



IPA 2025

Abschlussarbeit

Steuerverwaltungsplattform Prototyp

Vladyslav Astashyn
ajooda AG
Kapellgasse 6
6004 Luzern

Inhaltsverzeichnis

Vorwort	5
Organisation der Arbeitsergebnisse	6
Teil A: Projektumfeld und Aufgabenstellung	7
Umfeld	7
Ablauf	7
Aufbau	7
Funktionen	8
Ergänzte Funktionen	9
Umgang mit Anforderungen	9
Funktionstest	9
Dokumentation	9
Projektorganisation	10
Mittel und Methoden	11
Vorkenntnisse	11
Vorarbeiten	11
Arbeiten in den letzten 6 Monaten	11
Hilfestellung	11
Zeitplan	12
Arbeitsjournale	13
TAG 01 – 01.04.2024	13
TAG 02 – 02.04.2025	15
TAG 03 – 03.04.2025	16
TAG 04 – 04.04.2025	18
TAG 05 – 07.04.2025	19
TAG 06 – 08.04.2025	20
TAG 07 – 09.04.2025	22
TAG 08 – 10.04.2025	23
TAG 09 – 11.04.2025	24
TAG 10 – 14.04.2025	25
Teil B: Projekt	27
Kurzfassung	27
Ausgangssituation	27
Ziel	27
Umsetzung	27

Ergebnis	27
Einleitung	28
Informieren	29
Ziel der Aufgabenstellung	29
Vorgaben	29
Fragen	29
Planen	30
Realisierungskonzept	30
Usecase Diagramm	31
Mockups	32
Testkonzept	33
Entscheiden	38
Varianten	38
Entscheiden	38
Realisieren	39
React	39
Projekt Setup und Ordnerstruktur	40
Kundenübersicht umsetzen	45
localStorage integrieren	50
Rollen und Rechte umsetzen	52
Kundendetails umsetzen	54
Steuerjahre integrieren	58
Feedback und Bestätigung	64
Styling (CSS)	66
Zusatzfunktionen und Features	68
Kontrollieren	70
Testprotokoll	70
Testbericht	81
Auswerten	82
Reflexion	82
Danksagung	82
Anhang	83
Glossar	83
Quellenverzeichnis	84
Abbildungsverzeichnis	85

Anleitung Steuerverwaltungsplattform	87
--	----

Vorwort

Diese Dokumentation gehört zur IPA von Vladyslav Astashyn bei der Firma ajooda AG in Luzern. In dieser Dokumentation ist der Ablauf der Arbeit beschrieben und das Vorgehen dahinter.

die Dokumentation ist in zwei Teile gegliedert.

Im Teil A wird die detaillierte Aufgabenstellung und der Ablauf der Arbeit gezeigt. Es ist ersichtlich mit welchen Mitteln gearbeitet wurde und welche Vorkenntnisse vorhanden sind.

Im Teil B kommt dann die Projekt Dokumentation, in der die Arbeit beschrieben wird, was für Probleme aufgetreten sind und wie die Arbeit getestet wurde.

Durch die ganze Arbeit hinweg wurde mit der Projektmanagementmethode-IPERKA gearbeitet.

Die Reflexion und das Fazit stehen am Schluss der Dokumentation.

Ganz am Ende befindet sich noch der Anhang.

Organisation der Arbeitsergebnisse

Die Projektarbeit besteht aus der schriftlichen Dokumentation sowie dem Quellcode der Web-App.

Die Dokumentation und das Projektverzeichnis werden täglich jeweils auf zwei USB-Sticks gespeichert. So ist jederzeit sichergestellt, dass der Projektstand vom Tag davor immer zur Verfügung steht, selbst wenn ein USB-Stick ausfallen sollte.

Teil A: Projektumfeld und Aufgabenstellung

Umfeld

Die IPA wurde bei der ajooda AG durchgeführt, einem führenden digitalen Steuerberater mit Sitz in Luzern. Das Unternehmen bietet Privatpersonen und auch Unternehmen moderne Lösungen zur effizienten Bearbeitung von Steuererklärungen, sowie andere Services. Das geplante Projekt soll dabei helfen interne Abläufe weiter zu optimieren und die Digitale Kommunikation mit Kunden zu verbessern.

Ablauf

Nach der Analyse der Aufgabenstellung, der Kriterien und einer kurzen Besprechung mit dem Fachvorgesetzten werde ich das Projekt in einzelne Arbeitsschritte unterteilen, welche zu der IPERKA-Projektmanagementmethode passen. Basierend auf diesen Informationen werde ich einen passenden Zeitplan erstellen, welcher alle Punkte abdeckt.

Aufbau

Die Anwendung wird als Webapplikation mit React umgesetzt. Da es sich hier um einen Prototyp handelt verzichte ich komplett auf ein Backend. Die Daten werden lokal gespeichert. Die Applikation besteht aus zwei Hauptbereichen: einer Startseite, welche eine Ansicht der Kunden hat, welche für Admin und Advisor ersichtlich sind, sowie einer Detailseite mit den Kundendaten eines Clients und deren Steuerjahren welche für den Admin/Advisor als auch den Client sichtbar ist. Je nach Rolle stehen unterschiedliche Berechtigungen und Funktionen zur Verfügung. Nach einer Rücksprache mit dem Fachvorgesetzten wurde noch festgelegt das das Form als Popup erscheinen sollen.

Funktionen

Die Anwendung ist in zwei Bereiche unterteilt: eine Ansicht für Mitarbeitende (Admin/Advisor) in welcher die Kunden angezeigt werden und man diese in einer Suchleiste suchen kann und eine für Kundinnen und Kunden (Client) in welcher die Kundendaten und Steuerjahre angezeigt werden. Je nach Rolle stehen unterschiedliche Funktionen zur Verfügung.

Admin

- Kunden hinzufügen, anzeigen, bearbeiten und löschen
- Steuerjahre hinzufügen, bearbeiten und löschen
- Zugriff auf alle Kunden und deren Daten

Advisor

- Kunden hinzufügen, anzeigen und bearbeiten
- Steuerjahre hinzufügen und bearbeiten
- Kein Löschen von Kunden oder Steuerjahren möglich

Client

- Eigene Steuerjahre einsehen
- Eigene Kundendaten bearbeiten (z. B. Adresse, PLZ)
- Kein Zugriff auf andere Kunden oder deren Daten

Die erwähnten Funktionen sind die Kernanforderungen des Projekts und wurden daher priorisiert.

Ergänzte Funktionen

Nach der Besprechung sowie Rückfragen zu den Mockups wurden weitere Anforderungen klar und festgelegt. Diese tragen zur besseren Bedienbarkeit und Übersichtlichkeit der Anwendung bei.

Folgende Punkte wurden dabei festgelegt:

- Die Daten sollen im localStorage des Browsers gespeichert werden.
- Formulare zum Hinzufügen von Kunden und Steuerjahren sollen als Popup erscheinen falls möglich.
- Der Rollenwechsel soll statisch gelöst werden da kein Login- implementiert wird.
- Die Suchfunktion soll nach Vorname Nachname und E-Mail suchen können und es soll einen Button zum Schnellen Löschen der Eingabe geben.
- Auf der Seite der Kundendetails soll ein Button existieren, um zurück zur Startseite navigieren zu können.
- State Tax und Federal Tax sollen automatisch zu Total Tax gerechnet werden.
- Your Refund kann in Difference umbenannt werden und ergibt sich aus Total tax und Withholding tax paid, je nach Resultat soll die Zahl bei Diffrence grün mit der Unterschrift Refund oder rot mit der Unterschrift Remaining stehen.
- Die wichtigsten Buttons (z. B. Bearbeiten, Löschen, Speichern) sollen mit aussagekräftigen Icons dargestellt werden.
- Auf der Kundendetail Seite soll es möglich sein die Kundendaten ein und auszuklappen

Umgang mit Anforderungen

Die Anforderungen wurden während der Besprechung mit meinem Fachvorgesetzten gemeinsam definiert. Falls im Verlauf noch etwas dazukam oder sich etwas geändert hat, wurde das direkt besprochen und angepasst. Auch Feedback habe ich regelmässig eingeholt, um sicherzustellen, dass alles wie gewünscht umgesetzt wird.

Funktionstest

Die Funktionen werden während der Entwicklung der Anwendung manuell getestet. Dabei achte ich darauf, dass die Eingaben korrekt gespeichert, bearbeitet, angezeigt und gelöscht werden. Ausserdem werde ich überprüfen, ob die verschiedenen Rollen, die jeweils geplanten Rechte besitzen und die dazugehörigen Funktionen richtig funktionieren. Diese Tests werden während der Entwicklung so lange wiederholt, bis alle Funktionen so wie gewollt und ohne Fehler funktionieren.

Dokumentation

Die Dokumentation wird während der gesamten IPA-Zeit kontinuierlich mitgeschrieben.

Projektorganisation

Lehrbetrieb und Durchführungsort:

ajooda AG
Kappelgasse 6
6004 Luzern
+41 41 260 68 33
info@ajooda.ch

stellte die Infrastruktur zur Verfügung (Rechner, Software, Arbeitsumgebung) und ermöglichte die Durchführung vor Ort.

