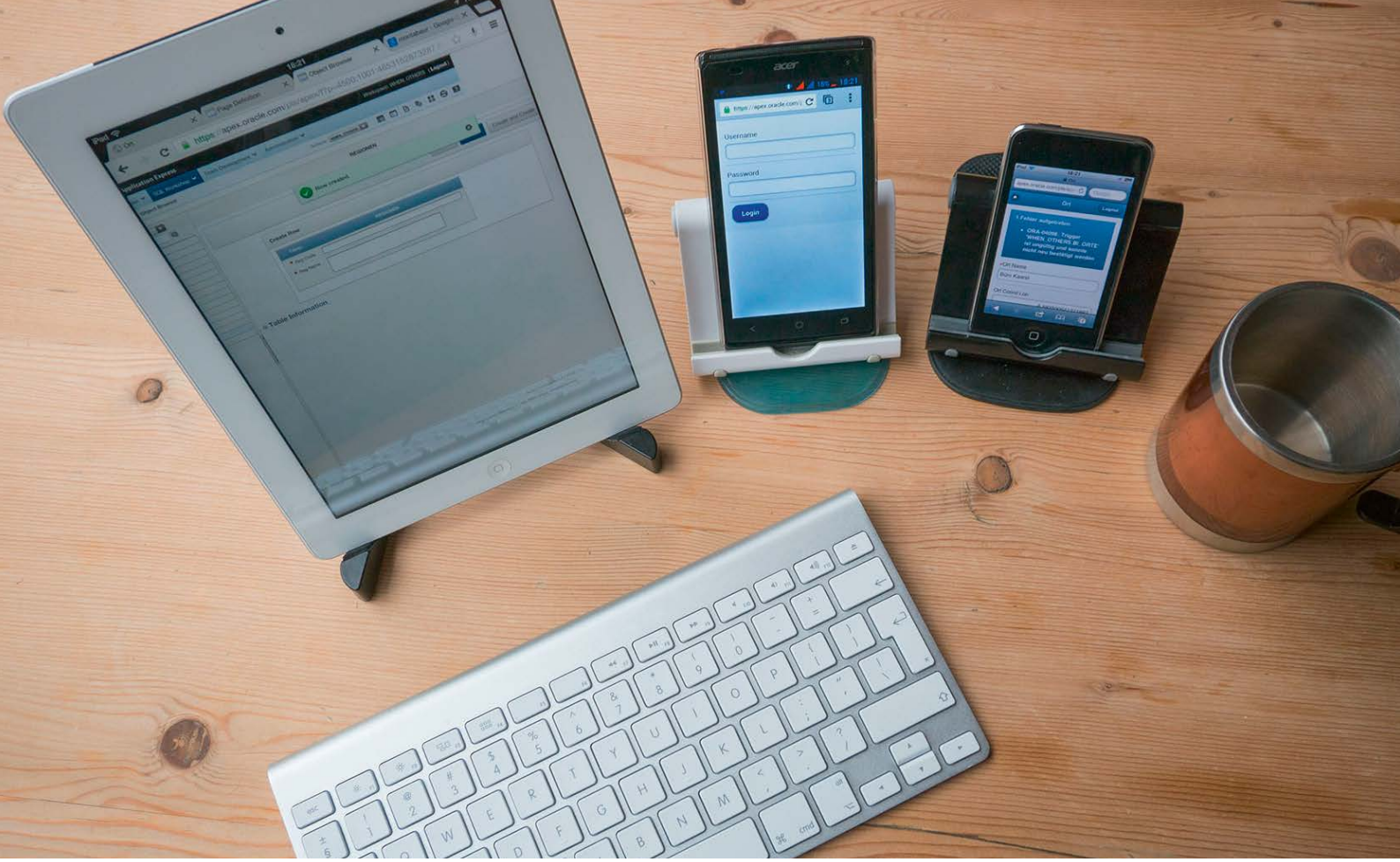

Buchführung mit Apex mobil erledigen

Andreas Wismann, WHENOTHERS

Desktop-Applikationen mit Oracle Application Express sind längst etablierter Standard in vielen Unternehmen. Die Zahl der Apex-Anwendungen, die speziell für Smartphones und Tablets programmiert werden, ist demgegenüber noch überschaubar – was aber nicht an der Qualität der Entwicklungsumgebung oder den erzielbaren Ergebnissen liegen dürfte, sondern an der (bis dato) vergleichsweise kleineren Zahl von Anwendungsfällen im Berufsleben.

