BASEL LANDSCHAFT
BILDUNGS-, KULTUR- UND SPORTDIREKTION
AMT FÜR BERUFSBILDUNG UND BERUFSBERATUNG

# Implementierung der User Experience eines Parkplatzreservationssystems mit React

09.03.2023 – 30.03.2023

**Kandidat**
Lima Raquel
**Betrieb (=Durchführungsort)**
Adobe Research (Schweiz) AG
Peter Merian-Strasse 80, 4052 Basel

T 0786682455 (am besten erreichbar)
G +41612265504
M rrlima.raquel@gmail.com

**BerufsbildnerIn/ Lehrfirma**
Jäggi Dominique
Adobe Research (Schweiz) AG
Barfüsserplatz 6, 4051 / Basel
T +41797424766 (am besten erreichbar)
G +41612265504
M djaeggi@adobe.com

**Verantwortliche Fachkraft**
Hertach Hannes
Adobe Research (Schweiz) AG
Peter Merian-Strasse 80, 4052 / Basel
T 0763974011 (am besten erreichbar)
G 061 226 55 00
M hhertach@adobe.com

**Hauptexperte**
Tobler Felix
T +41 76 399 00 72 (am besten erreichbar)
G +41 76 399 00 72
M felix.tobler@outlook.com

**Nebenexperte**
Georgy Michel
T 076 382 06 28
G 061 261 43 48
M mg@georgy.ch

Zuteilung Schulklasse
IAP 19-A

Zusätzliche Kandidaten-Informationen

# Implementierung der User Experience eines Parkplatzreservationssystems mit React

09.03.2023 – 30.03.2023

Kontakt Chefexperte (nur in Notfällen)
Daniel Gyger
T 079 260 15 13
M daniel.gyger@itec.ch

**Arbeitsbereiche**

- Applikationsentwicklung OO
- Macintosh OS
- JavaScript

# Implementierung der User Experience eines Parkplatzreservationssystems mit React

09.03.2023 – 30.03.2023

## Ausgangslage

Adobe zieht Anfang 2023 in ein neues Gebäude in Basel um. Am neuen Standort stehen den Mitarbeitenden eine begrenzte Anzahl Parkplätze zur Verfügung. Die Parkplätze müssen vor dem Gebrauch halbtagesweise reserviert werden. Die Reservation soll durch eine Web-Applikation erfolgen. Während dem Reservationsprozess sollen Nutzer/innen ein Datum, eine Uhrzeit und ein gespeichertes Fahrzeug wählen können (um jeder Reservation ein Kontrollschild zuweisen zu können).
Einige der Parkplätze verfügen zudem über eine Ladestation für Elektrofahrzeuge. Solange nicht alle Parkplätze ohne Ladestation belegt sind, sollte das System Fahrzeugen mit Verbrennungsmotor keine Reservationen von Parkplätzen mit Ladestation erlauben.

Nebst dem eigentlichen Reservationsprozess bietet das System auch ein Übersichts-Dashboard. In dieser Ansicht können Nutzer/innen sehen, wann sie Parkplätze reserviert haben, und existierende Reservationen wieder stornieren.

Zudem bietet das System für die Administratorin einen Überblick über den Status aller Parkplätze und Nutzer/innen. Im Admin-View gibt es auch weitere Aktionen, etwa kann ein Parkplatz für ein Datum als «disabled» markiert werden, wenn beispielsweise Bauarbeiten stattfinden.

Letztlich werden im Parkhaus über jedem Parkplatz Displays installiert, welche den Status des jeweiligen Parkplatzes im Reservationssystem anzeigt.

## Detaillierte Aufgabenstellung

In dieser Arbeit wird die Benutzeroberfläche des Parkplatzreservationssystems umgesetzt. Die Benutzeroberfläche wird in Javascript und dem React Framework implementiert. Die React-Applikation kommuniziert via HTTP API mit einem Backend. Da die Benutzeroberfläche relativ umfangreich ist, wird die Implementierung auf die wichtigsten Features begrenzt. Die Features, die umgesetzt werden müssen, sind in den folgenden Abschnitten beschrieben. Das Backend, die Datenbank und die Displays über dem Parkplatz sind nicht Teil dieser Arbeit.

