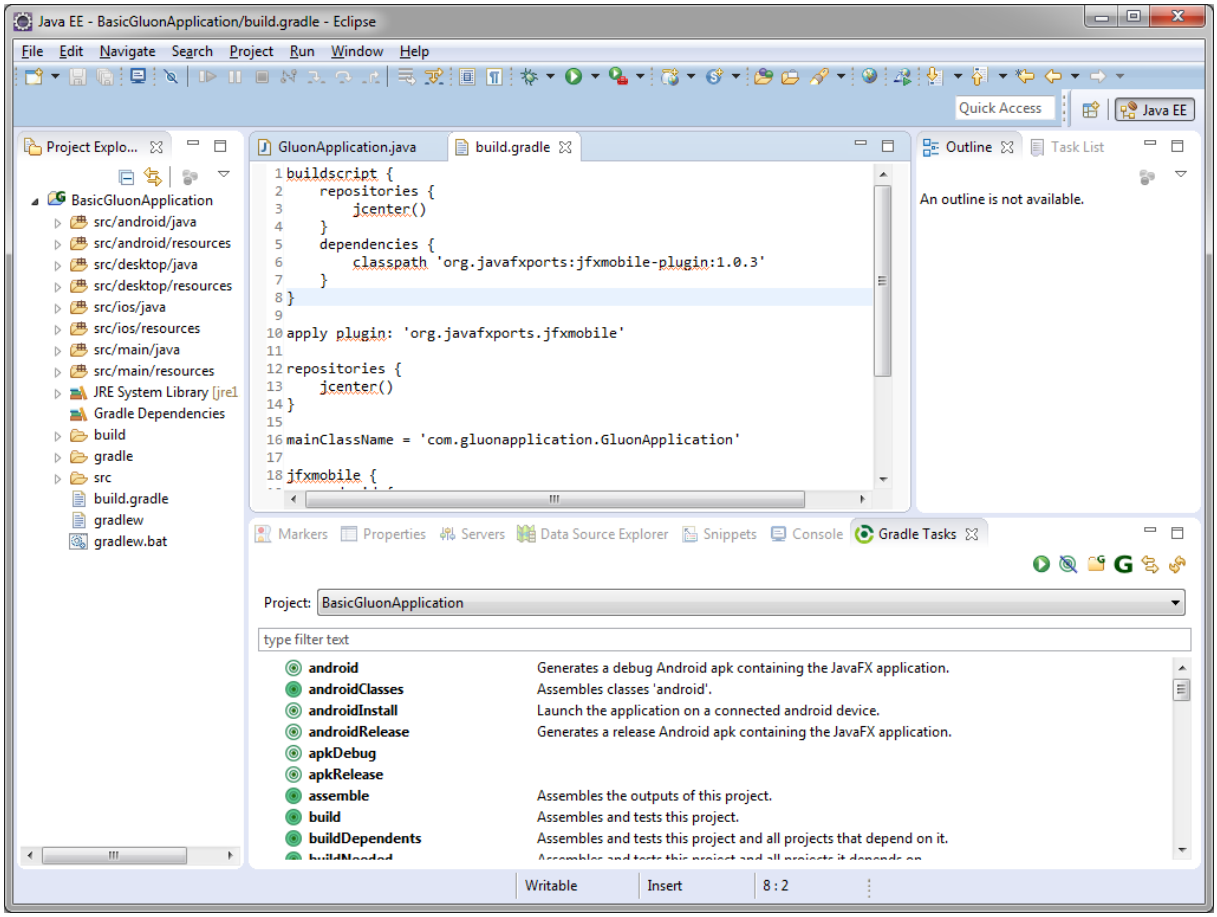

Abb. 1: Screenshot Java Source / Projektstruktur

Die Erklärung des Java Source Codes und der relevanten Klassen erfolgt Live im Vortrag.

