Handytickets für iOS erzeugen

Taschenticketautomat

Apple Wallet: Handytickets für iOS erzeugen

Fluggesellschaften bieten sie an, die Bahn und Konzertveranstalter: digitale Tickets, die man nicht ausdrucken muss und einfach auf dem Mobiltelefon mitnimmt. Apple hat sich dafür die App namens Wallet und ein Dateiformat mit raffinierten Zusatzfunktionen ausgedacht. So erzeugen Entwickler mit überschaubarem Aufwand eigene digitale Tickets, Gutscheine und Kundenkarten.

Von Jan Mahn

kompakt

- Digitale Tickets auf dem Handy sind praktisch und mehr als ein Ersatz für die Papierversion. Sie sind schnell griffbereit und informieren zum Beispiel über Planänderungen.
- Apple hat für iOS eine Ticket-App namens Wallet zusammen mit einem eigenen Datenformat entwickelt.
- Um als Entwickler Tickets zu erzeugen, braucht man theoretisch nur einen Texteditor und einen Apple-Entwickleraccount – für echte Tickets vom Fließband gibt es Open-Source-Generatoren.

Früher, eigentlich bis vor wenigen Jahren, waren sämtliche Eintrittskarten und Reisetickets noch durchgängig vom Anbieter gedruckte Dokumente, auf einem speziellen Papier mit Glitzerstreifen, damit sie nicht jeder nachbauen konnte. Als

nächste Evolutionsstufe folgten Tickets zum Selbstausdrucken. Die Echtheit bestätigt man seither nicht mehr über ein exklusives Spezialpapier, sondern über einen aufgedruckten Code, sei es ein Bar- oder ein QR-Code. Das Kontrollpersonal scannt ihn und ein mit einem Server verbundenes Lesegerät verrät, ob dieses Ticket gültig und noch nicht genutzt ist.

Anstatt den Code auszudrucken und ein zerknicktes A4-Ticket zum Konzert mitzunehmen, kann man den Code inzwischen auch direkt auf dem Mobiltelefon belassen und dessen Display beim Einscannen vorzeigen. Doch wer schon mal versucht hat, ein PDF-Ticket auf dem Handy aus der Mail-App zu fischen und so zu zoomen, dass man den QR-Code bequem scannen kann, der weiß, dass das nicht an den Komfort von Papier herankommt. Bequemer geht es mit einer Ticket-App. Apple hat sich schon 2012 die App Passbook ausgedacht und später in Wallet umbenannt. Neben Tickets liegen dort seitdem auch Apple-Pay-Kreditkarten. Die App funktioniert aus Kundensicht denkbar einfach: Man bekommt auf welchem Weg auch immer – per Mail, in einer anderen App oder auf einer Website – einen Link, um ein Ticket herunterzuladen oder direkt eine Ticket-Datei. Auf dem iOS-Gerät wird die Datei direkt in der Wallet-App geöffnet und die Karte dem Wallet hinzugefügt. Öffnet man den Link auf einem per iCloud verbundenen macOS-Computer, landet sie ebenfalls via Cloud-Magie auf dem iOS-Mobiltelefon. Und auch auf die Apple-Watch rutschen die digitalen Karten auf Wunsch. Gegen Vervielfältigung und Weitergeben sind die Pässe nicht geschützt, es können auch mehrere Konzertbesucher dasselbe Ticket auf ihr Telefon laden – so wie man auch ein PDF-Ticket mehrfach ausdrucken kann. Beim Einlass muss der Bar- oder QR-Code also immer mit einer Datenbank abgeglichen werden.

Die Einladung zu einer fiktiven Grillfeier mit Freunden steckt als digitales Ticket in der Apple-Wallet-App: Für Freunde mit Android-Telefonen muss man entweder ein eigenes Ticket ausstellen oder ihnen eine alternative App empfehlen.

Beim Einlass zum Konzert findet der Besucher all seine gültigen Tickets fein säuberlich sortiert in dieser App und der QR- oder Barcode ist immer an den Bildschirm angepasst. Doch damit nicht genug: Der Ticketanbieter kann das Ticket auch so präparieren, dass es zum Beispiel eine Stunde vorm Konzert beim Benutzer aufpoppt und er es nicht einmal mehr raussuchen muss. Und wenn die Show ausfällt oder sich der Flug verspätet, kann der Veranstalter alle Kunden mit Wallet-Ticket per Push-Benachrichtigung informieren, ohne dass diese eine separate App bräuchten. Als Alternative zu QR-Codes können sich die digitalen Tickets bei der Kontrolle auch per Near Field Communication (NFC) ausweisen.

