Shape

Description automatically generated with medium confidence

IPA Bericht

-

Implementierung der User Experience eines Parkplatzreservationssystems mit React

Kanditatin: Raquel Lima

Fachvorgesetzer: Hannes Hertach

Hauptexperte: Felix Tobler

Nebenexperte: Michel Georgy

Zeitraum: 9.03.2022-30.03.2022

Lehrfirma: Adobe Research (Schweiz) AG Peter Merian-Strasse 80 4052 Basel

# Dokumentenmanagement

Autorin: (RL)

Version: 0.2

Datum:

Dateiname:

## Versionierung

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Autorin | Änderung |
| 0.0 | 09.03.2022 | RL | Vorlage erstellt |
| 0.1 | 09.03.2022 | RL | Detaillierte Aufgabenstellung, Projektaufbauorganisation, |
| 0.2 | 10.03.2022 | RL |  |
| 0.4 |  | RL |  |
| 0.5 |  | RL |  |
| 0.6 |  | RL |  |
| 0.7 |  | RL |  |
| 0.8 |  | RL |  |
| 0.9 |  | RL |  |
| 0.10 |  | RL |  |
| 0.11 |  | RL |  |

Tabelle 1: Versionsverlauf

# Kurzfassung

## Ausgangssituation

## Umsetzung

## Ergebnis

Inhaltsverzeichnis

[Dokumentenmanagement 2](#_Toc129246682)

[Versionierung 2](#_Toc129246683)

[Kurzfassung 3](#_Toc129246684)

[Ausgangssituation 3](#_Toc129246685)

[Umsetzung 3](#_Toc129246686)

[Ergebnis 3](#_Toc129246687)

[Teil I: Obligatorische Kapitel 7](#_Toc129246688)

[1 Einleitung 7](#_Toc129246689)

[2 Detaillierte Aufgabenstellung 8](#_Toc129246690)

[3 Projektaufbauorganisation 14](#_Toc129246691)

[3.1 Rollen 14](#_Toc129246692)

[3.1.1 Kontaktdaten der Projektbeteiligten 14](#_Toc129246693)

[3.2 Organisation der Arbeitsergebnisse 14](#_Toc129246694)

[3.2.1 Datenablage 14](#_Toc129246695)

[3.2.2 Versionsverwaltung 14](#_Toc129246696)

[3.2.3 Sicherungskonzept 14](#_Toc129246697)

[3.2.4 Wiederherstellung der gesicherten Arbeitsergebnisse 14](#_Toc129246698)

[4 Projektmanagement 15](#_Toc129246699)

[4.1 Vergleich der Projektmethoden 15](#_Toc129246700)

[4.2 Wasserfallmodell 15](#_Toc129246701)

[4.2.1 Projektphasen 15](#_Toc129246702)

[4.2.2 Meilensteine 15](#_Toc129246703)

[4.3 Kanban 15](#_Toc129246704)

[5 Zeitplanung 16](#_Toc129246705)

[5.1 Aufbau der Zeitplan 16](#_Toc129246706)

[5.2 Abweichungen von Grobzeitplan 16](#_Toc129246707)

[5.3 Detaillierter Zeitplan 17](#_Toc129246708)

[6 Arbeitsjournale 18](#_Toc129246709)

[Tag 1 - Donnerstag, 9. März 2023 18](#_Toc129246710)

[7 Reflexion 20](#_Toc129246711)

[7.1 Soll/Ist Vergleich 20](#_Toc129246712)

[Teil II: Projektdokumentation 21](#_Toc129246713)

[1 Einleitung 21](#_Toc129246714)

[2 Analyse 22](#_Toc129246715)

[2.1 Anforderungsanalyse 22](#_Toc129246716)

[2.1.1 Ausgangslage 22](#_Toc129246717)

[2.1.2 Anforderungen (vielleicht in funktionale, nichtfunktionale, betriebliche Anforderungen) 22](#_Toc129246718)

[2.1.3 Anwendungsfalldiagramm 22](#_Toc129246719)

[2.1.4 Ablaufdiagramm 22](#_Toc129246720)

[2.2 Verfeinerung der Aufgabenstellung (maybe einfach in Arbeitsjournal erwähnen) 22](#_Toc129246721)

[2.3 Systemgrenzen 22](#_Toc129246722)

[2.4 Schnittstellen (Welche?) 22](#_Toc129246723)

[2.4.1 zur Aussenwelt 22](#_Toc129246724)

[2.4.2 Libraries 22](#_Toc129246725)

[3 Entwurf 23](#_Toc129246726)

[3.1 Technologien und Tools 23](#_Toc129246727)

[3.1.1 Programmiersprache 23](#_Toc129246728)

[3.1.2 Entwicklungsumgebung 23](#_Toc129246729)

[3.2 Konzept für die Entwicklung mit Git 23](#_Toc129246730)

[3.3 Technische Dokumentation Konzept 23](#_Toc129246731)