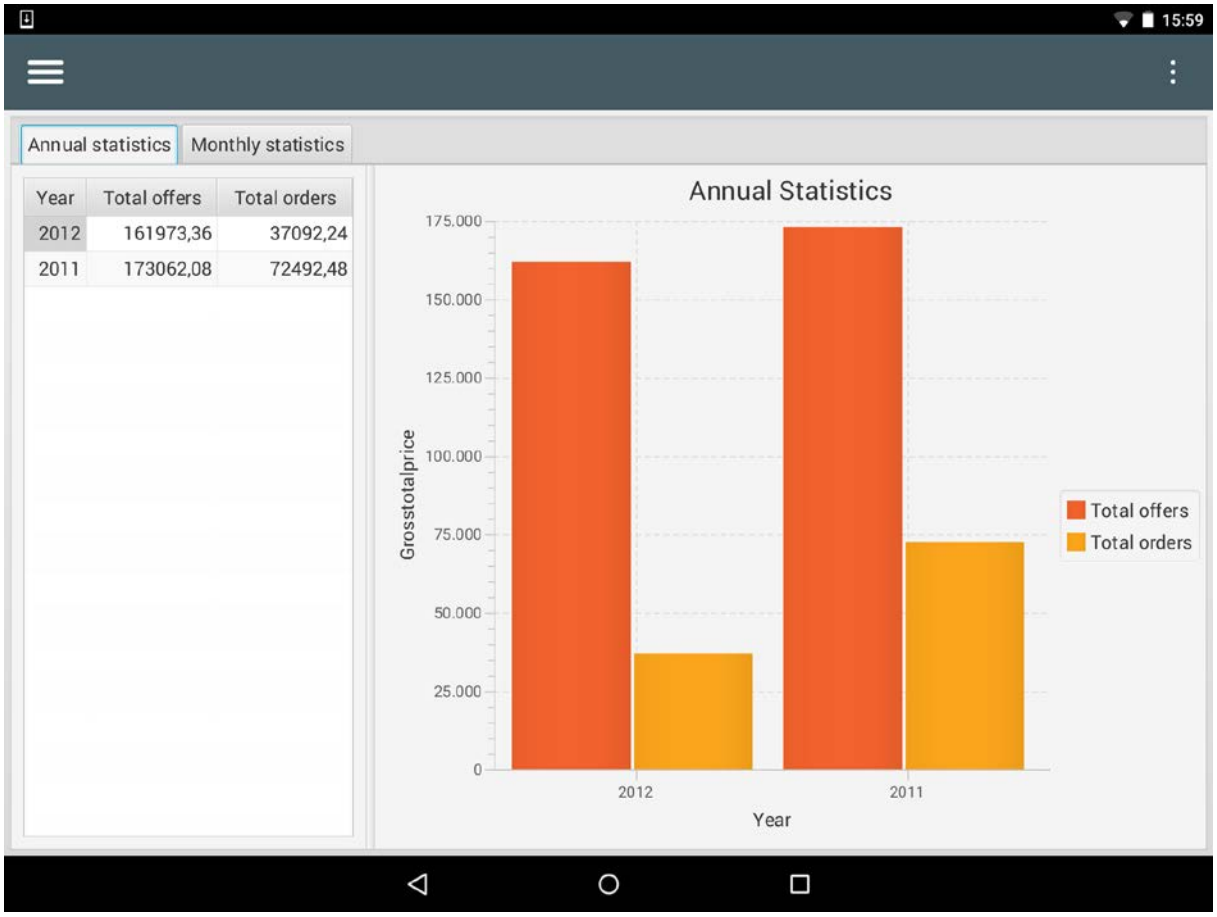

Abb. 2: Screenshot JavaFX Beispiel auf Android

Die Tücken der App Stores

Nachdem unsere Applikationen nun fertig ist, möchten wir Sie natürlich auch beispielsweise in den Google Play Store stellen. Dies sieht auf den ersten Blick einfacher aus als es tatsächlich ist. Das gradle plugin verwendet eine Konfigurationsdatei für die Erstellung der APKs oder IPKs. Für Android findet man die Einstellungen im AndroidManifest.xml. Dort ist für den Play Store die android:debuggable="false" Option auf false zu setzen, weil dies für öffentliche Apps nicht akzeptiert wird. Sinnvollerweise hinterlegt man auch mit android:icon="@mipmap/ic_launcher" das Verzeichnis wo sein App Icons in unterschiedlichen Auflösungen liegen. (mipmap-* ist eine Liste von Verzeichnissen mit den Images je Auflösung gemeint).