Kandidat:

Vladyslav Astashyn
Brunngrabenstrasse 24
4500 Solothurn
+41 78 889 27 69
astashyn.vladyslav@gmail.com

Ist für Planung, Umsetzung und Dokumentation des Projekts in Eigenverantwortung.

Berufsbildner/ Lehrfirma:

Frank Melber
Benedict-Schule Luzern
+41 41 227 01 15 direkt
+41 41 227 01 01
frank.melber@benedict.ch

Ist meine Ansprechperson von der Schule und unterstützte mich bei schulischen Fragen zur IPA.

Verantwortliche Fachkraft:

Veselin Lakic
ajooda AG
Kappelgasse 6
6004 Luzern
veselin.lakic@ajooda.ch

Gibt mir die Anforderungen vor, überprüfte Zwischenresultate und unterstützte mich fachlich.

Hauptexperte:

Marcel Arnold
marcel.arnold@dialog.ch
+41 79 780 65 81

Bewertet das Projekt im Rahmen der IPA-Abschlussprüfung.

Mittel und Methoden

Entwicklungsumgebung: Visual Studio Code (Windows 11)

Programmiersprache: TypeScript

Framework: React (nur Frontend, ohne Backend)

Styling: CSS (separate Stylesheets)

Datenbasis: Lokale Mock Daten (ggf. über LocalStorage gespeichert)

Benutzerrollen: Simulierte Rechte für Client, Advisor und Admin

Code Qualität: ESLint

Formatierung: Prettier Plugin in VSCode

Testing: Vitest für automatisierte Tests

KI Unterstützung: ChatGPT wurde beim Realisieren und Kontrollieren benutzt wegen Zeitmangel und fehlendem Wissen (wurde mit Text in Klammern markiert)

Vorkenntnisse

Grundkenntnisse in JavaScript, TypeScript, HTML und CSS. Dieses Wissen habe ich mir in meiner Ausbildung als auch in der Schule und Privat durch kleinere Projekte und Aufgaben angeeignet.

Vorarbeiten

Die Grundlagen von React habe ich mir im Voraus selbstständig angeeignet. Gemeinsam mit dem Fachverantwortlichen wurden Mockups erstellt, damit die Web-App eine benutzerfreundliche und übersichtliche Benutzeroberfläche bekommt.

Arbeiten in den letzten 6 Monaten

In den letzten sechs Monaten habe ich mich hauptsächlich mit der Frontend-Entwicklung beschäftigt. Ich habe unter anderem auch an einem eigenen Affiliate-System für ajooda gearbeitet, welches Daten verarbeitet und automatisch mit anderen Diensten verknüpft. Für unsere bestehende Webseite verschiedene Funktionen mit JavaScript umgesetzt, um die User experience zu verbessern. Zusätzlich war ich an der Weiterentwicklung und Gestaltung von Webseiten beteiligt. Durch diese Aufgaben konnte ich wertvolle Erfahrungen und Kenntnisse in verschiedenen Technologien sammeln und meine Fähigkeiten in der Frontend-Entwicklung weiter verbessern.

Hilfestellung

Der Fachverantwortliche steht bei Bedarf als Ansprechperson zu Verfügung, falls Fragen oder Unklarheiten während der Projektarbeit auftreten sollten.

Als Vorlage für meine IPA und um mich orientieren zu können wurde eine Vorlage von Herr Alpay Yildirim genommen.

Zeitplan

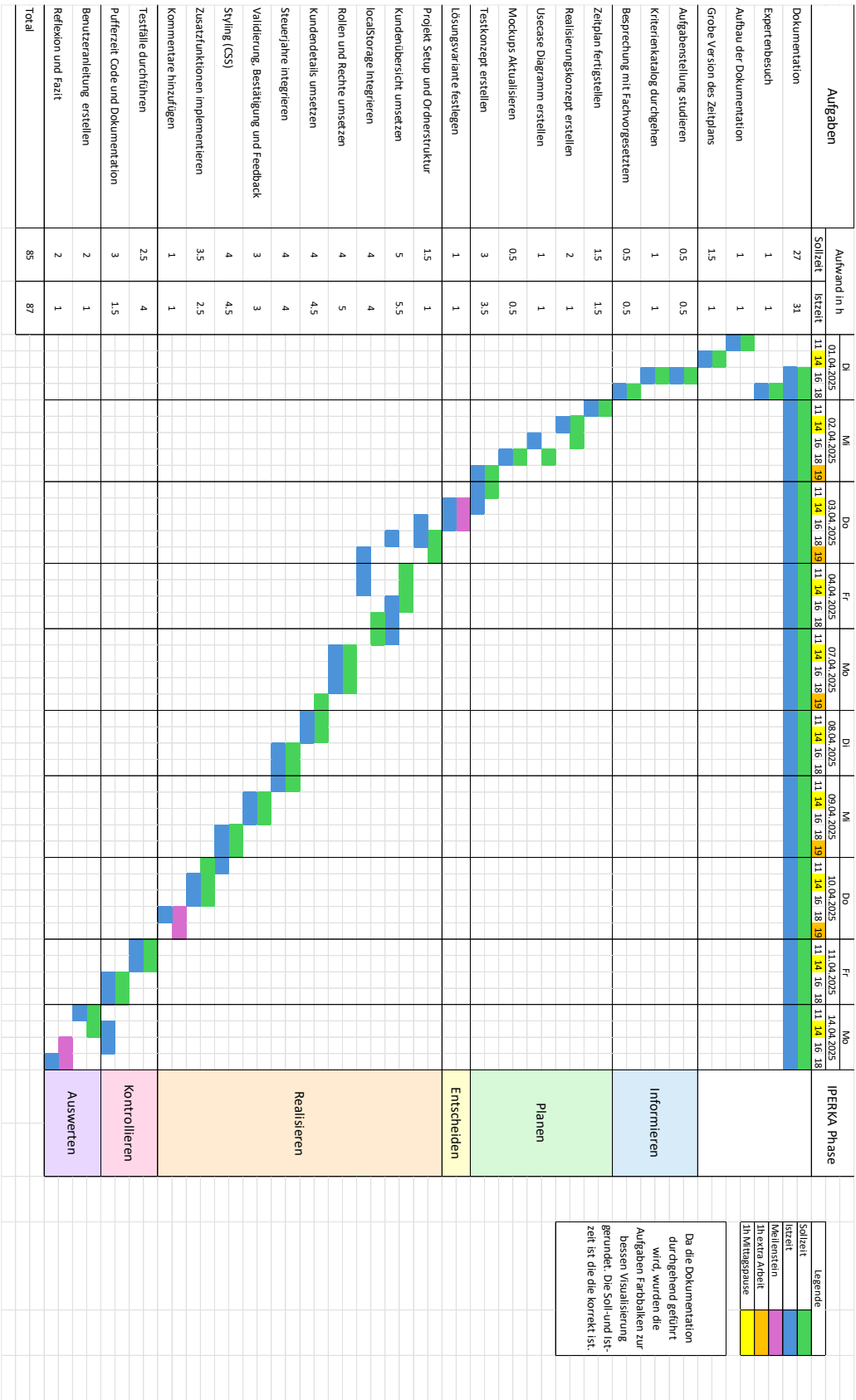


Abbildung 1 Zeitplan

Arbeitsjournale

Das Arbeitsjournal ist folgendermassen aufgebaut.

- Geplante Arbeiten
- Beschreibung dieser Arbeiten
- Falls Probleme entstanden sind, diese zu vermerken
- Für die Probleme Lösungssätze beschreiben
- Reflexion des Tages
- Verwendete Stunden pro Tag und die Gesamte Arbeitszeit und verbliebene Zeit

TAG 01 – 01.04.2024

Geplante Arbeiten:

- Aufbau der Dokumentation
- Grobe Version des Zeitplans
- Aufgabenstellung studieren
- Kriterienkatalog durchgehen
- Besprechung mit Fachvorgesetztem
- Exponentenbesuch
- Dokumentation beginnen

Erledigte Arbeiten:

- Aufbau der Dokumentation
- Grobe Version des Zeitplans
- Aufgabestellung studiert
- Kriterienkatalog analysiert
- Besprechung mit Fachvorgesetztem gehalten
- Exponentenbesuch
- Dokumentation begonnen

Aufbau der Dokumentation

Beschreibung:	Am Anfang wollte ich mal meine Dokumentation erstellen und das Grundgerüst dafür aufbauen damit ich an dieser auch arbeiten kann, ohne das nachträglich machen zu müssen. Das lief gut.
Zeit:	1h

Grobe Version des Zeitplans

Beschreibung:	Die Erstellung des Zeitplans verlief gut, ich habe mich dafür entschieden nur eine grobe Version zu machen, weil ich erstmal noch die Aufgabenstellung und den Kriterienkatalog nochmal analysieren und noch eine kurze Besprechung mit dem Fachvorgesetzten führen, bevor ich diesen beende. Hier bin ich auf keine Probleme gestossen.
Zeit:	1h