Eine OpenAPI Spezifikation ist vorhanden, welche die API des Backends beschreibt (https://github.com/berufsbildung-basel/parkit-spec) (auch im Anhang). Für die Entwicklung der Benutzeroberfläche steht eine Mock-API, welche die OpenAPI Spezifikation umsetzt, zur Verfügung (https://github.com/berufsbildung-basel/parkit-frontend-boilerplate) (auch im Anhang).

Ebenfalls besteht bereits ein Adobe Experience Designer (XD) Mockup des Designs und Layouts der Benutzeroberfläche (https://xd.adobe.com/view/43abcff4-bced-4b8b-ac03-e1e67d438d62-16f8/ - um den File im Browser anzusehen, ist ein Login mit einem Adobe-Konto erforderlich. Ein solches kann kostenlos erstellt werden, und um den File anzusehen werden keine speziellen Rechte

# Implementierung der User Experience eines Parkplatzreservationssystems mit React

09.03.2023 – 30.03.2023

benötigt. Alternativ ist der selbe File auch im Anhang auf PkOrg, und kann mit der Adobe XD Destop-Applikation geöffnet werden).

Die Requirements sind wie folgt:

Login:

Als erstes wird ein Login-Screen angezeigt, wenn ein/e neu/e Nutzer/in auf die Applikation zugreift. Die User-Authentication wird später mit Adobe's SSO integriert. SSO-Integration ist aber nicht im Umfang dieser Arbeit enthalten. Daher darf das Login mocked werden. Die Credentials können client-seitig überprüft werden, ob sie mit einem Test-User übereinstimmen, zum Beispiel Username test@adobe.com und Password testPassword. Wenn die Credentials nicht stimmen, muss eine entsprechende Fehlermeldung angezeigt werden. Wenn die Credentials stimmen, können sie client-seitig gespeichert werden, zB im Browser LocalStorage.

• Wenn die client-seitige Validierung erfolgreich war, muss der Username und das Passwort gespeichert werden (zum Beispiel im Browser Local Storage). Bei allen API-Requests muss der Username und das Passwort als http Basic Auth Header mitgeschickt werden.

Übersicht:

Wenn der User eingeloggt ist, wird eine Übersicht angezeigt. Diese Ansicht entspricht dem Adobe XD mockup. Auf der linken Seite wird eine Navigationsleiste angezeigt. Hier sind die Optionen «Dashboard», «Reservations», «Parking Overview» und «Vehicles» verfügbar

Auf der Page «Dashboard» wird folgendes angezeigt:

• Eine Übersichtstabelle aller zukünftigen Reservationen des Nutzers/der Nutzerin, in der jede Zeile Datum, Uhrzeit, Fahrzeugname und Kontrollschild einer Reservation zeigt
• Ein UI-Element zeigt die aktuelle Anzahl von verfügbaren Parkplätzen an
• Ein UI-Element zeigt die aktuelle Anzahl von kommenden Reservationen an
• Diese Informationen werden vom Endpunkt listReservations entnommen. Dieser Endpunkt liefert alle Reservationen, auch vergangene. Die Resultate müssen daher gefiltert werden.

Auf der Page «Reservations» wird folgendes angezeigt:

• Eine Tabelle zeigt alle Reservationen des Nutzers/der Nutzerin an. Es wird Datum, Uhrzeit,

# Implementierung der User Experience eines Parkplatzreservationssystems mit React

09.03.2023 – 30.03.2023

Fahrzeugname und Kontrollschild angezeigt.
• Diese Informationen kommen vom selben listReservations Endpunkt. Eine Spalte der Tabelle muss den Status der Reservation (ob vergangen, kommend oder storniert) anzeigen.
• Ein Button lässt den Nutzer die Tabelle nach Status filtern. In dem Fall werden nur vergangene, kommende oder stornierte Reservationen respektive angezeigt.
• Ein Button «Create Reservation» führt zur «Parking Overview» page.
• Stornierung: Für jede Reservation muss ein Button zur Stornierung existieren. Dieser ruft den API-Endpunkt cancelReservation auf.

