Open Education Platform for Management Schools

Publikationstyp: Lehrmaterialien

Entwicklung eines Web-basierten Desksharing-Tools

Version Nr. 1, 23. Mai 2024

Baldauf, Matthias
Müller, Sebastian
OST – Ostschweizer Fachhochschule

Publiziert auf: www.oepms.org
Unter doi: 10.25938/oepms.384

Open Education Platform
for Management Schools

Entwicklung eines Web-basierten Desksharing-Tools

Version Nr. 1, 23. Mai 2024

Baldauf, Matthias
Müller, Sebastian
OST – Ostschweizer Fachhochschule

Publikationsform:	Fallstudie
Institution:	OST – Ostschweizer Fachhochschule
Schlüsselbegriffe:	Web-Entwicklung; HTML; CSS; JavaScript; Web Services
Einsatzbereich:	Bachelorstudierende

Zitierweise nach APA:

Baldauf, M. & Müller, S. (2024). Entwicklung eines Web-basierten Desksharing-Tools. *Open Education Platform*. Doi: 10.25938/oepms.384

Open Education Platform
for Management Schools

Didaktische Reflexion:

“Entwicklung eines Web-basierten Desksharing-Tools”

Matthias Baldauf [a] und Sebastian Müller[b]

OST – Ostschweizer Fachhochschule
Institut für Informations- und Prozessmanagement
Rosenbergstrasse 59, 9001 St.Gallen

[a] matthias.baldauf@ost.ch, [b] sebastian.mueller@ost.ch

Abstract. Die vorliegende Fallstudie ist konzipiert für den Einsatz in Grundlagenmodulen zur Web-Entwicklung. Anhand des praktischen Beispiels können Studierende ihre Kenntnisse in HTML, CSS und JavaScript praktisch anwenden und festigen. Der Inhalt umfasst die Umsetzung einer Web-Anwendung zum Management von Arbeitsplätzen für mobile Mitarbeiter:innen einer Hochschule. Deren Informatikabteilung stellt ein Backend-System zur Verwaltung der Arbeitsplätze und Reservationen zur Verfügung. Die Studierenden werden mit der Umsetzung des webbasierten Frontends beauftragt. Um den meist unterschiedlichen Wissensstand der Studierenden zu berücksichtigen sowie spielerisch zur Umsetzung weiterer Funktionen zu motivieren, ist der Umfang der Anwendung in Pflicht- und Wahlaufgaben geteilt, für welche jeweils Punkte gesammelt werden können.

Inhaltsverzeichnis

1. Didaktischer Reflexionsbericht

1.1. Zielgruppe

Die vorliegende Fallstudie richtet sich an Bachelor-Studierende der Wirtschaftsinformatik (bzw. des Wirtschaftsingenieurwesens und der Informatik) an einer Fachhochschule oder Universität, die Basistechnologien in der Entwicklung von Web-Anwendungen erlernen möchten. Mit dieser Fallstudie können HTML-, CSS- und JavaScript-Kenntnisse praktisch angewendet werden.

1.2. Vorwissen

Zur Bearbeitung dieser Fallstudie sind Kenntnisse im Bereich der Software-Entwicklung bzw. im Speziellen von Web-Basistechnologien nötig. Um die beschriebenen Funktionalitäten umzusetzen, kommen HTML, CSS und JavaScript zum Einsatz. HTML (Hypertext Markup Language) ist eine Auszeichnungssprache zur Strukturierung einer Webseite, CSS (Cascading Style Sheets) ist ebenfalls eine Auszeichnungssprache und wird für Darstellungsvorgaben genutzt, JavaScript ist eine Skriptsprache für interaktive Funktionalitäten auf Webseiten.

Bei den Durchführungen wurde zudem Bootstrap als Framework eingesetzt, um ein responsives Webdesign umzusetzen. Dabei passt sich die Webseite an die Eigenschaften des aufrufenden Endgeräts, zumeist der Bildschirmgrösse, an.

Backend-Services wurden den Studierenden entweder zur Verfügung gestellt (eine eigens für die Fallstudie umgesetzte REST-Schnittstelle) oder durch Drittanbieter abgedeckt (bspw. ein tagesaktueller Währungskonverter). REST (Representational State Transfer) stellt einen De-Facto-Standard zur Nutzung von Webservices, beispielsweise zum Abruf von Daten, dar. Studierende lernen im Modul nicht, solche Dienste zu entwickeln, sondern sie via JavaScript zu nutzen.

