Ein digitales Ticketing-System für die JavaLand 2022

Ronan Le Tiec, escape GmbH

Endlich fand die JavaLand 2022 wieder in gewohnter Form statt. Doch in puncto Eventmanagement-Tools gab es einige Veränderungen – vor allem unter der Haube mit neuen Software-Bestandteilen. Wir nehmen das digitale Ticketing-System unter die Lupe, das die Organisatoren pünktlich zur JavaLand 2022 in Betrieb genommen haben.

In den Vorjahren fanden die rund 2.500 Besucher der JavaLand ihre Konferenztickets im Normalfall in der Post. Ziel war es, die Tageskasse vor Ort zu entlasten. Größtenteils gelang es. Trotzdem verzeichnete die Kasse in der Regel zu Beginn der Veranstaltung ein hohes Aufkommen. In der Schlange vor den Toren des Phantasialands gesellten sich viele Personen zu den Spontanbesuchern ohne Tickets – Badge vergessen oder verloren, Ticket nicht (rechtzeitig) eingetroffen, Lieferanschrift nicht mehr gültig, Rechnung noch nicht beglichen, Ersatzbesucher für eine verhinderte Person – das waren einige der Gründe fürs Anstehen.

Vorher/nachher

Mit der Auflage 2022 verzichtete die seit 2014 jährlich stattfindende JavaLand erstmalig auf den Ausdruck und Versand von Plastik-Zutrittsausweisen im Kreditkartenformat. Stattdessen erfolgte der Einlass komplett digital: Ein paar Wochen vor Veranstaltung konnten die Besucher ihr individuelles Ticket ganz bequem in ihrem persönlichen Event-Cockpit als Bild oder PDF aufrufen beziehungsweise herunterladen.

Wer das Ticket als Namensschild nutzen wollte, konnte die PDF-Datei als A4 ausdrucken, dieses nach doppelter Faltung in eine passende Hülle einführen und an einem Schlüsselband mit Karabiner um den Hals tragen. Für die Zutrittskontrolle selbst reichte es durchaus aus, das PDF beziehungsweise das Bild des QR-Codes auf einem mobilen Gerät seiner Wahl vorzuzeigen.

An den Eingängen der Veranstaltungslocation erledigte das mit Handscannern gewappnete JavaLand-Personal den Rest. Bis zu 30 Personen pro Minute wurden zu Stoßzeiten eingeschleust.

Web-Applikation für die Zutrittskontrolle

Die für die Zutrittskontrolle entwickelte Web-Applikation besteht aus einer JavaScript-Datei von knapp 300 Zeilen und nutzt die jQuery-Library. Die Software kommt im Frontend mit einer einzigen Seite aus, die sich im Look-and-Feel des Portals shop.doag.org präsentiert. Sie gehört zum Portfolio von shop.doag.org und nutzt weitere existierende Module.

Die Seite ist denkbar einfach aufgebaut *(siehe Abbildung 1)*. Trotzdem oder ausgerechnet deswegen sind auf konzeptioneller Ebene viele Gedanken eingeflossen, die für eine reibungslose und abgestimmte Logistik vor Ort sorgen.

Die Organisatoren der JavaLand sahen unterschiedliche Scanning-Stationen vor, die hauptsächlich zwei Zwecke erfüllen sollen:

- Zum einen sollten mehrere Stationen an den Eingängen in unterschiedlichen Szenarien und mit unterschiedlichen Produkten die Gültigkeit der Tickets überprüfen.
- Zum anderen sollten weitere Stationen die Ausgabe von Giveaways und Getränke-Marken im Park übernehmen.

Für diese Aufgaben kommt die Applikation mit einer Seite aus.

- Im zentralen Content-Bereich befindet sich ein Input-Feld mit Return-Button. Parallel zum Betrieb mit QR-Scanner sollte auch als Fallbacklösung eine manuelle Eingabe über eine auf dem Ticket abgebildete, menschenlesbare ID möglich sein.
- Nach getätigter Abfrage erscheint die Antwort darunter.