Soweit der betriebliche Alltag im Unternehmen zum Großteil über die stationäre Arbeit mit dem PC definiert ist, stellt die Verfügbarkeit einer mobilen Applikation eher ein „Nice to have"-Feature als ein entscheidendes Erfolgskriterium dar. Nun gehören jedoch zur Tätigkeit des Autors als Apex-Freelancer typischerweise auch umfangreiche Fahrt- beziehungsweise Reise-Anteile, sein betrieblicher Alltag spielt sich also sehr wohl auch mobil ab – Reisezeit ist Arbeitszeit. Jede Minute, die er auf dem Weg zum Kunden und zurück beruflich nutzen kann, wirkt sich „1:1" positiv auf sein Freizeitkontingent aus. Genau deshalb begann er vor einigen Monaten, eine mobile Apex-Applikation zu konzipieren und exakt auf seine Bedürfnisse zuzuschneiden, mit der er seine tägliche Buchführung auch unterwegs erledigen kann. Der Artikel zeigt seine Erfahrungen während der Entwicklungs- und Nutzungszeit; gleichzeitig dient er als Anleitung zum Erstellen einer ersten eigenen mobilen Apex-App *(siehe Abbildung 1)*.

Von der Idee zum Entwurf

Die Vorgeschichte: Für seine Buchführung und Projektverwaltung hatte der Autor natürlich längst eine Apex-Applikation für den Desktop geschrieben, die auf einer (in der eigenen Cloud gehosteten) 11*g*-Express-Edition läuft und die er unterwegs auch „1:1" auf seinem iPad verwenden kann. Doch es stellte sich mit der Zeit heraus, dass ein Tablet viel zu groß ist, wenn er im überfüllten Regionalexpress nur einen Stehplatz erwischt hatte. Außerdem ist das Smartphone praktisch immer dabei, das iPad hingegen nicht unbedingt.

Die Idee hinter dieser mobilen HTML-App ist einfach: Sobald er unterwegs ist und mindestens zwei Minuten Zeit übrig sowie eine Hand frei hat, nimmt er sein Android-Smartphone zur Hand (alternativ einen iPod touch der 3. Generation – also noch nicht einmal ein ultramodernes Gerät) und arbeitet häppchenweise die Aufgaben ab, für die er sonst am Schreibtisch säße, um sie aus seinem althergebrachten, kugelschreibergeführten Projekt-No-

tizbuch in die Datenbank zu übertragen. Typische Einträge sind:

- Mit welchem Verkehrsmittel reise ich heute, wie weit, von wo nach wo?
- Woran arbeite ich und auf welches Teilprojekt buche ich den Aufwand?
- Welche neue Kontaktdaten gibt es zu übertragen?
- Mit wem habe ich gesprochen, worüber, wen muss ich noch kontaktieren?
- Welche Notizen möchte ich erfassen?
- Welche Ausgaben hatte ich?

Gemeinsamer Nenner all dieser Daten sind die vielfältigen Zuordnungsmöglichkeiten zwischen Orten, Personen, Projekten, Tätigkeiten, Notizen, Fahrtkilometern und sonstigen Einträgen.

Los geht's

Eine neue mobile Apex-Applikation zu erstellen, quasi nach dem „Hallo Welt"-Verfahren, ist seit der Version 4.2.x genauso einfach wie das Erzeugen einer Desktop-App und nimmt kaum länger als fünf Minuten in Anspruch. Zum unkomplizierten Ausprobieren eignet sich hervorragend die allseits bekannte Apex-Test-Plattform *(siehe http://apex.oracle.com)*.

