# Projektdokumentation Software Projekt

~ Max Knerrich, Benjamin Mehl

## 1. Inhalt des Projekts

Unser Projekt war eine spannende Kooperation mit der renommierten Eventagentur 0711. Gemeinsam konzipierten wir eine Cross-Plattform App für die angesagte Eventreihe "Saturdays" im bekannten Club Perkins Park in Stuttgart. Die App bietet ein praktisches Check-In System, mit dem die Teilnehmer Punkte sammeln und attraktive Prämien erhalten können, wie zum Beispiel kostenlose Getränke und vieles mehr.

Darüber hinaus enthält die App umfassende Informationen zu den anstehenden Events sowie den DJs. Nutzer finden hier außerdem Links zur Reservierungsplattform und zu weiteren interessanten Inhalten, wie Social Media Konten der Veranstalter.

Um die Events und Prämien gezielt verwalten zu können, haben wir eine intuitive Webanwendung entwickelt, die es den Mitarbeitern der Eventagentur ermöglicht, diese Daten komfortabel zu verwalten. Zusätzlich bietet die Admin Webanwendung die Möglichkeit, Push-Benachrichtigungen an die App-User zu senden und sie so an bevorstehende Events oder andere Highlights zu erinnern.

## 2. Wie wurde es umgesetzt?

Um uns auf die Nutzerfreundlichkeit der beiden Anwendungen zu fokussieren, haben wir von Anfang an beschlossen, ein BaaS (Backend as a Service) einzusetzen. Wir erkannten, dass es in diesem Bereich verschiedene Optionen gibt, darunter Firebase, Supabase und Appwrite. Daher führten wir eine Recherche durch, um herauszufinden, welche Option am besten den Anforderungen unseres Projekts entspricht.

Unsere Hauptkriterien waren dabei, dass wir nicht an einen bestimmten Cloud Provider wie Amazon Web Services oder Google Cloud Platform gebunden sein wollten. Stattdessen wollten wir das Backend selbst hosten können, um maximale

Flexibilität zu gewährleisten. Zudem war es uns wichtig, dass die Entwicklung mit der Backend-Schnittstelle einfach und unkompliziert gestaltet ist.

Firebase wurde damit von vornherein aus den Optionen ausgeschlossen, da es ausschließlich von Google angeboten wird und keine Möglichkeit besteht, es selbst zu hosten. Supabase wiederum erschien uns in Bezug auf das selbstständige Hosting als komplexer und weniger geeignet für unser Projekt. Schließlich entschieden wir uns daher für Appwrite.

Durch die Entscheidung für Appwrite waren wir in der Lage, unsere Anforderungen vollständig zu erfüllen. Die Integration verlief reibungslos, und die einfache Handhabung der Backend-Schnittstelle erleichterte die Entwicklung erheblich. Mit Appwrite konnten wir deutlich Zeit sparen in Themen wie Sicherheit oder Monitoring unseres Backends.

Da wir beide bereits viel Erfahrung mit Webtechnologien und Javascript-Frameworks hatten, suchten wir nach einer Möglichkeit, dieses Wissen in die App-Entwicklung einzubringen. Wir entschieden uns für Capacitor, ein Framework, das es ermöglicht, Cross-Plattform-Apps für Android und iOS mithilfe von HTML, Javascript und CSS zu erstellen. Um uns mit der Entwicklung in Capacitor vertraut zu machen, entwickelten wir einen Prototypen, in dem wir erste Native APIs wie einen QR Code Scanner einsetzten. Dabei stellten wir fest, dass wir dank Capacitor nicht auf unsere vertrauten Web-Technologien verzichten mussten, was den Entwicklungsprozess schnell und effektiv gestaltete.

## 3. Wie war der Projektverlauf?

Um ein umfassendes Verständnis für den Umfang unseres Projektes zu entwickeln, führten wir zunächst eine Online-Besprechung mit unserem Partner durch. Dabei erhielten wir erste Einblicke in die zu entwickelnde Applikation. Anschließend trafen wir uns persönlich vor Ort, um sicherzustellen, dass unsere Anforderungen mit den Erwartungen unseres Partners übereinstimmen. In diesem Treffen diskutierten wir auch erste Ansätze zur Umsetzung.

Wie bereits erwähnt, erstellten wir ein Proof of Concept in Form eines Prototypen, um die wesentlichen Funktionen unserer App zu demonstrieren. Hierbei konzentrierten wir uns auf die Umsetzung der wichtigsten Features. Dabei wagten wir uns an die Integration der Schnittstelle zu Appwrite, das Scannen und Auslesen von QR-Codes sowie das Auslesen von NFC-Tags. Allerdings stellten wir fest, dass das Plugin für NFC-Tags im Hinblick auf iOS und die Entwicklung für Apple-Geräte einige Schwierigkeiten aufwies. Auch in den neuesten

Android-Versionen erfordert es leichte Anpassungen, um reibungslos zu funktionieren.