1.3. Lernziele

Nachfolgende Lernziele sollen mit dieser Fallstudie erreicht werden:

Die Studierenden können:

- Web-Technologien anwenden und vertiefen
    - Die im Unterricht kennengelernten Web-Technologien (HTML, CSS und JavaScript) können in der Fallstudie angewendet werden.
    - Bei auftretenden Problemen können die in der Softwareentwicklung bekannten Online-Hilfestellungen verwendet werden, um Codefehler selbständig zu beheben.
- Web-Projekte basierend auf erhobenen Anforderungen umsetzen
    - Anforderungen können aus User Stories abgeleitet und unter Anwendung der kennengelernten Web-Technologien in einem Web-Projekt umgesetzt werden.
- Softwareprojekte strukturieren und dokumentieren
    - Die drei Web-Technologien werden in der Projektstruktur logisch voneinander getrennt und im Code miteinander verknüpft.
    - Durch eine sinnvolle Dokumentation ist sichergestellt, dass der geschriebene Code nachvollziehbar ist.
- Mit existierenden Web-Service-Schnittstellen kommunizieren und Antworten verarbeiten
    - Die im Unterricht kennengelernten und für die Fallstudie zur Verfügung gestellten REST-Services werden genutzt.

1.4. Lehrplan und Leitfaden

Diese Fallstudie begleitete ein Modul in der zweiten Semesterhälfte. Die erste Semesterhälfte wurde zur Vermittlung der benötigten Programmierfähigkeiten eingesetzt. Die benötigten Konzepte und Funktionalitäten sind den Studierenden also zu Beginn dieser Fallstudie vermittelt worden und wurden jeweils durch kleine Einzelübungen vertieft. Der Einsatz für eine konkrete Anwendung und das Zusammenspiel unterschiedlicher Komponenten bringen jedoch neue Herausforderungen mit sich.

Die Studierenden bearbeiteten die Fallstudie zur Anwendung und Vertiefung ihrer erworbenen Kenntnisse nach einer Einführung der Fallstudie (ca. eine Lektion) ab der zweiten Semesterhälfte begleitend zum Unterricht als Einzelarbeit. Drei Tutoriate à vier Lektionen wurden zur Verfügung gestellt, um individuelle Fragen und Probleme in «Mini-Coachings» mit den Studierenden zu besprechen. Diese wurden jeweils durch ca. die Hälfte der Studierenden in Anspruch genommen.

Insbesondere für Studierende ohne Vorkenntnisse in der Webentwicklung hat es sich bewährt, die Fallstudie anhand der Aufträge 1-3 zu strukturieren. Da die Aufträge jeweils aufeinander aufbauen, kann mit Auftrag 1 gestartet werden. Die anfängliche Strukturierung des Projekts kommt den Studierenden auch bei der Bearbeitung der Aufgaben 2 und 3 zugute. Das ebenfalls in Auftrag 1 zu erstellende Design-Konzept mit Mockups hilft zusätzlich dabei, spätere aufwändige Änderungen zu vermeiden. Auftrag 1 endet dann mit dem Erstellen der statischen Inhalte (Navigation, Start- und allfällige Unterseiten, Formulare).

In Auftrag 2 werden die statischen Inhalte nun um dynamisch eingebundene ergänzt. Arbeitsplätze, getätigte Reservierungen und Kosten, als Wahlaufgabe auch eine Karte, werden über unterschiedliche REST-Schnittstellen abgerufen und in die vorher erarbeiteten statischen Inhalte integriert.

Vor Auftrag 3 sollten die Inhalte der Webseite grundsätzlich fertig sein. Nun müssen noch die bereits erstellten Formulare korrekt abgesendet und die Antworten der Backend-Services verarbeitet und benutzerfreundlich dargestellt werden.

Die Dokumentation des Codes sollte im besten Fall fortlaufend erfolgen. Darauf können die Studierenden auch hin und wieder hingewiesen werden, da erfahrungsgemäss ein Teil der Studierenden dazu neigt, die Dokumentation zu vernachlässigen und dann versucht, diese zum Schluss noch nachzuholen, was die Qualität der Dokumentation verschlechtern kann.

Nach Einreichung der Fallstudie wurde diese bewertet (siehe Abschnitt 1.6) und den Studierenden ein Feedbackbogen mit Erläuterungen zur Bewertung gesendet.