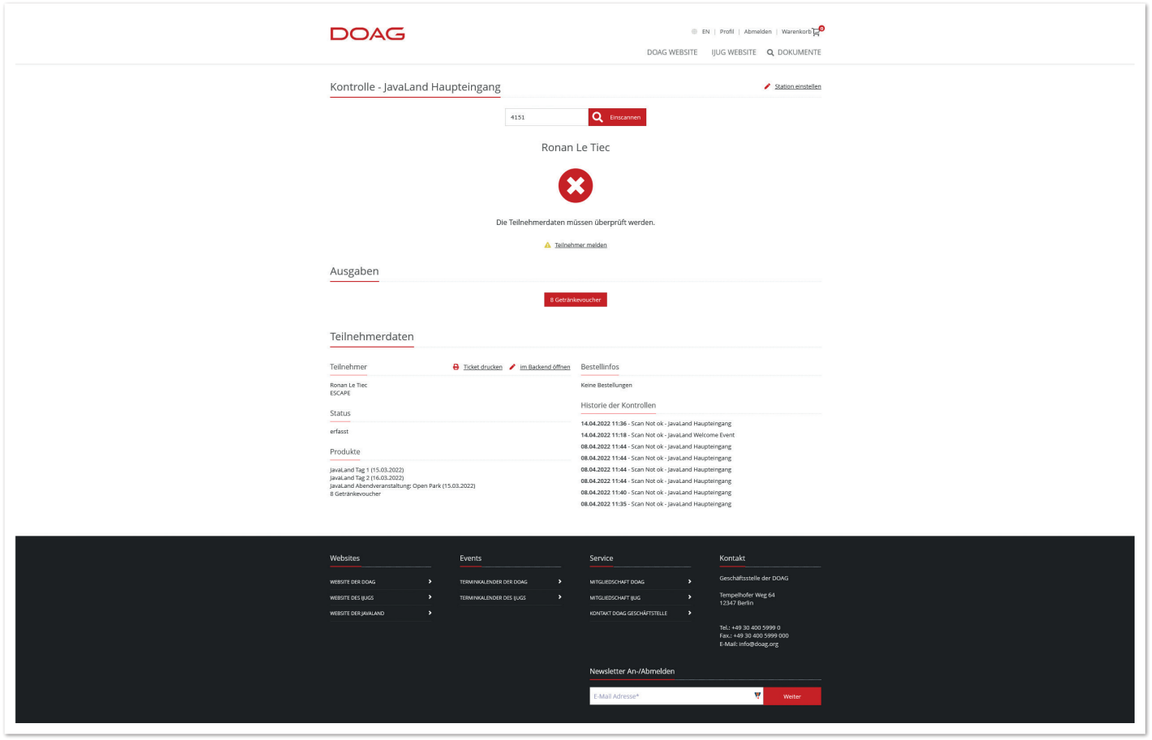

Abbildung 1: So sieht die Maske aus, mit der die JavaLand-Mitwirkenden an der Zutrittskontrolle arbeiten.

- Unter „Teilnehmerdaten“ werden alle Informationen und Aktionen zusammengefasst, die für die Arbeit an den unterschiedlichen Stationen nötig sind.

Darüber hinaus sollten die von den Scan-Stationen vernommenen Aktivitäten aufgezeichnet werden. Hier sollten die Daten über folgende Aspekte Auskunft geben können:

- Verdacht des Ticketmissbrauchs aufgrund von erhöhten Scanning-Aktivitäten überwachen
- Aussagekräftige Zahlen über die Teilnehmerströme nahezu in Real-Time
  - Optimierung der Ressourcenplanung an den Scanning-Stationen
  - Mitteilung über Catering-Zahlen
  - Unterstützung der Referentenbetreuung in den Vortragsräumen

Einfache, intuitive Bedienung

Die rund 50 Helfer der JavaLand konnten dementsprechend mithilfe ihrer mitgeteilten Benutzerdaten die Web-Applikation aufrufen. Da an den Zutrittsstationen vorrangig ehrenamtliche Helfer unterstützten, die im Normalfall nicht mit internen Vorgängen vertraut sind, legte das Team bei der Konzeption großen Wert auf eine einfache und intuitive Bedienung: Die Scan-Stationen sollten ohne Vorkenntnisse benutzt werden können.

