Kapitel 2
App-Typen

Von Projekt zu Projekt werden unterschiedliche Anforderungen an mobile Apps gestellt. Spätestens, wenn Sie eine Cross-Platform-Unterstützung benötigen, kann die Wahl des App-Typs signifikante Auswirkungen auf Kosten und Wartbarkeit haben.

Es gibt eine Vielzahl an Optionen bei der Erstellung mobiler Apps. Abhängig davon, welche Kenntnisse die Entwicklerteams besitzen, und welche Funktionalität benötigt wird, muss man sich für einen bestimmten App-Typ entscheiden. Doch nicht alle App-Typen werden von den SAP Mobile Services unterstützt. Und abhängig von der gewählten Option müssen unterschiedliche Werkzeuge bei der Entwicklung verwendet werden.

Native Apps

Der bekannteste Typ sind *native Apps*. Dabei handelt es sich um Apps, die gezielt in einer Programmiersprache entwickelt werden, die von dem mobilen Betriebssystem unterstützt wird, auf dem die fertige App laufen soll. Auf diese Art von Apps gehen wir in Abschnitt 2.1, »Native Apps«, ein.

Hybride Apps

In vielen Unternehmen sind Entwicklerteams mit Kenntnissen in der Webentwicklung im Einsatz. Das ist vor allem dann der Fall, wenn ein Großteil der internen Apps webbasiert ist. Diese Apps können von jedem Browser aus aufgerufen werden – egal, ob auf einem Windows-Betriebssystem oder auf macOS. Genau an dieser Stelle setzen *hybride Apps* an. Sie sind eine Mischung aus Web-App und nativer App. Hybride Apps laufen in einem nativen Container und ermöglichen es, auf viele Funktionen des darunterliegenden Betriebssystems zuzugreifen. In Abschnitt 2.2 führen wir Sie in die Entwicklung hybrider Apps auf Basis der SAP Mobile Services ein.

SAP-Fiori-Apps

Eine Vielzahl an SAP-Kunden hat *SAP-Fiori-Apps* und damit auch das *SAP Fiori Launchpad* im Einsatz. Viele SAP-Fiori-Apps werden von SAP bereits im Standard ausgeliefert, es gibt aber auch die Möglichkeit, eigene Apps zu entwickeln. Das SAP Fiori Launchpad ist der zentrale Einstiegspunkt für diese Apps und ermöglicht Ihnen den rollenbasierten Zugriff. Es eignet sich auch für den Einsatz auf mobilen Endgeräten.

SAP Fiori Client

Grundsätzlich kann das SAP Fiori Launchpad direkt aus dem Browser heraus gestartet werden, was jedoch hinsichtlich der Benutzerfreundlichkeit

und der Sicherheit zu wünschen übrig lässt. SAP bietet mit dem *SAP Fiori Client* jedoch die Möglichkeit, aus einer nativen App heraus auf das SAP Fiori Launchpad zuzugreifen. In Abschnitt 2.3 bringen wir Ihnen den SAP Fiori Client näher.

Mobile Development Kit

SAP bietet mit dem *Mobile Development Kit* noch eine weitere Möglichkeit der App-Entwicklung. Mit diesem SDK können metadatengetriebene Apps entwickelt und über die SAP Mobile Services bereitgestellt werden. Die Entwicklung erfolgt in JavaScript und ermöglicht unter anderem offlinefähige Apps, die eine native Benutzererfahrung bieten. Das Mobile Development Kit wird in Abschnitt 2.4 vorgestellt.

SAP Mobile Cards

Viele Nutzer*innen mobiler Endgeräte sind mit sogenannten *Wallet-Apps* vertraut. Dabei handelt es sich um Apps, die als Laufzeitcontainer dienen und einfach strukturierte Informationen wie Flug-, Hotel- oder Mietwagenbuchungen, aber beispielsweise auch Mitgliedskarten darstellen können. Diese Möglichkeit bietet sich in Form der *Mobile Cards* auch auf Basis der SAP Mobile Services. Die Entwicklung erfolgt unter Verwendung von Webtechnologien. Die als Laufzeitumgebung dienende App *SAP Mobile Cards* können Sie über die bekannten App Stores beziehen. In Abschnitt 2.5 bringen wir Ihnen die Konzepte von SAP Mobile Cards näher.

2.1 Native Apps

Native Apps ermöglichen es, auf alle vom Betriebssystem bereitgestellten APIs zuzugreifen. Mit diesen Apps kann man daher jegliche Funktion der mobilen Endgeräte verwenden. SAP legt seinen Fokus bei den nativen Apps aufgrund ihrer Marktdominanz auf die Betriebssysteme Apple iOS und Google Android.

Apple iOS

Apple unterstützt auf seinen iOS-Geräten die Programmiersprachen *Objective-C* und *Swift*. Objective-C war vor der Einführung von iOS bereits in der Mac-Entwicklung im Einsatz und war auf dem mobilen Betriebssystem iOS von Beginn an vorhanden. Im Laufe der Zeit und mit steigender Popularität der iOS-Plattform sind laufend neue Anforderungen an die Programmiersprache gestellt worden. Apple hat darauf mit einer neuen, modernen Programmiersprache namens *Swift* reagiert. Mittlerweile wird der Großteil der Apps in Swift entwickelt.

Als Entwicklungswerkzeug setzt Apple auf das hauseigene *Xcode*. Die iOS-Entwicklung mit Xcode hat gegenüber der Android-Entwicklung den Nachteil, dass Xcode nur auf dem Betriebssystem macOS ausgeführt werden kann. Das bedeutet, dass Sie Ihre Entwicklerteams zwingend mit einer

Apple-Infrastruktur ausstatten müssen. Als weiterer Nachteil hat sich erwiesen, dass ein Großteil der Entwickler*innen eine neue Programmiersprache lernen muss, da es sich weder bei Objective-C noch bei Swift außerhalb des Apple-Universums um gängige Programmiersprachen handelt. Für die Entwicklung nativer iOS-Apps bietet SAP mit dem *SAP BTP SDK for iOS* eine Ergänzung zum Swift SDK an, die wir Ihnen in Abschnitt 2.1.1 vorstellen.

Google Android

Die Entwicklung für das Betriebssystem Google Android findet in den Programmiersprachen *Java* und *Kotlin* statt. Java wurde von Beginn an von der Android-Plattform unterstützt.

In den ersten Jahren haben Entwickler*innen meist auf die populäre Eclipse IDE als Entwicklerwerkzeug gesetzt. Im Laufe der Zeit hat Google mit dem *Android Studio* eine auf der *IntelliJ IDE* basierende Entwicklungsumgebung bereitgestellt. Aufgrund der Tatsache, dass die Android-Entwicklung auf Java basiert ist, war die Plattform bei Entwickler*innen sehr beliebt. Eine Vielzahl an Entwickler*innen mit Java-Kenntnissen tummelt sich auf dem Markt, die sich lediglich mit den Android-APIs vertraut machen muss, um mit der Entwicklung von Apps starten zu können.