[3.4 Code Style Konzept 23](#_Toc129246732)

[3.4.1 Namenskonvention 23](#_Toc129246733)

[3.4.2 Kommentare 23](#_Toc129246734)

[3.4.3 Formatierung 23](#_Toc129246735)

[3.4.4 Prinzipien 23](#_Toc129246736)

[3.5 Testkonzept 23](#_Toc129246737)

[3.5.1 Fehlerbehandlung 23](#_Toc129246738)

[3.6 Checkliste ende vom tag(entweder da oder bei organisation der arbeitsergebnisse) 23](#_Toc129246739)

[3.7 Benutzeroberfläche 23](#_Toc129246740)

[4 Implementierung 24](#_Toc129246741)

[4.1 Entwicklungsumgebung 24](#_Toc129246742)

[4.1.1 Vorarbeit(Bestandene ) 24](#_Toc129246743)

[4.1.2 Während der arbeit 24](#_Toc129246744)

[4.1.3 Verzeichnisstruktur 24](#_Toc129246745)

[4.2 React 24](#_Toc129246746)

[4.2.1 React Router 24](#_Toc129246747)

[4.2.2 React Komponeten 24](#_Toc129246748)

[4.2.3 React State Management 24](#_Toc129246749)

[4.2.4 React-Daten-Fetching 24](#_Toc129246750)

[4.3 Benutzeroberflächen 24](#_Toc129246751)

[4.3.1 Mui 24](#_Toc129246752)

[4.3.2 User Experience 24](#_Toc129246753)

[4.3.3 Fehlermeldungen 24](#_Toc129246754)

[4.4 API-Abfragen 24](#_Toc129246755)

[4.5 Verwendete Bibliotheken 24](#_Toc129246756)

[5 Testing 25](#_Toc129246757)

[5.1 Übersicht der Ergebnisse 25](#_Toc129246758)

[5.2 Testprotokoll 25](#_Toc129246759)

[5.3 Fazit 25](#_Toc129246760)

[5.4 Neue Arbeiten 25](#_Toc129246761)

[6 Schlusswort 26](#_Toc129246762)

[7 Danksagung 27](#_Toc129246763)

[Teil III: Anhang 28](#_Toc129246764)

[1 Verzeichnisse 28](#_Toc129246765)

[1.1 Quellenverzeichnis 28](#_Toc129246766)

[1.2 Tabellenverzeichnis 29](#_Toc129246767)

[1.3 Abbildungsverzeichnis 30](#_Toc129246768)

[2 Glossar 31](#_Toc129246769)

1. Obligatorische Kapitel

# Einleitung

# Detaillierte Aufgabenstellung

## Titel der Arbeit

Implementierung der User Experience eines Parkplatzreservationssystems mit React

## 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 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, 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. 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

## 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

# Projektaufbauorganisation

## Rollen

**Kandidatin:** Der Prüfungskandidat, nachfolgend auch KAND genannt, ist der sogenannte Projektleiter und ist verantwortlich für die Durchführung des Projekts. Zeitplan sowie Projektziele sollten dabei eingehalten werden.

**Verantwortliche Fachkraft:** Die verantwortliche Fachperson, nachfolgend VF genannt, ist zuständig für die Definition des Projektes und begleitet den Kandidaten in der Zeitspanne des Projektes. Tauchen allfällige Fragen bezüglich des Auftrags auf, steht diese Person zur Unterstützung bereit. Die VF stellt den betrieblichen Teil des Expertenteams dar.

**Expertenteam:** Das Expertenteam begleitet den Kandidaten sowie die VF bei dem Projekt. Es gibt einen Hauptexperten, nachfolgend HE genannt, und einen Nebenexperten, nachfolgend NE genannt.