Jede Applikation muss signiert werden. Im Standard build wird ein Debug Zertifikat verwendet. Dieses müssen wir durch ein offizielles Zertifikat mit Privat und Public Key ersetzten. Am besten bereitet man mit den keytool ein entsprechendes Zertifikat unter einen Alias in einem keystore file vor. Auf dieses kann dann im build verwiesen werden.

```
jfxmobile {
    android {
        signingConfig {
            storeFile file("path/to/my-release-key.keystore")
            storePassword 'STORE_PASSWORD'
            keyAlias 'KEY_ALIAS'
            keyPassword 'KEY_PASSWORD'
        }
        manifest = 'lib/android/AndroidManifest.xml'
        resDirectory = 'src/android/resources'
```

```
    }
}
```
Unter iOS sind ähnliche Schritte notwendig, wobei die Zertifikate mit dem DevCenter erstellt werden müssen. Die Anleitung dafür findet sich im Dev Center (https://developer.apple.com/support/certificates/)

Desktop, Web und Mobile Business Applikationen entwerfen
Es ist möglich JavaFX Applikationen auf mobilen Geräten anzubieten, wer aber seine bestehende Desktop Applikation einfach 1:1 in den App Store stellt, wird bei den Benutzern auf kein positives Feedback stoßen.

Was ist also zu beachten?