Aufgabenstellung studieren

Beschreibung:	Bei diesem Punkt habe ich mir kurz Zeit genommen, um die Aufgabenstellung zu studieren damit ich sie etwas später nochmal mit dem Fachvorgesetzten durchgehen kann.
Zeit:	0.5h

Kriterienkatalog durchgehen

Beschreibung:	Ich habe mir hier Zeit genommen und die Kriterien durchzugehen und zu verstehen, damit ich den Experten Fragen kann, falls was unklar ist.
Zeit:	1h

Besprechung mit Fachvorgesetztem

Beschreibung:	Um die Aufgabenstellung und die Mockups komplett zu verstehen wollte ich noch eine Rücksprache mit meine Fachvorgesetzten führen, hierbei habe ich alles, was nicht klar war erklärt bekommen.
Zeit:	0.5h

Expertenbesuch

Beschreibung:	Um genau 16:00 Uhr stand der Expertenbesuch an mit Herr Arnold, dieser war leider krank also fand das Gespräch Online statt. Er erklärte mir mehr über meine IPA, erwähnte auf welche Punkte ich achten soll und wo ich punkten kann, was zu tun ist bei unvorhersehbaren Ereignissen und mir noch paar Infos über die Präsentation, Demonstration und das Fachgespräch gegeben.
Zeit:	1h

Dokumentation

Beschreibung:	Ich habe an der Dokumentation kontinuierlich durch den ganzen Tag gearbeitet und habe am Ende des Tages noch das Arbeitsjournal geschrieben. Ich schliesse das Erstellen der Backups auch in diesen Schritt ein.
Zeit:	3h

Reflexion

Ich bin gut gestartet und motiviert an die Arbeit rangegangen.

Der Start war etwas holprig, aber ich habe schnell den Faden gefunden und konnte konzentriert meine Aufgaben planen und durchführen. Das Gespräch mit Herr Arnold war informativ und hilfreich, dadurch habe ich einige wichtige Punkte für später vermerkt. Leider war ich etwas nervös beim Gespräch und bin nicht selbstbewusst rübergekommen, aber das soll mich nicht davon abhalten eine gute IPA-Arbeit zu machen.

Tagesaufwand: 8h

Gesamtaufwand: 8h

Restzeit: 77h

TAG 02 – 02.04.2025

Geplante Arbeiten:

- Zeitplan fertigstellen
- Realisierungskonzept erstellen
- UsecaseDiagramm erstellen
- Testkonzepte erstellen
- Dokumentation weiterführen

Erledigte Arbeiten:

- Zeitplan fertiggestellt
- Realisierungskonzept erstellt
- Usecase Diagramm erstellt
- Mockups aktualisiert
- Testkonzepte erstellt (in Progress)
- Dokumentation weitergeführt

Zeitplan fertigstellen

Beschreibung:	Das Fertigstellen des Zeitplans lief ok ich war mir nicht sicher, wie ich den Ablauf von meinem Projekt gestalten soll, habe es aber ziemlich gut hingekriegt würde ich meinen.
Problem:	Da ich die Dokumentation durchgehend führe sind die Aufgaben Blöcke nicht gut aufgegangen.
Lösung:	Ich habe die Zeitblöcke gerundet damit das Visuell besser aussieht und das in der Legende des Zeitplans vermerkt.
Zeit:	1.5h

Realisierungskonzept erstellen

Beschreibung:	Das Erstellen des Realisierungskonzept verlief reibungslos und ohne Schwierigkeiten und ich war schneller fertig als geplant.
Zeit:	1h

Usecase Diagramm

Beschreibung:	Hier habe ich ein Usecase Diagramm erstellt, um die Wichtigsten Rollen und deren Funktionen innerhalb der Projekts übersichtlich darzustellen
Zeit:	1h

Mockups aktualisieren

Beschreibung:	Bei diesem Teil habe ich die bestehenden Mockups, die schon erstellt waren, nochmal mit meinem Fachvorgesetzten durchgegangen und einige Änderungen dran vorgenommen.
Zeit:	0.5h

Testkonzepte erstellen

Beschreibung:	Das Erstellen der Testfälle war etwas schwierig, da es zum Teil schwer war abzuschätzen, was ich testen soll und wie ich das am besten mache. Hier konnte ich noch nicht so viel machen und habe mich den Grossteil dieser Zeit informiert und schlau gemacht.
Zeit:	0.5h

Dokumentation weiterführen

Beschreibung:	Das Weiterführen der Dokumentation lief gut, ich konnte vieles machen und die Backups wurden erfolgreich erstellt.
Zeit:	4.5h

Reflexion

Das Fertigstellen des Zeitplans lief ok, ich war mir nicht sicher, wie ich die Zeit einteilen soll, deshalb habe ich mehr Zeit für die Realisierung des Projektes eingeteilt und habe mir noch erlaubt extra Zeit in meinen Zeitplan einzubauen. Da die IPA bis zu 90 Stunden gehen darf, beim Erstellen des Realisierungskonzepts konnte ich etwas Zeit sparen, da dieser bei mir nicht zu umfangreich ist. Etwas später im Verlauf vom Tag habe ich mich entschieden den Zeitplan nochmal anzupassen und die Mockups in meine IPA zu integrieren, dort habe ich wieder etwas Zeit verloren da das nicht geplant war. Was heute gut funktioniert hat war das Schreiben der Dokumentation, ich konnte viel schreiben und bin sehr weit gekommen. Heute war ein Erfolgreicher Tag und ich bin gespannt auf Morgen.

Tagesaufwand: 9h

Gesamtaufwand: 17h

Restzeit: 68h

TAG 03 – 03.04.2025

Geplante Arbeiten:

- Testkonzept erstellen
- Lösungsvariante festlegen
- Projekt Setup und Ordnerstruktur
- Kundenübersicht umsetzen
- Dokumentation weiterführen

Erledigte Arbeiten:

- Testkonzept erstellt
- Lösungsvariante festgelegt
- Projekt Setup und Ordnerstruktur aufgesetzt
- Kundenübersicht umsetzen (in Progress)
- localStorage integrieren (in Progress)
- Dokumentation weitergeführt

Testkonzept erstellen

Beschreibung:	Gestern und heute habe ich mich zum Thema Testing und Testpyramide informiert, mit diesen Informationen konnte ich dann geeignete Integrationstests erstellen.
Zeit:	3h

Lösungsvariante festlegen

Beschreibung:	Das Festlegen der Lösungsvariante war recht leicht da nicht viele Entscheidungen treffen musste.
Zeit:	1h

Projekt Setup und Ordnerstruktur

Beschreibung:	Zum Erstellen des Projekts habe ich mir einen kurzen Youtube Guide angeschaut und noch eine strukturierte Ordnerstruktur erstellt.
Zeit:	1h

Kundenübersicht umsetzen

Beschreibung:	Ich habe angefangen die Kundenübersicht zu erstellen, das hat auch ganz gut geklappt, jedoch bin ich auf ein Problem gestossen.
Problem:	Um die Funktionen umzusetzen, die ich für die Kundenübersicht brauche, wird localStorage benötigt.
Lösung:	Ich habe mich entschieden das Umsetzen der Kundenübersicht zu stoppen und mit der Implementation des localStorage zu beginnen.
Zeit:	1h

localStorage integrieren

Beschreibung:	Ich habe angefangen den LocalStorage zu implementieren da ich diesen für meine Kundenübersicht brauche.
Zeit:	1h

Dokumentation weiterführen

Beschreibung:	Ich habe die Dokumentation weitergeführt und Backups erstellt.
Zeit:	2h

Reflexion

Heute war der Beginn der Realisierungsphase jedoch bin ich da auf ein Problem gestossen, und zwar das es keinen Sinn machen würde den localStorage erst nach dem Abschluss der Umsetzung der Kundenübersicht zu machen, deshalb habe ich mich entschieden die Implementation des localStorage Parallel zur Umsetzung der Kundenübersicht zu machen und das so in meinem Zeitplan einzutragen, ich sehe hier eigentlich kein Problem da IPERKA eine flexible Projektmanagement Methode ist aber das wird sich am Ende rausstellen. Ich bin zufrieden mit dem Heutigen Tag.

Tagesaufwand: 9h

Gesamtaufwand: 26h

Restzeit: 59h

TAG 04 – 04.04.2025

Geplante Arbeiten:

- Kundenübersicht Umsetzen
- localStorage Integrieren
- Dokumentation weiterführen

Erledigte Arbeiten:

- Kundenübersicht Umsetzen (in Progress)
- localStorage integriert
- Dokumentation weitergeführt

Kundenübersicht Umsetzen

Beschreibung:	Heute habe ich einen Grossteil der Zeit an der Umsetzung der Kundenübersicht zusammen mit der Implementation des localStorage gearbeitet.
Zeit:	3h

localStorage Integrieren

Beschreibung:	Hier habe ich Funktionen erstellt, welche es mir ermöglichen den localStorage zu verwenden
Problem:	Nach zwei Stunden Recherche und Programmieren hatte ich noch kein funktionierendes Ergebnis und bin nicht weitergekommen
Lösung:	Ich habe hier ChatGPT genutzt und mich von ihm inspirieren lassen.
Zeit:	3h

Dokumentation weiterführen

Beschreibung:	Das Weiterführen der Dokumentation lief hervorragend, ich habe damit begonnen den Code zu dokumentieren, konnte jedoch nicht sehr viel machen, die Backups wurden auch erfolgreich erstellt.
Zeit:	2h

Reflexion

Der heutige Tag verlief gut, ich konnte an der Kundenübersicht und am localStorage arbeiten und bin recht weit gekommen, leider musste ich bei der Integration des localStorage ChatGPT benutzen, da ich es allein sonst wohl nicht geschafft hätte und mich nicht an meine Zeiten halten könnte, was noch fehlt ist die Suchleiste in der Kundenübersicht, diese werde ich am Montag umsetzen und fertigstellen. Ich habe anfangen Funktionen aus der Kundenübersicht in der Dokumentation detailliert zu beschreiben und am Ende des Tages das Arbeitsjournal geschrieben und die Backups erstellt. Die erste Woche ist durch und ich konnte vieles erreichen, ich freue mich auf nächste Woche.