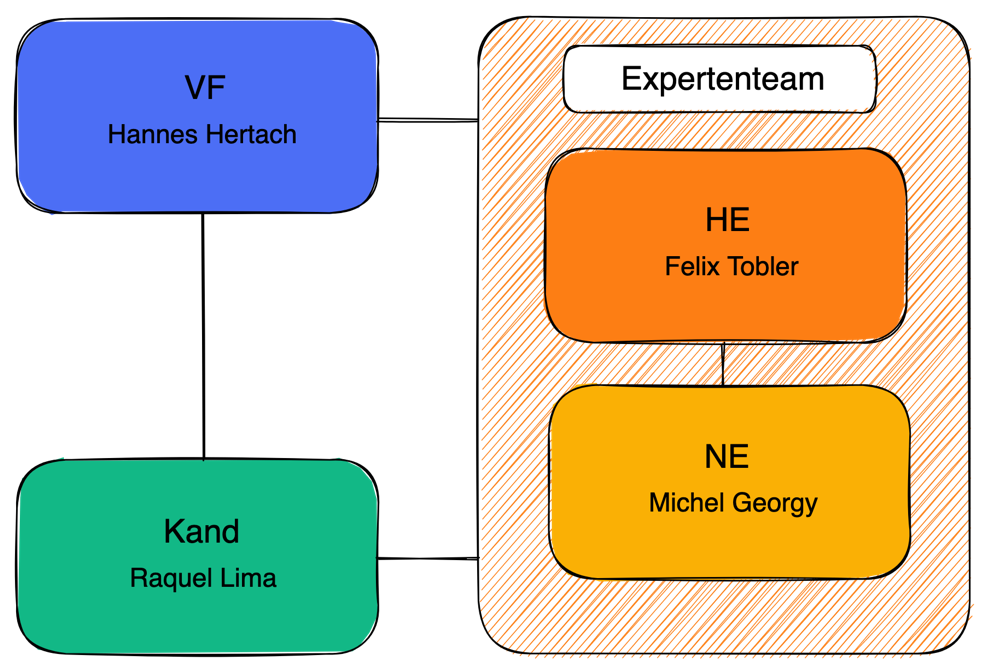


Abbildung 1: Aufbauorganisation der IPA

### Kontaktdaten der Projektbeteiligten

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Rolle | Telefon | E-Mail |
| Raquel Lima | Kanditatin | 0786682455 | rrlima.raquel@gmail.com |
| Hannes Hertach | Fachvorgesetzer | 0763974011 | hhertach@adobe.com |
| Felix Tobler | Hauptexperte | 0763990072 | felix.tobler@outlook.com |
| Michel Georgy | Nebenexperte | 0763820628 | mg@georgy.com |

Tabelle 2: Kontaktdaten der Projektbeteiligten

## Organisation der Arbeitsergebnisse

Während der Arbeit an einem Projekt ist es wichtig, ein Konzept für die Organisation der Arbeitsergebnisse zu entwerfen. Die Erstellung eines strukturierten und logischen Ablage- und Sicherungskonzepts stellt sicher, dass die Arbeit immer gefunden wird und im Falle eines Verlusts die Daten wiederhergestellt werden können.

### Datenablage

Während der IPA werden alle Dateien lokal und auf verschiedenen Online-Plattformen gespeichert. Als Online-Plattformen werden Sharepoint, Google Drive und Github verwendet.

Um den Speicherort aller Dokumente leicht zu finden, wurden sie unten dokumentiert:

**Dokumente**

|  |  |
| --- | --- |
| **Ort** | **Pfad** |
| Lokal | /Users/lima/Desktop/IPA-Lima/ |
| Sharepoint | https://adobe-my.sharepoint.com/personal/djaeggi\_adobe\_com |
| Google Drive | https://drive.google.com/drive/u/1/folders/17Wjyc1nSy0BCYCRphPEe3cQZhjcVg6Rf |
| Github | https://github.com/raquelima/IPA-Lima/tree/main/docs |

Tabelle 3: Dokumentenablage

**Quellcode**

|  |  |
| --- | --- |
| **Ort** | **Pfad** |
| Lokal | /Users/lima/Desktop/IPA-Lima/Repository |
| Github | https://github.com/raquelima/IPA-Lima |

Tabelle 4: Quellcodeablage

### Ordnerstruktur

Bei allen oben genannten Plattformen sind den Dokumenten in der folgenden Ordnerstruktur organisiert:

A picture containing graphical user interface

Description automatically generated

Abbildung 2: Ordnerstruktur der Dokumente

### Versionsverwaltung

Die Versionsverwaltung ist sowohl bei Dokumenten als auch bei der Softwareentwicklung sehr wichtig, da sie es ermöglicht, Änderungen zu verfolgen, frühere Versionen zu vergleichen und Änderungen bei Bedarf zurückzunehmen. Die Dateien werden während der IPA in den gewählten Online-Plattformen wie folgt versioniert:

|  |  |  |
| --- | --- | --- |
| **Daten** | **Plattform** | **Versionierung** |
| Quellcode | Git | Während der Entwicklung wird auf einem Dev-«Branch» gearbeitet. Der Quellcode wird gesichert und versioniert, wenn ein Fortschritt im Code erkennbar ist. Der Umfang des Codefortschritts wird in der Entwurfsphase im Kapitel 3.2 Konzept für die Entwicklung mit Git definiert. Der Dev-«Branch» wird täglich am Ende des Tages in den main-«Branch» gemerged. |
| Bericht, Zeitplan | Sharepoint | Der Bericht und der Zeiplan wurden in Sharepoint hochgeladen. Durch Öffnen des in Sharepoint gespeicherten Dokuments mit der Word-App und Aktivieren der Option AutoSave werden die Dokumente gespeichert und alle paar Minuten wird eine neue Version erstellt. |
| Google Drive | Jeden Tag am Ende des Tages wird die neueste Version auf Google Drive hochgeladen. |
| Git | Jeden Tag am Ende des Tages wird die neueste Version auf Git in den main-«Branch» hochgeladen |