1.5. Erfahrungen mit der Fallstudie

Diese Fallstudie wurde zweimal mit Wirtschaftsinformatik-Studierenden auf Bachelor-Stufe im Modul «Web Development Fundamentals» bzw. dessen Vorgängermodul (als Teil der Vertiefungsrichtung «Business Software Development») durchgeführt und als Leistungsnachweis eingesetzt.

Grundsätzlich gelang den Studierenden die Umsetzung statischer HTML- und CSS-Elemente Anwendung gut. Mehr Aufwand bekundeten sie meist bezüglich der dynamischen Erzeugung und der Manipulation von HTML-Elementen via JavaScript. Konzepte wie Schleifen wurden im Unterricht oftmals verstanden, die konkrete Anwendung in der Fallstudie bereitete jedoch einigen Studierenden Mühe. Diese Themen wurden auch in den erwähnten Unterrichtseinheiten oft nachgefragt und anschliessend nochmals repetiert. Hilfreich war es, bei der zweiten Durchführung diese identifizierten Schwächen mit zusätzlichen Übungen in der ersten Hälfte des Semesters gezielt zu adressieren.

Auch aus den Rückmeldungen der Studierenden wurde klar, dass die HTML- und CSS-Teile der Fallstudie vergleichsweise rasch gelöst werden konnten. Die JavaScript-Funktionalitäten wurden als deutlich grössere Herausforderung wahrgenommen. Dies widerspiegelten auch die eingereichten Lösungen der Studierenden.

Grundsätzlich sind die Funktionen des Systems bewusst nur sehr grob definiert. Es werden die Hauptfunktionalitäten anhand von Userstorys und Aufgaben definiert, Details der Umsetzung bleiben aber den Studierenden überlassen. Dies erwies sich in mehreren Fällen als positiv, da verschiedene Herangehensweisen der Studierenden möglich sind und ein gewisser Gestaltungsspielraum besteht.

Eine grosse Herausforderung für die Dozierenden im Modul sind unterschiedliche Vorkenntnisse der Studierenden: Während manche lediglich bei (teils länger zurückliegenden) Einführungsmodulen mit Programmiergrundlagen konfrontiert wurden, haben andere in Hobbyprojekten selbst Erfahrungen mit Web-Technologien gesammelt bzw. sind einige wenige beruflich als Entwickler (eventuell sogar im Bereich Web-Frontends) tätig. Das Anbieten von Wahlaufgaben hat sich dabei bewährt, auch erfahreneren Studierenden kleine Herausforderungen zu bieten und deren Kenntnisse über die Grundlagen hinaus zu erweitern. Während für die Lösung der Fallstudie bewusst nur die Basistechnologien HTML, CSS und (reines) JavaScript eingesetzt werden sollen, wurde in begründeten Fällen einzelnen Studierenden die Nutzung weiterer Technologien erlaubt. So wurde beispielsweise einem Studenten, der als Entwickler arbeitete und mit den Basistechnologien bereits sehr vertraut war, auf Anfrage die Nutzung von vue.js erlaubt. Er hatte bisher damit keine Erfahrung gesammelt und konnte die Fallstudie somit zur Einarbeitung und Auseinandersetzung mit dem Framework nutzen und neue Kenntnisse erwerben.

1.6. Bewertungsvorschlag

Die Bewertung erfolgt transparent nach Punktezahlen pro (verpflichtenden und optionalen) Aufträgen (vgl. Kapitel 2.5):

- Auftrag 1 – Pflicht: max. 4 Punkte
- Auftrag 2 – Pflicht: max. 6 Punkte
- Auftrag 2 – Wahl: max. 4 Punkte
- Auftrag 3 – Pflicht: max. 6 Punkte
- Auftrag 3 – Wahl: max. 4 Punkte

Bei der Punktevergabe pro Aufgabe können folgende Beurteilungskriterien betrachtet werden:

- Funktionalität
- Umsetzungsqualität
- Code-Stil

Bei der Erklärung der Fallstudie vor der Klasse wird darauf hingewiesen, dass im Falle von offensichtlich kopiertem Code die Note 1 resultieren würde.

Für die Projektdokumentation werden bis zu 6 Punkte vergeben. In Summe lassen sich somit maximal 30 Punkte erreichen. 17 Punkte entsprechen einer 4.0, ab 28 Punkten wird eine 6.0 vergeben.