Proprietär, aber transparent

Der Funktionsumfang hat Sie überzeugt und Sie wollen als Entwickler direkt loslegen, solche Tickets zu erzeugen? Dann haben wir zuerst ein paar nicht so gute Nachrichten: Das Format für die Wallet-Tickets hat sich Apple im stillen Kämmerlein ausgedacht. Es gibt also keinen (Web-)Standard und Apple kümmert sich auch nicht um den Teil der Nutzerschaft, der ohne iPhone aus dem Haus geht, sondern lieber ein Android-Gerät zum Konzert mitnimmt. Wie die Tickets dennoch aufs Android-Telefon gelangen, lesen Sie auf Seite 154.

Und Android?

Auch auf dem Android-Telefon kann man die für Apples Betriebssysteme erstellten Tickets öffnen. Im Play Store gibt es die kostenlose App WalletPasses. Sie stammt von einem Entwickleraccount namens „WalletPasses Alliance“. Das klingt wie ein unabhängiger Herstellerverband, scheint aber ein Fantasiename zu sein. Außerhalb des Stores tritt ein Verband dieses Namens nirgends in Erscheinung. Auch Apple hat damit nichts zu tun. Die App funktioniert aber, kann Apples Datenformat lesen und zeigt die Passes nahezu im gleichen Layout wie die Original-App an.

Dass Apple in diesem Bereich jahrelang die Nase vorn hatte, schien auch Google zu stören. Auf der Entwicklerkonferenz Google I/O im Mai 2022 stellte das Unternehmen seine Pläne für Google Wallet vor. Nicht nur der Name erinnert an Apples App, auch das Konzept: Das Bezahlen mit Google Pay soll künftig nur eine Funktion der digitalen Brieftasche namens Google Wallet sein. Daneben sollen Kundenkarten, Tickets und Impfpässe ein digitales Zuhause bekommen. Die Entwickerdokumentation für diese neuen Funktionen ist unter der Adresse developers.google.com/wallet zu finden.

Nur weil das Format von Apple entwickelt wurde, bedeutet das zum Glück aber nicht, dass nur Apple-Software solche Tickets

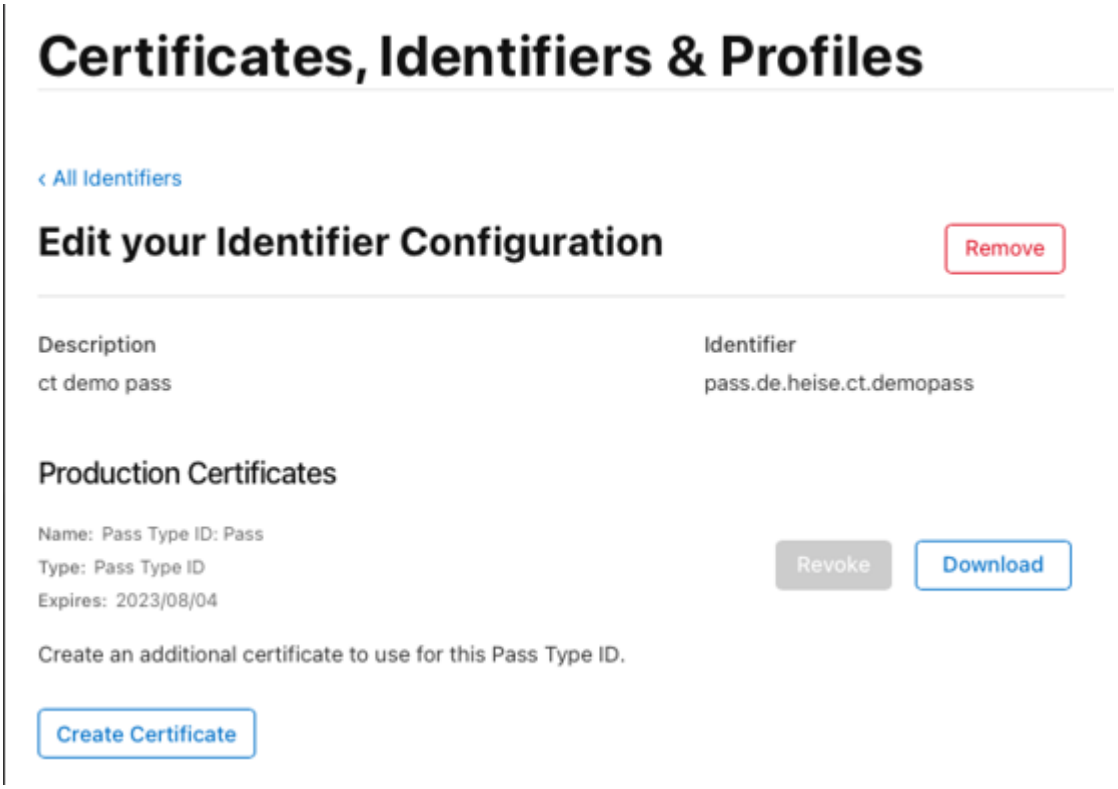
erzeugen kann. Und anders als bei Transaktionen über den App-Store nimmt Apple auch keine Provision für jedes ausgestellte Ticket. Stattdessen ist das Format ausführlich dokumentiert (siehe ct.de/yw3e) und jeder mit einem Texteditor und einem Zip-Werkzeug könnte Tickets erzeugen. Es gibt jedoch einen Haken: Damit die Tickets, in Apples Terminologie Pass genannt, beim Kunden funktionieren, muss man sie digital signieren. Dafür braucht man ein Apple-Entwickler-Zertifikat, das man nur bekommt, wenn man sich als Entwickler im Apple-Developer-Programm registriert. Der Account berechtigt auch dazu, Apps in den Store zu bringen und damit Geld zu verdienen. 99 US-Dollar (und mit Steuern auch 99 Euro) kostet ein solcher Account für Einzelpersonen im Jahr. Wie Sie Apple-Entwickler werden und an ein Zertifikat zum Signieren der Tickets kommen, lesen Sie auf Seite 155.