Google hat im Jahr 2017 mit Kotlin eine neue Programmiersprache für die Entwicklung auf der Android-Plattform vorgestellt. Kotlin ist eine moderne Programmiersprache, die viele Funktionen bietet, die in Java ebenso wie in anderen Programmiersprachen gefehlt haben. Sie tragen dazu bei, die Entwicklerproduktivität zu steigern. Für die Entwicklung nativer Android-Apps bietet SAP das *SAP BTP SDK for Android*, das wir Ihnen in Abschnitt 2.1.2 präsentieren.

Cross-Platform-Apps

Ein wesentlicher Nachteil der nativen Entwicklung sind die unterschiedlichen Programmiersprachen und Tools, die je nach Zielplattform erlernt und verwendet werden müssen. Möchten Sie als Anbieter*in einer App einen signifikanten Anteil der Endgeräte unterstützen, muss die App sowohl für Android als auch für iOS entwickelt werden. Die Erfahrung zeigt, dass dies in den meisten Fällen durch verschiedene Entwicklerteams erfolgt. Es gibt jedoch auch diverse Frameworks, die eine *Cross-Platform-Entwicklung* ermöglichen. Zu den populärsten Frameworks zählen *Xamarin*, *Flutter* und *React Native*. Hier entwickeln Sie abhängig vom Framework in einer eigenen Programmiersprache (z. B. *Dart* für Flutter), und der Quellcode wird über einen sogenannten *Transpiler* in die jeweilige native Sprache übersetzt. Als Ergebnis haben Sie eine native App. Die SAP Mobile Services können bei diesem Entwicklungsansatz lediglich über die REST API angebun-

den werden. Weder das SAP BTP SDK for iOS noch das SAP BTP SDK for Android können für die Cross-Platform-Entwicklung verwendet werden. SAP positioniert stattdessen das MDK für die Entwicklung von Cross-Platform-Apps. Aus diesem Grund wird in diesem Buch nicht näher auf die Entwicklung mit anderen Frameworks eingegangen.

SAP Fiori

Das SAP BTP SDK for iOS, das SAP BTP SDK for Android und hybride SAP-Apps haben eine Gemeinsamkeit: Sie basieren auf SAP Fiori. SAP Fiori ist eine Designsprache, die es ermöglicht, Unternehmens-Apps mit einer hervorragenden Benutzererfahrung zu erstellen. Mit SAP Fiori fand ein Paradigmenwechsel in der SAP-Entwicklung statt, weg von monolithischen Anwendungen hin zu leichtgewichtigen Apps, die auf die Rollen der Nutzer*innen zugeschnitten sind.

Designrichtlinien

SAP-Fiori-Apps halten sich auf allen Zielplattformen an die folgenden Designrichtlinien:

- **Sie sind rollenbasiert**
  Die Apps stellen die richtigen Informationen den richten Personen zur richtigen Zeit und am richtigen Ort zur Verfügung.
- **Sie sind adaptiv**
  Die Apps ermöglichen es den Nutzer*innen, dort zu arbeiten, wo sie wollen, und auf dem Gerät ihrer Wahl.
- **Sie sind kohärent**
  SAP Fiori bietet in allen Unternehmens-Apps die gleiche intuitive und konsistente Benutzererfahrung.
- **Sie sind einfach**
  SAP Fiori unterstützt die Nutzer*innen dabei, sich auf das Wesentliche zu konzentrieren.
- **Sie sind reizvoll**
  Das SAP-Fiori-Design bereichert die Arbeitserfahrung der Nutzer*innen.

Zusammengefasst setzt SAP Fiori einen neuen Standard für Benutzerfreundlichkeit in Unternehmens-Apps, indem unnötige Komplexität beseitigt wird. Nachdem sich SAP Fiori bei den SAP-Kunden großer Beliebtheit erfreute, hat SAP im Jahr 2016 bekannt gegeben, zukünftig ein eigenes SDK für die Entwicklung nativer iOS-Apps bereitzustellen. Das Ergebnis ist das SAP BTP SDK for iOS. Im Jahr 2018 wurde auch ein SDK für die Entwicklung nativer Android-Apps veröffentlicht, das SAP BTP SDK for Android. Ziel beider SDKs ist es, den Entwicklungsprozess zu beschleunigen und dafür zu sorgen, dass einheitliche Empfehlungen für die Entwicklung nativer Apps

gegeben werden. Zusätzlich können Sie als Entwickler*in von Funktionalität profitieren, um Anforderungen typischer Unternehmens-Apps wie sichere Kommunikation, Authentifizierung, Onboarding und Offlinedatenhaltung umzusetzen. Die SAP Mobile Services spielen dabei eine zentrale Rolle.

2.1.1 SAP BTP SDK for iOS

2016 ging SAP eine strategische Partnerschaft mit Apple ein. Mit dem iPhone und dem iPad hatte Apple sich auch das Unternehmensumfeld erobert. Das SAP BTP SDK for iOS unterstützt Sie bei der effizienten Entwicklung von Apps, basierend auf der Swift-Programmiersprache. Das SDK ist kein Ersatz für das *Swift SDK*, sondern ergänzt dieses lediglich.

Komponenten

Das SAP BTP SDK for iOS setzt sich aus den folgenden Komponenten zusammen:

- **SDK**
  Eine Sammlung an Bibliotheken, die von Entwickler*innen in den iOS-Apps verwendet werden und SAP-spezifische Funktionen abdecken.
- **Assistant**
  Der *Assistant* ist eine App für macOS, die es ermöglicht, Xcode-Projekte zu erzeugen, die alle grundlegenden Funktionen für das Zusammenspiel mit den SAP Mobile Services beinhalten.
- **SAP Fiori Mentor**
  SAP Fiori Mentor ist eine iOS-App, die den Entwicklungsprozess vereinfacht. In dieser App können die verschiedenen *Controls*, d. h. die verschiedenen Elemente der mobilen Benutzeroberflächen, getestet und angepasst werden. Der Swift-Quellcode des angepassten Controls kann in weiterer Folge mittels *AirDrop* direkt auf den Mac des Entwicklers oder der Entwicklerin und von dort in das Xcode-Projekt übernommen werden. AirDrop ist eine Technologie von Apple, die es ermöglicht, Daten zwischen sich vertrauenden Geräten drahtlos mittels WiFi oder Bluetooth zu übertragen.
- **SAP Fiori for iOS Design Guidelines**
  Es wird eine einheitliche Designsprache bereitgestellt, die es den Entwickler*innen ermöglicht, optisch ansprechende native mobile Apps für iOS zu entwickeln. Die aktuelle Version der *SAP Fiori for iOS Design Guidelines* finden Sie unter der URL *https://experience.sap.com/fiori-design-ios/*.