Auf der Page «Parking Overview» wird folgendes angezeigt:

• Eine grafische Übersicht aller Parkplätze, und deren Status (frei, besetzt, vom Administrator als disabled markiert, Elektrofahrzeug-Ladestation vorhanden oder nicht). Das Datum für die Status-Anzeige kann durch ein Date-Picker geändert werden (max. 2 Wochen in die Zukunft). Diese Informationen stammen von den Endpunkten /parking-spots/availability und /parking-spots
• Durch einen Klick auf ein freies Parkfeld öffnet sich rechts ein Panel zur Erstellung einer Reservation. Hier kann der/die Nutzer/in die Dauer der Reservation wählen (Vormittag, Nachmittag, ganzer Tag), sowie das entsprechende Fahrzeug (Endpunkt /vehicles)

Der Link zu «Vehicles» öffnet rechts ein Panel, wie bei «Profile».

• Hier werden alle Fahrzeuge, die dem Nutzer/der Nutzerin gehören, angezeigt. Dazu wird der Endpunkt /vehicles aufgerufen. Die Fahrzeuge haben einen Namen und ein Kontrollschild.
• Durch einen Button neben jedem Fahrzeug kann das Fahrzeug gelöscht werden. Dazu wird die Operation /removeVehicle aufgerufen.
• Durch einen Button kann ein weiteres Fahrzeug hinzugefügt werden. Dazu wird die Operation createVehicle aufgerufen.
Oben rechts ist ein Profile-Image des Users/der Userin. Dieses Bild darf für die Arbeit ein Placeholder sein (hard-coded). Durch den Klick auf dieses Bild wird folgendes angezeigt:
• Es öffnet sich rechts ein Panel, der Informationen zum aktuellen Nutzer anzeigt: E-Mail-Adresse, Name und Benutzername. Dies wird dem Endpunkt /users/{id} entnommen. Als User-ID Parameter für den API-Request darf in der IPA der Benutzername verwendet werden, da diese ID später aus dem SSO-System stammt.

Testing:

Wichtig ist, dass unter unvorhergesehenen Applikationszuständen eine gute User Experience

gewährleistet ist. Der/die Nutzer/in muss weitergeführt werden, wenn die API eine Fehlermeldung oder keine Daten zurückgibt. Alle möglichen Fehlermeldungen der API sind im OpenAPI File dokumentiert. Da die API eine grosse Anzahl an potentiellen Fehlern liefern kann, müssen nur folgende 5 Fälle abgedeckt werden (diese Erwargung ist auch als invividuelles Kreterium I2 dokumentiert):

• Bei allen Requests muss der/die Nutzer/in zuück zum Login-Screen geführt werden, wenn eine 401 Response erhalten wird.
• Wenn keine kommenden Reservationen im System sind, darf auf der Übersichtssseite nicht eine leere Tabelle angezeigt werden, sondern eine entsprechende Mitteilung, dass keine Reservationen gefunden wurden.
• Beim Endpunkt «createReservation» muss der Status-Code 409 – Conflict abgedeckt sein, und in der Benutzeroberfläche eine entsprechende Fehlermeldung angezeigt werden.
• Zwei zusätzliche Error-Responses der API, welche von der Kandidatin als sinnvoll empfunden werden, müssen abgedeckt sein.

Um diese Fehler in der Mock-API zu testen, können im Request Header verwendet werden, um eine gewisse Antwort der API zu erzwingen. Die Boilerplate-Applikation und das Mock-Backend (beides Teil der Vorarbeit) sind so eingerichtet, dass alle variablen im Browser Local Storage, welche mit «x-test-» starten, als Header ans Mock-Backend geschickt werden. Das Mock-Backend versteht bereits folgende Header, welche während der Vorarbeit noch ergänzt werden müssen, um die zwei zusätzlichen Testfälle der Kandiadtin zu unterstützen:

• x-test-response-status: Das Backend antwortet mit diesem Response-Code
• x-test-response-text: Das Backend antwortet mit diesem Text als Status-Text
• x-test-empty-response: Der Endpunkt «listReservations» antwortet mit einem leeren Array als Body

Dokumentation:

Als technische Dokumentation muss das Github-Repository einen Readme-File in Markdown Format enthalten. Dieses Readme dient dazu, dass andere Entwickler nach der IPA den Code verstehen sowie daran arbeiten können. Der Readme File muss kurz den Zweck der Applikation beschreiben, eine grobe Übersicht über die Architektur aufzeigen und eine Anleitung zur Installation beinhalten.