Tabelle 5: Versionsverwaltung

**Quellcodesicherungsbefehle**

Für die Versionierung mit Git ist es wichtig, einige Befehle zu kennen. Im Folgenden sind die am häufigsten verwendeten aufgeführt:

|  |  |
| --- | --- |
| Befehl | Zweck |
| git add . | Inhalt aller veränderter Dateien wird zum Git Index hinzugefügt |
| git commit –m «...» | Die Veränderungen werden im lokalen Repository mit einer Nachricht gespeichert |
| git push | Das remote Repository wird mit den lokalen Veränderungen aktualisiert |

Tabelle 6: Quellcodesicherungsbefehle

**Namenskonvention des Berichts**

Das Word-Dokument wird nach seiner letzten Version mit folgendem Format benannt:

*Version\_IPA-Bericht\_Nachname-Vorname*

Siehe *Tabelle 1: Versionsverlauf* für eine Übersicht über alle vorhandenen Versionen.

### Wiederherstellung der gesicherten Arbeitsergebnisse

**Wiederherstellung mit Sharepoint und Google Drive**

Sharepoint und Google Drive bieten einen Versionsverlauf für jedes gespeicherte Dokument. Um eine Version des Dokuments wiederherzustellen, wählt man die gewünschte Version aus und drückt auf „restore“ oder „download“.

**Wiederherstellung mit Git**

Um eine bestimmte Version wiederherzustellen, kann man mit dem Befehl „*git clone»* das entfernte Repository lokal im gewünschten Ordner.

Sobald man das Repository lokal hat, kann man mit dem Befehl «*git log»* die Git-Commit-Historie ansehen.

Um einen bestimmten Commit wiederherzustellen, benötigen wur den folgenden Befehl:

*«git checkout <commit>»*

Die Wiederherstellung der Backups wurde an folgenden Tagen überprüft:

|  |  |
| --- | --- |
| **Datum** | **Status** |
|  |  |
|  |  |
|  |  |

### Checkliste für die tägliche Datensicherung

Wie bereits in den letzten Kapiteln beschrieben, müssen am Ende eines jeden Tages viele Sicherungsmassnahmen durchgeführt werden, um die Sicherheit der Daten zu gewährleisten. Um diesen Vorgang etwas zu erleichtern und damit nichts vergessen wird, wurde folgende Checkliste erstellt.

* Versionsverlauf und Dateiname des Berichts aktualisieren.
* Überprüfen, ob die neuste Version des Berichtes und des Zeitplans auf Sharepoint stimmt.
* Bericht, Zeitplan und Arbeitsjournal sowie alle bearbeiteten Dokumente auf Drive hochladen.
* Bericht, Zeitplan und Arbeitsjournal sowie alle bearbeiteten Dokumente auf Github hochladen.
* Falls in der Entwicklungsphase: Dev-„Branch“ in main-„Branch“ mergen.

# Projektmanagement

Im Rahmen der IPA muss eine Projektmanagement-Methode gewählt werden, die das Projekt zusammenhält, Qualität gewährleistet und sicherstellt, dass die Ziele termingerecht erreicht werden. Es gibt eine breite Palette an Projektmanagementmethoden, doch welche ist am besten für mein Projekt geeignet? Im folgenden Kapitel werde ich die mir bekanntesten Projektmethoden vergleichen, nämlich: Scrum, das Wasserfallmodell und Kanban. Schliesslich wird die am besten geeignete Projektmethode ausgewählt und auf das Projekt angewandt.