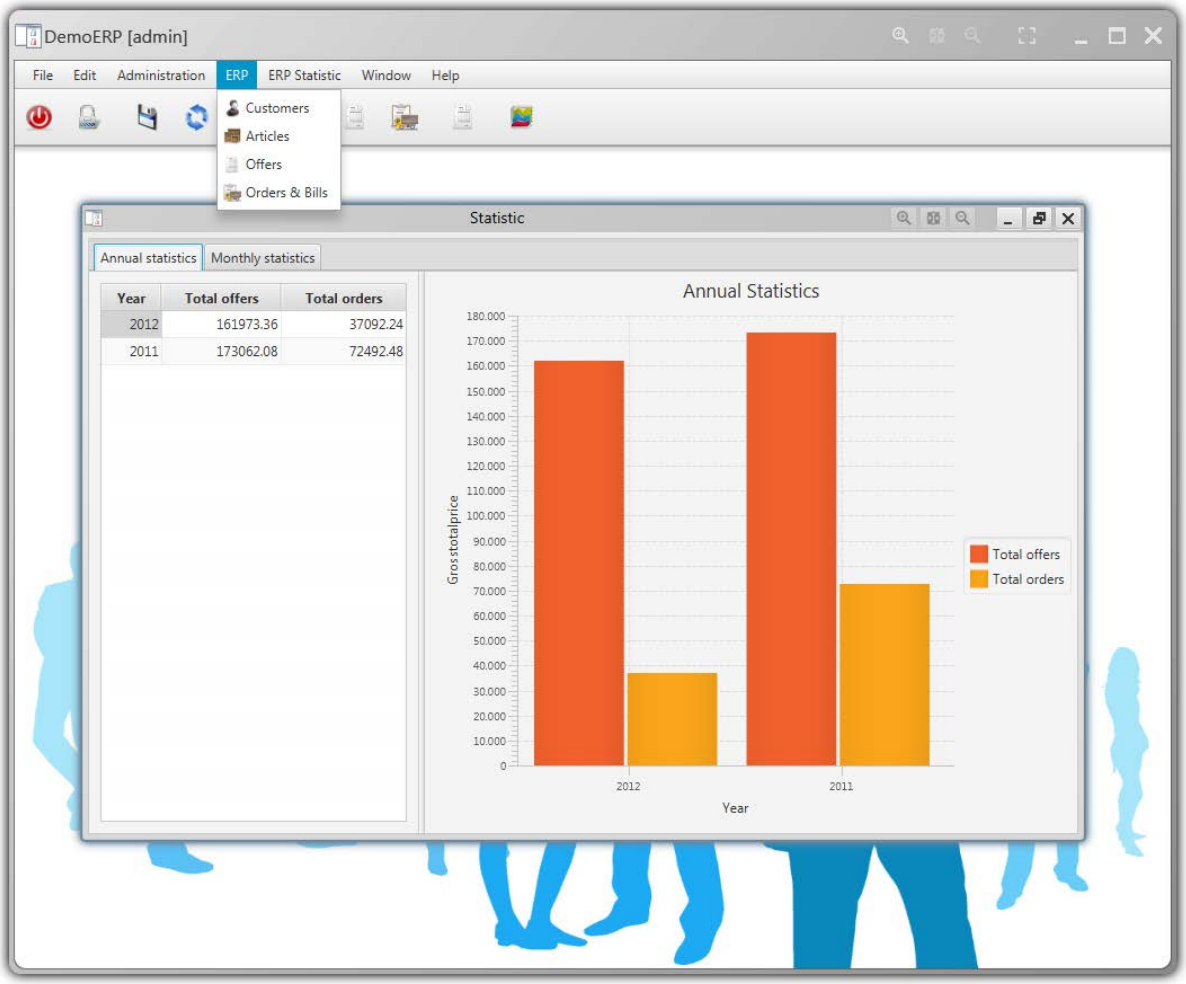

Abb. 2: Screenshot JavaFX Desktop Applikation

Typische Desktop Applikationen wurden gerne als MDI Applikationen umgesetzt. Die Funktionalitäten werden über Fenster zur Verfügung gestellt. Diese können dabei über ein Menü bzw. über eine Toolbar aufgerufen werden. Im Menü werden meist verschiedene Standardfunktionen, wie Hilfe, Login/Logout, Passwort ändern, Speichern/Verwerfen von Änderungen und vieles mehr angeboten.

Die Bedienung der Masken erfolgt am Desktop PC mit der Maus sowie mittels Tastatur. Die Erfassung von vielen Informationen ist per Tastatur schnell und einfach möglich. Auch die Navigation zwischen oft benötigen Funktionen erfolgt typischerweise über die Tastatur.

Wenn die Fenster nicht mit extrem vielen Feldern und Reitern überzogen sind, lassen sich solche Applikationen technisch 1:1 auch im Webbrowser mit HTML/CSS/Javascript lösen. Doch wenn dieselbe Maske auch auf einen Tablet über den Browser bedient werden soll, wird der Benutzer an der Bedienung der Fenster mittels Touchsteuerung scheitern. Typischerweise wird die Navigation von Business Webapplikationen über ein typisches Webmenü (links, oben oder rechts angeordnet) umgesetzt. Der Inhalt der Masken wird mittig im Browser angezeigt. Um nun mehrere Screens gleichzeitig zu öffnen, werden diese über das Menü meist mittels „im eigenen Tab öffnen“ Funktion aufgerufen. Damit erfolgt die Verwaltung von mehreren Screens über die Reiter des Browsers.