Der Prototyp erwies sich als äußerst bedeutsam für unsere nachfolgende Projektplanung. Er verschaffte uns einen groben Überblick darüber, welche Features mehr Zeit in Anspruch nehmen würden als andere. Somit konnten wir unsere Planung besser ausrichten.

Gleichzeitig zur Erstellung unseres Prototyps haben wir ein Wireframe für die App entworfen und darauf aufbauend einen High-Fidelity-Prototypen erstellt. Diesen konnten wir anschließend auch mit unserem Partner von 0711 teilen, um abzustimmen, welche Aspekte noch Anpassungen benötigen. Dieser Schritt war von großer Bedeutung, um bereits vor der eigentlichen Entwicklung der App User Flows zu testen. Darüber hinaus ermöglichte uns das ausgearbeitete Design eine zügige Weiterentwicklung und ersparte uns zeitraubende Entscheidungen während der Entwicklungsphase der Anwendung.

Anschließend begannen wir damit, das Design in der App umzusetzen. Wie bereits erwähnt, verlief dieser Schritt äußerst zügig, wodurch wir im Anschluss ausreichend Zeit für die Entwicklung unseres Backends und anderer Funktionen wie Push-Benachrichtigungen zur Verfügung hatten. In Bezug auf die Backend-Logik begannen wir damit, Serverless-Funktionen zu erstellen. Parallel dazu arbeiteten wir an Aspekten wie Authentifizierung und dem Administrationsbereich. Sobald der Administrationsbereich grundlegende CRUD-Funktionen aufwies, war es möglich, die App im Zusammenspiel mit dem Administrationsbereich zu testen. Dies half dabei, Unstimmigkeiten zu erkennen und zu beheben.

Nachdem wir die anfänglich definierten Features der App umgesetzt hatten, konnten wir uns auf die Behebung von Fehlern und das Durchführen von weiterem Refactoring konzentrieren. Dadurch steigerten wir die Wartbarkeit und Benutzerfreundlichkeit der App.

## 4. Was haben wir gelernt?

### Push-Notifications nur über externe Services

Eine Anforderung an die App war das senden von Push-Notificationen an die User.

Zu Beginn gingen wir davon aus, dass dies einfach über eine Native API innerhalb der App möglich wäre, nach einer Recherche fanden wir aber heraus, dass man zum senden von Push Notifications einen externen Provider benötigt. Schlussendlich entschieden wir uns dann für Firebase Cloud Messaging entschieden, da dies auch unabhängig von Firebase BaaS verwendbar ist

**Kommunikation mit Kunden**

Eine Erkenntnis, die wir aus dem Projekt gezogen haben, ist wie wichtig eine klare Absprache zu Beginn eines Projekts ist. Insbesondere in Bezug auf Farben, visuelle Ästhetik, Fortschritt und Funktionalitäten sollte eine umfassende Klarheit herrschen. Indem diese Aspekte zu Beginn detailliert besprochen und festgelegt werden – praktisch einfriert –, lassen sich spätere Missverständnisse und unerwartete Änderungen minimieren. Dies verhindert nicht nur Verzögerungen im späteren Verlauf des Projekts, sondern fördert auch ein reibungsloses Zusammenspiel zwischen den Kundenerwartungen und dem tatsächlichen Fortschritt.

Die Erfahrung hat gezeigt, dass regelmäßige Check-ins mit dem Kunden ein Eckpfeiler für erfolgreiche Projektabschlüsse sind. Die Einrichtung eines klaren Plans für regelmäßige Überprüfungen – beispielsweise im Zwei-Wochen-Rhythmus – ermöglicht es, den Fortschritt zu zeigen, Feedback einzuholen und gegebenenfalls Anpassungen vorzunehmen. Dies schafft nicht nur Transparenz, sondern stärkt auch das Vertrauen zwischen uns und dem Kunden. Der geplante Austausch hilft, mögliche Unstimmigkeiten frühzeitig zu identifizieren und gemeinsam Lösungen zu finden, wodurch das Projekt auf Kurs gehalten wird und potenzielle Verzögerungen vermieden werden.

**Ökosystem und Features von Technologien vor Beginn sorgfältig prüfen**

Eine weitere wichtige Erkenntnis für uns war, dass es wichtig ist, die gewählten Technologien sorgfältig für alle benötigten Features und Plugins zu prüfen. Wir haben festgestellt, dass es Fälle geben kann, in denen notwendige Plugins, wie zum Beispiel ein NFC-Plugin für Capacitor, entweder nicht verfügbar sind oder nicht den gewünschten Qualitätsstandards entsprechen, da sie eventuell deprecated sind. In einem Fall mussten wir ein älteres, nicht mehr aktiv entwickeltes Plugin für neuere Versionen von Android und iOS patchen, um die benötigte Funktionalität sicherzustellen. Diese Erfahrung unterstreicht die Wichtigkeit der sorgfältigen Auswahl von Drittanbieter-Tools & Technologien. Hätten wir unsere gewählten Technologien sorgfältiger geprüft, hätten wir eventuell eine andere Entscheidung getroffen (z.B. React Native statt Capacitor) was uns eventuell Arbeit erspart hätte.