Man wählt den „Create new application"-Assistenten und entscheidet sich bei der Frage nach dem User Interface einfach für „jQuery Mobile Smartphone" – im Prinzip ist es das schon. Bei den Themes ergibt sich übrigens keine Qual der Wahl wie bei den Desktop-Applikationen: Man nimmt das einzige für Smartphones verfügbare Theme mit der Nummer 50. Das scheinbare Fehlen weiterer Themes für mobile Anwendungen stellt kein Manko dar, im Gegenteil, denn jQuery-mobile-Anwendungen werden visuell bei Bedarf per CSS und JavaScript verändert, nicht durch neue Templates.

Damit die Freude über die neue Anwendung durch das wiederholte Auftauchen eines Login-Fensters im Smartphone-Browser nicht gebremst wird, empfiehlt es sich, für die ersten Gehversuche unter „Authentication Scheme" die Einstellung „No Authentication" zu wählen, somit kommt die Anwendung ohne Anmeldung aus. Man fügt nun der leeren Anwendung nach Belieben Reports, List Views oder Formulare hinzu und testet das Verhalten der App mit dem Smartphone, wozu lediglich die Apex-URL aus der Test-Umgebung im favorisierten Handy-Browser erforderlich ist. Tipp: Man erzeugt von dieser URL ein Verknüpfungs-Icon auf dem Home-Bildschirm, um jederzeit bequemen Zugriff auf die Anwendung zu haben.

Der entscheidende Unterschied

Neben den offensichtlichen Unterschieden wie Bildschirmgröße, Vorhandensein einer Maus (oder nicht) sowie unterschiedlich arbeitenden Browsern gibt es einen strategisch entscheidenden Vorteil von mobilen Apps gegenüber Desktop-Applikationen: Das Gerät weiß, wo es sich befindet, während es bedient wird. Die Verfügbarkeit von Geokoordinaten war seinerzeit ein treibender Faktor bei der Entwicklung aller Smartphone-Betriebssysteme. Der Browser hat diese Informationen zur Hand, sofern der Benutzer die Verwendung des Geolocation-Dienstes zugelassen hat. Es ist derselbe Dialog, den auch Applikationen wie Facebook, Twitter & Co. einblenden, wenn es darum geht, diese Informationen nutzen zu dürfen *(siehe Abbildung 2)*.

Der Browser stellt diese Frage genau dann, wenn die Web-Applikation zum ersten Mal auf diese Daten zugreifen möchte. Genauer gesagt erscheint diese Frage sogar doppelt:

1. Bei der allerersten Gelegenheit, wenn erstmals irgendeine Webseite diese Daten verwenden möchte – quasi als Türöffner für den Browser selbst
2. Erneut für jede weitere Webseite, die ihrerseits auf Geolocation-Daten zugreifen will

Diese Geodaten erleichtern die Dateneingabe in einer Projekt-Buchführungs-App enorm. Nahezu alle Tabellen (wie „ORTE", „PROJEKTE", „REISEKOSTEN", „TERMINE", „BUCHUNGEN") enthalten zwei zusätzliche Koordinatenfelder („GEO_COORD_LAT number" und „GEO_COORD_LON number") zum Abspeichern des Breiten- und Längengrads des Aufenthaltsorts, der zu diesem Datensatz gehört. Der Autor hat bewusst diesen Ansatz gewählt und nicht den Datentyp „SDO_GEOMETRY" aus Oracle Spatial, um unabhängig von der Datenbank-Edition zu sein. Genau diese beiden Zahlenwerte werden vom Browser ja schließlich auch geliefert. Eine Transformation in Spatial-Koordinaten ist jederzeit problemlos möglich.

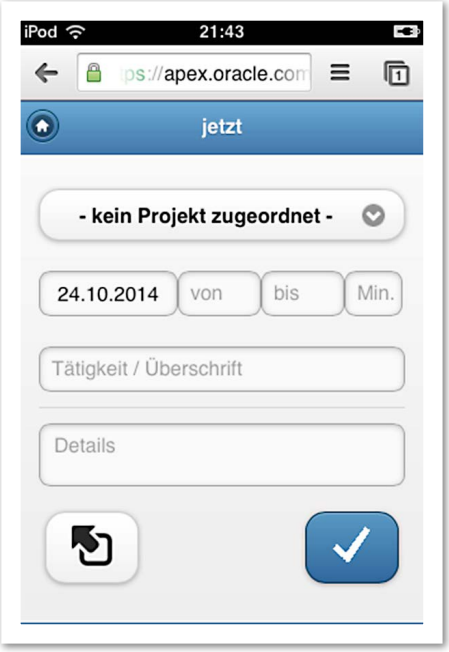

Abbildung1: Eingabemaske „Tätigkeitserfassung"