## Vergleich der Projektmethoden

## Wasserfallmodell

### Projektphasen

Analyse

Entwurf

Implementierung

Testing

Projektabschluss

### Meilensteine

Analyse abgeschlossen

Planung abgeschlossen

Entwurf abgeschlossen

Implementierung abgeschlossen

Testing abgeschlossen

Ipa bereit zur Abgabe

## Kanban

# Zeitplanung

## Aufbau der Zeitplan

Pufferzeit und übergreifende Zeit erklären

## Abweichungen zum Grobzeitplan

## Detaillierter Zeitplan

# Arbeitsjournale

## Tag 1 - Donnerstag, 9. März 2023

**Arbeitszeiten: 09:00-12:00/13:00-18:00**

****

**Tagesüberblick**

Heute war der erste Tag meiner IPA, der wie geplant um 9 Uhr begann. Ich startete den Tag mit der Analyse der Aufgabe. Ich war mir nicht sicher, wie ich anfangen sollte, also informierte ich mich im Internet darüber, was in der Analysephase zu tun ist. Ich erfuhr, dass dieser Phase eine Machbarkeitsstudie und eine Anforderungsdefinition umfasst. Ich beschloss, die Machbarkeitsstudie zu überspringen, da diese bereits vor der IPA durchgeführt wurde. Für die Anforderungsdefinition habe ich eine Ist-Analyse gemacht und ein Soll-Konzept erstellt. Während dieses Prozesses bemerkte ich, dass es einen Fehler in der Aufgabenstellung gab. Ich habe dies im Soll-Konzept in der Fahrzeuge Ansicht erwähnt und korrigiert. Um sicher zu gehen, dass dies der richtige Weg ist, habe ich meinen Berufsbildner gefragt, ob das in Ordnung ist. Nach Fertigstellung des Soll-Konzeptes habe ich die Systemgrenzen beschrieben. Danach begann ich mit der Arbeit an einem Ablaufdiagramm, das ich erfolgreich abschliessen konnte. Als letzten Schritt der Anforderungsanalyse habe ich dann die Anforderungen für dieses Projekt definiert. Später habe ich dann meine Schnittstellen dokumentiert. Vor dem Mittagessen konnte ich die Projektaufbauorganisation beschreiben. Am Nachmittag begann ich mit der Definition von Tätigkeiten, Phasen und Meilensteinen für das Projekt, die ich dann in meinem Zeitplan auf Excel darstellte. Nachdem ich mit meinem Zeitplan fertig war, erstellte ich mein Kanban mit Meilensteinen auf GitHub.

**Hilfsmittel**

* Für die Analysephase habe ich einen Artikel von einer Website gelesen (Web-Entwicklung, 2023). Aus dieser Quelle habe ich nur Informationen aus der Analysephase übernommen
* Ich fragte meinen Berufsbildner, Dominique Jäggi, was der richtige Weg ist, um einen Fehler in der Aufgabenstellung zu behandeln, worauf er antwortete, dass ich es richtig gemacht habe.

**Reflexion**

Heute begann ich den Tag ruhig. Ich fühlte mich bereit, meine IPA zu beginnen. Ich war mir nicht sicher, wie ich mit der Analyse der Aufgabestellung beginnen sollte, aber ich blieb ruhig und recherchierte. Dokumentieren ist nicht meine Stärke, also war es sehr stressig für mich, so viel Text in so kurzer Zeit zu schreiben. Im Laufe des Tages fühlte ich mich mehr und mehr gestresst. In solchen Momenten half es, eine Pause zu machen und einfach die Augen zu schliessen. Für die Anforderungsanalyse habe ich eine Stunde länger gebraucht als geplant, was mich wirklich genervt hat, aber am Ende des Tages war das kein Problem, weil andere Aktivitäten weniger Zeit in Anspruch nahmen als geplant. Es war ein sehr stressiger Tag, aber ich bin froh, dass ich meinen ersten Tag hinter mir habe.

Datum, Unterschrift Kandidatin: Datum, Unterschrift Fachvorgesetzer:

9.03.23, 9.03.23,

## Tag 2 - Donnerstag, 10. März 2023

**Arbeitszeiten: 09:00-12:00/13:00-18:00**

****

**Tagesüberblick**

Heute habe ich wieder einmal pünktlich um 9 Uhr mit der Arbeit begonnen. Bei der Durchsicht meiner gestrigen Arbeit ist mir aufgefallen, dass ich etwas vergessen habe, was ich gestern noch machen wollte.

Bei der Analyse der Ausgangssituation sind mir ein paar Dinge aufgefallen, die verbessert werden müssen. Zum Beispiel sollte der Benutzer nicht in der Lage sein, eine Reservierung zu erstellen, ohne vorher ein gespeichertes Auto im System zu haben. Dies sollte in der Implementierungsphase berücksichtigt werden. Ausserdem sind mir noch ein paar andere Kleinigkeiten in den Mockups aufgefallen. Heute habe ich also den Tag damit begonnen, schnell das Kapitel Verfeinerung zu schreiben. Der Plan ist, später eine Lösung für die in diesem Kapitel erwähnten Dinge zu entwerfen, damit sie später implementiert werden können. Im obigen Zeitplan wurde diese Tätigkeit zusammen mit der Organisation der Arbeitsergebnisse gezählt. Auch wenn die Verfeinerung nicht geplant war, verstösst es nicht gegen die gewählte Projektmethode, da ich mich noch in der Analysephase befinde.

Danach habe ich mir überlegt, wie ich meine Arbeitsergebnisse organisieren möchte.

Da es in Sharepoint bereits einen Ordner mit Vorarbeiten gab, habe ich diesen beibehalten. Ausserdem habe ich beschlossen, meine Dokumente noch in Google Drive zu speichern und zu versionieren, denn im Gegensatz zu Sharepoint, das mehrere Versionen pro Tag erstellt, gibt es in Google Drive nur eine pro Tag. Vor der IPA gab es schon ein Github-Repository, also werde ich dort meinen Code und meine Dokumente speichern. Ich habe dafür gesorgt, dass alle diese Ordner dieselbe Ordnerstruktur haben.

Spontan habe ich eine Checkliste für die Backups erstellt, um diese Aufgabe zu erleichtern. Leider habe ich für die ganze Organisation eine Stunde länger gebraucht als geplant.