Im oberen Content-Bereich entschied man sich für ein naheliegendes Ampelsystem: Bei Grün wird Einlass gewährt, bei Rot wiederum „kein Zutritt“ mit dem Hinweis „Bitte begeben Sie sich zur Tageskasse“. Für Sonderfälle, die es auf Veranstaltungen immer gibt, wurde eine gelbe Ampel mit Kommentarfeld vorgesehen, das zur Kommunikation zwischen den unterschiedlichen Stationen und der Tageskasse genutzt werden konnte. Weiter unten blendet die Applikation diverse Detailinformationen in zwei gleichwertigen Spalten ein.

Performance & Aktualität

An den Kontrollstationen spielt der Faktor Zeit eine entscheidende Rolle: Viele Besucher kommen kurz vor Beginn der Veranstaltung an und möchten trotzdem pünktlich im ersten Vortrag sitzen. Hier ist ein Bottleneck vorprogrammiert.

Deswegen sollte die Applikation an den Scan-Stationen ein zügiges Tempo vorlegen. So legten wir bei der Entwicklung viel Wert auf Performance und effizient geschriebene Datenbank-Abfragen. Mit Fokus auf Aktualität und Schnelligkeit entschieden wir uns dazu, bestimmte Elemente der Seite wie Header und Footer zu cachen. Alle Informationen, die für die Zutrittskontrolle notwendig waren, sollte die Applikation wiederum frisch vom Server per AJAX-Call einholen: Informationen zum Teilnehmer, zur entsprechenden Buchung, zum Zahlungsstatus und den dazugehörenden Produkten, inklusive Leistungszeitraum.

Auf der anderen Seite stellte sich die Frage der Sicherheit. Hier erweiterten wir das bereits existierende „Rollen & Rechte“-Konzept um die entsprechende Rolle „Zutrittskontrolle“ und sorgten für einen passwortgeschützten Zugriff auf die Seite.

Web-App aus den Hosentaschen bedienen

Nun müssen wir uns vorstellen, dass das Zutrittspersonal nicht unbedingt an einem bequem und schön eingerichteten Arbeitsplatz sitzt. Vielmehr arbeiten die Mitwirkenden an provisorisch aufgebauten Stehtischen, an denen Laptop und Scanner aufgestellt sind. Die Helfer stehen Mitte März in den frühen Morgenstunden in der Kälte. Die Zutrittskontrolle sollte auch mit Handschuhen oder aus der Wärme der Jackentasche aus funktionieren: Wir wählten Scanner, die im Tastatur-Wedge-Modus arbeiten können (Daten in virtuelle Tastatureingaben umwandeln) und über einen Dauermodus sowie einen Standfuß verfügten – die Applikation sollte diese Form der Arbeit unterstützen.

Triviale, aber wichtige Details für eine gute und effiziente Arbeit unter solchen Bedingungen:

- Die Benutzer sollten sich nicht darum bemühen müssen, das Input-Feld anzuklicken, bevor sie den QR-Scanner betätigen. Vielmehr sollte hier die Applikation den Datenfluss des Scanners abfangen und in das Input-Feld eintragen.
- Aus demselben Grund sollte bei einer Scanner-Nutzung kein manuelles Bestätigen nötig sein, um die Abfrage abzufeuern.

Ein gut funktionierender QR-Code ist ein Code mit einer guten Auflösung. Dafür sollte er so wenig Details und Informationen wie möglich abbilden. Wir entschieden uns, die Anfragen auf die Domain *https://qr.doag.org* auszulagern.