Abbildung 2: Freigabe des Geolocation-Dienstes für den Web-Browser

Diese Koordinaten sind so wertvoll, weil zu jedem Formular-Eintrag auch der Ort mitgespeichert wird, an dem die Daten-Eingabe erfolgt. Formulare können dadurch erkennen, zu welchem Projekt die Daten erfasst werden sollen. Zu allen Einträgen in der Tabelle „PROJEKTE" existieren nämlich Einsatzorte in der Tabelle

```
/* Diese Funktion unter „Execute When Page Loads“ aufrufen */
function storePosition () {
  navigator.geolocation.getCurrentPosition(
    setPosition, positionError);
}

/* Übergibt die Koordinaten an das Formular */
function setPosition (position) {
  $('input[id$="COORD_LAT"]').val(!position?
    '' : ('' + position.coords.latitude).replace('.',','));
  $('input[id$="COORD_LON"]').val(!position?
    '' : ('' + position.coords.longitude).replace('.',','));
}

/* Ermöglicht die Reaktion auf Fehler */
function positionError(error) {
  var geoMessage;
  switch(error.code) {
    case error.PERMISSION_DENIED:
      geoMessage = "Bitte die Ortungsdienste aktivieren"
      break;
    case error.POSITION_UNAVAILABLE:
      geoMessage = "Ortsdaten sind nicht verfügbar"
      break;
    case error.TIMEOUT:
      geoMessage = "Warten auf Ortsdaten abgebrochen"
      break;
    case error.UNKNOWN_ERROR:
      geoMessage = "Fehler beim Ermitteln der Ortsdaten"
  }
  if(!!geoMessage)
    // beispielsweise eine Benutzermeldung ausgeben:
    alert(geoMessage); // ... oder einfach ignorieren
```

Listing 1

Abbildung 3: Verdeckter Login-Button im Anmelde-Bildschirm, Gerät: iPod touch 3. Generation, Betriebssystem: IOS 4, Browser: Chrome)

„ORTE“, denen der Autor bereits bei ihrer Erfassung (oder notfalls später durch Nachschlagen in Google Maps) die richtigen Koordinaten zugeordnet hat. Wenn er sich beim Kunden im Westerwald aufhält, erscheinen in seiner mobilen Dateneingabe-Maske andere Default-Zuordnungen, als wenn er sich in Düsseldorf befindet. Tippt er beispielsweise im Bonner Hauptbahnhof auf „Bahnfahrt erfassen“, dann reicht genau dieser Fingertipp, um seinem Buchführungssystem mitzuteilen, dass er an diesem Morgen mit der Bahn zum Projekt-Einsatz gefahren ist, und damit natürlich auch, zu welchem. Tippt er tags darauf auf „Pkw erfassen“, dann ist genau dies die Grundlage für eine spätere Reisekosten-Kilometerbuchung. So bequem sollten mobile Apps immer funktionieren.

Geo-Koordinaten werden in HTML5-Browsern mittels JavaScript angefordert, zum Beispiel beim Öffnen einer HTML-Seite. Innerhalb weniger Sekunden stehen diese dann zur Verfügung (sofern dies gestattet worden ist) und können im Hintergrund als Werte von „Hidden Items“ gesetzt werden, die dann auf dem klassischen Weg (per Submit des Formulars) an den Server gesendet werden. Es muss allerdings sichergestellt sein, dass ein Formular auch ohne Positions-Informationen korrekt arbeitet, denn in Abhängigkeit von der Netzqualität und anderen Parametern stehen diese Daten nicht immer ad hoc zur Verfügung.

Der jQuery-Code in *Listing 1* kann im Seiten-Template untergebracht sein und übergibt dann bei jedem Seitenaufruf die momentanen Koordinaten an die beiden versteckten Formularfelder, deren Apex-Itemname jeweils mit „COORD_ALT“ beziehungsweise „COORD_LON“ endet.