1.7. Lösungsvorschlag und Backend-System

Auf Anfrage stellt das Autorenteam verifizierten Dozierenden gerne die fürs das Backend-System genutzten Skripte zur Verfügung. Auch Lösungsvorschläge können auf Anfrage zugeschickt werden.

Aufgrund des fortlaufenden Einsatzes dieser Fallstudie als Leistungsnachweis und der Open Access Strategie dieser Plattform muss auf die Veröffentlichung von entsprechendem Material verzichtet werden.

Open Education Platform
for Management Schools

Fallstudie:

“Entwicklung eines Web-basierten Desksharing-Tools”

Matthias Baldauf[a] und Sebastian Müller[b]

OST – Ostschweizer Fachhochschule
Institut für Informations- und Prozessmanagement
Rosenbergstrasse 59, 9001 St.Gallen

[a] matthias.baldauf@ost.ch, [b] sebastian.mueller@ost.ch

Abstract. Die vorliegende Fallstudie ist konzipiert für den Einsatz in Grundlagenmodulen zur Web-Entwicklung. Anhand des praktischen Beispiels können Studierende ihre Kenntnisse in HTML, CSS und JavaScript praktisch anwenden und festigen. Der Inhalt umfasst die Umsetzung einer Web-Anwendung zum Management von Arbeitsplätzen für mobile Mitarbeiter:innen einer Hochschule. Deren Informatikabteilung stellt ein Backend-System zur Verwaltung der Arbeitsplätze und Reservationen zur Verfügung. Die Studierenden werden mit der Umsetzung des webbasierten Frontends beauftragt. Um den meist unterschiedlichen Wissensstand der Studierenden zu berücksichtigen sowie spielerisch zur Umsetzung weiterer Funktionen zu motivieren, ist der Umfang der Anwendung in Pflicht- und Wahlaufgaben geteilt, für welche jeweils Punkte gesammelt werden können.

Fallstudie

1.8. Einführung und Situationsanalyse

Eine Schweizer Hochschule steht vor folgendem Problem: Aufgrund des stetigen Wachstums der Anzahl Mitarbeitenden kam es immer wieder zu Platzproblemen in den bestehenden Räumlichkeiten. Teilweise mussten Abteilungen in externe Büros umziehen, welche zu diesem Zweck angemietet werden. Seit der Corona-Pandemie und der damit einhergehenden zeitweisen Homeoffice-Pflicht hat sich jedoch das Bild in den Büros der Hochschule verändert. Viele Mitarbeitende arbeiten weiterhin aus dem Homeoffice oder sind im Unterricht, sodass einzelne Arbeitsplätze tageweise unbenutzt bleiben. Dies ist auch den Mitarbeitenden der Hochschulverwaltung aufgefallen. Das Potenzial, vielleicht sogar angemietete Büroräumlichkeiten wieder aufzulösen und die bestehenden Arbeitsplätze effizienter zu nutzen ist offensichtlich.

Aktuell haben alle Angestellten mit einem Beschäftigungsgrad von über 40% das Anrecht auf einen eigenen Arbeitsplatz. Weiter stehen für Lehrbeauftragte ohne Festanstellung an der Hochschule Arbeitsplätze bereit, die beispielweise in Freilektionen durch diese genutzt werden können. Das Facility-Management ist für die Einrichtung der Arbeitsplätze zuständig und verwaltet die Möbel der Hochschule.

1.9. Status

Um der Arbeitsplatzknappheit entgegenzuwirken und die Auslastung der bestehenden Arbeitsplätze zu verbessern, möchte die Hochschulverwaltung auf ein in diversen Unternehmen bereits praktiziertes Mittel zurückgreifen: flexible Arbeitsplätze bzw. «Desksharing». Mitarbeitende mit geringen Pensen oder vermehrtem Homeoffice verzichten freiwillig auf einen ihnen fix zugewiesenen Schreibtisch. Dafür stehen die so frei gewordenen Arbeitsplätze jenen Personen zur Verfügung, welche nur tageweise vor Ort in Büros arbeiten. Zusätzlich wird das Minimalpensum für das Anrecht auf einen eigenen Arbeitsplatz auf 50% erhöht. Damit stehen nun ausreichend Arbeitsplätze zur Verfügung, die für Desksharing genutzt werden können.