Wer bereits QR-Scanner in einer Web-App genutzt hat, weiß, dass es eine kleine Herausforderung darstellen kann: Der QR-Scanner findet im Tastatur-Wedge-Modus Anwendung. Betrachtet man unseren QR-Code als String, so schickt das Gerät jedes Zeichen als Tastatureingabe an den Computer. Das hat Vor- und Nachteile.

Ein Plus ist, dass man kein Plug-in oder sonstige Apps braucht, um den QR-Scanner zu verwenden. Man muss aber wiederum zwischen einer von Menschen generierten und einer QR-generierten Tastatureingabe differenzieren. Nach welchen Kriterien kann das erkannt werden? Ein wichtiger Faktor ist die Geschwindigkeit der Eingabe.

Wir haben zwei unterschiedliche Scanner ausgewählt – ein Modell aufgrund der handlichen Größe zum Mitnehmen, ein Modell für den Dauermodus und den Standfuß. Je nach QR-Scanner variiert die Pause zwischen zwei Zeichen-Eingaben zwischen 20 Millisekunden und 150 Millisekunden. Die Eingabegeschwindigkeit ist sogar je nach Modell einstellbar. Wenn ein Mensch es schafft, so schnell zu tippen, dann mit Sicherheit nur kurzfristig. Dementsprechend haben wir ein valides Kriterium für die Unterscheidung der Tastatur-Eingaben.

In unserer Web-Applikation fangen wir jede Tastatureingabe ab, ganz ungeachtet dessen, wo der Cursor-Fokus liegt. Dadurch muss der Benutzer nicht explizit ins Input-Feld klicken.

Bei jeder Eingabe tragen wir das entsprechende Zeichen in eine zentrale Variable ein. In den meisten Fällen wird das Standardverhalten nicht unterbunden. So merkt der Benutzer nicht, dass seine Eingabe abgefangen wird. Wir nutzen ein Time-out von 200 Millisekunden. Werden diese zwischen zwei Eingaben überschritten, beginnt die Analyse der Zeichenkette, die wir zwischengespeichert haben.

Nun zurück zu unseren QR-Codes: Diese haben eine wohlgeformte URL, mit der die Software arbeiten kann. Wenn ein Mensch indes etwas eintippt, dann ignoriert das die Applikation. Die Zeichenkette

wird in diesem Fall nicht als Befehl anerkannt *(siehe dazu Listing 1)*.

Flexibilität & Mobilität als Antwort zum Flaschenhals

Wir sprachen das Thema Flaschenhals bei der Zutrittskontrolle bereits an: Viele Personen müssen in kürzester Zeit eingeschleust werden. Mit performanten Abfragen und Caching kann die App alle zwei Sekunden einen Besucher scannen und im Idealfall bei positiver Rückmeldung durchwinken. Wenn jedoch einige voll besetzte Shuttle-Busse zeitgleich vor dem Eingang halten, kann eine performante Software nur bedingt weiterhelfen.

Um diese Besucherpeaks abzufangen, ist eine punktuelle Aufstockung des Personals nötig. Die App skalierte zwar, aber sie brauchte weitere Eigenschaften und Fähigkeiten, um eine adäquate Antwort zu bieten: Flexibilität und Mobilität waren hier die Stichwörter.

Ein Crew-Mitglied sollte von einer Station mit wenig Zulauf oder einer Aufgabe mit niedriger Priorität abgezogen werden können, um an einer Kontrollstation punktuell zu unterstützen. Deswegen waren neben den stationären Geräten auch mobile Endgeräte eingeplant – ganz nach dem verbreiteten Prinzip des BYOD („Bring your own Device").

Aus diesem Grund ist die Web-Applikation in der Desktop-Ansicht sehr schlicht gestaltet: Hier verzichteten wir bewusst auf eine aufwendige Gestaltung und wählten einen Mobile-first-Ansatz.