Tagesaufwand: 8h

Gesamtaufwand: 34h

Restzeit: 50h

TAG 05 – 07.04.2025

Geplante Arbeiten:

- Kundenübersicht umsetzen
- Rollen und Rechte umsetzen
- Kundendetails umsetzen
- Dokumentation weiterführen

Erledigte Arbeiten:

- Kundenübersicht fertiggestellt
- Rollen und Rechte umgesetzt
- Dokumentation weitergeführt

Kundenübersicht umsetzen

Beschreibung:	Heute habe ich als erstes die Kundenübersicht fertiggestellt, ich habe hauptsächlich an der Suchfunktion der Kunden gearbeitet.
Zeit:	1.5h

Rollen und Rechte umsetzen

Beschreibung:	Hier habe ich länger gebraucht als geplant war, da ich mir zu Beginn unsicher war, wie ich die Rollen und Berechtigungen sauber umsetzen soll. Also habe ich mich ein gutes Stück der Zeit zusätzlich informiert und die Hilfe von ChatGPT geholt, um mich inspirieren zu lassen, so konnte ich eine flexible Lösung entwickeln die gut in mein Projekt passt.
Zeit:	5h

Dokumentation weiterführen

Beschreibung:	Heute habe ich mich drauf fokussiert die Teile vom Code die ich noch nicht hatte fertig zu dokumentieren, am Ende des Tages habe ich noch das Arbeitsjournal gemacht und die Backups erstellt.
Zeit:	2.5h

Reflexion

Ich habe motiviert in die Woche gestartet und habe als erstes die Kundenübersicht fertig umgesetzt. Bei der Umsetzung der Rollen und Rechte hatte ich anfangs Schwierigkeiten eine saubere Lösung zu finden, was mich auch viel Zeit gekostet hat, dann habe ich mich entschieden bei diesem Teil mit Hilfe von ChatGPT zu realisieren, das hat auch gut funktioniert und ich konnte eine Lösung finden und umsetzen welche flexibel und erweiterbar ist. In der Restlichen Zeit habe ich die Dokumentation weitergeführt und dokumentiert was noch offen war. Leider bin ich nicht dazu gekommen mit der Umsetzung der Kundendetails zu beginnen wie geplant war, aber Ich konnte heute vieles zu Rollen und Rechte lernen und bin insgesamt zufrieden mit dem heutigen Tag.

Tagesaufwand: 9h

Gesamtaufwand: 43h

Restzeit: 42h

TAG 06 – 08.04.2025

Geplante Arbeiten:

- Kundendetails umsetzen
- Steuerjahre integrieren
- Dokumentation weiterführen

Erledigte Arbeiten:

- Kundendetails fertiggestellt
- Steuerjahre integrieren (in Progress)
- Dokumentation weitergeführt

Kundendetails umsetzen

Beschreibung:	Heute habe ich angefangen die Kundendetails Seite zu erstellen, jedoch schnell gemerkt das ich nicht auf die Kundendetails Seite komme, also habe mit Hilfe von ChatGPT react-router-dom eingebaut. Sonst hatte ich hier keine Probleme.
Problem:	Ich konnte nicht zur Kundendetails Seite navigieren.
Lösung	Ich habe Navigation mit Hilfe von react-router-dom erstellt.
Zeit:	4.5h

Steuerjahre implementieren

Beschreibung:	Ich habe angefangen den Steuerjahre teil zu Implementieren und konnte die Ansicht und paar Funktionen umsetzen.
Zeit:	3.5h

Dokumentation weiterführen

Beschreibung:	Ich habe die Dokumentation weitergeführt als auch das Arbeitsjournal erstellt und die Backups gemacht.
Zeit:	1h

Reflexion

Heute habe ich 1 Stunde länger als geplant gearbeitet, denn beim Umsetzen der Kundendetails Seite ist schnell ein Problem entstand. Und zwar dass ich keine Navigation zu der Seite habe, Also habe ich hier mit Zusammenarbeit von ChatGPT eine Navigation gebaut. Sonst war die Umsetzung der Kundendetails Seite selbst kein Problem. Dank der einen Stunde konnte ich auch heute recht viel vom Steuerjahr teil realisieren. Da ich heute eine Stunde länger am Projekt gearbeitet habe, bin ich +- im Zeitplan drin. Ich bin gespannt auf Morgen.

Tagesaufwand: 9h

Gesamtaufwand: 52h

Restzeit: 33h

TAG 07 – 09.04.2025

Geplante Arbeiten:

- Steuerjahre Integrieren
- Validierung, Bestätigung und Feedback
- Styling (CSS)
Dokumentation weiterführen

Erledigte Arbeiten:

- Steuerjahre fertiggestellt
- Validierung, Bestätigung und Feedback eingebaut
- Styling (CSS) (in Progress)
- Dokumentation weitergeführt

Steuerjahre Integrieren

Beschreibung:	Ich habe das Formular zum Hinzufügen der Steuerjahre fertiggestellt und das Ganze in die Kundendetails Seite implementiert.
Zeit:	0.5h

Validierung, Bestätigung und Feedback

Beschreibung:	Hier habe Ich Validierung für die Formulare erstellt, Bestätigung eingebaut und Feedback Nachrichten platziert, wo es nötig ist. Hier traten keine Probleme auf.
Zeit:	3h

Styling (CSS)

Beschreibung:	Heute habe ich angefangen den CSS für mein Projekt zu realisieren, Ich bin recht weit gekommen bin jedoch nicht komplett fertig.
Zeit:	3h

Dokumentation weiterführen

Beschreibung:	Ich habe weiter an der Dokumentation gearbeitet, hier habe ich ein paar Änderungen an der Beschreibung meines Codes gemacht da es nicht ganz so war ich es ich möchte. Am Ende habe ich noch das Arbeitsjournal gemacht und die Backups erstellt.
Zeit:	2.5h

Reflexion

Ich habe heute eine kleine Änderung am Zeitplan vorgenommen, und zwar habe ich die Validierung, Bestätigung und das Feedback als auch das CSS den Zusatzfunktionen vorgezogen, da bei den Zusatzfunktionen, die ich noch machen muss, da es von Vorteil wäre, wenn das CSS schon gemacht ist. die Zusatzfunktionen haben auch nicht so eine hohe Priorität, die Anwendung funktioniert momentan sehr gut auch ohne diese. Momentan bin ich wie es aussieht etwas hintendrein im Zeitplan bin jedoch zuversichtlich das ich das Morgen Nacharbeiten kann. Sonst verlief der Tag gut und ich konnte viel erledigen.

Tagesaufwand: 9h

Gesamtaufwand: 61h

Restzeit: 24h

TAG 08 – 10.04.2025

Geplante Arbeiten:

- Zusatzfunktionen fertigstellen
- Kommentare hinzufügen
- Dokumentation weiterführen

Erledigte Arbeiten:

- CSS Fertiggestellt
- Zusatzfunktionen fertiggestellt
- Kommentare hinzugefügt
- Dokumentation weitergeführt

Styling (CSS)

Beschreibung:	Als erstes habe ich das CSS für meine Anwendung fertiggestellt, hier gab es keine Probleme. Ich habe das Ganze noch dem Fachvorgesetzten gezeigt dieser war sehr zufrieden.
Zeit:	1.5h

Zusatzfunktionen Implementieren

Beschreibung:	Ich habe zusätzliche Zusatzfunktionen für mein Projekt gemacht, hier konnte ich etwas Zeit sparen.
Zeit:	2.5h

Kommentare hinzufügen

Beschreibung:	In diesem Teil habe ich meinen Code kommentiert.
Zeit:	1h

Dokumentation weiterführen

Beschreibung:	Den Rest der Zeit habe ich damit verbraucht meine Dokumentation zu verbessern, vor allem beim Beschreiben des Codes. Am Ende des Tages habe ich noch das Arbeitsjournal und die Backups erstellt.
Zeit:	4h

Reflexion

Der heutige Tag lief gut Ich konnte das CSS und die Zusatzfunktionen komplett fertigstellen und Kommentare in meinen Code hinzufügen, ich habe nach Abschluss das Design noch dem Fachvorgesetzten gezeigt und dieser meinte das das, was ich gemacht habe, besser ist als die Mockups. Somit ist die Realisierungsphase meines Projekts komplett fertig und ich kann mich

morgen komplett auf die Tests Fokussieren. Heute habe ich auch noch einen grossen Teil der Zeit an meiner Dokumentation gearbeitet und den Inhalt verbessert und aktualisiert. Ich bin erleichtert das ich die Realisierungsphase einigermassen ohne Probleme durchführen konnte und freue mich auf Morgen.