Adaptive Design

Ein für Entwickler*innen sehr interessanter Aspekt ist das adaptive Design. Dadurch werden die Inhalte der App dynamisch neu angeordnet und hinsichtlich ihrer Größe angepasst, wenn die App auf Geräten mit unterschiedlichen Formfaktoren läuft oder sich die Ausrichtung des Geräts ändert. Dazu werden die beiden Größenklassen regulär (*Regular*) und kompakt (*Compact*) definiert. Der *Formfaktor* spiegelt die Größe und Form sowie die Anordnung der Kernkomponenten wider, wie beispielsweise der Bedienknöpfe auf den verschiedenen Gerätetypen. Die reguläre Größenklasse bezieht sich typischerweise auf das iPad, auf dem viel Platz zur Verfügung steht. Die kompakte Größenklasse hingegen bezieht sich auf das iPhone, auf dem Inhalte in einer komprimierten, minimalistischen Form dargestellt werden. In Abbildung 2.1 wird eine App im regulären Design auf einem iPad dargestellt.

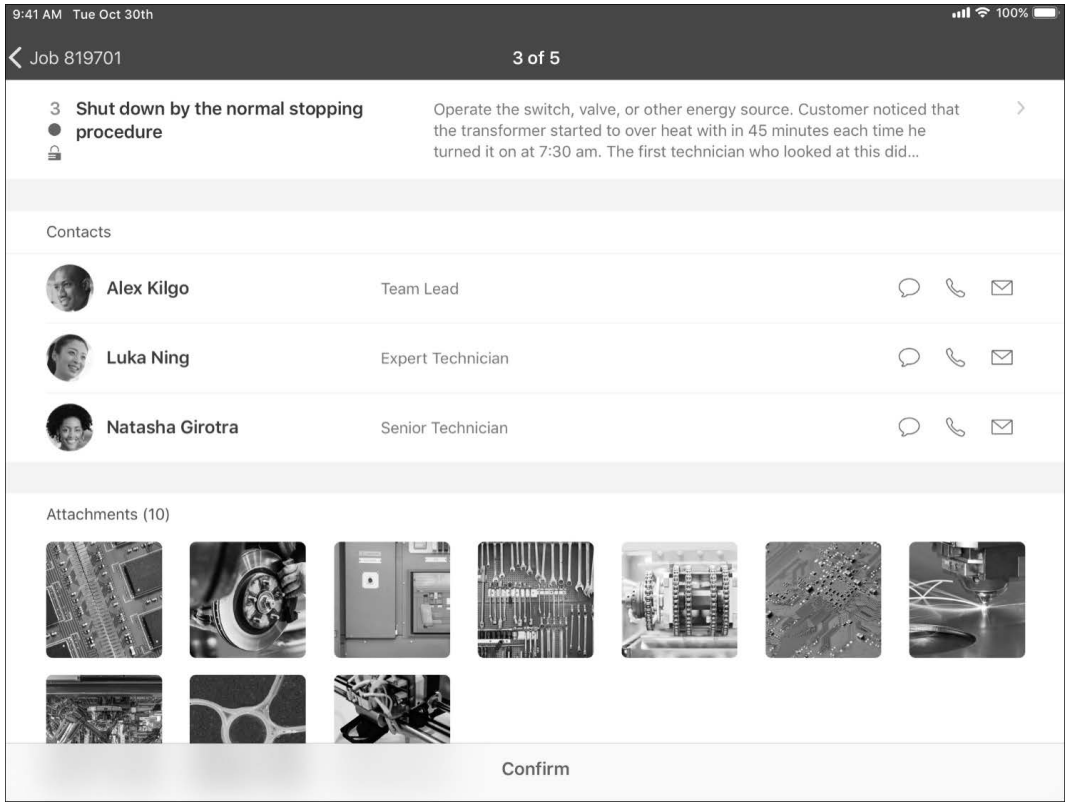

Abbildung 2.1 SAP BTP SDK for iOS, reguläres Design
(Quelle: SAP Fiori Mentor App)

In Abbildung 2.2 wird dieselbe App auf einem iPhone im kompakten Design dargestellt. Sie werden erkennen, dass bestimmte Informationen ausgeblendet wurden, um die Benutzeroberfläche nicht zu überladen und um eine bessere Benutzererfahrung zu ermöglichen.

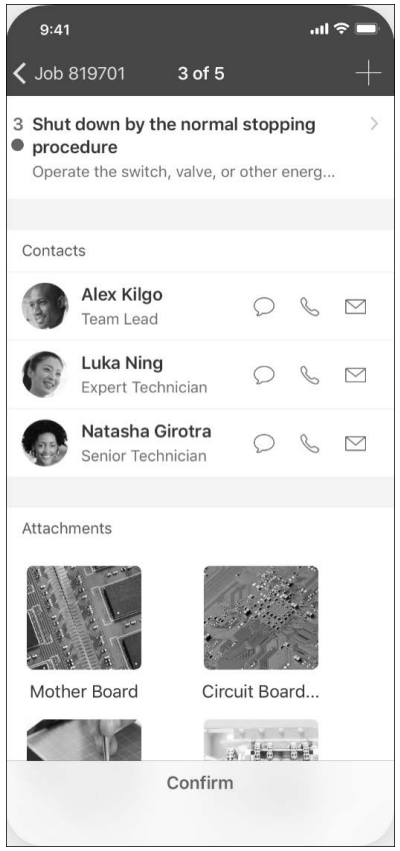

Abbildung 2.2 SAP BTP SDK for iOS, kompaktes Design
(Quelle: SAP Fiori Mentor App)

Frameworks

Das SDK beinhaltet Komponenten, die abhängig vom Anwendungszweck in Frameworks verpackt sind. Sie können diese Frameworks verwenden, um Apps vollumfänglich zu entwickeln. Folgende Frameworks sind im SDK enthalten:

- **SAPFoundation**
  Das `SAPFoundation`-Framework beinhaltet Allzweckkomponenten, die sich einerseits nahtlos in das Swift SDK integrieren und andererseits die Integration in die SAP Mobile Services ermöglichen. Dadurch wird sichergestellt, dass Ihnen Funktionalität wie Sicherheit, Authentifizierung, Konnektivität zu Backend-Systemen und Datenpufferung direkt zur Verwendung in Ihren eigenen Apps zur Verfügung steht.