Developer-Account und Zertifikat beschaffen

Ein Apple-Developer-Account ist Voraussetzung, um Passes für die Wallet signieren zu können. Nebenbei dürfen Entwickler mit einem solchen Account auch Apps in den App Store bringen – 99 Euro kostet die Mitgliedschaft für ein Jahr. Um Mitglied zu werden, braucht man eine gewöhnliche Apple-ID, die man als normaler Nutzer zum Beispiel für die App-Stores und Apple Music nutzt. Der Login muss durch einen zweiten Faktor abgesichert sein. Los geht die Registrierung unter der Adresse developer.apple.com/enroll. Der Assistent fragt zunächst nach der Art des Accounts – im einfachsten Fall meldet man sich als Privatperson an. Firmenangehörige, die mit dem Account auch Apps unter Firmennamen verkaufen wollen, müssen den anderen Knopf betätigen. Die Fragen des Assistenten sind weitestgehend schnell beantwortet – etwas verwirrend ist die Anforderung, die eigenen Kontaktdaten zweimal einzutippen, einmal davon „Romanized“. Das ist für asiatische Entwickler gedacht, die ihre Daten einmal in ihren Schriftzeichen und noch einmal im lateinischen Buchstabensystem eingeben sollen. Europäer füllen beide Abschnitte des Formulars einfach identisch aus.

Am Ende des Prozesses geht es ans Bezahlen per Kreditkarte, die PayPal-Option war bei uns ausgeblendet. 99 Euro und ein paar Minuten Wartezeit später bekommt man eine Mail und ist offiziell Apple-Entwickler. Unter der Adresse developer.apple.com/account findet man ab jetzt den internen Bereich.

Für jeden Typ von digitalen Tickets, aber nicht für jedes einzelne Ticket, brauchen Sie einen sogenannten Identifier, der im internen Bereich hinterlegt ist. Eine Fluggesellschaft bräuchte also für all ihre Flugtickets einen solchen Identifier. Wenn sie über die Wallet auch Essensgutscheine verteilen wollte, müsste sie einen neuen anlegen – und zu jedem Identifier braucht man ein von Apple signiertes Zertifikat. Damit kann man stets beliebig viele Tickets signieren.



Ein Identifier und ein passendes Zertifikat sind die Voraussetzung, um gültige Tickets für die Wallet zu erzeugen. Anlegen kann man sie im Developer-Portal mit einem kostenpflichtigen Account.
Klicken Sie zum Anlegen eines Identifiers im Entwicklerportal auf „Certificates, Identifiers & Profiles“ und dort unter

Identifiers auf das blaue Plus. Sie brauchen ein Objekt vom Typ „Pass Type IDs“. Anschließend geben Sie dem Identifier eine Beschreibung wie „Mein Demoticket“ und denken sich eine eindeutige Bezeichnung in Form einer umgekehrten Domain aus, wir haben zum Test pass.de.heise.ct.demopass gewählt. Das Formular ergänzt den ersten Block pass. automatisch, die Domain müssen Sie auch nicht besitzen – es geht nur darum, einen eindeutigen String zu generieren, den niemand sonst hat. Hat man den Assistenten durchgeklickt, liegt der Identifier bereit und kann mit einem Zertifikat verknüpft werden.