Tagesaufwand: 9h

Gesamtaufwand: 70h

Restzeit: 15h

TAG 09 – 11.04.2025

Geplante Arbeiten:

- Testfälle durchführen
- Fehleranalyse und Behebung
- Dokumentation weiterführen

Erledigte Arbeiten:

- Testfälle durchgeführt
- Buffer Dokumentation
- Dokumentation weiterführen

Testfälle durchführen

Beschreibung:	Heute habe ich als erstes meine Testfälle durchgeführt, hierzu habe ich mich kurz informiert, wie ich diese durchführen soll und bin auf das Vitest Framework gestossen und habe mit Unterstützung von ChatGPT die Tests durchgeführt. Dieser Teil hat länger als geplant gedauert.
Zeit:	4h

Buffer Dokumentation

Beschreibung:	Ich musste meinen Testkonzept teil leicht anpassen und ergänzen und habe sonst noch Korrekturen und Ergänzungen an meiner Dokumentation durchgeführt.
Zeit:	3h

Dokumentation weiterführen

Beschreibung:	Ich habe das Testprotokoll erstellt und weiter an der Dokumentation gearbeitet, am ende des Tages habe ich noch das Arbeitsjournal und die Backups erstellt.
Zeit:	2h

Reflexion

Heute war die Kontrollieren Phase in meinem Projekt dran, dazu habe ich automatisierte Tests mit dem Vitest Framework gemacht. Die Tests habe mehr Zeit in Anspruch genommen als geplant jedoch liefen sie alle Fehlerfrei und ich bin erleichtert. Da meine Tests Fehlerfrei sind und ich somit keine Korrektur am Code vornehmen muss habe ich den Fehleranalyse und Behebung Block in meinem Zeitplan zu Pufferzeit Code und Dokumentation geändert und habe diese Zeit genutzt um weiter an der Dokumentation zu Arbeiten. Da Ich noch ein paar Änderungen an der Dokumentation machen wollte habe ich mich heute entschieden eine Stunde länger zu Arbeiten. Heute war der zweitletzte Tag der Durchführung meiner IPA und es bleibt nur noch der Montag übrig, aber allgemein habe ich ein gutes Gefühl.

Tagesaufwand: 9h

Gesamtaufwand: 79h

Restzeit: 6h

TAG 10 – 14.04.2025

Geplante Arbeiten:

- Benutzeranleitung erstellen
- Reflexion und Fazit
- Dokumentation beenden

Erledigte Arbeiten:

- Benutzeranleitung erstellt
- Reflexion und Fazit erstellt
- Pufferzeit Code und Dokumentation
- Dokumentation beendet

Benutzeranleitung erstellen

Beschreibung:	Ich habe eine kurze Benutzeranleitung für mein Projekt mit den wichtigsten Funktionen geschrieben und diese anschliessend einem Mitarbeiter gezeigt und geprüft ob für ihn alles klar war.
Zeit:	1h

Pufferzeit Code

Beschreibung:	In diesem Schritt bin ich nochmals durch meinen Code durchgegangen und kleine Änderungen an der Benennung des Codes vorgenommen. Da ich hier Teilweise nicht sehr klare Namen gegeben hatte.
Zeit:	1.5h

Pufferzeit Dokumentation

Beschreibung:	Ich habe die Dokumentation erweitert, Rechtschreibfehler korrigiert und das ganze besser formatiert.
Probleme:	Rechtschreibfehler
Lösung:	Korrekturen vorgenommen & Feedback von den Hilfestellungen umgesetzt.
Hilfestellung	Pantos Dejan
Zeit:	2.5h

Reflexion und Fazit schreiben

Beschreibung:	Hier habe ich eine Reflexion für mein Projekt und eine Danksagung erstellt, hier gab es keine Probleme.
Zeit:	1h

Dokumentation beenden

Beschreibung:	In Diesem Finalen Schritt habe ich das Arbeitsjournal geschrieben und alles Abgabebereit gemacht.
Zeit:	1h

Reflexion

Heute war der letzte Tag meiner IPA, ich konnte alle für heute geplanten Aufgaben erledigen und die Dokumentation abschliessen.

Tagesaufwand: 8h

Gesamtaufwand: 87h

Restzeit: 0h

Teil B: Projekt

Kurzfassung

Diese Kurzfassung richtet sich an Leser mit Fachwissen in der Informatik und gibt einen klaren Überblick über die IPA von Vladyslav Astashyn bei der ajooda AG.

Ausgangssituation

Aktuell versenden die Mitarbeiter von der ajooda AG Steuerelemente direkt per E-Mail. Das führt dazu, dass Dokumente von Kunden oft verloren gehen und Mitarbeiter zusätzlich Zeit investieren müssen, um Unterlagen erneut zu versenden.

Ziel

Ziel dieses Projekts ist die Entwicklung eines Prototyps einer webbasierten Steuerverwaltungsplattform, mit der Kunden und Mitarbeiter Steuerunterlagen zentral und ohne direkte Kommunikation über eine benutzerfreundliche Web-App verwalten und abrufen können.

Umsetzung

Die Steuerverwaltungsplattform wird als reine Frontend-WebApplikation mit dem React Framework entwickelt. Die Applikation wird mit TypeScript umgesetzt und CSS wird für das Styling der Applikation benutzt. Die Entwicklung wird in Visual Studio Code stattfinden.

Da kein Backend verwendet wird, wird die Datenspeicherung lokal im Browser mit localStorage gespeichert. Für die Simulation echter Nutzer werden Mock Daten verwendet.

Die Applikation wird in mehrere Komponenten gegliedert wie z. B. Startseite, Detailseite der Kunden oder Formulare für das Hinzufügen von Kunden und Steuerjahren.

Die Benutzeroberfläche berücksichtigt drei verschiedene Rollen (Admin,Advisor,Client) welche statisch verteilt werden da kein Backend vorgesehen ist.

Ergebnis

Die Web-App erfüllt alle vom Arbeitsgeber genannten Funktionen und Features.

Kunden können ihre Daten und Steuerjahre sehen und ihre Daten bearbeiten. Mitarbeiter haben die Möglichkeit Kundendaten und Steuerjahre zu bearbeiten und hinzuzufügen und auch die Möglichkeit die einzelnen Kunden in einer Liste zu sehen. Admins verfügen noch über die Option Steuerjahre und Kunden zu löschen und kann das machen das der Mitarbeiter auch kann. Die Anwendung bietet eine Benutzerfreundliche Oberfläche mit einfacher Navigation zum Verwalten von Kundendaten und deren Steuerjahre.

Einleitung

In diesem Abschnitt beschäftige ich mich mit dem Aufbau des Projekts. Das Projekt ist nach der IPERKA-Methode aufgebaut, welche für eine gegliederte Struktur und Flexibilität für den Projektablauf bietet. Hierzu sind die 6 Folgenden Phasen notwendig.

Informieren:

In dieser Phase informiert man sich so gut wie möglich über den Auftrag und dessen Aufgabenstellung, um alle nötigen Informationen zu erhalten und die offenen Fragen zu klären. Dies ist wichtig, um einen Zeitplan erstellen zu können und um zu verstehen, was genau bei diesem Auftrag erwartet wird.

Planen:

Bei dieser Phase geht es darum sich mit den erhaltenen Informationen einen Plan zur Realisierung des Projekts zu erstellen. Hier werden Zeitplan sowie ein Realisierungskonzept zur Umsetzung des Projekts erstellt.

Entscheiden:

In diesem Abschnitt entscheidet man sich für einen Lösungsweg welcher zur Realisierung des Projekts. Dabei wird festgelegt, wie das Projekt technisch umgesetzt werden soll z. B. mit welchen Technologien man arbeiten wird.

Realisieren:

Wenn der Auftrag definiert ist, man einen klaren Ablauf hat und sich für alles entschieden hat beginnt die Realisierungsphase des Projekts. Die Arbeitsabläufe werden protokolliert und die Ist-Werte werden im Zeitplan eingetragen.

Kontrollieren:

Das Resultat der Realisierungsphase wird hier getestet mit den erstellten Testkonzepten, aber erst wenn die Realisierungsphase komplett beendet wird. Fehlerhafte Tests werden dokumentiert und direkt behoben.

Auswerten:

Am Schluss des Projektes wird eine Reflexion geschrieben über die Arbeit und die Erfahrungen, die man dabei gesammelt hat. Hier geht man durch die ganze Arbeit durch und schaut was gut gelaufen ist was nicht so gut gelaufen ist und wo Verbesserungspotenzial besteht. Das hilft auch bei der Umsetzung für zukünftige Projekte.

Informieren

Das Informieren ist die erste Phase der IPERKA Projektmanagement Methode.