Methoden und Funktionen, welche Business-Logik umsetzen, sowie React Function Components (props müssen dokumentiert sein), müssen mit JSDOC die Inputs und Outputs dokumentieren.

# Implementierung der User Experience eines Parkplatzreservationssystems mit React

09.03.2023 – 30.03.2023

Davon ausgenommen sind return-values von React-Components.

Die technische Dokumentation (Kommentare, JSDoc, Readme), sowie aller Code, wird auf Englisch geschrieben.

Als Teil der schriftlichen Arbeit müssen Entscheidungen, die während der Implementation getroffen werden, dokumentiert und begründet werden.

Out of Scope:

Folgende Punkte werden bei der IPA bewusst nicht bearbeitet:
• Build System: Die Kandidatin muss verstehen, wieso ein Build-System benötigt wird, und muss die Commands, die für die Arbeit benötigt werden, selbstständig verwenden können. Allerdings ist das Build-System für die Arbeit vorgegeben, und sie muss selbst keine Konfigurationsänderungen vornehmen können.
• Das Backend, die Datenbank und die Displays über dem Parkplatz sind nicht Teil dieser Arbeit. Die Datenbank und Displays sind einfach abzugrenzen - diese werden von dieser Arbeit nie berührt, da der Zugriff auf diese nur durch das Backend erfolgt. Dieser Zugriff ist für das Frontend (=diese Arbeit) trasparent. Das Frontend interagiert mit dem Backend durch eine API. Die API zum Backend wird als Mock-API implementiert. Dies erfolgt während der Vorarbeit.
• Das Admin-Dashboard ist nicht Teil der Arbeit und wird nicht implementiert.
• Integration mit Adobe’s SSO Login System ist nicht Teil der Arbeit, daher darf der Authentifizierungsprozess client-seitig mocked werden.
• Responsive Design: Es wird nur die Desktop-Version implementiert
• Deployment: Die Applikation wird nur lokal entwickelt. Sie muss nicht auf einen Web-Server deployed werden oder als Produktions-Build gebaut werden.

## Mittel und Methoden

• MacOs
• Node und NPM-Umgebung
• Chrome Browser
• VSCode IDE
• Code-Versionierung durch Git
• Die Benutzeroberfläche wird in React und Javascript implementiert
• Die Grafischen Elemente im User Interface wird mit dem React MUI Framework implementiert

# Implementierung der User Experience eines Parkplatzreservationssystems mit React

09.03.2023 – 30.03.2023

## Vorkenntnisse

• Die Kandidatin hat bereits mit React gearbeitet, sowohl an Projektarbeiten sowie auch als Teil ihres Teams
• Sie hat aus einem Projekt Erfahrung mit der MUI Component-Library, welche sie für die Arbeit verwenden wird.
• Sie hat Erfahrung in der Frontend-Entwicklung und der Benützung von HTTP RESTful APIs.

## Vorarbeiten

• Es besteht ein Git-Repository welches den React-Boilerplate (Build-System, Linter) enthält. Ebenfalls besteht die Mock-API im selben Git-Repository.
• In der Mock-API wurden bereits für die benötigten Endpunkte Handler-Funkionen geschrieben
• Es wurden UI-Mocks in Adobe XD bereits erstellt, als Vorlage für die Benutzeroberfläche
• Ein Dependency-Injection Framework (mit React-Context API) ist bereits im React-Code enthalten, um Dependencies wie den API-Client in der Applikation global verfügbar zu machen.

## Neue Lerninhalte

• Die Kandidatin hat zuvor noch nie mit einer Mock-API gearbeitet.
• Sie hat noch nie OpenAPI-Spezifikationen verwendet.
• Sie hat nur wenig Erfahrung mit Testing
• Sie hat bisher mit React Class Components gearbeitet. Die React Hooks API ist ihr neu. Obwohl nicht vorgeschrieben ist, dass sie diese API verwendet, ist der Kontakt dazu möglicherweise nicht vermeidbar, zum Beispiel bei der Verwendung von gewissen Libraries.