Gerätepark und Browser-Vielfalt

Bei einer selbst erstellten Applikation zum Eigengebrauch kommen in diesem Fall als Abnehmer nur drei Geräte-Typen zum Einsatz: iPad, iPod und Android-Smart-

phone. Eine wesentlich größere Zahl an verschiedenen Testgeräten ist allerdings ein Muss, sobald man Anwendungen für mobile Endgeräte in größerem Stil entwickelt. Ein Beweis für diese Behauptung ist bereits der Login-Bildschirm seiner Anwendung, wenn der Autor sie mit dem iPod touch öffnet *(siehe Abbildung 3)*.

Die Login-Schaltfläche wird aufgrund der geringen Bildschirm-Abmessungen des iPod/iPhone-Bildschirms von der eingeblendeten Tastatur verdeckt. Hier muss mit CSS nachgearbeitet werden. Bei anderen Geräten tritt dieses Problem nicht auf, da dort mehr vertikale Bildschirmpixel zur Verfügung stehen. Der Safari-Browser im iPod verhält sich ebenfalls gutmütiger, weil er das Eingabefeld elegant nach oben schiebt, sobald die Tastatur-Oberkante dem Cursor zu nahe kommt. Bildschirm-Ausgaben und Formulare von mobilen Apps sollten also mit so vielen unterschiedlichen Geräte-, Betriebssystem- und Browser-Kombinationen wie möglich getestet werden. Man ist in jedem Fall gut beraten, die infrage kommenden Zusammenstellungen mit dem Kunden (falls möglich) im Voraus zu vereinbaren und dabei möglichst eng einzugrenzen *(siehe Abbildung 4)*.

Abschied von liebgewonnenen Gepflogenheiten

In einer Apex-Applikation, die der Autor für den Desktop-Einsatz entwickelt, verwendet er typischerweise Reports, Standard-Formulare und Tabular Forms. Diese Elemente sind für den Einsatz in mobilen Anwendungen nur bedingt geeignet. Interaktive Reports sind (aus gutem Grund) überhaupt nicht für jQuery-mobile-Anwendungen verfügbar: Auf dem kleinen Smartphone-Display wären sie aufgrund der Fülle von Optionen und der erforderlichen Darstellungsbreite schlicht und ergreifend unbedienbar. Das Apex-Entwicklerteam hat somit gut daran getan, die Entwickler hier gar nicht erst in Versuchung zu führen. Stattdessen ist (ausschließlich) im Mobile-Theme eine jQuery-Mobile-Komponente namens „List View“ verfügbar, die auf die speziellen Erfordernisse mobiler Anwendungen zugeschnitten ist *(siehe Abbildung 5)*.

Auf den erfahrenen Desktop-Entwickler, die seine erste mobile Anwendung entwickelt, wartet genau hier ein gewisser Kulturschock: Es ist nämlich unumgänglich, sich von breiten Listendarstellungen zu verabschieden, in denen der Benutzer am Full-HD-tauglichen Monitor nach Herzenslust scrollen kann (oder eher gesagt, muss …), um die gewünschte Information zu finden. Auf den winzigen Bildschirmen mobiler Endgeräte ist eben kein Platz für Überflüssiges. Das Maß aller Dinge sind kurze und knackige Überschriften sowie reduzierte Informationen, dafür große und gut bedienbare Schaltflächen und Navigationselemente. Wie im Screenshot der List View zu sehen ist, führen bereits mehr als 20 Zeichen dazu, dass der Listentext abgeschnitten wird – deshalb wird im Datumsfeld bereits auf die Jahresangabe verzichtet; nur das Allerwichtigste wird angezeigt. Auch Tabular Forms kann man getrost aus seinem mobilen Smartphone-Repertoire streichen, sofern man kein CSS-Guru ist, preisverdächtige Bedienkonzepte erfinden möchte oder die Benutzer der App überfordern will.