In der Stunde vor dem Mittagessen habe ich mit dem Vergleich von Projektmethoden begonnen, um zu zeigen, warum die Wasserfallmethode die perfekte Methode für dieser IPA ist und wie ich sie mit Kanban ergänzen kann. Mit dem Abschluss dieser Aufgabe habe ich den zweiten geplanten Meilenstein erreicht und die erste Phase des definierten Wasserfallmodells abgeschlossen.

Nach dem Mittagessen begann ich die Entwurfsphase, indem ich die Technologien definierte, die zur Erfüllung der in der Anforderungsanalyse definierten Anforderungen eingesetzt werden sollen.

Nach dem Mittagessen begann ich mit der Entwurfsphase, indem ich die Technologien definierte, die zur Erfüllung der in der Anforderungsanalyse festgelegten Anforderungen eingesetzt werden sollen. Auch dies dauerte eine Stunde länger als erwartet.

Danach beeilte ich mich und beendete schnell die Entwicklung der Konzepte, die in der Umsetzungsphase zum Einsatz kommen werden. Mit nur zwei weiteren Stunden begann ich mit dem Schreiben des letzten Konzepts und war unsicher, ob ich heute noch mit dem Testkonzept beginnen könnte. Ich hatte Recht, was mich in Verzug bringt. Für den nächsten Mittwoch ist eine weitere Stunde geplant. Ich hoffe, dass ich es nächste Woche noch rechtzeitig fertigstellen kann.

**Hilfsmittel**

Für das Kapitel 3.2.4 Wiederherstellung der gesicherten Arbeitsergebnisse musste ich im Internet nachschlagen, was die genauen Git-Befehle sind. (Otieno, 2023)

Für das Kapitel *4.1 Vergleich der Projektmethodenmusste* habe ich ein Artikel gelesen (Fresco, 2023)

**Reflexion**

Heute begann ich den Tag nervöser als gestern, denn heute wusste ich, wie viel Arbeit vor mir lag. Dieses Gefühl hat sich im Laufe des Tages gelegt, weil es mir Freude bereitet, wenn ich sehe, dass das Dokument eine schöne Form bekommt. Heute war ich auch zuversichtlicher, kleine Pausen zu machen. Ich habe mir ein bisschen Sorgen gemacht, weil ich so lange mit der ganzen Organisation gebraucht habe, aber das ist sehr wichtig und hat Priorität, also ist es okay. Es war ein gutes Gefühl, die erste Phase des Projekts abzuschliessen. Da ich mich für eine lineare Projektmanagementmethode entschieden habe, kann ich nach Abschluss dieser Phase nicht wirklich zurückgehen, was mich nervös macht, aber ich bin immer noch zuversichtlich, dass es die richtige Entscheidung war, besonders nachdem ich Kapitel 4.1 Vergleich der Projektmethoden geschrieben habe.

Datum, Unterschrift Kandidatin: Datum, Unterschrift Fachvorgesetzer:

9.03.23, 9.03.23,

# Reflexion

## Soll/Ist Vergleich

1. Projektdokumentation

# Einleitung

# Analyse

Die erste Phase des in Kapitel *2 Detaillierte Aufgabenstellung* entwickelten Wasserfallmodells ist die Analyse. In diesem Kapitel wird die Ausgangslage eingehend analysiert. Das Ziel dieser Phase ist es, alle Anforderungen zu sammeln…

## Anforderungsanalyse

Durch die Anforderungsanalyse werden alle Anforderungen für die Erfüllung der Aufgabe definiert…

### Ist-Zustand

Adobe ist ein internationales Softwareunternehmen, welches derzeit 120 Mitarbeiter in Basel beschäftigt. Aufgrund des kürzlich erfolgten Umzugs des Firmensitzes zu Peter Merian stehen den Mitarbeitern 8 Parkplätze zur Verfügung. Um Nutzungskonflikte zu vermeiden, müssen die Parkplätze vor der Nutzung reserviert werden. Dazu wird ein Reservationssystem benötigt. Für das System wurde bereits eine API-Spezifikation (Jäggi, 2023) und ein Mock-Server erstellt, die in Kapitel *2.3 Schnittstellen* beschrieben sind. Die Mockups wurden ebenfalls entworfen, welche im Anhang oder online (Lima, ParkIT, 2023) zu finden sind. Nun muss noch die Benutzeroberfläche implementiert werden.