Klicken Sie den Eintrag in der Liste an und wählen dann „Create Certificate“. Der Assistent erwartet von Ihnen einen Namen (nur für Ihre eigene Sortierung) und den Upload eines CSR, eines „Certificate Signing Request“. Das ist eine Datei, die man auf dem lokalen Computer zusammen mit einem geheimen Schlüssel erzeugt. Der Schlüssel bleibt immer auf dem eigenen Gerät, der CSR ist eine schriftliche Bitte, den öffentlichen Schlüssel zu signieren. Als Antwort auf den CSR bekommen Sie von Apple einen signierten öffentlichen Schlüssel zurück.

Am schnellsten kommen Mac-Nutzer an einen passenden CSR (wen wundert es). Sie öffnen das Systemprogramm Schlüsselbundverwaltung und wählen oben links in der Menüleiste „Schlüsselbundverwaltung/Zertifikatsassistent/Zertifikat einer Zertifizierungsinstanz anfordern …“. Der Assistent hat nicht viele Fragen. Man gibt eine E-Mail-Adresse (wird nicht im Zertifikat eingebaut und ist nicht sichtbar) und einen Namen (Freitext, zum Beispiel „Wallet-Tickets“) ein. Das dritte Feld für die Mailadresse der Zertifizierungsstelle bleibt leer. Anschließend „Auf der Festplatte sichern“ anwählen, den Haken „Eigene Schlüsselpaarinformationen festlegen“ aktivieren und den Assistenten abschließen. Der CSR liegt dann im Dokumente-Ordner. Der private Schlüssel heißt Wallet-Tickets, liegt im Schlüsselbund und kann dort später exportiert werden, wenn man ihn braucht.

Unter Linux (und unter Windows im WSL) ist es ebenfalls möglich, an einen CSR zu kommen, wenn auch nicht von Apple dokumentiert. Führen Sie einfach folgenden Befehl aus:

```
openssl req -nodes -newkey rsa:2048 -keyout apple_pass.key -out CertificateSigningRequest.certSigningRequest
```

Die meisten Fragen des Assistenten können Sie überspringen, nur den Ländercode (DE) und die E-Mail-Adresse sollten Sie setzen. Auch bei diesem Verfahren landen die Daten nicht im Zertifikat und werden nicht angezeigt. OpenSSL erzeugt den CSR und den Schlüssel in der Datei apple_pass.key.

Unabhängig vom Betriebssystem schnappen Sie sich am Ende die erzeugte Datei mit dem sperrigen Namen CertificateSigningRequest.certSigningRequest und laden Sie, zurück im Developer-Portal, im Feld für den CSR hoch. Postwendend erhalten Sie eine Datei namens pass.cer mit Ihrem Zertifikat zurück.

Datenstrukturkunde

Ein Pass ist technisch nur eine Zip-Datei mit ein paar verpflichtenden Inhalten. Nach dem Verpacken mit einem Zip-Programm muss man die Dateiendung .zip lediglich durch .pkpass ersetzen und kann sie anschließend verschicken. Um zu verstehen, wie die Tickets funktionieren, erfahren Sie zunächst, welche Dateien im Ordner liegen müssen – im Anschluss erhalten Sie Tipps, wie Sie das Generieren in bestehende Prozesse einbauen können. Denn obwohl man die Pässe theoretisch per Hand zusammenbauen bauen könnte, Spaß macht das nicht.

Alle Dateien, die zu einem Pass verpackt werden sollen, müssen in einem Ordner liegen. In diesen Ordner gehört verpflichtend eine Datei namens pass.json. Die enthält im JSON-Format alle Texte und Einstellungen. Zunächst sind da ein paar Pflichtfelder:

```
{
 "passTypeIdentifier":
          "pass.de.heise.ct.demopass",
 "serialNumber": "1234567890",
 "formatVersion": 1,
 "organizationName": "c’t magazin"
 "description": "Einladung",
 "teamIdentifier": "<Ihre ID>",
 [...]
}
```

Der passTypeIdentifier ist die Zeichenkette, die Sie zuvor im Developer-Bereich angelegt haben. Die Seriennummer (serialNumber) müssen Sie als Aussteller generieren und sich darum kümmern, dass sie für alle Pässe mit einem Identifier eindeutig ist – mit einer Datenbank im Hintergrund, die Ihre Tickets verwaltet, sollte das kein Problem sein. Vorgaben zum Format gibt es nicht, viele von uns untersuchte Tickets großer Aussteller enthielten in diesem Feld UUIDs [1].

Die formatVersion ist schnell erklärt, dafür ist aktuell nur der Wert 1 zulässig. Die description soll laut Dokumentation keine Inhalte des Tickets (wie den Namen des Inhabers) enthalten. Es handelt sich um eine Beschreibung, die beispielsweise von den Assistenzwerkzeugen sehbehinderter Nutzer vorgelesen werden kann. Den teamIdentifier müssen Sie aus der Developer-Plattform kopieren. Sie finden ihn, indem Sie unter developer.apple.com/account links auf Membership klicken. Es handelt sich um eine zehnstellige Zeichenkette aus Großbuchstaben und Zahlen.