- **SAPCommons**
  Das `SAPCommons`-Framework stellt grundlegende Funktionen innerhalb der App bereit, wie Logging und Nutzungs-Reporting. Diese Funktionen sind nahtlos in die SAP Mobile Services integriert. Sie können somit in den SAP Mobile Services auf die Logs und Nutzungs-Reports Ihrer App zugreifen.
- **SAPOData**
  Das `SAPOData`-Framework ermöglicht Ihnen den Onlinezugriff auf Backend-Systeme mittels OData. Dieses Framework sollten Sie verwenden, falls Sie aktuelle Daten aus dem Backend-System benötigen und eine stabile Internetverbindung besteht. Lokale Datenspeicherung und Pufferung werden von diesem Framework nicht unterstützt.
- **SAPOfflineOData**
  Das `SAPOfflineOData`-Framework bieten Ihnen die Möglichkeit, offlinefähige Apps zu entwickeln. Die Daten werden mittels OData über eine Onlineverbindung aus dem Backend-System geladen und in einen Offline-OData-Store geschrieben. In der Folge führt die App alle Operationen gegen den Offline-Store durch, wodurch Sie die App über einen längeren Zeitraum hinweg ohne Internetverbindung verwenden können. Sobald der App wieder eine Internetverbindung zur Verfügung steht, können die Änderungen an das Backend-System übermittelt werden.
- **SAPFiori**
  Das `SAPFiori`-Framework beinhaltet Komponenten, die von den Nutzer*innen der App zur Interaktion verwendet werden. Dieses Framework stellt Ihnen die zuvor erwähnte SAP-Fiori-for-iOS-Designsprache zur Verfügung und ermöglicht es Ihnen damit, den Anwender*innen eine sehr gute Benutzererfahrung zu bieten.

Floorplans

Das SDK nimmt Ihnen auch an einer anderen Stelle einiges an Arbeit ab: SAP definiert sogenannte *Floorplans*, d. h. unterschiedliche Layouttypen, in denen die verwendeten Steuerelemente jeweils in einer bestimmten, auf den Anwendungsfall angepassten Struktur angeordnet werden. Die Floorplans definieren auch, wie mit bestimmten App-Typen, z. B. analytischen Apps oder Report-Ergebnislisten, umzugehen ist und wie die Navigation aufgebaut sein sollte.

Mac Catalyst

Mit Version 5.0 des SDK wurde mit der Unterstützung von *Mac Catalyst* eine weitere sehr spannende Funktionalität eingeführt. Sie können Mac Catalyst verwenden, um Ihren iPhone- und iPad-Apps eine Mac-Version bereitzustellen. Sie haben damit für iOS und macOS einen gemeinsamen Quellcode.

Um Mac Catalyst zu verwenden, müssen Sie bei der Erstellung des Projekts mithilfe des Assistants lediglich das Kennzeichen für die Aktivierung von Mac Catalyst setzen. Sie können die Mac-Catalyst-Unterstützung auch nachträglich in bereits bestehenden Apps nutzen.

In Kapitel 4, »Entwicklung einer nativen App mit dem SAP BTP SDK for iOS«, führen wir Sie anhand eines Beispiels in die Entwicklung mit dem SDK ein. Wir zeigen Ihnen, wie Sie eine App sowohl mithilfe des Assistants als auch direkt in Xcode erstellen und die SAP-Frameworks einbinden können.

2.1.2 SAP BTP SDK for Android

Als sich neben dem iOS-Betriebssystem auch Android im Unternehmensumfeld etablierte, stieg auch die Nachfrage nach Apps mit einer nativen SAP Fiori UX auf Android. 2018 veröffentlichte SAP die erste Version des SAP BTP SDK for Android (damals unter dem Namen *SAP Cloud Platform SDK for Android*). Das SDK wurde in Java entwickelt, kann aber sowohl für Java-basierte als auch für Kotlin-basierte Apps verwendet werden. Es stellt keinen Ersatz für das allgemeine Android SDK dar, sondern ergänzt bzw. erweitert dieses lediglich.

Komponenten

Das SAP BTP SDK for Android setzt sich aus den folgenden Komponenten zusammen:

- **SDK**
  Das SDK stellt eine Sammlung an Bibliotheken bereit, die von Entwicklerteams in Android-Apps verwendet werden können und die SAP-spezifische Funktionen abdecken.
- **Wizard**
  Der *Wizard* ist ein Plug-in für das Android Studio, das die Entwicklung vereinfacht, indem Quellcode generiert wird, der bereits alle grundlegenden Funktionen für das Zusammenspiel mit den SAP Mobile Services beinhaltet.
- **SAP Fiori Mentor**
  Die App SAP Fiori Mentor, die den Entwicklungsprozess vereinfacht, gibt es auch als Android-App. In der App können die verschiedenen Controls getestet und angepasst werden. Der Java- oder Kotlin-Quellcode des angepassten Controls sowie die zugehörigen XML-Daten zur Erstellung der Views können beispielsweise über AirDrop auf einen Mac übernommen und von dort in ein Android-Studio-Projekt integriert werden. Die Entwicklung direkt auf den mobilen Geräten ist aktuell noch nicht möglich.

- **SAP Fiori for Android Design Guidelines**
  Auch für Android gibt es eine einheitliche Designsprache, die es Entwickler*innen ermöglicht, optisch ansprechende, native mobile Apps im Stil von SAP Fiori zu entwickeln. Die aktuelle Version der Design Guidelines finden Sie unter: *https://experience.sap.com/fiori-design-android/*

Adaptives Design

Das adaptive Design wird auch im SDK for Android bereitgestellt. Dadurch werden die Inhalte der App dynamisch neu angeordnet und hinsichtlich ihrer Größe angepasst, wenn die App auf Geräten mit unterschiedlichen Formfaktoren läuft oder sich die Ausrichtung des Geräts ändert. Dazu werden auch hier die beiden Größenklassen regulär (Regular) und kompakt (Compact) definiert. Die reguläre Größenklasse bezieht sich, unabhängig vom Hersteller, typischerweise auf Tablets, auf denen viel Platz zur Verfügung steht. Die kompakte Größenklasse hingegen bezieht sich, ebenfalls unabhängig vom Gerätehersteller, auf Smartphones, auf denen Inhalte in einer komprimierten, minimalistischen Form dargestellt werden.

2.2 Hybride Apps

Die Entwicklung hybrider Apps ist aus der Anforderung entstanden, eine gemeinsame Codebasis für alle Betriebssysteme verwenden zu können. Die Entwicklung solcher Apps erfolgt mit den Webtechnologien HTML5, JavaScript und CSS.

Funktionsweise

Grundsätzlich ist eine hybride App eine Web-App, die auf native Funktionen des mobilen Endgeräts zugreifen kann. Die Idee dahinter ist einfach, aber genial: Die native App, der sogenannte *App-Container*, stellt einen Webbrowser bereit, in dem die Web-App ausgeführt wird. Eigentlich verhält sich die App also wie jeder beliebige Webbrowser, der auf dem mobilen Endgerät zur Verfügung steht. Die App greift hart codiert auf die bestimmte Web-App zu. Somit entfällt die Notwendigkeit, die Adressleiste des Browsers zu füllen und damit auch eine potenzielle Fehlerquelle. Im Bereich hybrider Apps hat sich das Framework *Apache Cordova* (*https://cordova.apache.org/*) als Standard etabliert.