Das Durchlesen der Aufgabenstellung sowie das Klären von offenen Fragen sind die wichtigsten Punkte, um an das gewünschte Ergebnis zu kommen, ebenfalls kann man sich schon hier Gedanken machen, was mögliche Probleme wären.

Ziel der Aufgabenstellung

Das Ziel dieses Projekts ist es, eine benutzerfreundliche Webapplikation zu entwickeln, mit der Kunden und Steuerdaten verwaltet und dargestellt werden können. Die Anwendung soll es möglich machen, Kunden und Steuerjahre hinzuzufügen, zu bearbeiten und anzuzeigen. Dabei werden unterschiedliche Benutzerrollen berücksichtigt, um Funktionen gezielt freizugeben oder zu beschränken.

Vorgaben

Das Projekt wird als reines Frontend Projekt mit React entwickelt, ein Backend wird nicht integriert.

Die Benutzeroberfläche ist in zwei Ansichten aufgeteilt, die Admin/Advisor View und die Client View. Es gibt insgesamt drei Benutzerrollen diese wären Admin, Advisor und Client diese besitzen jeweils verschiedene rechte.

Die Daten werden mithilfe von Mock Daten simuliert, welche im Projekt lokal gespeichert sind.

Fragen

Frage an Veselin Lakic (01.04.2025): Soll ich JavaScript oder TypeScript für React nutzen?

Antwort: Das ist dir Frei überlassen informiere dich dazu und entscheide selbst, was bei diesem Projekt Sinn machen würde.

Frage an Veselin Lakic (01.04.2025): Was soll Teil XY auf den Mockups machen?

Antwort: Teil XY ist für XY Funktion damit die Benutzeroberfläche besser bedienbar ist, dies ist aus meiner Sicht ein Must-have-Feature bei solchen Anwendungen.

(Die zusätzlichen Funktionen wurden im teil „Funktionen“ ins Detail beschrieben)

Planen

Das Planen ist die zweite Phase der IPERKA-Methode.

In dieser Phase wird der Zeitplan fertiggestellt, ein Realisierungskonzept erstellt sowie ein Testkonzept für die spätere Kontrolle. Außerdem wird ein Usecase Diagramm zur Visualisierung der Anwendung erstellt.

Realisierungskonzept

Die Anwendung wird mit dem React Framework in Visual Studio Code entwickelt. Für die Programmierung werde ich TypeScript verwenden und für das Styling CSS.

Das Projekt wird in mehrere Komponenten mit einer klaren Ordnerstruktur aufgeteilt:

HomePage mit der Kunden Übersicht für admin/advisor

ClientDetailsPage für die Kundendaten und Steuer Jahre

Zwei Formulare zum Hinzufügen von Kunden und Steuerdaten

Geteilte Komponenten wie den Rollenwechsler

Die Daten werden im localStorage des Browsers gespeichert. Zur Simulation von einer DB werde ich Mock Daten in einer Datei im Projekt einbauen.

Das Projekt wird drei Rollen haben (Admin, Advisor, Client) diese werden mit dem Rollenwechsler gewechselt und je nach Rolle werden bestimmte Funktionen zur Verfügung stehen.

Die Umsetzung wird in 10 Schritten erfolgen, wie in meinem Zeitplan dargestellt.

1. Projektstruktur erstellen
2. Startseite umsetzen
3. LocalStorage integrieren
4. Rollen und Rechte einbauen
5. Kundendetailseite erstellen
6. Steuerjahre hinzufügen
7. Zusatzfunktionen einbauen (z. B. Ein-/Ausklappen)
8. Validierung und Bestätigungen
9. Styling mit CSS
10. Kommentare schreiben und Tests durchführen

Im Verlauf der Realisierung vom Projekt werde ich Sachen immer wieder testen, um sicherzustellen, dass es so funktioniert wie geplant. Da es sich um eine Neuentwicklung handelt, werden keine Anpassungen an einer bestehenden Applikation nötig sein.

Usecase Diagramm

Um die benutzerrollen und Hauptfunktionen der Anwendung darzustellen habe ich ein Usecase Diagramm erstellt. Das wurde mit draw.io gemacht.

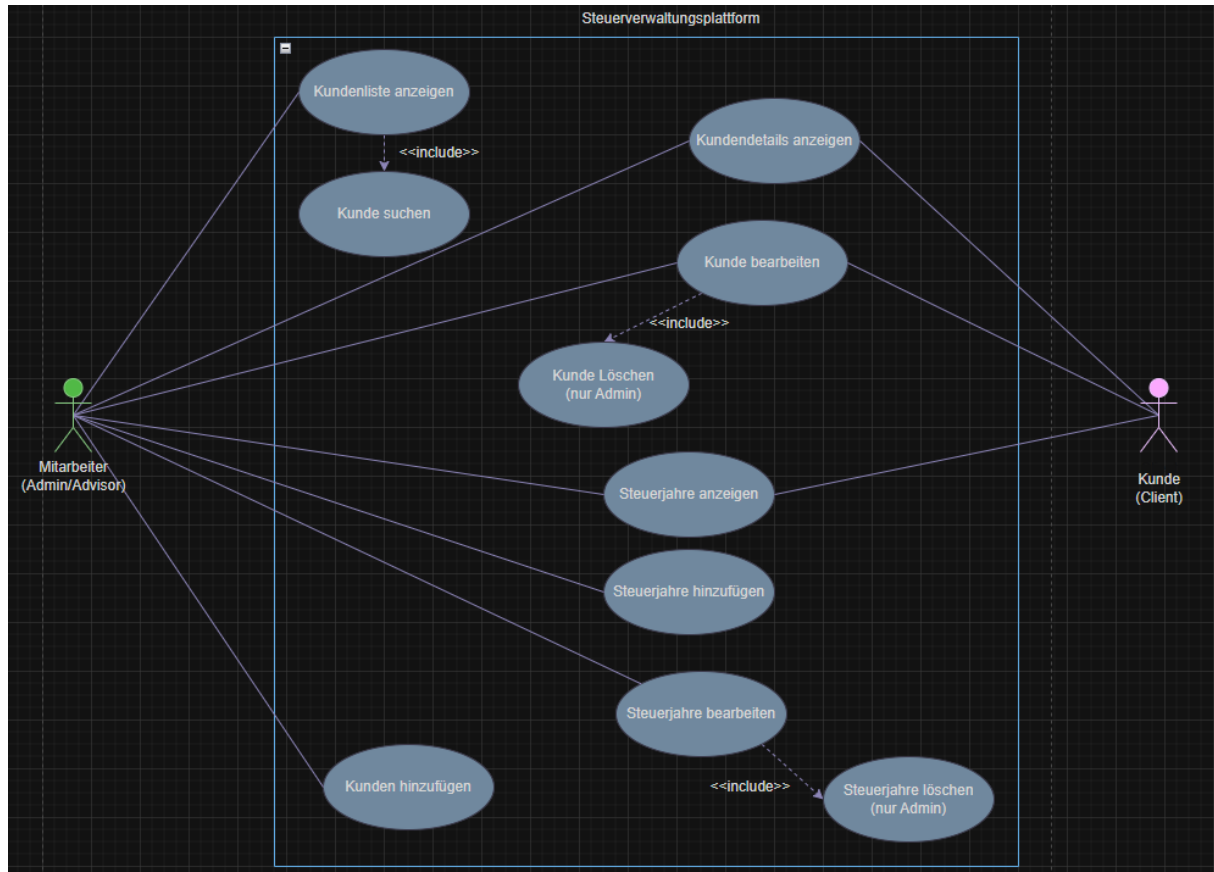


Abbildung 2 Usecase Diagramm

Mockups

Die Mockups waren grösstenteils im Voraus schon erstellt. Ich habe kleine Sachen angepasst wie die Sprache auf English geändert und die Komponenten besser und kompakter positioniert. Die Änderungen wurde mit Figma gemacht und mit meinem Fachvorgesetzten besprochen, damit das Design besser zu den Anforderungen passt.