## Arbeiten in den letzten 6 Monaten

Eigenes Projekt: Arbeit mit React und MUI an einer Chat-Applikation über ca 8 Tage
Arbeit mit React an einem Adobe-Produkt als Teil eines Scrum-Teams über ca 4 Monate
Schularbeit: Youtube Video Download Applikation in React implementiert

# Individuelle Kriterien

Auf den folgenden Seiten werden die individuellen Kriterien aufgeführt, welche durch die verantwortliche Fachkraft für diese IPA festgelegt wurden.

# Individuelle Kriterien

| **Leitfrage**<br>1 | **JSDoc Dokumentation**<br><br>Methoden und Funktionen sind mit JSDoc Kommentaren (input-output types) dokumentiert. Erwartung:<br>- Alle Methoden und Funktionen mit Business-Logic, und React Function Components, haben JSDoc kommentare (vollständigkeit)<br>- Die Types in den JSDocs sind korrekt: Wenn die Funktion mit den dokumentiereten Types aufgerufen wird, entsteht kein Type-Fehler während der Ausführung<br>- Die Kommentare dokumentieren präzise und prägnant den Zweck der Funktion in 1-3 (max) Zeilen. |
|---|---|
| Gütestufe 3 | Alle 3 Punkte sind erreicht. |
| Gütestufe 2 | 2 von 3 Punkten sind erreicht. |
| Gütestufe 1 | 1 von 3 Punkten sind erreicht. |
| Gütestufe 0 | Es bestehen keine JSDoc Kommentare. |

Notizen

# Individuelle Kriterien

| **Leitfrage**<br>2 | **Fehlerbehandlung - User Experience**<br><br>Die Benutzeroberfläche soll eine gute User Experience gewährleisten, auch wenn die API Fehlermeldungen liefert. Um dieses Ziel zu erreichen, müssen in dieser Arbeit folgende API-Fehlermeldungen abgedeckt werden:<br>• Bei allen Requests soll der/die Nutzer/in zuück zum Login-Screen geführt werden, wenn eine 401 Response erhalten wird.<br>• Wenn keine kommenden Reservationen im System sind, soll auf der Übersichtsseite nicht eine leere Tabelle angezeigt werden, sondern eine entsprechende Mitteilung, dass keine Reservationen gefunden wurden.<br>• Beim Endpunkt «createReservation» muss der Status-Code 409 – Conflict abgedeckt sein, und in der Benutzeroberfläche eine entsprechende Fehlermeldung angezeigt werden.<br>• Zwei zusätzliche Error-Responses der API, welche von der Kandidatin gewählt werden, müssen abgedeckt sein. |
|---|---|
| Gütestufe 3 | 5 Test-Cases sind abgedeckt. |
| Gütestufe 2 | 3/5 sind abgedeckt. |
| Gütestufe 1 | 1/5 sind abgedeckt |
| Gütestufe 0 | Keine der Fehlermeldungen sind abgedeckt. |

Notizen

# Individuelle Kriterien

# Individuelle Kriterien

| **Leitfrage**<br>3 | **Git**<br><br>Die Kandidatin verwendet Git korrekt. Folgende Punkte sollen erreicht werden:<br>- Es wird oft committed. Der Umfang der jeweiligen Commit-Messages beschränkt sich auf wenige features, welche noch in der commit message prägnant beschrieben werden können.<br>- Die Commit-Messages verwenden "imperative-style" (also, werden im Präsens geschriben). zB. "add function to configure database", und NICHT "added function to configure database).<br>- Die Entwicklung findet auf einem Dev branch statt, welcher einmal pro Tag in den main Branch merged wird. |
|---|---|
| Gütestufe 3 | 3 Punkte erreicht. |
| Gütestufe 2 | 2 Punkte erreicht. |
| Gütestufe 1 | 1 Punkt erreicht. |
| Gütestufe 0 | Keine Punkte erreicht. |