Plug-ins

Der Zugriff auf native Funktionen wird über *Plug-ins* bereitgestellt. Ein solches Plug-in wird nativ für das jeweilige Betriebssystem entwickelt und stellt der Web-App über eine *JavaScript Bridge*, d. h. eine JavaScript-API, bereit. Aus der Web-App heraus können Sie diese API direkt in JavaScript konsumieren. Beim Erstellen der Container-App werden die verwendeten Plug-ins mit eingebunden. Der *WebView* ist eine Steuerelement, das betriebssystemspezifisch einen Browser bereitstellt, der in Apps integriert werden

kann. Die Architektur hybrider Apps wird in Abbildung 2.3 grafisch dargestellt.

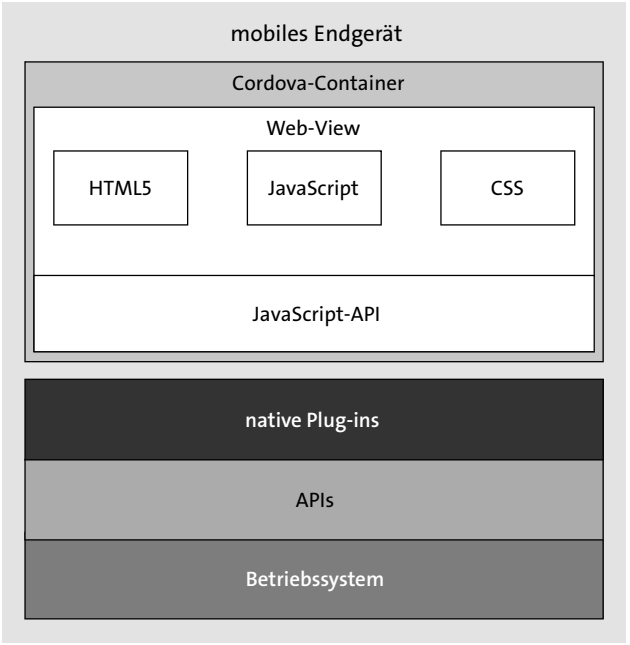

Abbildung 2.3 Architektur hybrider Apps

Hybrid SDK

SAP liefert in Kombination mit den SAP Mobile Services das *Hybrid SDK* aus, das auch unter dem Namen Kapsel bekannt ist. Als Basis verwendet SAP den App-Container Apache Cordova und liefert eigene Plug-ins aus. SAP verwendet den Cordova-Container im Standard ohne Modifikationen, d. h., Sie können die Kapsel-Plug-ins mit jedem Plug-in eines Drittanbieters kombinieren. Damit können Sie unternehmstaugliche Apps erstellen, die sich nahtlos in die SAP Mobile Services integrieren.

Bereitgestellte Plug-ins

Das Hybrid SDK beinhaltet folgende Plug-ins für iOS und Android:

- AppUpdate: Ermöglicht das Aktualisieren der eingebetteten Web-App, ohne das Erfordernis, eine neue Container-App zu erstellen.
- Attachment Viewer: Verbessert das Benutzererlebnis bei der Darstellung von Anhängen, wie beispielsweise PDF- oder Word-Dokumenten.
- AuthProxy: Ermöglicht eine Basic Authentication und eine zertifikatsbasierte Authentifizierung aus der Container-App heraus.
- Barcode Scanner: Ermöglicht das rudimentäre Scannen von Barcodes.
- Calendar: Ermöglicht die Verwendung und den Zugriff auf den betriebssystemspezifischen Kalender.

- Certificate Delivery: Ermöglicht die Kommunikation mit dem SAP Mobile Secure Server, um Zertifikate auszutauschen.
- Encrypted Storage: Ermöglicht das verschlüsselte Speichern von Daten auf dem mobilen Gerät.
- End-to-End Trace: Ermöglicht die Aufzeichnung von Anfragen vom Client über die SAP Mobile Services bis hin zum Backend-System.
- Federation Provider: ein Zertifikatsanbieter, der eine App-übergreifende Authentifizierung mittels Client-Zertifikaten ermöglicht
- Logger: Ermöglicht es, Informationen wegzuspeichern, die in der Fehlersuche hilfreich sind.
- Logon: Unterstützt die Authentifizierung und den Onboarding-Prozess.
- Offline OData: Ermöglicht die Offlinedatenhaltung auf dem mobilen Endgerät.
- Printer: Ermöglicht den Druck von Dokumenten.
- Push: Ermöglicht den Empfang von Push-Nachrichten.
- Settings: Ermöglicht es, Einstellungen, die in den SAP Mobile Services durchgeführt werden, auf das mobile Endgerät zu laden.
- Usage: Ermöglicht es, Informationen über die Nutzung der Apps zu sammeln.

Darüber hinaus steht das *Consent-Plug-in* ausschließlich für Android zur Verfügung. Es ermöglicht die Zustimmung der Nutzer*innen vor dem Zugriff auf persönliche Daten, wie beispielsweise Termine im Kalender oder Kontakte. Das *Keychain-Certificate-Plug-in* kann nur für iOS eingesetzt werden. Es ermöglicht die Ablage von Zertifikaten im App-spezifischen Zertifikatsspeicher.

Bezugsquellen von Plug-ins

Neben den von SAP bereitgestellten gibt es eine Vielzahl von Plug-ins, die unter diversen Lizenzen bereitgestellt werden. Wir empfehlen Ihnen, vor der Verwendung eines Plug-ins eines Drittanbieters zu prüfen, ob die Lizenz eine kommerzielle Nutzung zulässt.

Sie haben auch die Möglichkeit, eigene Plug-ins zu entwickeln. Dazu ist es genau wie bei der Entwicklung nativer Apps erforderlich, eine der unterstützten Programmiersprachen des darunterliegenden Betriebssystems zu verwenden.

Grundsätzlich können Sie eine hybride App auch ohne Verwendung der SAP Mobile Services entwickeln. In diesem Fall müssen Sie sich jedoch selbst

um Funktionen wie das Onboarding, die Authentifizierung, Single Sign-on, Push-Benachrichtigungen, Backend-Zugriff, Logging usw. kümmern.