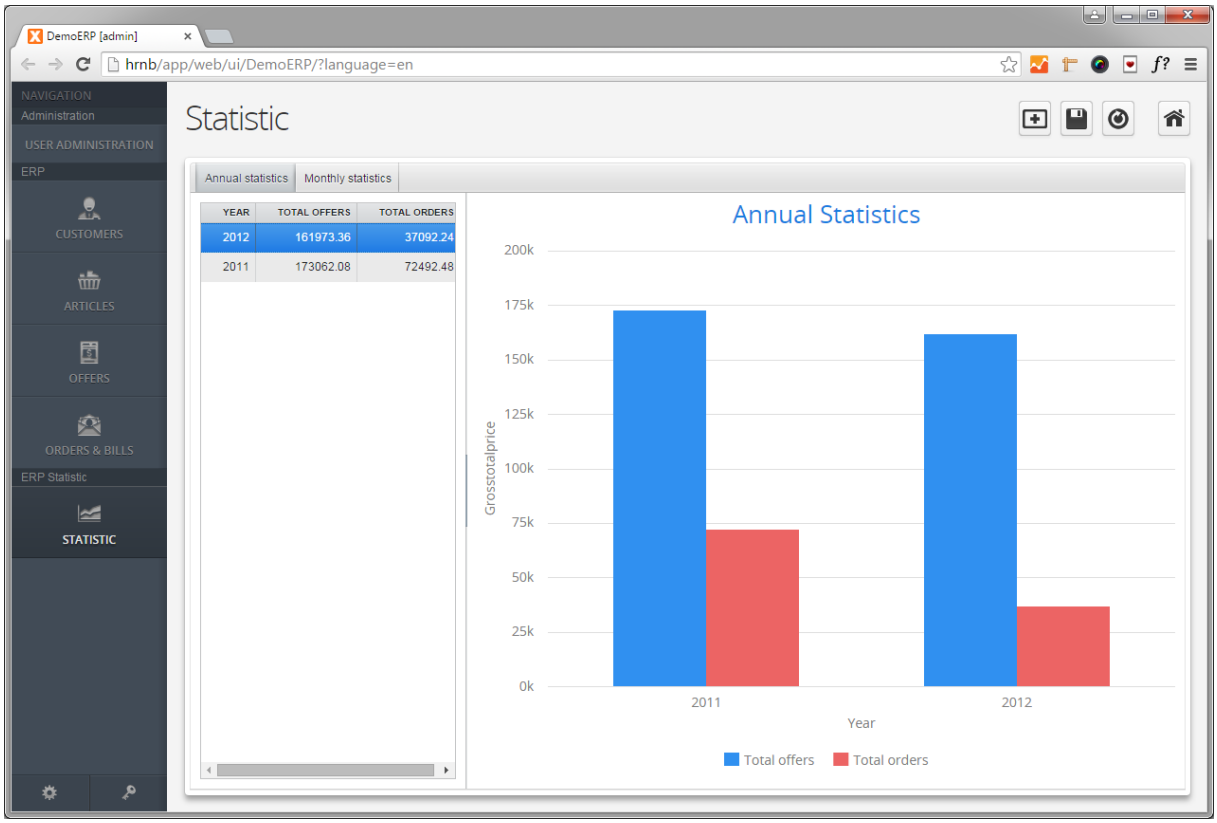

Abb. 2: Screenshot Web Applikation

Bei Web Applikationen werden die gleichzeitig sichtbaren Informationen reduziert. Klar weniger ist einfacher verständlich, lesbar und sieht kombiniert mit einen feinen Layout, Bildern und Fonts einfach besser aus. Weniger Informationen schafft man durch Drilldown. Daher sieht der Benutzer zuerst ein personalisiertes Dashboard, von dem er z.B. in einen tabellarischen Übersichts-/Filter-Screen gelangt. Von dort kann er dann meist zu weiteren Details oder zu verknüpften Informationen in weiteren Screens navigieren. Dies führt gegebenenfalls zu mehr Klicks als in einer Desktop Applikation um zu den gewünschten Informationen zu gelangen. Falls dies für bestimmte Anwendungsfälle relevant ist, können hier wieder überfüllter Legacy Masken umgesetzt werden oder es werden einfach mehrere speziell für die verschiedenen Anwendungsfälle konzipierte Masken umgesetzt.