Es folgen auf der obersten Ebene des JSON-Objekts viele freiwillige Angaben, die man auch weglassen kann. Diese bestimmen unter anderen, wie das digitale Dokument aussehen soll. Ans eigene Farbschema passt man das Ticket mit backgroundColor, foregroundColor und labelColor an. Die Werte müssen wie in CSS als RGB-Farbe angegeben sein, andere Schreibweisen (wie die hexadezimale) sind nicht erlaubt. Ein Eintrag für ein leuchtendes Orange als Hintergrund sieht zum

Beispiel wie folgt aus:

"backgroundColor": "rgb(255,125,0)"

Außerdem kann man auf der obersten Ebene festlegen, wann das Dokument relevant ist, also beim Nutzer automatisch auf dem Sperrbildschirm auftauchen soll. Bei einem Konzertticket könnte das kurz vor dem Einlass sein. Wenn es eine eindeutige Zeit gibt, ist das der einfachste Weg, das Ticket automatisch in den Vordergrund zu rücken. Das Datumsformat muss dem W3C-Standard für Datum und Uhrzeit entsprechen, am einfachsten gibt man die Zeit in UTC an (signalisiert durch das Z am Ende):

"relevantDate":"2022-08-10T10:00Z"

Was für Eintrittskarten gut funktioniert, ist bei anderen Dokumenten gar nicht so nützlich. In einem Wallet-Pass kann zum Beispiel auch eine Kundenkarte stecken. Für solche hat sich Apple ein besonderes Mittel zur Kundenbindung ausgedacht – für datenschutzbewusste Europäer mag die aber übergriffig wirken, weshalb man sie sparsam einsetzen sollte. Im Pass kann man Geokoordinaten sowie eine Distanz hinterlegen. Nähert sich der Kunde einem Bereich um einen Laden, taucht der Pass mit einem Hinweistext auf. Dabei erhalten Sie als Aussteller keine Benachrichtigung und die Verarbeitung geschieht lokal auf dem Telefon – aber das weiß der datenschutzbewusste Kunde nicht und könnte sich verfolgt fühlen. Technisch funktioniert das wie folgt:

```
"locations": [
 {
 "latitude": 52.3859153,
 "longitude": 9.80959388,
 "relevantText": "Kommen Sie herein!"
 }
],
"maxDistance": 100
```

Unter locations können Sie bis zu zehn Orte mit ihren

Koordinaten hinterlegen, die maxDistance ist der Abstand in Metern. Unterschreitet das Telefon diesen Abstand, erscheint der Pass mit dem unter relevantText angegebenen Werbetext. Haben Sie zu viele Geschäfte, um sie alle unter locations einzutragen, gibt es eine Alternative: Sie können auch Bluetooth-Beacons in den Läden aufhängen und die digitalen Tickets darauf reagieren lassen – die Dokumentation (siehe ct.de/yw3e) verrät, welche Datenfelder dafür vorgesehen sind. Wer es mit solchen Kundenbindungsmaßahmen übertreibt, muss damit rechnen, dass die Kunden die virtuelle Kundenkarte schnell wieder aus der Wallet-App werfen.

Das Rauswerfen funktioniert aber nicht nur manuell, wenn der Nutzer vom Pass genervt ist, sondern auch automatisch, wenn ein Ticket nicht mehr aktuell ist. Gerade Veranstaltungs- und Reisetickets möchte man am Tag darauf nicht mehr in der App sehen (oder höchstens in einer Art Papierkorb). Mit einem expirationDate (im W3C-Datumsformat wie auch das relevantDate) nehmen Sie den Kunden das Aufräumen ab.

Karten für alle Gelegenheiten

Nach diesen verpflichtenden und freiwilligen Angaben müssen Sie sich entscheiden, um welchen Typ digitales Dokument es sich handelt und diesen in der Datei pass.json eintragen (mehr dazu später). Zur Auswahl stehen:

- eventTicket: Eintrittskarte zu einer Veranstaltung
- boardingPass: Ticket für Flüge, Bus- oder Bahnreisen
- coupon: Gutschein oder Rabattcoupon
- storeCard: Kunden- oder Mitgliedskarte
- generic: sonstige digitale Karten

Für Verkehrsunternehmen, Veranstaltungsorganisatoren, Ladenbetreiber und Fitnessstudios gibt es somit passende Designvorlagen, für alle anderen Dokumente ist generic gedacht. Mit der Wahl eines Typs entscheidet man sich für ein Layout, im Prinzip funktionieren aber alle Typen identisch und

unterscheiden sich bei den Funktionen nur in Nuancen. Als Ersteller darf man vier Bereiche mit Texten füllen – in der Infografik auf Seite 156 haben wir die Abschnitte markiert und beschriftet. Im Bereich oben rechts (headerFields) kann man bei einer Veranstaltung zum Beispiel das Datum gut sichtbar hinterlegen. Die Wallet-App simuliert ja eine Brieftasche und stapelt mehrere Tickets hintereinander. Diese Kopfzeile ist immer lesbar und wird nicht von anderen Karten verdeckt.