Cloud Build

In Kapitel 7, »Entwicklung einer hybriden App mit dem Hybrid Application Toolkit«, zeigen wir Ihnen anhand eines Praxisbeispiels, wie Sie eine hybride App im SAP-Umfeld entwickeln können. Als Entwicklungswerkzeug verwenden wir die SAP Web IDE. Die SAP Web IDE bietet eine Integration in die SAP Mobile Services, die es ermöglicht, einen sogenannten *Cloud Build* durchzuführen. Damit können Sie direkt aus der SAP Web IDE heraus eine hybride App erzeugen, die auf der SAP BTP in der Cloud betrieben wird. Von einem Cloud Build wird gesprochen, da der Build der App vollständig in der Cloud erfolgt und keine lokalen Entwicklungswerkzeuge oder lokale Infrastruktur erforderlich sind. Als Ergebnis erhalten Sie eine Datei vom Typ APK (*Android Package*) für die Android-Plattform und eine Datei vom Typ IPA (*iOS Application*) für die iOS-Plattform.

Lokaler Build

Der Cloud Build ist auch mit einigen Einschränkungen verbunden, auf die wir in Kapitel 12, »Integration von Drittanbieterfunktionalität«, noch eingehen. Daher haben Sie alternativ die Möglichkeit, das SAP Mobile Platform SDK bzw. das darin enthaltene Kapsel SDK über den SAP Service Marketplace zu beziehen und einen lokalen Build auf Ihrem Rechner oder einem Server durchzuführen.

2.3 SAP Fiori Client

SAP Fiori wurde im Mai 2013 von SAP mit den folgenden Worten angekündigt (Quelle: SAP, *http://s-prs.de/v812701*):

> *»Eine Sammlung von Anwendungen, die einfach und benutzerfreundlich sind und eine intuitive Benutzererfahrung für weit verbreitete und häufig genutzte SAP-Softwarefunktionen auf einer Vielzahl von Geräten – Desktop, Tablet, Smartphone – bieten, um die Arbeit zu erleichtern«.*

SAP Fiori

Dieses Versprechen wurde gehalten. Nur ein Jahr später lieferte SAP mehr als 300 SAP-Fiori-Standard-Apps aus. Anfangs wurde SAP Fiori ausschließlich in der *SAP Business Suite* verwendet, später auch in der Nachfolgerlösung *SAP S/4HANA*. 2016 kündigte SAP an, SAP Fiori als Designsprache aller zukünftigen Entwicklungen zu etablieren.

Im Laufe der Jahre hatte SAP die Anbieter führender SaaS-Lösungen (Software as a Service) übernommen. Die größte Herausforderung neben der Integration dieser Lösungen in das SAP-Portfolio war es, eine konsistente

Benutzererfahrung über die verschiedenen Lösungen hinaus zu bieten. Um dieses Ziel zu erreichen, hat SAP in verschiedenen Phasen eine Harmonisierung des Aussehens und Verhaltens der verschiedenen Produktoberflächen vollzogen.

SAP Fiori Launchpad

Wir haben im Zuge der nativen Apps bereits die SAP-Fiori-Designrichtlinien erwähnt, die auf der SAP-Website *https://experience.sap.com/fiori-design-web/* beschrieben werden. Ein grundlegendes Konzept von SAP Fiori ist das SAP Fiori Launchpad, der zentrale Einstiegspunkt in alle SAP-Fiori-Apps. Das SAP Fiori Launchpad verwenden Sie unabhängig davon, ob Sie über ein mobiles Endgerät oder von einem Desktop aus auf die Apps zugreifen. Es stellt rollenbasiert sogenannte *Kacheln* für den Zugriff auf die einzelnen Apps dar und bietet Ihnen Möglichkeiten der Personalisierung. Durch einen Klick auf eine Kachel starten Sie die dahinterliegende App. Auf den Kacheln selbst können auch dynamische Inhalte wie Key Performance Indicators (KPIs) oder Diagramme dargestellt werden. In Abbildung 2.4 sehen Sie ein Beispiel für ein SAP Fiori Launchpad und verschiedene Kacheltypen.

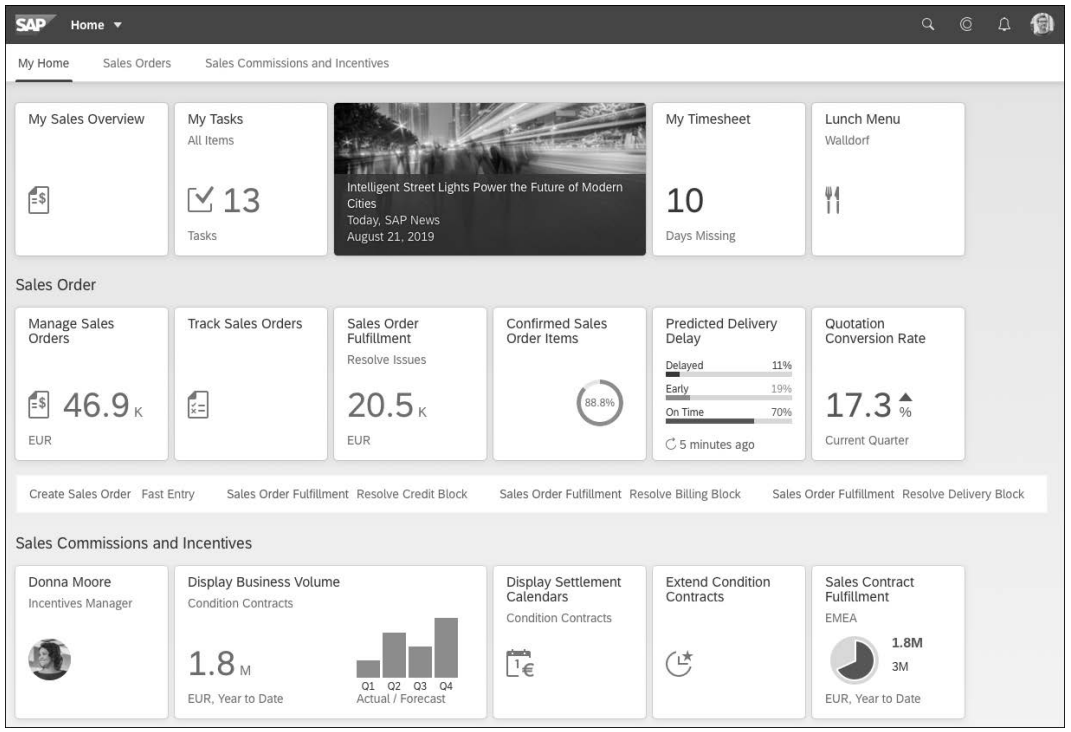

Abbildung 2.4 SAP Fiori Launchpad (Quelle: SAP Fiori Design Guidelines)

Bereitstellungsoptionen

Für das SAP Fiori Launchpad bietet SAP unterschiedliche Bereitstellungsoptionen an. Typischerweise wird es in On-Premise-Systemen mit einer SAP

Business Suite oder einem SAP-S/4HANA-System bereitgestellt. Es besteht auch die Möglichkeit, das SAP Fiori Launchpad auf einem eigenen *SAP NetWeaver Application Server (AS) for ABAP* laufen zu lassen. Für den externen Zugriff Ihrer Mitarbeitenden oder den Zugriff durch Kunden und Geschäftspartner bietet SAP darüber hinaus die Möglichkeit, das SAP Fiori Launchpad in der SAP BTP bereitzustellen. Dabei werden sowohl die Neo- als auch die Cloud-Foundry-Umgebung unterstützt.