Homepage:

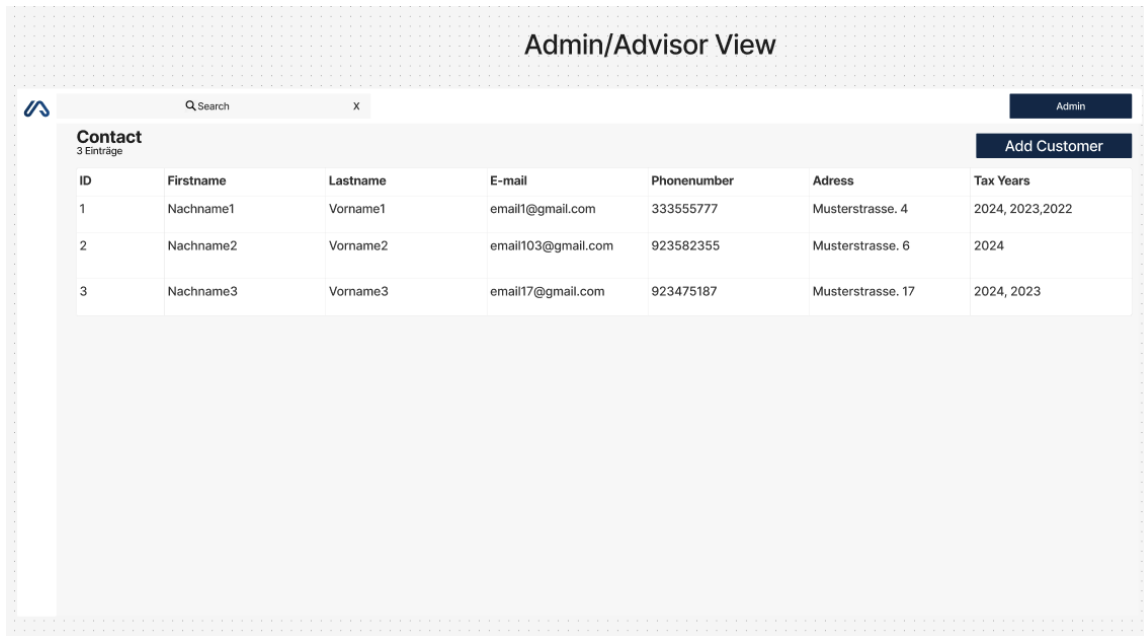


Abbildung 3 Mockups Startseite

Customerdetailspage:

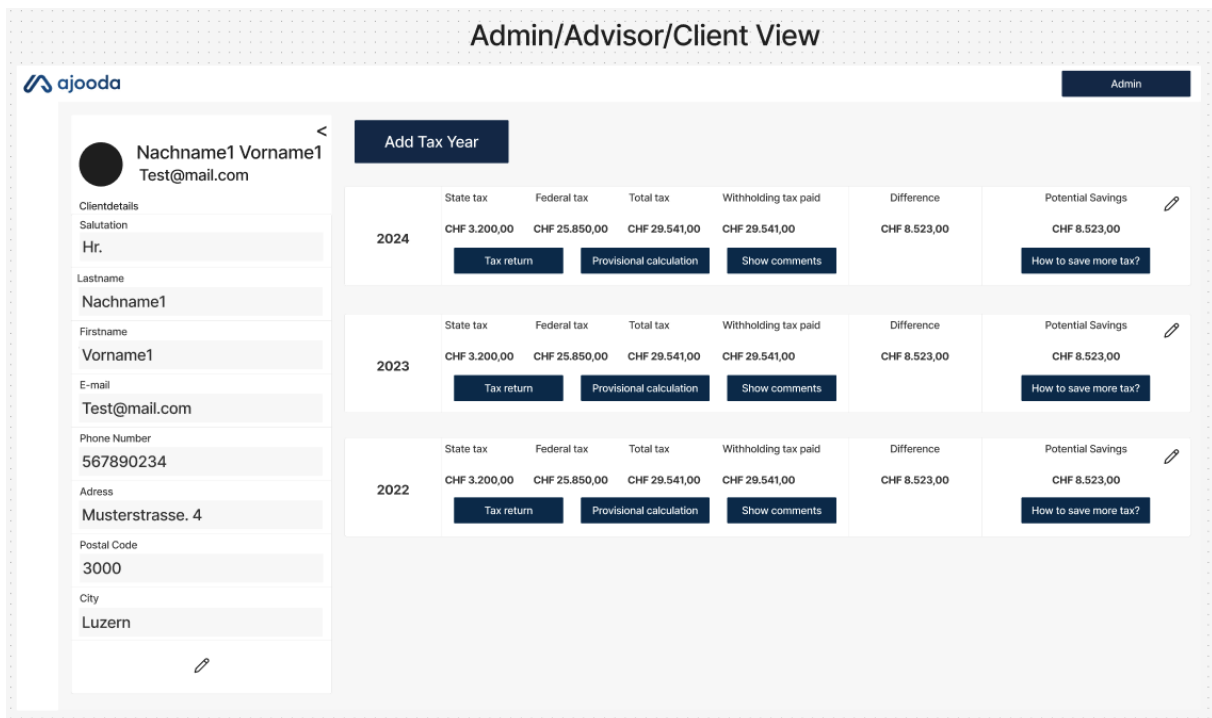


Abbildung 4 Mockups Kundendetailsseite

Testkonzept

Status:	In Arbeit
Programmname:	Steuerverwaltungsplattform
Projektleiter:	Vladyslav Astashyn
Version:	1.1
Datum:	02.04.2025
Auftraggeber:	ajooda AG

Testobjekte

Nr.	Objekt	Beschreibung
1	Startseite	Die Startseite der Anwendung wo alle Kunden zu sehen sind.
2	Kundendetailseite	Die Kundendetailseite, wo die kompletten Kundendetails und die Steuerjahre des Kunden sind.
3	Kundenformular	Formular zum Hinzufügen eines neuen Kunden, befindet sich auf der Startseite.
4	Steuerjahrformular	Formular zum Hinzufügen eines neuen Steuerjahres für einen Kunden.
5	Suchfunktion	Suchleiste für die Suche der Kunden in der Liste.
6	Rollenlogik	Je nach Rolle, stehen einem verschiedene Funktionen zur Verfügung.
7	Navigation	Navigation zwischen Startseite und Kundendetailsseite.

Testarten

Nr.	Testart	Beschreibung
1	Integration	Integration-tests

In meinem Projekt kommen hauptsächlich Integrationstests zum Einsatz, weil es sich hier um eine reine Frontend-Anwendung ohne Backend handelt. Integrationstests sind automatisierte Tests, welche das Zusammenspiel von mehreren Funktionen oder Komponenten prüfen.

Mängelklassifizierung

Nr.	Testart	Beschreibung
0	mängelfrei	Einwandfrei und anforderungsgerecht
1	belangloser Mangel	Verwendung möglich, Mangel sollte aber nicht vorkommen
2	leichter Mangel	Verwendung möglich, aber leicht eingeschränkt
3	schwerer Mangel	Verwendung stark eingeschränkt
4	kritischer Mangel	Anwendung unbrauchbar, Funktionalität fehlt, evtl. sicherheitsrelevant

Testumgebung

Die Tests laufen lokal mit dem Testframework Vitest, welches das Nutzerverhalten im Browser automatisiert simuliert.

Laptop Specs

Windows 11 Home (Version 24H2)

Intel(R) Core(TM) 7 150U 1.80GHz

16.0 GB RAM

64-Bit-Betriebssystem

Testinfrastruktur

Testseite im Browser

Testdaten

Die Testdaten befinden sich im Projektverzeichnis der Anwendung

Testfallbeschreibung

ID/Bezeichnung	T-1 INT	Kunde hinzufügen
Beschreibung	Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden.	
Testvoraussetzung	Startseite und Kundenformular sind fertiggestellt	
Testschritte	<ol style="list-style-type: none"> 1. Auf "Add Client" klicken 2. Formular ganz ausfüllen 3. Auf Bestätigen klicken 	
Erwartetes Ergebnis	Die Kundendaten werden in die Liste hinzugefügt.	

ID/Bezeichnung	T-2 INT	Kunde löschen
Beschreibung	Beim Klicken des Lösch Buttons wird der Kunde gelöscht.	
Testvoraussetzung	Startseite fertiggestellt und Rolle: Admin implementiert	
Testschritte	<ol style="list-style-type: none"> 1. "Kunde löschen" Klicken 2. Prüfen, ob der gelöschte Kunde aus der Liste gelöscht wird 	
Erwartetes Ergebnis	Der Kunde wird aus der Liste gelöscht.	

ID/Bezeichnung	T-3 INT	Kundendaten bearbeiten
Beschreibung	Die Daten eines Kunden werden bearbeitet und gespeichert.	
Testvoraussetzung	Kundendetails und Detailseite fertiggestellt, Rolle: Admin/Advisor/Client implementiert	
Testschritte	<ol style="list-style-type: none"> 1. Zur Detailseite navigieren 2. Auf "Bearbeiten" Button Klicken 3. Daten des Kunden ändern 4. Änderung speichern 5. Prüfen, ob die Änderung übernommen wurde 	
Erwartetes Ergebnis	Die geänderten Daten werden gespeichert und korrekt angezeigt.	

ID/Bezeichnung	T-4 INT	Steuerjahr löschen
Beschreibung	Beim Klicken des löscht Buttons der Steuerjahre wird das Steuerjahr gelöscht.	
Testvoraussetzung	Kundendetails und Steuerjahre fertiggestellt, Rolle Admin implementiert.	
Testschritte	<ol style="list-style-type: none"> 1. Zu Detailseite navigieren 2. Auf "Steuerjahr löschen" klicken 3. Prüfen, ob das Steuerjahr aus der Liste entfernt, wurde 	
Erwartetes Ergebnis	Steuerjahr wird korrekt gelöscht.	

ID/Bezeichnung	T-5 INT	Steuerjahr bearbeiten
Beschreibung	Die Daten des Steuerjahres eines Kunden werden bearbeitet und korrekt gespeichert.	
Testvoraussetzung	Kundendetails und Steuerjahre fertiggestellt	
Testschritte	<ol style="list-style-type: none"> 1. Zu Detailseite navigieren 2. Steuerjahr bearbeiten klicken 3. Steuerjahr Werte ändern 4. Änderung speichern 5. Prüfen, ob die Änderung übernommen wurde 	
Erwartetes Ergebnis	Die Werte des Steuerjahres werden korrekt geändert und gespeichert.	