Konfiguration der Stationen per QR-Code

Genauso einfach sollte auch die Konfiguration der Stationen für das mobile Personal sein. Hat ein Benutzer die Rolle „Zutrittskontrolle" inne, ist er in der Lage, seine zugeteilte Scan-Station in den App-Einstellungen über eine Dropdown-Liste vorzunehmen. Aber es sollte noch einfacher sein: An den Orten, an denen gescannt werden sollte, fanden die Nutzer einen Aushang mit QR-Code. Nach Anmeldung auf die Website mit der entsprechenden Rolle konnten sie den QR-Code nach dem gleichen Prinzip wie bei den Tickets einscannen. Werfen wir wieder einen Blick in den Code *(siehe Listing 2)*. Wir waren bei der wohlgeformten URL stehengeblieben, die unsere QR-Codes generieren. Ein direkter Aufruf der URL kam bei unseren Überlegungen nicht infrage. Er hätte ein Neuladen der Seite mit sich gezogen. Aus Performance-Gründen galt es, andere Wege zu finden.

Deswegen erfolgt die Analyse der QR-Code-URL clientseitig. Ein regulärer Ausdruck übernimmt die Analyse der URL. Eine Weiterverarbeitung erfolgt nur unter der Bedingung, dass diese wohlgeformt ist. Ist dies der Fall, so extrahiert die Software folgende Informationen:

- Typ der Anfrage: Handelt es sich um eine Scan-Station oder um einen Teilnehmer?
- Wie lautet die passende ID?

Erkennt die Applikation eine Teilnehmerabfrage, so ruft sie die zentrale Funktion zur Analyse der Teilnehmerinformationen auf. Dabei trägt sie die Teilnehmer-ID in das Input-Feld ein. Per AJAX werden alle Informationen nachgeladen.

Wird eine Scan-Station erkannt, ruft die Software die zentrale Funktion zur Einstellung der Station auf. Parallel dazu wird die Station-Aktualisierung in einem Cookie festgeschrieben, sodass sich der Browser die Station-Auswahl auch nach einer Arbeitsunterbrechung merkt.

Auch hier achtete das Team penibel auf eine für den Nutzer einfache Bedienung: Die Stationen kann nur der Organisator im Backend einrichten. Dort werden auch die entsprechenden QR-Codes generiert und heruntergeladen. Die zahlreichen Helfer kriegen diese Arbeitsschritte nicht mit und können sich auf ihre Aufgabe konzentrieren.

```
this.initBarcodeScanner = function()
{
  self.scannerInput = "";
  $(document).on('keydown', self.handleKeyPress);
}

this.handleKeyPress = function(event)
{
  // clear the time out and start it again
  clearTimeout(self.inputKeyTimeout);
  self.inputKeyTimeout = setTimeout(self.analyzeKeyInput, 200);

  // we save the key pressed in a central buffer if it's a
  // single key (and not for instance 'Enter')
  event = event.originalEvent;
  if(event.key.length == 1)
  {
     self.scannerInput += event.key;
  }

  var _focused = $(document.activeElement);
  var _inputting = _focused.get(0).tagName.toLowerCase()==="textarea" || _focused.get(0).tagName.
toLowerCase()==="input";

  // we prevent some default behaviour in certain conditions.
  if (!_inputting && (self.scannerInput.length > 2) || _inputting && event.key == "Enter")
  {
     event.preventDefault();
     return;
  }
};
```

Listing 1: So analysiert die Applikation die Tastatureingaben


```
this.analyzeKeyInput = function()
{
  let regex = /^http[s]?:\/\/qr\.[^\/]+\/(p|cs)([0-9]+)$/g;

  // do we have a qr-URL?
  if (self.scannerInput.match(regex))
  {
     // extract the request type and id
     var result = regex.exec(self.scannerInput);
     var type = result[1];
     var id = result[2];

     //reset the participant input
     self.element.find('[name=participantIdSelection]').val("");

     // trigger the update of the participant infos or station
     if(type == "p")
     {
        self.updateParticipant(id);
     }
     else if(type == "cs")
     {
        self.updateStation(id);
     }
  }
  self.scannerInput = "";
}
```