Da auch mit diesem neuen Konzept die Arbeitsplätze begrenzt sind, und niemand ins Büro reisen soll, um dort keinen freien Arbeitsplatz zu finden, soll ein Tool für das «Desksharing Management» umgesetzt werden. Die Idee der Hochschulverwaltung: Mitarbeitende reservieren sich ganz einfach einen Arbeitsplatz für die Zeit, zu welcher sie einen benötigen. Bildschirm, Dockingstation, Maus und Tastatur liegen bereit. Alle Mitarbeitenden verfügen bereits über einen Laptop, welcher mit dieser Infrastruktur an jedem Arbeitsplatz direkt einsatzbereit ist. Dies ermöglicht zusätzlich eine verbrauchergerechte Abrechnung der Infrastruktur-Kosten und verhindert, dass die einzelnen Organisationseinheiten für leere Arbeitsplätze bezahlen müssen.

Die Hochschulverwaltung beauftragt den Rektorats-Stab mit der Evaluation und Einführung eines entsprechenden Tools. In Absprache mit der internen Informatik wird der Entschluss getroffen, dass das Tool als Eigenentwicklung der Hochschule umgesetzt werden soll, da die am Markt verfügbaren Tools nicht sämtliche Ansprüche der Hochschule erfüllen. Die Idee: der Rektorats-Stab kümmert sich um die Anforderungserhebung und das Testen von ersten Prototypen mit ausgewählten Mitarbeitenden. Die interne Informatik stellt alle nötigen Informationen in einer Datenbank zur Verfügung und kümmert sich um die Verwaltung und Speicherung der Informationen. Für die Entwicklung und Umsetzung der Benutzeroberfläche sollen Studierende aus den Studiengängen der Hochschule eingesetzt werden.

1.10. Herausforderungen

1.10.1. User Stories

In Fokusgruppen mit ausgewählten Mitarbeitenden und den weiteren Anspruchsgruppen wurden durch den Rektorats-Stab Anforderungen an das Tool ausgearbeitet. Die Anforderungen wurden in folgenden User Stories festgehalten.

User Story 1: Arbeitsplätze einsehen

Als Mitarbeiter:in möchte ich alle buchbaren flexiblen Arbeitsplätze im Unternehmen ansehen können, um mich über deren Status informieren zu können. Dies umfasst:

- Darstellung der Arbeitsplätze
- Darstellung relevanter Zusatzinformationen je Arbeitsplatz
- Visuelle Kennzeichnung, wenn ein Arbeitsplatz dauerhaft nicht verfügbar ist (bspw. aufgrund einer Reparatur)

User Story 2: Reservierungen einsehen

Als Mitarbeiter:in möchte ich die aktuellen Reservierungen der Arbeitsplätze ansehen können, um einen für mich passenden, freien Zeitslot finden zu können. Dies umfasst:

- Auswahl eines Arbeitsplatzes
- Angabe eines Zeitraums
- Darstellung der Reservierungen des Arbeitsplatzes in diesem Zeitraum

User Story 3: Reservierung tätigen

Als Mitarbeiter:in möchte ich eine Reservierung absenden können, um einen Arbeitsplatz für einen bestimmten Zeitraum zu reservieren. Dies umfasst:

- Auswahl des Arbeitsplatzes und gewünschten Zeitraums
- Eingabe von Vor- und Nachname und E-Mail-Adresse
- Validierung der Eingaben

1.10.2. Anforderungen der Hochschule

Nebst den Anforderungen der Mitarbeitenden haben auch weitere Stakeholder bzw. Abteilungen innerhalb der Hochschule Anforderungen an das Desksharing-Tool. Folgende Anforderungen und Hinweise müssen nebst den formulierten User Stories bei der Realisierung des neuen Systems berücksichtigt werden:

Informatikabteilung:

- Responsive Design: Das neue Reservationssystem soll sowohl auf Smartphones und auf Desktop-PCs mit Monitoren ansprechend aussehen und bedient werden können.
- Browserkompatibilität: Durch die Informatikabteilung werden ausschliesslich Windows-Geräte unterstützt. Auf diesen Geräten ist Google Chrome installiert und als Standardbrowser eingerichtet. Für andere Betriebssysteme oder Browser wird kein Support angeboten, weshalb

die Prüfung der Kompatibilität des Reservationssystems für weitere Browser nicht geprüft werden muss.

- Die Informatikabteilung kümmert sich um die Schnittstellen zur Beantragung und Löschung von Reservationen, der Abfrage von bereits getätigten Reservierungen und der Speicherung und Verwaltung sämtlicher Arbeitsplätze und Reservationen in einer dafür aufgesetzten Datenbank. Die Dokumentation der entsprechenden Schnittstellen und Datenformate wird zur Verfügung gestellt.