### Soll-Konzept

Das Soll-Konzept beschreibt die Fragestellung "Wo wollen wir hin?"

|  |  |
| --- | --- |
| Ansicht | Funktionen |
| **Login** | Die Mitarbeiter müssen sich mittels SSO-Authentifizierung anmelden können. Falls die Authentifizierung fehlschlägt, muss eine Fehlermeldung angezeigt werden. |
| **„Dashboard“** | Sobald der Benutzer angemeldet ist, soll er die Dashboard-Ansicht mit allen Informationen sehen, wie sie in den oben erwähnten Mockups dargestellt ist. Auf dem Dashboard kann der Benutzer anstehende Reservierungen betrachten und stornieren. Mit einem Button wird der Benutzer direkt zur Parkplatzübersicht führt, um eine Reservierung zu erstellen. |
| **„Reservations“** | In der Reservierungsansicht muss der Benutzer alle seine Reservierungen und deren Details sehen können. Mit einem Button muss der Nutzer die Reservierungen nach ihrem Status filtern können. Sowie beim Dashboard-Ansicht gibt es einen weiteren Button, der den Benutzer direkt zur Parkplatzübersicht führt. |
| **„Parking overview“** | In der Parkplatzübersicht kann der Benutzer seine Reservierung erstellen. Hier gibt es eine Übersicht über die Parkplätze. Durch Auswahl einer Uhrzeit und eines Datums ändert sich die Ansicht der Parkplätze. Sobald der Nutzer einen freien Parkplatz ausgewählt hat, öffnet sich ein Panel, in dem er die Dauer der Reservierung und das Fahrzeug auswählen kann und seine Reservation abschliessen kann. |
| **„Vehicles“** | In der ursprünglichen Aufgabe in Kapitel 2.3 steht "Der Link zu «Vehicles» öffnet rechts ein Panel, wie bei «Profile»". Das war ein Fehler des Auftraggebers. Die Fahrzeugansicht wird wie die Reservierungsansicht funktionieren, wie man in den Mockups sehen kann. Der Benutzer muss alle seine gespeicherten Fahrzeuge und Details sehen. Er muss auch in der Lage sein, sie zu löschen und neue Fahrzeuge zu erstellen. |
| **„Admin“** | In der Admin-Ansicht soll es weitere Aktionen geben, wie zum Beispiel die Deaktivierung von Parkplätzen. Diese Ansicht wurde nicht in die Mockups aufgenommen, da sie im Rahmen der IPA nicht implementiert wird. |

**Navigation**

Der eingeloggte Benutzer kann jederzeit über die Navigation an der Seite zwischen der Reservierungsansicht, der Parkplatzübersicht und der Fahrzeugansicht navigieren. Ausserdem kann er auf seine Profilansicht zugreifen und sich abmelden.

**Design**

Die zu implementierende UI muss mit den vorgegebenen Mockups identisch sein. Die Screenshots der Mockups wurden aus Gründen der Übersichtlichkeit nicht in der Ausgangslage aufgenommen. Wie bereits erwähnt, können Sie diese unter diesem [Link](https://xd.adobe.com/view/43abcff4-bced-4b8b-ac03-e1e67d438d62-16f8/) oder im Anhang einsehen.

**Fehlerbehandlung**

Während der Nutzung des Reservierungssystems könnten mehrere Fehler auftreten. Um Benutzerfreundlichkeit zu gewährleisten, muss die Benutzeroberfläche wie folgende mit Fehler umgehen:

|  |  |  |
| --- | --- | --- |
| **Status-Code** | **Name** | **Umgang** |
| 401 | „Unauthorized Access“ | Der User muss zurück zum Login-Screen geführt werden |
| 409 | „Conflict“ | Fehlermeldung anzeigen |

Zusätzlich müssen noch mindestens zwei weitere Fehlerfälle abgedeckt werden. Diese werden in der Entwurf Phase im Kapitel bla definiert.

### Systemgrenzen

Damit die Benutzeroberfläche in den Rahmen der IPA implementiert werden kann, müssen einige Features eingeschränkt werden. Für die IPA gilt den oberen Soll-Konzept mit den folgenden Ausnahmen:

* Der Login wird nicht mit Adobe’s SSO integriert. Die Benutzername und Passwort werden client-seitig überprüft und mit dem im Kapitel 3.8 festgelegt Testbenutzer verglichen. Die Eingaben werden dann im Localstorage gespeichert
* Die „Admin“-Ansicht wird nicht implementiert
* Wie erwähnt in der Ausgangslage, gibt es keinen echten Backend und Datenbank, sondern einen Mockserver.

### Ablaufdiagramm

Zur besseren Übersicht der Aufgabenstellung wurde folgendes Ablaufdiagramm erstellt.

### Anwendungsfälle

### Anforderungen

Die React-Applikation kommuniziert via HTTP API mit einem Backend

**Funktionale Anforderungen**

|  |  |  |
| --- | --- | --- |
| **ID** | **Anforderung** | **Kurze Beschreibung** |
|  | User Login |  |
|  | Aktuelle Informationen - Dashboard |  |
|  | Bevorstehende Reservierungen - Dashboard |  |
|  | Anstehende Reservierungen stornieren - Dashboard |  |
|  | „Reservierung erstellen“ Button - Dashboard |  |
|  | Anzeige der Reservierungen - Reservations |  |
|  | Reservierungen filtern - Reservations |  |
|  | Reservierungen stornieren - Reservation |  |