Die zentralen Informationen, die groß und fett geschrieben werden sollen, liegen in primaryFields und erstrecken sich oben quer über das Ticket. Die nächste Ebene (kleiner dargestellt und mit mehreren Einträgen nebeneinander) heißt secondaryFields, noch kleiner werden auxiliaryFields dargestellt. Die virtuelle Variante der Ticketrückseite heißt bei Apple backFields und ist gedacht für Geschäftsbedingungen und anderes Kleingedrucktes. Angezeigt werden die Daten nur, wenn der Nutzer die drei Punkte oben rechts auf einem Ticket antippt.

Apple hat sich dagegen entschieden, Datenmodelle für allerlei Gelegenheiten mit starren Feldern zu spezifizieren (etwa Datum, Sitzplatz, Start, Ziel, Reisender, Gate, Bahnsteig). Stattdessen kann man in den Bereichen alle Angaben in Schlüssel-Werte-Paaren frei definieren und statt eines Konzerttickets auch eine digitale Kantinenkarte bauen.

Zunächst legt man den gewählten Typ im JSON-Dokument auf der obersten Ebene als Objekt an, darunter Objekte für die Felder, die man nutzen möchte (primaryFields, secondaryFields …). Sie sind allesamt optional. In den Feldern platziert man seine Inhalte jeweils mit einem Schlüssel (key), einer Beschriftung (label) und dem Wert (value). Das Label steht dann im fertigen Pass über oder unter dem Wert.

Bei einem Veranstaltungsticket für eine fiktive Grillfeier mit Freunden kann das zum Beispiel folgendermaßen aussehen:

```
"eventTicket": {
 "headerFields": [
  {
   "key": "seat",
   "label": "SITZPLATZ",
   "value": "Bierbank 1"
  }
 ],
 "primaryFields": [
  {
   "key": "event",
   "label": "VERANSTALTUNG",
   "value": "Grillfeier mit Freunden"
  }
```

```
 ],
 "secondaryFields": [
  {
   "key": "location",
   "label": "ORT",
   "value": "Garten"
  }
 ],
 "auxiliaryFields": [
 {
   "key": "food",
   "label": "MITBRINGEN",
   "value": "Nudelsalat"
  }
 ],
 "backFields": [
  {
   "key": "terms",
   "label": "Bedingungen",
   "value": "Bitte bis 17:00 absagen"
  }
 ]
}
```

Bei der Vergabe der Schlüsselnamen ist man frei, sie müssen nur einmalig sein. Falls Sie zufällig in die Verlegenheit kommen sollten, Tickets für ein Reiseunternehmen zu programmieren, gibt es noch eine Besonderheit. Sobald Sie das Objekt boardingPass angelegt haben, müssen Sie darin verpflichtend einen transitType angeben. Im Ticket erscheint dann zwischen den beiden primären Informationen in primaryFields ein passendes Icon. Ein Ausschnitt aus einem Flugticket kann zum Beispiel wie folgt aussehen:

```
"boardingPass": {
  "transitType": "PKTransitTypeAir",
  "primaryFields": [
   {
    "key": "from",
    "label": "VON",
    "value": "HAJ"
```

```
   },
   {
    "key": "to",
    "label": "NACH",
    "value": "LYR"
   }
  ]
}
```

Die Buchstaben PK stehen für PassKit, so heißt das Framework für die Tickets und auch für den Zugriff auf Apple Pay. Neben TypeAir gibt es noch TypeBoat, TypeBus, TypeGeneric und TypeTrain.

Codes und Bilder

Sind all diese Werte ausgefüllt, gibt es noch zwei weitere optionale Aufgaben. Auf der einen Seite sind das die Bilddateien für Logos. Dabei kann (muss aber nicht) sich ein Grafiker genauso austoben wie bei den Favicons für Websites und noch Varianten für unterschiedlich große Bildschirme erzeugen (mit @2x am Ende des Dateinamens). Die Arbeit fällt pro Unternehmen nur einmal an. In der Tabelle auf Seite 158 sehen Sie, welche Bildchen Sie anlegen und in den Ordner neben die pass.json legen können. Alle Bildchen sind optional.