Finanzen & Controlling:

- Für eine transparente Verrechnung der belegten Arbeitsplätze an die jeweiligen Organisationseinheiten, sollen die Kosten für die Arbeitsplätze jeweils für einen reservierten Zeitraum bereits bei der Reservation angezeigt werden. Die Kosten pro Stunde für die verschiedenen Arbeitsplätze werden zur Verfügung gestellt.
- Da einzelne Arbeitsplätze auch für Mitarbeitende von Partnerhochschulen zur Verfügung stehen, und Kolleginnen und Kollegen aus dem grenznahen Ausland erwartet werden, sollen die Preise sowohl in Schweizer Franken als auch in Euro angezeigt werden können.

Facility-Management & Empfang:

- Zur Vermeidung einer Anhäufung von Fragen im Stil von «Wo finde ich Arbeitsplatz XY?» und der Zusatzbelastung, wenn ständig Personen zu ihrem reservierten Arbeitsplatz geführt werden müssen, soll bei der Reservierung eine Karte angezeigt werden, auf welcher der Standort des jeweiligen Arbeitsplatztes vermerkt ist. Die entsprechenden Koordinaten werden vom Facility Management geliefert.
- Aus den Erfahrungen der bereits im Einsatz stehenden Raumreservation wissen die beiden Organisationseinheiten, dass unvollständig ausgefüllte Reservationen oder solche mit fehlerhafter E-Mail-Adresse immer zu Mehraufwand führen. Deshalb sollen bei der Reservierung folgende Angaben zwingend gemacht werden müssen:
    - Zu reservierender Arbeitsplatz
    - E-Mail-Adresse (mit korrektem Format)
    - Start der Reservierung
    - Ende der Reservierung

1.11. Aufträge

Um die Lösung der Fallstudie zu strukturieren und das web-basierte Frontend schrittweise zu entwickeln, werden die sich aus den User Stories ergebenden Anforderungen durch folgende Aufträge zusätzlich ausformuliert.

Auftrag 1 (Pflicht): Grundgerüst (max. 4 Punkte)

Als Einstieg in die Fallstudie wird als erstes das Grundgerüst für die späteren Inhalte geschaffen. Dies umfasst:

- Erstellen der Projektstruktur
  - Hauptdatei index.html und Verzeichnisse für Bilder, Scripts, CSS-Files etc.
  - Erstellen von ersten benötigten Dateien
- Design-Konzept
  - Skizzen zum User Interface (im Projektbericht festzuhalten):
  - Darstellung der Inhalte auf Smartphones und grösseren Screens
  - Nutzung von Unterseiten
  - Navigation der User durch die Anwendung
- Umsetzung des Responsiven Layouts mit Anpassungen an die jeweiligen Bildschirmgrössen
- Erstellen von Web-Formularen

Auftrag 2 (Pflicht): Dynamische Elemente (max. 6 Punkte)

Aus den Anforderungen und den zur Verfügung gestellten Schnittstellen ergeben sich Elemente, welche dynamisch in die Webseite eingebunden werden müssen. Diese umfassen:

- Darstellung der Arbeitsplätze
- Darstellung der getätigten Reservierungen
- Kosten pro Stunde
  - Anzeige in CHF und EUR
  - Aktueller EUR Währungskurs abgerufen über REST-Schnittstelle, z.B. über https://exchangerate.host

Auftrag 2 (Wahlaufgaben)

- Dynamische Anzeige der exakten Kosten für den ausgewählten Zeitraum (2 Punkte)
- Einbindung einer interaktiven Karte (z.B. von https://leafletjs.com), um den Ort, wo sich ein Arbeitsplatz befindet, grafisch darzustellen (2 Punkte)

Auftrag 3: Daten versenden (max. 6 Punkte)

Benutzerinnen und Benutzer des Arbeitsplatz-Reservationssystems werden Formulare ausfüllen, um freie Arbeitsplätze abzufragen oder Arbeitsplätze für bestimmte Zeiträume zu reservieren. Diese Daten müssen über die entsprechenden Schnittstellen an das Backend-System gesendet werden. Daneben soll die häufige Nutzung des Systems erleichtert werden.