Listing 2: Die Analyse unserer URL anhand eines regulären Ausdrucks

Ausgabestationen

Nach dem gleichen Prinzip funktioniert die Ausgabe von Give-aways. Hierfür erstellen und konfigurieren die Organisatoren Produkte und Unterprodukte im Backend und weisen diese den bestimmten Teilnehmern zu. Die Mitarbeiter der Ausgabestationen sehen für jedes definierte Produkt einen Button, der entweder rot oder ausgegraut ist, je nachdem, ob ein Anspruch besteht und/oder die Ausgabe stattgefunden hat.

Teilnehmer scannt sein Badge selbst – na und?

Eine kleine, aber wichtige Anekdote zum Schluss: Heutzutage hat jeder einen mobilen Scanner in der Hosen- oder Handtasche. Deswegen stellte sich folgende Frage: Wie gehen wir mit Scan-Vorgängen um, die nicht im Kontext unserer Web-Applikation erfolgen?

Hier ist die Antwort: Ein Teilnehmer scannt sein eigenes Ticket mit dem Smartphone. Die Scanner-App öffnet den Browser und die QR-URL wird geladen. Die URL hat zu diesem Zeitpunkt folgende Form: *https://qr.doag.org/p123*

Im ersten Schritt erfolgt eine Weiterleitung zur Domain unserer Kontroll-App. Wir befinden uns nun serverseitig im Kontext der Webseite, immer noch im QR-Request-Handling. Unsere URL sieht dann folgendermaßen aus: *https://shop.doag.org/qr/p123*

Der Server prüft anschließend den Login-Zustand des Nutzers. Ist dieser nicht eingeloggt, leitet die Teilnehmerabfrage zur entsprechenden Event-Seite weiter. Ein Teilnehmer erhält also über das Scannen des eigenen Badges Informationen zur Veranstaltung.

Ist der Nutzer eingeloggt, dann prüft die Applikation seine Rechte. Wenn er die Zutrittskontrolle einsehen darf, wird er zu der Seite weitergeleitet. In diesem Schritt wird die URL um zusätzliche Hash-Parameter ergänzt. So erfährt unsere Web-Applikation, welche Teilnehmer-Daten aufgerufen oder welche Station eingestellt werden soll. Hier ein Beispiel dafür: *https://shop.doag.org/urlzutrittskontrolle/#participantId.123*

```
1. https://qr.doag.org/p123
2. https://shop.doag.org/qr/p123
3. https://shop.doag.org/urlzutrittskontrolle/#partici
pantId.123
```

Listing 3: Die Verwandlung einer URL

Fazit

Manchmal ist die größte Herausforderung in einem Projekt, die Dinge einfach zu gestalten. Diese kleine Applikation ist ein Paradebeispiel dafür. Ein gut lesbarer, wiederverwendbarer Code ist wichtig. Genauso wichtig ist es als Entwickler, gute Kenntnisse über den Kontext zu haben, in dem die Applikation angewandt werden soll. In unserem Fall hoffen wir, dass wir zur Customer Experience der JavaLand beitragen konnten.

Ronan Le Tiec
escape GmbH
ronan.letiec@escape-germany.de

Ronan ist Senior Full-Stack Web Developer und arbeitet seit zwölf Jahren bei der Digitalagentur escape. Er war an der Entwicklung des firmeneigenen Werkzeugkastens beteiligt. Der gebürtige Franzose hat eine kleine Schwäche für Gestaltung, worüber Backend-affine Kollegen sehr glücklich sind. Sein Schwerpunkt liegt inzwischen in der agilen Anforderungsanalyse und -Planung in Kundenprojekten. Ronan hat in Frankreich, Deutschland und Spanien Computer Sciences Engineering studiert und fühlt sich sowohl beim Coden als auch in sozialen Interaktionen in mehreren Sprachen wohl.