Bei mobilen Applikationen ist die Bedienung über die Tastatur nur eingeschränkt möglich. Daher können realistisch trotz aller Eingabeunterstützungen nur wenige Informationen eingegeben werden. Daher muss die Benutzerführung mit wenigen Klicks oder Gesten am Touchscreens optimiert werden.

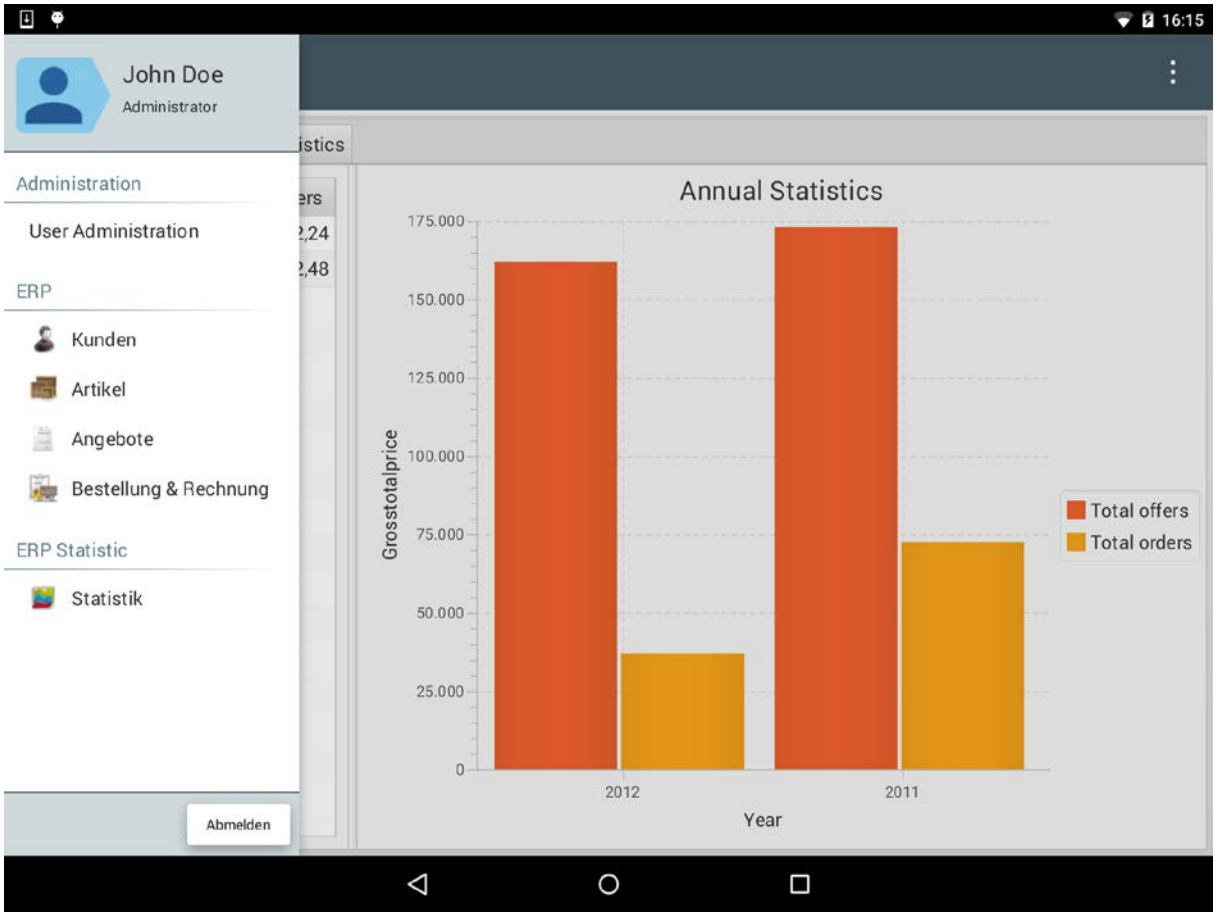

Abb. 2: Screenshot JavaFX Mobile Applikation - Menü

Auf eine Business Applikation umgelegt, bedeutet dies, dass nach dem erfolgreichen Login, das Menü der Applikation dargestellt wird. Z.B. Als gruppierte Liste der Screens