- Formularvalidierung: Das Absenden eines Reservierungsantrags darf nur möglich sein, wenn alle Pflichtfelder gemäss den Anforderungen und Schnittstellen-Dokumentation befüllt sind und die Inhalte dem jeweils korrekten Format entsprechen.
- Verarbeiten der Antwort: Benutzende sollen «verständliche» Rückmeldungen nach den von ihnen getätigten Aktionen erhalten (bspw.: «Arbeitsplatz erfolgreich reserviert!»)
- Nach einer erfolgreichen Reservierung sollen Name und E-Mail-Adresse des Reservierenden lokal abgespeichert werden und die entsprechenden Felder beim nächsten Aufruf damit vorausgefüllt werden, um den Aufwand bei mehrfachen Arbeitsplatzreservierungen zu minimieren.

Auftrag 3 (Wahlaufgaben)

- Reservationen sollen durch den Aufruf des entsprechenden Services auch storniert werden können (2 Punkte)
- Zum einfachen Eintragen einer Reservation in den eigenen Kalender soll mittels https://calndr.link/ dynamisch für eine erfolgreiche Buchung eine ics-Datei erstellt und dieses verlinkt werden.

Nach erfolgreicher Bearbeitung dieser Fallstudie sollen folgende Lieferobjekte abgegeben werden:

- Lauffähiges Reservationssystem als eigenständige Webseite (max. 24 Punkte):
  - Abgabe als ZIP-Datei. Darin enthalten: index.html sowie sämtliche eingebundenen lokalen Daten (CSS-Dateien, JavaScript-Dateien, Bilder, etc.)
  - Anforderung: ZIP-Datei wird entpackt, enthaltene index.html wird vom lokalen Dateisystem aus in Google Chrome geöffnet, System funktioniert.
  - Der erstellte Code ist sauber strukturiert und so dokumentiert, sodass die Funktionsweise der verschiedenen Komponenten (insbesondere der JavaScript-Funktionen) nachvollziehbar ist.

- Schriftlicher Projektbericht (min. 6 Seiten ohne Titelseite, max. 6 Punkte):
  - Dokumentiert Projektstruktur und Zusammenspiel der Komponenten
  - Dokumentiert pro Auftrag Vorgehensweise und Überlegungen
  - Reflektiert über persönliche Herausforderungen und Lösungen

1.12. Dokumentation der Schnittstelle

Wie bereits beschrieben, kümmert sich die Informatikabteilung der Hochschule um das Backend und stellt eine entsprechende Schnittstelle zur Verfügung. Folgende fünf REST-Services werden dabei angeboten:

GET /desks

Zur Abfrage aller Tische stellt dieser Service ein Array mit sämtlichen Tischen als JSON-Objekte zur Verfügung. Die Antwort ist in folgendem Format zu erwarten:

```
[
    {
        "id": "1",                                  // eindeutige Tisch-Id
        "name": "D7-4",                             // Name des Tisches
        "available": "1",                           // 1 = verfügbar, 0 = nicht verf.
        "address": [anonymisiert],                  // Adresse
        "price": "65.2",                            // Preis pro Stunde in CHF
        "lat": [anonymisiert],                      // Breitengrad des Standorts
        "lon": [anonymisiert],                      // Längengrad des Standorts
        "comment": "2 screens"                      // Optionaler Kommentar zum Tisch
    },
    {
        "id": "2",
        "name": "D7-3",
        "available": "0",
        "address": [anonymisiert],
        "price": "78.5",
        "lat": [anonymisiert],
        "lon": [anonymisiert],
        "comment": ""
    },
    ...
]
```

GET /freedesks

Für einen festzulegenden Zeitraum können alle freien Tische mit folgender Anfrage abgefragt werden:

Parameter	Beschreibung	Beispiel
start	Start des Zeitraums, Format YYYY-MM-DDTHH:MM:SS (Pflicht)	2023-02-01T08:00:00
end	Ende des Zeitraums, Format YYYY-MM-DD (Pflicht)	2023-02-03T12:00:00
studid	Eindeutige Studierenden-ID (Pflicht)	1234

Die Studierenden-ID (studid) dient nur zur Unterscheidung der Reservierungen pro Student/in in der Datenbank. Diese kann frei gewählt werden, sollte dann aber für alle Aufrufe gleich verwendet werden. Die Antwort ist in folgendem Format zu erwarten:

```
[
    {
        "id": "1",                                  // eindeutige Tisch-Id
        "name": "D7-4",                             // Name des Tisches
        "available": "1",                           // 1 = verfügbar, 0 = nicht verf.
        "address": [anonymisiert],                  // Adresse
        "price": "65.2",                            // Preis pro Stunde in CHF
```

```
        "lat": [anonymisiert],                          // Breitengrad des Standorts
        "lon": [anonymisiert],                          // Längengrad des Standorts
        "comment": "2 screens"                          // Optionaler Kommentar zum Tisch
    }
]
```