SAP Fiori Client

Neben dem browserbasierten Zugriff auf das SAP Fiori Launchpad und die Apps besteht für mobile Endgeräte die Möglichkeit des Zugriffs über den SAP Fiori Client. Dieser Laufzeitcontainer für das SAP Fiori Launchpad wird Ihnen in Form einer nativen mobilen App bereitgestellt. Damit erfüllt der SAP Fiori Client auch die Kriterien hybrider Apps, die Sie in Abschnitt 2.2 kennengelernt haben. Damit bietet er gegenüber dem browserbasierten Zugriff auf das SAP Fiori Launchpad den Vorteil, dass er mit vorinstallierten Plug-ins ausgeliefert wird, wodurch die Möglichkeit des Zugriffs auf native Funktionalitäten auf den mobilen Endgeräten besteht. Der SAP Fiori Client kann über den Apple App Store, den Google Play Store oder den Enterprise App Store Ihrer Enterprise-Mobility-Management-Lösung bereitgestellt werden. In Abbildung 2.5 sehen Sie den Zugriff auf das SAP Fiori Launchpad über den Standard-SAP-Fiori-Client.

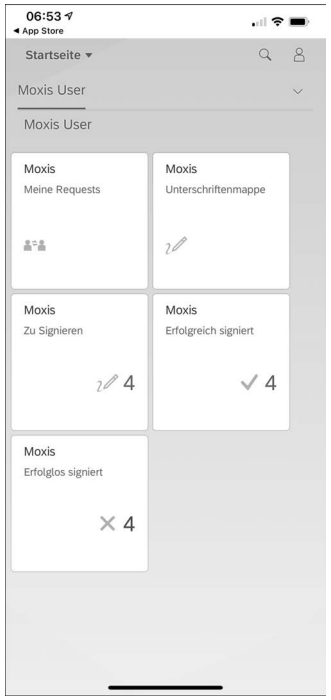

Abbildung 2.5 Zugriff auf das SAP Fiori Launchpad über den SAP Fiori Client

Kundeneigener SAP Fiori Client

Der SAP Fiori Client kann das SAP Fiori Launchpad entweder direkt oder unter Verwendung der SAP Mobile Platform oder der SAP Mobile Services aufrufen. Neben dem Bezug des SAP Fiori Clients als Standardlösung über die App Stores von Apple und Google besteht die Möglichkeit, einen *Custom Build* unter Verwendung der SAP Mobile Services durchzuführen und den SAP Fiori Client so den eigenen Anforderungen anzupassen. Wenn Sie den SAP Fiori Client selbst erstellen, besteht jedoch die Einschränkung, dass Sie keine Plug-ins von Drittanbietern verwenden können. Das Offline-OData-Plug-in des Kapsel SDK kann im SAP Fiori Client nicht verwendet werden, d. h., der SAP Fiori Client dient ausschließlich dem Onlinezugriff auf das SAP Fiori Launchpad.

Verwenden Sie den SAP Fiori Client unabhängig von den SAP Mobile Services, können Sie direkt auf das SAP Fiori Launchpad zugreifen. Sie haben die Möglichkeit, den SAP Fiori Client über das Hybrid SDK lokal zu erstellen. Damit können Sie neben den Standard-Plug-ins dieses SDKs auch Plug-ins von Drittanbietern integrieren. Das Hybrid SDK stellt auch die Basis für die Erstellung eines kundeneigenen SAP Fiori Clients über eine *Continuous Integration Pipeline*, wie beispielsweise *Apache Jenkins*, dar. Eine Pipeline ermöglicht es Ihnen, mehrstufige Prozesse beim Zusammenbauen der App abzubilden. So werden üblicherweise Prüfungen des Quellcodes und automatisierte Tests durchlaufen. Der Build wird abgebrochen, sobald in einem der Tests ein Fehler auftritt. Damit können Sie Ihren Entwicklungsprozess standardisieren und dafür sorgen, dass Ihren eigenen Richtlinien entsprechend vorgegangen wird.

Sicherheitseinstellungen

Der SAP Fiori Client wird im Standard mit der Möglichkeit ausgeliefert, Sicherheitseinstellungen vorzunehmen. So kann beispielsweise ein *App-Code* oder *Passcode* definiert werden, der beim Starten des SAP Fiori Clients eingegeben werden muss. Abbildung 2.4 zeigt eine entsprechende Abfrage auf der Startseite. Optional und abhängig vom verwendeten mobilen Endgerät kann der SAP Fiori Client auch mit biometrischen Verfahren wie *Face ID* entsperrt werden. Mit diesem Verfahren schalten Sie das mobile Endgerät durch einen Scan Ihres Gesichts frei. Beim Einrichten des Geräts wird die Face ID mit den biometrischen Merkmalen Ihres Gesichts verknüpft.

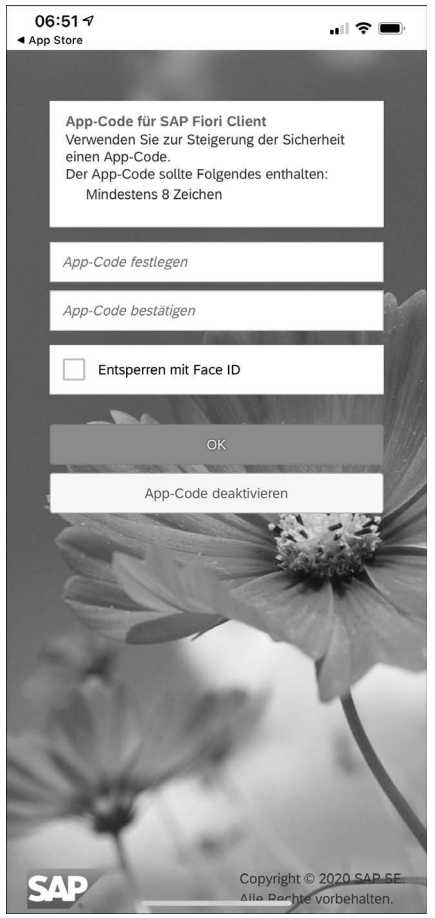

Abbildung 2.6 Beispiele für Sicherheitseinstellungen beim Aufruf des SAP Fiori Clients

2.4 Mobile Development Kit

Das *Mobile Development Kit* (MDK) der SAP Mobile Services bietet Ihnen die Möglichkeit, plattformübergreifende Apps (*Cross-Platform-Apps*) zu entwickeln. Dadurch können Sie mit einer Codebasis sowohl die Android- als auch die iOS-Plattform bedienen. Es handelt sich bei solchen Apps um metadatengetriebene Apps.