Nachdem der Benutzer die gewünschte Funktion gewählt hat, wird im ein Überblick zu dieser Funktion angezeigt z.B.: die Liste der Projekte usw. Jetzt muss er mit wenigen Klicks zu seinen Projekt kommen. Z.B.: Mit Klick auf das gewünschte Projekt oder durch Eingabe des Namens in einem Volltextfilterfeld um zum Projekt zu gelangen.

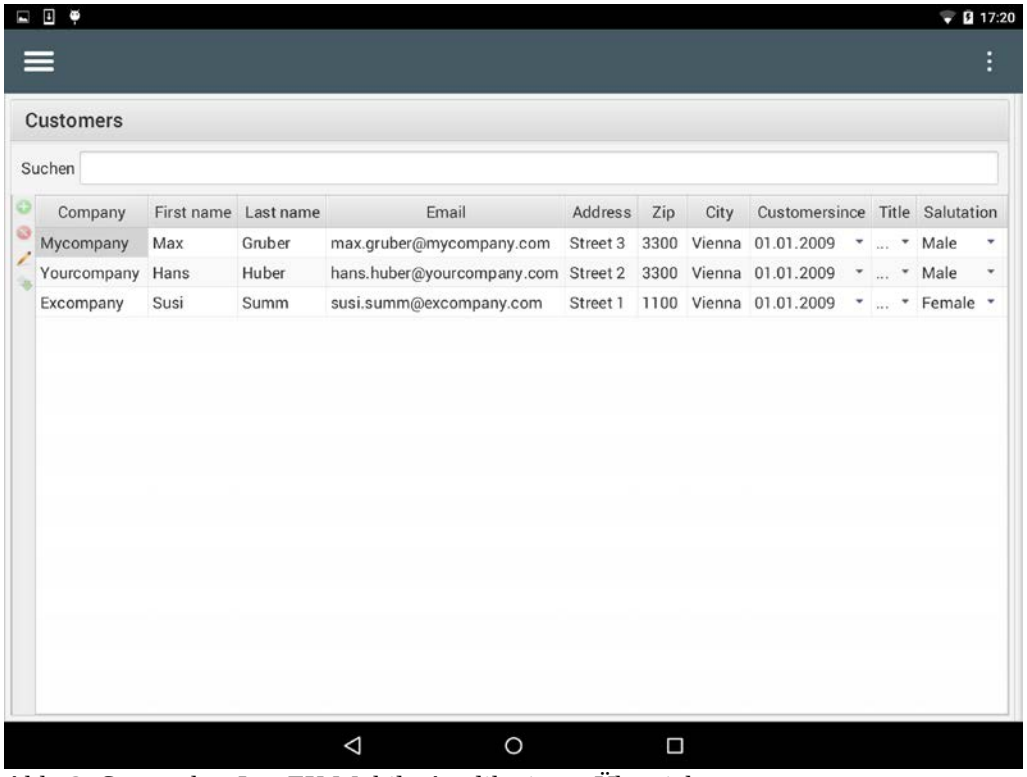

Abb. 2: Screenshot JavaFX Mobile Applikation – Übersicht

Danach kann er bei seinem Projekt ein paar Details verändern. Z.B. Ansprechpartner ändern, Todo hinzufügen, usw.