**Nichtfunktionale Anforderungen**

|  |  |  |
| --- | --- | --- |
| **ID** | **Anforderung** | **Kurze Beschreibung** |
|  |  |  |
|  |  |  |
|  |  |  |

**Betriebliche Anforderungen**

|  |  |  |
| --- | --- | --- |
| **ID** | **Anforderung** | **Kurze Beschreibung** |
|  |  |  |
|  |  |  |
|  |  |  |

### Verfeinerung der Aufgabenstellung (maybe einfach in Arbeitsjournal erwähnen)

Change mockup to “today’s information” to make clear to user

Ui/ux flowchart - also change parking numbers to correspond to the real numbers - change that user can only create reservations when he has a vehicle

## Schnittstellen (Welche?)

### zur Aussenwelt-API

### Mockserver

### Libraries

Mui

# Entwurf

## Technologien und Tools

Um die Anforderungen der Analysephase zu erfüllen, werden folgende Technologien und Tools verwendet

### Programmiersprache

Javascript, React, MUI

### Entwicklungsumgebung

IDE

Buildsystem

Abhängigkeitsverwaltung

Code Formater

Laufzeitumgebung der Mockserver

JavaDoc

|  |  |
| --- | --- |
| Funktion | Tool und Version |
|  |  |
|  |  |

## Konzept für die Entwicklung mit Git

Auf ein dev branch arbeiten und einmal pro tag mergen

## Technische Dokumentation Konzept

Hannes: 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.

Die technische Dokumentation (Lima, Eigen erstellter Inhalt) (Kommentare, Readme), sowie aller Code, wird auf Englisch geschrieben.

## Code Style Konzept

### Namenskonvention

Hannes: Variabel-namen beschreiben den Zweck der Variable prägnant

### Kommentare

Hannes: Kommentare werden da wo nötig eingesetzt, 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.  
Methoden und Funktionen, welche Business-Logik umsetzen, sowie React Function Components (props müssen dokumentiert sein), müssen mit JSDOC die Inputs und Outputs dokumentieren. Davon ausgenommen sind return-values von React-Components.

### Formatierung

anhand eines Code-Styles formatiert und in sich stimmig (Indentation immer gleich, überall single oder double quotes und kein mix, etc.)

### Prinzipien

## Testkonzept

Absichtlich vor der Implementierungsphase

Test user erwähnen

### Fehlerbehandlung

## Benutzeroberfläche -> Schnittstellen

Mockups – da beschreiben dass es schon welche gibt und auf anhang zeigen mit screenshots und erwähnen was ich bei der implementierung ändern würde

Schnittstellen

Fehlermeldungen

### Entwurf der Lösung für kapitel verfeinerung

# Implementierung

## Entwicklungsumgebung

### Vorarbeit(Bestandene )

**API**

**Dev/Mock Server**

**Linter**

### Während der arbeit

Git (Branch erstellen und mergen)

**Javadoc**

**Mui installieren**

**Readme (vielleicht nicht da)**

### Verzeichnisstruktur

## React

Aufzeigen wo es erklärt wurde

### React Router

### React Komponeten

### React State Management

### React-Daten-Fetching

## Benutzeroberflächen

### Mui

### User Experience

### Fehlermeldungen

## API-Abfragen

## Verwendete Bibliotheken

# Testing

## Übersicht der Ergebnisse

## Testprotokoll

## Fazit

## Neue Arbeiten

# Schlusswort

Reflexion

Zeitplan

Weiteres Vorgehen

# Danksagung

1. Anhang

# Verzeichnisse

## Quellenverzeichnis

Jäggi, D. (9. März 2023). *parkit-spec*. Von Github: https://github.com/berufsbildung-basel/parkit-spec abgerufen

Lima, R. (9. März 2023). *ParkIT*. Von xd.adobe.com: https://xd.adobe.com/view/43abcff4-bced-4b8b-ac03-e1e67d438d62-16f8/ abgerufen

Web-Entwicklung. (9. März 2023). *Das Wasserfallmodell*. Von IONOS: https://www.ionos.de/digitalguide/websites/web-entwicklung/wasserfallmodell/ abgerufen

## Tabellenverzeichnis

[Tabelle 1: Versionsverlauf 2](#_Toc129286163)

[Tabelle 2: Kontaktdaten der Projektbeteiligten 14](#_Toc129286164)

[Tabelle 3: Dokumentenablage 15](#_Toc129286165)

[Tabelle 4: Quellcodeablage 15](#_Toc129286166)

[Tabelle 5: Versionsverwaltung 15](#_Toc129286167)

[Tabelle 6: Glossar 37](#_Toc129286168)

## Abbildungsverzeichnis

[Abbildung 1: Aufbauorganisation der IPA 14](#_Toc129286171)

# Mockups

# Kanban Board

Tag 1

Graphical user interface, application, Teams

Description automatically generated

# Glossar

|  |  |
| --- | --- |
| Begriff | Definition |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tabelle 7: Glossar