Entwicklung von MDK-Apps

Mit dem MDK entwickelte Apps bieten den Anwender*innen die gewohnte Performance und das Verhalten einer nativen App. Für eine MDK-App erstellen Sie die erforderlichen Metadaten im Editor und programmieren die

Geschäftslogik in JavaScript. Zur Laufzeit werden die Artefakte in nativen Code übersetzt. Damit erstellen Sie Ihre App plattformunabhängig und deployen sie danach über die SAP Mobile Services.

Bereitstellung von MDK-Apps

Die Nutzer*innen erhalten eine native iOS- oder Android-App auf ihren Geräten. Sie können sich diese App wie ein Abspielgerät vorstellen, das jegliche MDK-App ausführen kann. Die »Abspiel«-App wird von SAP unter dem Namen *SAP Mobile Services Client* in den App Stores bereitgestellt. Die Nutzer*innen können diese App direkt auf ihre Endgeräte laden. Die Verteilung der App auf die Endgeräte kann aber auch in diesem Fall über ein Mobile Device Management erfolgen, wenn das gewünscht wird. Dies bietet Ihnen an dieser Stelle noch mehr Flexibilität, da die App über das Mobile Device Management vorkonfiguriert werden kann.

Der SAP Mobile Services Client verbindet sich mit den SAP Mobile Services und lädt die MDK-Inhalte. Da die App von SAP bereitgestellt wird, müssen Sie sich nicht um den App-Signierungsprozess kümmern. Bei Updates muss keine neue Version der Client-App geladen werden, sondern es ist lediglich erforderlich, die Inhalte aus der Entwicklungsumgebung heraus in die SAP Mobile Services zu pushen. Danach stehen die neuen Inhalte sofort zur Verfügung. Die Entwicklung von MDK-Apps wird in Kapitel 6, »Entwicklung einer hybriden App mit dem Mobile Development Kit«, detailliert behandelt.

Vor- und Nachteile der MDK-Apps

Das MDK hat einen weiteren Vorteil: Sie können die gleichen Metadaten, die in der nativen App verwendet werden, auch im Web bereitstellen, um dieselbe App im Browser auszuführen. Das MDK hat aber auch einen Nachteil: Sie können innerhalb des SAP Mobile Services Client nur auf eine MDK-App referenzieren. Es gibt aktuell keine Möglichkeit, mehrere MDK-Apps parallel auszuführen.

Entwicklungsumgebungen

Die Entwicklung kann sowohl in der Neo-Umgebung in der SAP Web IDE als auch in der Cloud-Foundry-Umgebung im SAP Business Application Studio erfolgen. Dabei wird ein grafischer Editor verwendet. Die Apps werden danach in die SAP Mobile Services auf der SAP BTP deployt. Die Entwicklung ist sehr intuitiv, und Ihnen steht eine Vielzahl an Funktionen wie Offlinefähigkeit oder Push-Benachrichtigungen zur Verfügung.

Vorlagen

Bei MDK-Apps können Sie zwischen mehreren Projekttypen wählen, für die Vorlagen bereitgestellt werden. Durch die Auswahl der richtigen Vorlage können Sie sich den initialen Entwicklungsaufwand sparen. In Tabelle 2.1 haben wir alle verfügbaren Projekttypen aufgelistet.

Projekttyp	Beschreibung
Empty	Eine leere Vorlage beinhaltet keine wichtigen Dateien, bis auf eine leere Hauptseite und zwei Aktionen, nämlich zum Ausloggen und Schließen der Hauptseite. Bei dieser Vorlage gibt es auch keine standardmäßige Servicekonfiguration für den Datenaustausch.
List Report	Eine List-Report-App bietet neben einer Servicekonfigurationen eine Hauptseite mit einem Listenelement. Die Servicekonfiguration umfasst Aktionen, die auch mit einigen Ereignissen im Lebenszyklus der App verknüpft werden können. Damit können Sie beispielsweise eine Aktion ausführen, wenn die App in den Offlinezustand wechselt oder eine Synchronisierung der Daten mit dem Server abgeschlossen wurde.
Base	Vorlagen, die auf den Projekttyp **Base** beruhen, beinhalten bis auf die Servicekonfiguration inklusive Aktionen keine weiteren nennenswerten Dateien.
CRUD	Diese Vorlage ist die umfangreichste und beinhaltet die meisten Dateien. Neben einer Servicekonfiguration erhalten Sie eine Auswahl aller verfügbaren Entitäten. Für jede ausgewählte Entität werden alle fünf CRUDQ-Funktionen (Create, Read, Update, Delete, Query) unterstützt. Es werden sowohl vollständige Seiten als auch Aktionen mit Regeln für den Löschvorgang generiert.

Tabelle 2.1 MDK-Projekttypen

2.5 SAP Mobile Cards

Wallet-Apps

Wallet-Apps erfreuen sich bei den Nutzer*innen großer Beliebtheit – egal, ob es sich um Kundenkarten von Hotelketten, Fluglinien oder Mietwagenanbietern handelt oder um digitale Flug- oder Bahntickets. Ziel dieser Apps ist es, Inhalte einfach und effizient auf den mobilen Geräten der Nutzer*innen verfügbar zu machen, ohne für jeden Anwendungsfall eine eigene App installieren zu müssen.

Diesen Trend hat auch SAP erkannt und mit SAP Mobile Cards eine ideale Lösung gefunden. Die Mobile Cards sind einfache HTML-Anwendungen, die in der App SAP Mobile Cards dargestellt werden. Die App wird von SAP in den App Stores bereitgestellt. Die Nutzer*innen können SAP Mobile Cards direkt auf ihre Endgeräte herunterladen. Die Verteilung der App auf die Endgeräte ist auch in diesem Fall über ein Mobile Device Management möglich.

Entwicklung von Mobile Cards

Die Entwicklung der Mobile Cards erfolgt in der Neo-Umgebung der SAP BTP direkt innerhalb der SAP Mobile Services. In der Cloud-Foundry-Umgebung wird das SAP Business Application Studio verwendet. Eine Entwicklung innerhalb der SAP Mobile Services ist in der Cloud-Foundry-Umgebung nicht vorgesehen.

Die Mobile Cards unterstützten Push-Benachrichtigungen, die Sie aus dem Backend an die SAP Mobile Services senden können und die von dort aus an die App SAP Mobile Cards weitergeleitet werden.

Stärken der Mobile Cards

Ihre Stärken können die Mobile Cards vor allem bei der Darstellung einfacher Inhalte ausspielen. Entwickler*innen benötigen lediglich HTML-Kenntnisse. Es gibt keine Möglichkeit, Mobile Cards mit JavaScript zu entwickelt, womit auch die Grenzen der Mobile Cards klar abgesteckt sind.