Letzte Baustelle ist der Code, mit dem Sie die Echtheit des Dokuments prüfen. Das kann ein Barcode sein, ein QR-Code oder auch die NFC-Schnittstelle des Mobiltelefons. Für letztere Technik ist zusätzlicher Zertifikatsaufwand nötig, dazu sei auf die Dokumentation verwiesen (siehe ct.de/yw3e). Bar- und QR-Codes dagegen sind schnell eingebunden und gehören auf die oberste Ebene im JSON-Objekt. Sie sind wie die meisten Felder optional:

```
barcodes": [
 {
  "altText": "123 Mustermann",
  "format": "PKBarcodeFormat",
  "messageEncoding": "iso-8859-1",
```

```
  "message": "123MUSTERMANN"
 }
]
```

Die Wahl des Formats hängt vor allem von Ihrer bestehenden Leser-Infrastruktur ab. In einen QR-Code passen die meisten Daten, neben PKBarcodeFormatQR sind auch FormatPDF417 und FormatAztec vorgesehen. Apple verweist darauf, dass man heutzutage unter dem Schlüsel barcodes eine Liste mit potenziell mehreren Einträgen anlegen soll. Ab iOS 9.0 wird diese Schreibweise verwendet. Früher legte man einen einzelnen Barcode unter barcode ab. Das ist seit dem Start von iOS 9.0 offiziell abgekündigt und wäre nur noch für iOS 8 und noch ältere Systeme nötig. Bei unseren Tests fanden wir etwa bei einem Ticket der Lufthansa aus dem Jahr 2022 ausschließlich den alten Eintrag barcode im Singular.

Bilddateien für Logos und Hintergründe		
Dateiname	Zweck	Format[1]
background.png	Großflächiges Hintergrundbild	180 × 220
footer.png	Kleiner Streifen über dem Barcode	286 × 15

Bilddateien für Logos und Hintergründe		
Dateiname	Zweck	Format[1]
icon.png	Quadratisches Logo, wenn das Ticket auf dem Home-Screen angezeigt wird	29 × 29
logo.png	Logo des Unternehmens, wird oben links neben dem headerFields angezeigt	160 × 50 (oder schmaler)
strip.png	Zusätzlicher Hintergrund für den Bereich primaryFields	375 × 123
thumbnail.png	Zusätzliches Logo, wird bei Veranstaltungstickets auf der Vorderseite rechts dargestellt	90 × 90
[1] In Pixeln. Zusätzlich kann man alle Bilder für hochauflösende Bildschirme noch in hochskalierten Varianten anlegen und die Dateinamen mit @2x.png kennzeichnen.		

Zur Prüfung

Die Datei pass.json und die Bildchen liegen im Ordner, damit ist das Ticket fast fertig zum Verpacken. Eine letzte verpflichtende Datei fehlt aber noch. Sie heißt manifest.json und dient der Signaturprüfung. Dafür enthält sie ein Inhaltsvezeichnis aller Dateien im Ordner im JSON-Format. Stark gekürzt sieht sie zum Beispiel so aus:

```
{
  "pass.json": "74342dc0649fc1 [...]",
  "icon.png":  "e0f0bcd503f611 [...]",
  "logo.png":  "66a0989dcf0c5c [...]",
  [...]
}
```

Jede Datei, die zum Pass gehört, bekommt hier einen Eintrag mit ihrem Dateinamen und dem SHA1-Hash ihres Inhalts. Die Idee dahinter: Beim Verpacken erzeugt der Ersteller mithilfe seines Apple-Zertifikats für diese Manifest-Datei eine Signatur im Format „PKCS #7 Detached“, das zum Beispiel auch bei E-Mails mit S/MIME zum Einsatz kommt. Die erzeugte Signatur legt er mit dem Dateinamen signature mit in den Ordner. Beim Laden vollzieht die Wallet-App auf dem Telefon diese Schritte zum Test einmal nach und bildet ebenfalls alle Hash-Werte aller Dateien. Hätte zwischendurch jemand eine der Dateien auch nur um ein Bit manipuliert, wäre schlagartig ihr SHA1-Hash ungültig, dadurch die Manifest-Datei und schließlich auch die Signatur.

Auch Reisetickets und Kundenkarten finden in der Wallet-App

ihren Platz.
Den gesamten Signaturprozess darf man aber nicht überbewerten – er stellt letztlich nur sicher, dass ein registrierter Entwickler, der den passTypeIdentifier bei Apple angelegt hat, das Ticket mit seinem Schlüssel signiert hat. Am Einlass beim Konzert oder am Flughafen sagt das aber überhaupt nichts aus, den Identifier sieht das Kontrollpersonal nicht einmal und die Authentifizierung erfolgt lediglich über den QR- oder Barcode. Jeder mit Apple-Account kann Wallet-Pässe basteln und signieren, die wie Bahn- oder Lufthansa-Tickets aussehen. Herausfinden kann man den wahren passTypeIdentifier nur, wenn man den Pass in Ruhe auf einem PC untersucht.