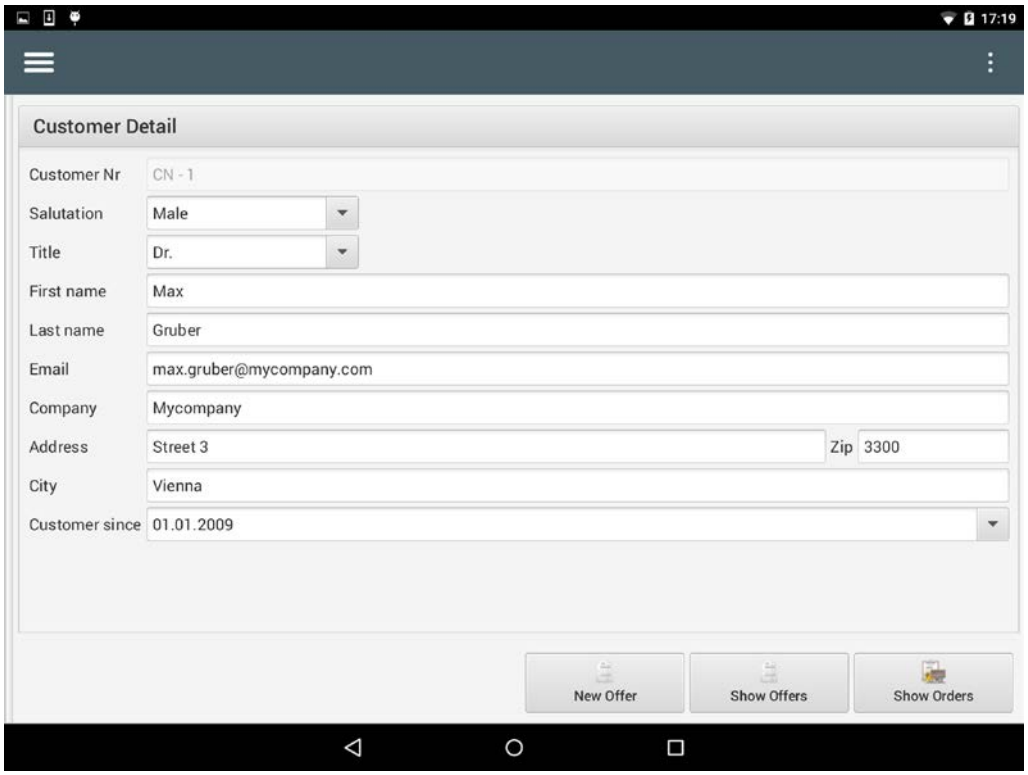

Abb. 2: Screenshot JavaFX Mobile Applikation – Details

In der mobilen Welt ist daher auch ein Drilldown der Weg um die Informationen pro Screen zu reduzieren, und so über wenige Klicks die gewünschte Funktion durchzuführen.

JVx - Ein Framework für alle UI Plattformen

Wenn man für Kunden nun für alle UI Plattformen Applikationen anbieten können muss, stellt sich die Frage wie man das mit der bestehenden Mannschaft umsetzen kann. Wenn Sie für jede UI Plattform eine unterschiedliches Framework und Entwicklungsumgebung einsetzen, ist sehr viel an Know How aufzubauen. Wenn Sie hier nur auf ein Open Source Framework zurückgreifen, können Sie den Komplexitätsgrad reduzieren und dies mit Ihrem bestehenden Team umsetzen. Ein anderer nicht zu vernachlässigender Aspekt ist, wird JavaFX wirklich die Zukunft? Oder ist es doch das Web Framework XY? Bei Einsatz von JVx können Sie den Wechsel jederzeit durchführen, das ist gerade im Legacy Umfeld relevant um eine Langzeitstabilität zu erreichen.

Kontaktadresse:
Roland Hörmann
SIB Visions GmbH
Wehlistr. 29 / Stiege 1 / 2.Stock
A-1200 Wien
Österreich

Telefon:	+43 1 934 6009 616	
Fax:	+43 1 934 6009 999	
E-Mail	roland.hoermann@sibvisions.com	
Internet:	www.sibvisions.comFax:	+49 (0) 12-345 6788
E-Mail	Ihre@adresse.de	
Internet:	www.adresse.de	