GET /bookings

Für einen anzugebenden Tisch und einen festzulegenden Zeitraum können sämtliche Buchungen mit folgender Anfrage abgefragt werden:

Parameter	Beschreibung	Beispiel
deskid	ID des Tisches (Pflicht)	1
start	Start des Zeitraums, Format YYYY-MM-DD (Pflicht)	2023-02-07
end	Ende des Zeitraums, Format YYYY-MM-DD (Pflicht)	2023-02-09
studid	Eindeutige Studierenden-ID (Pflicht)	1234

Die JSON-Antwort ist in folgendem Format zu erwarten:

```
[
    {
        "id": "707",
        "deskid": "1",
        "start": "2023-02-08T08:00:00+0100",
        "end": "2023-02-08T10:00:00+0100",
        "user": "John Doe",
        "email": "john.doe@examp.le",
        "studid": "12345"
    }
]
```

POST /booking

Um einen Arbeitsplatz zu reservieren kann folgende Anfrage gesendet werden:

Parameter (Name des Formular-Felds)	Beschreibung	Beispiel
deskid	ID des Tisches (Pflicht)	1
user	Name des Users (Pflicht)	John Doe
email	E-Mail des Users (Pflicht)	john.doe@examp.le
start	Start des Zeitraums, Format YYYY-MM-DDTHH:MM:SS (Pflicht)	2023-02-08T08:00:00
end	Ende des Zeitraums, Format YYYY-MM-DDHH:MM:SS (Pflicht)	2023-02-08T10:00:00
studid	Eindeutige Studierenden-ID (Pflicht)	1234

Die Antwort im Falle einer erfolgreichen Reservierung (Tisch ist zum angegebenen Zeitraum noch verfügbar), ist in folgendem Format zu erwarten:

```
{
    "success": true,
    "message": "booking successful"
}
```

Falls versucht wird, einen Tisch zu reservieren, welcher zur angegebenen Zeit bereits belegt ist, wird folgende Antwort gesendet:

```
{
    "success": false,
    "errorcode": 150,
    "message": "not available in given period"
}
```

DELETE /booking

Um eine bereits getätigte Reservierung wieder zu löschen, kann folgende Anfrage gesendet werden:

Parameter (Name des Formular-Felds)	Beschreibung	Beispiel
id	ID der zu löschenden Reservierung (Pflicht)	1
studid	Eindeutige Studierenden-ID (Pflicht)	1234

Die Antwort im Falle einer erfolgreichen Löschung ist in folgendem Format zu erwarten:

```
{
    "success": true,
    "message": "booking canceled!"
}
```

Falls versucht wird eine nicht existierende Reservierung zu löschen, wird folgende Antwort erhalten:

```
{
    "success": false,
    "errorcode": 150,
    "message": "no booking found"
}
```

1.13. Empfohlene Software, Online-Hilfe und Literatur

Den Studierenden werden folgende Software und Online-Tools empfohlen:

- Entwicklungsumgebung
  - *Visual Studio Code* (https://code.visualstudio.com/Download) oder
  - ein Text-Editor wie z.B. *Sublime Text* (https://www.sublimetext.com/)
- *Postman* zum Testen der Schnittstellen (https://postman.com)
- *Codepen* für Übungen während des Unterrichts (https://codepen.io/)

Weiter werden folgende Angebote zur Online-Hilfe empfohlen:

- https://www.w3schools.com/
- https://developer.mozilla.org/en-US/docs/Web
- https://wiki.selfhtml.org/
- https://javascript.info/
- https://stackoverflow.com/

Ergänzende Bücher, optional:

Fuchs, P. (2020). *HTML5 und CSS3 für Einsteiger - Der leichte Weg zur eigenen Webseite.* Berlin: BMU Verlag.

Fuchs, P. (2020). *JavaScript Programmieren für Einsteiger: Der leichte Weg zum JavaScript-Experten.* Berlin: BMU Verlag.

Wenz, C., & Prevezanos, C. (2018). *HTML5 und CSS3 - Start ohne Vorwissen* (2. Ausg.). Burgthann: Markt+Technik Verlag GmbH.