Mobile Dateneingabe-Masken erfordern pragmatische Überlegungen schon während der Planung. Bestimmte Anordnungen von Eingabe- und Bedien-Elementen werden vom Benutzer als positiv empfunden, andere eher nicht. So ist beispielsweise das vertikale Scrollen für die meisten User überhaupt kein Problem und wird auch intuitiv angewendet, horizontales Scrollen ist hingegen praktisch ein „No-Go“. Das liegt einerseits an den Gegebenheiten des Betriebssystems (seitliches Ziehen von Bildschirmen hat beispielsweise in mobilen Betriebssystemen von Microsoft eine komplett andere Bedeutung als unter Android oder IOS), andererseits sind es auch die anatomischen Gegebenheiten der menschlichen Hand, die beim Halten des Smartphones eher schlecht als recht eine Links-Rechts-Bewegung mit dem Daumen ausführen kann. Aus demselben Grund werden die prominenten Schaltflächen möglichst dem Aktionsradius des Daumens entsprechend angeordnet; in einer für Rechtshänder konzipierten App sollten sich also links oben oder links unten keine bedienungsintensiven Elemente befinden *(siehe Abbildung 6)*.

Der Autor bevorzugt grafische Schaltflächen anstatt textbasierter Buttons; gerade bei mobilen Applikationen lässt sich so die Bedienung recht intuitiv gestalten. Außerdem sind solche Icons bildschirmtechnisch günstiger zu platzieren, wenn sie quadratische Abmessungen haben und etwa die Größe eines Daumenabdrucks besitzen.

Abbildung 4: Der Home-Bildschirm der Anwendung. Die häufiger benutzten Schaltflächen für Dokumentation, Tätigkeitsnachweise, Memos und Reisekosten sind größer als die anderen Symbole

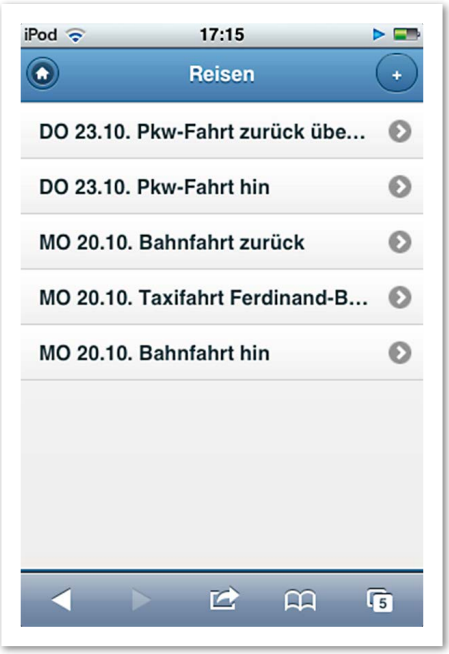

Abbildung 5: Die jQuery-mobile-Komponente „List View“

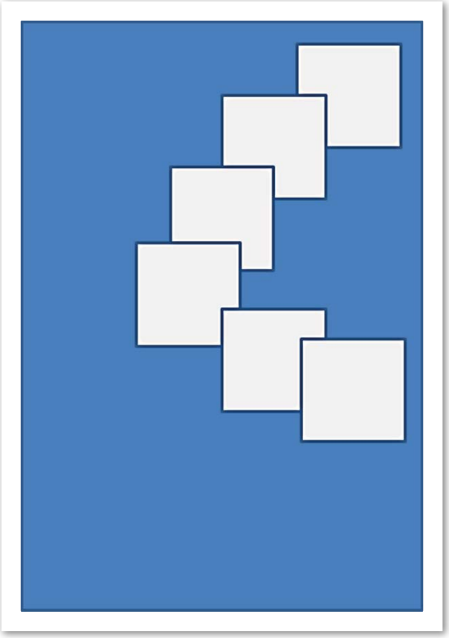

Abbildung 6: Diese Bildschirm-Koordinaten sind für die Platzierung von wichtigen Interaktionselementen bei Rechtshändern ideal

Abbildung 7: Die Eingabemaske zur Erfassung einer geschäftlichen Fahrt: wenig Text, große selbsterklärende Schaltflächen

Aus Platzgründen verzichtet er in Eingabemasken zudem auf Feldbeschriftungen („Label“) und verwendet stattdessen lieber Platzhalter-Text („Placeholder“), solange die Maske dabei verständlich bleibt. Der Grat zwischen Effizienz und zu knapp gehaltener Benutzerführung ist dabei allerdings schmal. Hier gilt es, in wichtigen Projekten die Meinung der Benutzer einzuholen und (im Idealfall) professionell durchgeführte Usability-Tests auszuwerten.