Daraus folgt auch ein ganz legaler Praxistipp für Entwickler, die eigene Pässe bauen und sich inspirieren lassen wollen: Sie sollten einen Pass eines bekannten Anbieters auf dem Telefon öffnen, oben rechts auf die drei Punkte klicken und ihn sich per AirDrop oder Mail an den PC schicken. Dort ändert man einfach die Dateiendung .pkpass in .zip und entpackt das Archiv. Auf diese Weise lässt sich erfahren, wie die großen Anbieter ihre Pässe bauen.

Vom Fließband

Mit diesen Erklärungen und etwas Recherche in bestehenden Tickets sind Sie in der Lage, eigene Tickets und Karten für viele Gelegenheiten zu gestalten. Erklärungen zu allen Funktionen, die Apple sonst noch in den Passes versteckt hat, finden Sie in der Dokumentation über ct.de/yw3e. Nützlich für einige Szenarien, wenn auch nicht mal eben eingerichtet, sind die Themen Mehrsprachigkeit und Updates über einen Server – wie Sie letztere Funktion zum Laufen bringen, würde den Umfang dieses Artikels deutlich sprengen. Bevor Sie sich an solche Herausforderungen machen, sollten Sie Ihren ersten gültigen und signierten Pass erzeugen.

Wir empfehlen dringend, den Signaturprozess mit der Manifest-Datei nicht per Hand durchzuspielen. Die Debugging-

Möglichkeiten sind auf den ersten Blick begrenzt. Entweder ein Ticket ist gültig oder das Telefon weigert sich mit einer nichtssagenden Fehlermeldung. Wer ohnehin schon im Apple-Universum entwickelt, kommt mit dem iOS-Simulator von XCode immerhin an detailliertere Fehlermeldungen im System-Log.

Machen Sie sich das Leben einfacher und suchen Sie nach einer Apple-Wallet-Bibliothek in der Programmiersprache Ihres Vertrauens und erzeugen Pässe lieber programmatisch. Die Bibliotheken funktionieren alle ähnlich: Sie erwarten den privaten Schlüssel und das Zertifikat. Im Programmcode hinterlegt man statische Werte, die bei allen Tickets identisch sind, als Vorlage. Die dynamischen Inhalte (ID, Reisedatum, Name des Reisenden …) werden dann zur Laufzeit eingebacken. Heraus fällt eine verpackte und signierte pkpass-Datei.

Über ct.de/yw3e finden Sie eine Sammlung mit Bibliotheken für verschiedene Programmiersprachen. Für private Experimente kann man anhand der Dokumentationen und mit Kenntnis der jeweiligen Sprache schnell einen Prototypen zusammenstricken. Im Unternehmenseinsatz ist das Generieren solcher Tickets ein Musterbeispiel für den Einsatz eines Microservices – zum Beispiel in Form eines kleinen Containers, der intern ein HTTP-API anbietet und von einem anderen System (das Buchungen verarbeitet) die variablen Ticket-Daten als JSON-Daten annimmt. Zurück liefert der Microservice einen fertigen Pass. Wenn Sie sich für eine solche Umsetzung interessieren, werfen Sie mal einen Blick auf unser Repository zum Artikel – da haben wir einen solchen Microservice als Docker-Container zusammengebaut. Übergeben müssen Sie dem Container nur Ihr Zertifikat und Ihren privaten Schlüssel.

Fazit

Mit seiner Wallet-App und dem eigenen Dateiformat hat Apple ein echtes Alltagsproblem gut und recht flexiblel gelöst. Leider gilt dasselbe wie bei Apps in der Mobilgeräte-Welt:

Möchte man die Android-Nutzer nicht ausschließen, muss man wohl oder übel verschiedene Pässe erzeugen. Apple und Google gehen in diesem Bereich getrennte Wege, statt Standards zu vereinbaren.

Hat man die Hürde mit Entwickler-Account und Zertifikat überwunden, ist der Rest der Arbeit einfaches JSON-Handwerk. Beim Datenformat gilt dankenswerterweise: Vieles kann, wenig muss. Um ein Grundgerüst zu einem Pass zu verschnüren, sollte man dann zu einer Softwarebibliothek in der Sprache des Vertrauens greifen. So steht eigenen digitalen Tickets nichts mehr im Weg. *(jam@ct.de)*

1. Literatur
2. Jan Mahn, Zufall schlägt das System, Ein Plädoyer für UUIDs in Datenbanken, c’t 6/2022, S. 138

Dokumentation und Beispiel-Container: ct.de/yw3e