Notizen

# Individuelle Kriterien

| **Leitfrage**<br>4 | **Code Style**<br><br>Ist der Code gut leserlich?<br>- Variabel-namen beschreiben den Zweck der Variable prägnant<br>- Kommentare werden da wo nötig engesetzt, um eine Code-Stelle einfacher leserlich zu machen. Es wird auf Kommentare verzichtet wenn der Code von sich aus logisch ist und besser mit gut ausgewählten Namen dokumentiert werden kann.<br>- Der Code ist anhand eines Code-Styles formattiert und in sich stimmig (Indentation immer gleich, überall single oder double quotes und kein mix, etc.). |
|---|---|
| Gütestufe 3 | Alle 3 Punkte sind voll erfüllt. |
| Gütestufe 2 | Alle 3 Punkte sind erfüllt, bis auf 1-5 Ausnahmen. |
| Gütestufe 1 | Ein Punkt ist gar nicht erfüllt. 2 Punkte sind mit 1-5 Ausnahmen erfüllt. |
| Gütestufe 0 | Es sind weniger als 2 Punkte erfüllt. |

Notizen

# Individuelle Kriterien

| **Leitfrage**<br>5 | **Programmierung von React-Komponenten**<br><br>- Der Code wurde in logische Kompnenten aufgeteilt<br>- Die Komponenten sind wiederverwendbar und nach dem DRY Prinzip aufgebaut<br>Wurde der Code in logische React-Komponenten aufgeteilt? |
|---|---|
| Gütestufe 3 | Der Code ist auf logische art und weise in Komponenten aufgeteilt, welche ggf. wiederverwendet werden können. Die Komponenten wurden sauber aufgeteilt in 1) Komponenten, die Business-Logic umsetzen und State besitzen, und 2) in stateless Komponenten die nur für Rendering verantwortlich sind. |
| Gütestufe 2 | Der Code ist auf logische art und weise in Komponenten aufgeteilt, welche ggf. wiederverwendet werden können. In maximal 1-2 Fällen haben Komponenten mehrere Verantwortlichkeiten (zB mehrere unverwandte UI-Elemente rendern). |
| Gütestufe 1 | Die aufteilung in Komponenten ist ungeügend. Komponenten haben mehrere Verantwortlichkeiten (zB mehrere unverwandte UI-Elemente rendern), und können schlecht wiederverwendet werden. |
| Gütestufe 0 | Die aufteilung in Komponenten ist nicht erfolgt. Der gesamte code ist in 1-2 Komponenten. |

Notizen

# Individuelle Kriterien

# Individuelle Kriterien

| **Leitfrage**<br>6 | **Programmierung von React-State-Management**<br><br>Wie wurde Applikations-State in React gehandhabt? |
|---|---|
| Gütestufe 3 | Die Applikation setzt State nach React-Standards um wie bei Stufe 2. State wurde gezielt nur in gewissen Komponenten eingesetzt, um den Daten-Flow leicht verständlich zu machen.<br>setState wird nur wenn nötig aufgerufen, um Rerenders zu vermeiden - Variablen, die nicht dringend im State sein müssen, sollen von State abgeleitet werden (derived state). |
| Gütestufe 2 | Die Applikation setzt State nach React-Standards um. Es ist ein Daten-Flow zu erkennen: State wird als Props an Komponenten, welche für das Rendering zuständig sind, weitergegeben. State kann durch callbacks, welche als Props and Rendering-Komponenten gegeben werden, updated werden. |
| Gütestufe 1 | Es ist kein klarer Daten-Flow durch die Applikation zu erkennen. State wird unüberlegt in verschiedenen Teilen der Applikation gespeichert. |
| Gütestufe 0 | State-Management wurde nicht genügend umgesetzt - Keine der anderen Stufen wurde erreicht. |

Notizen

# Individuelle Kriterien

# Individuelle Kriterien

| **Leitfrage**<br>7 | **Programmierung von React-Daten-Fetching**<br><br>Wurde Data-Fetching aus der API in React korrekt umgesetzt?<br>- Daten werden nur fetched wenn nötig<br>- Die UI bleibt während dem Data-Fetching voll funktionsfähig (management von Loading-State)<br>- Das Data-Fetching wird in der korrekten Lifecycle-Methode/hook der Komponente umgesetzt<br>- Die fetch-API/swagger-Client werden korrekt verwendet |
|---|---|
| Gütestufe 3 | Alle Punkte erreicht. |
| Gütestufe 2 | Drei Punkte erreicht |
| Gütestufe 1 | Zwei Punkte erreicht. |
| Gütestufe 0 | Weniger als 2 Punkte erreicht. |

Notizen