Ein weiterer Schritt in Richtung Benutzer-Akzeptanz stellen Abkürzungen dar. Es passieren häufig Schreibfehler beim Eintippen von Text auf der winzigen Bildschirmtastatur, deshalb verwendet der Autor gerne Abkürzungen (beispielsweise „d“ für „Düsseldorf“), wenn er in seinen Formularen Daten eingibt. Diese Kürzel werden beim Verlassen des Eingabefeldes durch Dynamic Actions aufgelöst. Es lohnt sich, mit den Benutzern solche Kürzel-Mappings zu erarbeiten, denn einmal gelernt, erfreuen sich diese Shortcuts noch größerer Beliebtheit als Dropdown-Listen, die einen Bedienschritt mehr erforderlich machen (Tippen zum Öffnen der Liste, Tippen zum Auswählen des Werts, *siehe Abbildung 7)*.

Für die häufigsten, immer wiederkehrenden Eingaben kann man sogar das Wischen (engl. „Swipe“) verwenden, das auf Desktop-Browsern gar nicht zur Verfügung steht. In den Dynamic Actions steht erfreulicherweise auch dieser Event-Typ zur Auswahl. So könnte etwa ein „Swipe Right“ in einem Zeit-Eingabefeld die aktuelle Uhrzeit eintragen, ein anderes Beispiel wäre der Heimatort des Anwenders beim Erfassen der täglichen Fahrtkilometer zum Arbeitsplatz.

Hand in Hand mit der Desktop-Anwendung

Eines der Formulare seiner mobilen App ermöglicht es dem Autor, Buchungslisten seines Girokontos abzuarbeiten, indem er nacheinander jede Zahlung dem passenden Buchhaltungskonto zuordnet. Alle unbearbeiteten Kontobewegungen befinden sich in einer Liste, deren Einträge er jeweils antippt, nach der Auswahl eines Kontos eventuell mit einem kurzen Buchungstext versieht und sogleich zum nächsten Listeneintrag weitergeleitet wird. Dadurch kann er diesen unbeliebten Teil der Buchhaltung sozusagen im Handumdrehen erledigen. Die Geschäftsvorfälle lädt er zuvor mit einer Apex-Desktop-Applikation per Excel-Import in die Anwendungstabellen, um die bearbeiteten Einträge am Monatsende ebenfalls in dieser Applikation zu überprüfen (schließlich arbeiten beide Applikationen auf exakt denselben Tabellen) und sie dann in sein Buchhaltungsprogramm zu exportieren.

Fazit

Um erfolgreich Anwendungen für mobile Geräte zu entwickeln, sind (mehr noch als im Desktop-Bereich) Kenntnisse in CSS und JavaScript beziehungsweise jQuery hilfreich, wenn auch nicht zwingend erforderlich. Der Apex Application Builder ermöglicht es einmal mehr, ohne HTML-Kenntnisse gebrauchsfertige Formulare auch für mobile Applikationen zu entwickeln. Insbesondere die „List View“-Komponente erfüllt in hervorragender Weise die Erfordernisse einer mobilen Anwendung, vor allem weil sie vielfältig konfigurierbar ist.

Serverseitig, also im Bereich der SQL- und PL/SQL-Programmierung, gibt es zwischen Desktop- und mobilen Apex-Applikationen nicht den geringsten Unterschied. Bieten Sie Ihren Anwendern durch geschickte Formularprogrammierung ein gewisses Maß an Komfort.

Gerade für die Bildschirmgestaltung mobiler Apps gilt: Weniger ist mehr, man sollte die Anwendung auf möglichst vielen Geräten testen und, so oft es geht, das Feedback der Benutzer einholen. Vielleicht der wichtigste Tipp: Eigene Programmier-Erfahrung mit Übungs-Apps sammeln, bevor man produktiv ins mobile Rennen geht, und diese Apps über einen längeren Zeitraum auch selbst verwenden.

Andreas Wismann
wismann@when-others.com