ID/Bezeichnung	T-6 UNIT	Steuerjahr hinzufügen
Beschreibung	Nach dem Ausfüllen des Steuerjahr Formulars wird das Steuerjahr korrekt hinzugefügt	
Testvoraussetzung	Kundendetails und Steuerjahre fertiggestellt	
Testschritte	<ol style="list-style-type: none"> 1. Zu Detailseite navigieren 2. Auf "Steuerjahre hinzufügen" klicken 3. Formular ausfüllen 4. Formular abschicken 5. Prüfen, ob das Steuerjahr Korrekt hinzugefügt wurde 	
Erwartetes Ergebnis	Die Steuerjahr Daten werden korrekt hinzugefügt	

ID/Bezeichnung	T-7 INT	Kunde hinzufügen und in Details aufrufen
Beschreibung	Ein Kunde wird hinzugefügt und über die Tabelle aufgerufen.	
Testvoraussetzung	Startseite und Detailseite fertiggestellt	
Testschritte	<ol style="list-style-type: none"> 1. Auf "Add Client" klicken 2. Formular korrekt ausfüllen und bestätigen 3. Auf den Kunden in der Tabelle klicken 4. Prüfen, ob die Detailseite angezeigt wird 	
Erwartetes Ergebnis	Die neu eingegebenen Kundendaten werden korrekt in der Detailseite angezeigt	

ID/Bezeichnung	T-8 INT	Kunde bearbeiten und Suchfunktion nutzen
Beschreibung	Einen Kunden bearbeiten und mit der Suchfunktion suchen.	
Testvoraussetzung	Startseite und Detailseite fertiggestellt, Rolle Admin implementiert.	
Testschritte	<ol style="list-style-type: none"> 1. Zu Kundendetails navigieren 2. Kunde bearbeiten und den Namen ändern 3. Zurück zur Startseite navigieren 4. Geänderten Namen in der Suchleiste eingeben 	
Erwartetes Ergebnis	Der bearbeitete Kunde wird mit der Suchfunktion korrekt angezeigt.	

ID/Bezeichnung	T-9 INT	Rolle beeinflusst Benutzeroberfläche und Rechte
Beschreibung	Beim Wechseln der Rolle ändert sich die Benutzeroberfläche und die Rechte	
Testvoraussetzung	Startseite fertiggestellt, Rolle: Admin/Advisor implementiert	
Testschritte	<ol style="list-style-type: none"> 1. Prüfen, ob Löschfunktion sichtbar ist 2. Rolle zu Advisor wechseln 3. Prüfen, ob die Löschfunktion verschwindet 	
Erwartetes Ergebnis	Der Button wird je nach Rolle korrekt angezeigt.	

ID/Bezeichnung	T-10 INT	Kunde hinzufügen und Steuerjahr hinzufügen
Beschreibung	Kunde wird hinzugefügt und danach ein Steuerjahr	
Testvoraussetzung	Startseite und Detailseite fertiggestellt	
Testschritte	<ol style="list-style-type: none"> 1. Auf „Add Client“ klicken 2. Kundenformular ausfüllen und bestätigen 3. Auf den neuen Kunden in der Tabelle klicken 4. Auf „Steuerjahr hinzufügen“ klicken 5. Steuerjahr Formular ausfüllen 6. Steuerjahr Formular abschicken 7. Prüfen, ob beide Einträge korrekt übernommen wurden 	
Erwartetes Ergebnis	Der neue Kunde wird korrekt hinzugefügt und angezeigt, das neue Steuerjahr wird korrekt hinzugefügt und angezeigt	

ID/Bezeichnung	T-11 INT	Steuerjahr bearbeiten und korrekt auf Startseite anzeigen
Beschreibung	Ein Steuerjahr wird bearbeitet dann wird zur Startseite navigiert und das bearbeitete Steuerjahr wird angezeigt.	
Testvoraussetzung	Startseite und Detailseite fertiggestellt	
Testschritte	<ol style="list-style-type: none"> 1. Zur Detailseite navigieren 2. Steuerjahr bearbeiten klicken 3. Werte des Steuerjahres ändern 4. Geänderte werte Speichern 5. Zurück zur Startseite navigieren 6. Prüfen, ob das geänderte Steuerjahr korrekt angezeigt wird 	
Erwartetes Ergebnis	Das bearbeitete Steuerjahr wird auf der Startseite korrekt angezeigt	

ID/Bezeichnung	T-12 INT	Kunde löschen und erneuten Zugriff prüfen
Beschreibung	Ein Kunde wird gelöscht und kann nicht mehr über die Detailseite aufgerufen werden	
Testvoraussetzung	Startseite, Detailseite und localStorage fertiggestellt	
Testschritte	<ol style="list-style-type: none"> 1. Zur Detailseite eines Kunden navigieren 2. URL-Link kopieren 3. Kunden löschen 4. URL-Link einfügen 5. Prüfen, ob der Kunde noch erreichbar ist über die Detailseite 	
Erwartetes Ergebnis	Der gelöschte Kunde ist über einen URL-Link nicht erreichbar und wird nicht angezeigt	

Entscheiden

Das Entscheiden ist die dritte Phase der IPERKA-Methode.

In dieser Phase wird entschieden, welche Lösungsvariante sich am besten für das Projekt eignet und ob diese so auch umsetzbar ist.

Varianten

Während der Planung habe ich mir überlegt, wie ich gewisse Funktionen und Features umsetzen werde. Ein Punkt war die Rollenzuteilung, dort waren die Optionen ein richtiges backend Login, was direkt wegfällt da das Projekt ein reines Frontend-Projekt werden soll.

Dann war noch eine Option das Login selbst zu erstellen oder einfach ein UI-Element zu machen, in welchem man die Rollen direkt wechseln kann.

Die Darstellung der Formulare war auch ein Thema, wie und wo ich diese platzieren soll, eine Variante wäre diese auf eine eigene Seite zu packen oder diese als Popup zu machen.

Bei der Datenspeicherung hatte ich nicht viele Optionen, da es sich eben um ein reines Frontend-Projekt handelt. Eine Datenbank wäre zwar die beste Lösung, kam hier aber nicht in Frage. Deshalb habe ich mich entschieden die Daten im LocalStorage des Browsers zu speichern.

Ein weiterer Punkt war die Wahl zwischen JavaScript und TypeScript, da beides mit React funktioniert, dazu habe ich mich zu Beginn informiert, was bei meinem Projekt am meisten Sinn machen würde.

Entscheiden

Bei den Rollen habe ich mich für ein einfaches UI-Element entschieden, mit dem man die Rolle direkt wechseln kann, ich fand das ausreichend für mein Projekt da das zum Simulieren der Rollen reicht und dieser Teil bei Späteren Weiterentwicklung des Projekts ersetzt wird.

Die Formulare werde ich als Popup umsetzen, weil das etwas übersichtlicher ist und der User auf der gleichen Seite bleibt und dafür keine extra Seite angelegt werden muss.

Die Daten werden im LocalStorage gespeichert, weil es für ein reines Frontend-Projekt mit Mock Daten die einfachste und sinnvollste Lösung ist.

Bei der Sprache habe ich mir für TypeScript entschieden, da man damit Fehler frühzeitig erkennt, vor allem bei Props und der Typisierung von States was die Entwicklung sicherer macht und klarer macht.

Realisieren

Das Realisieren ist die vierte Phase der IPERKA-Methode.

In dieser Phase wird das Projekt umgesetzt. Funktionen und Elemente die wichtig sind werden hier eingebaut und beschrieben.

React

React ist eine weit verbreitete Open-Source-JavaScript-Bibliothek, die sich auf die Entwicklung von Benutzeroberflächen spezialisiert hat. Sie wurde ursprünglich von Facebook entwickelt und wird heute von einer großen Entwickler-Community weiter gepflegt und weiterentwickelt.

React bietet eine moderne, komponentenbasierte Architektur, mit der sich komplexe Benutzeroberflächen in kleine, wiederverwendbare Teile zerlegen lassen. Diese Komponenten können unabhängig voneinander entwickelt, getestet und gewartet werden.

Einige der zentralen Merkmale von React sind:

- **JSX:** Eine Syntaxerweiterung für JavaScript, die HTML-ähnlichen Code direkt im JavaScript ermöglicht und dadurch die Entwicklung intuitiver macht.
- **Virtuelles DOM:** React arbeitet mit einem sogenannten Virtual DOM, das für effizientes Rendering sorgt und nur die Komponenten aktualisiert, die sich wirklich geändert haben.
- **Hooks:** Seit React 16.8 bieten Hooks eine elegante Möglichkeit, Zustände und Seiteneffekte in Funktionskomponenten zu verwalten.

React lässt sich sehr gut mit **TypeScript** kombinieren. Dies bringt den Vorteil, dass Props, States und andere Typen frühzeitig geprüft werden können, was die Codequalität verbessert und Fehler reduziert.

React wird weltweit in unzähligen Projekten eingesetzt – von kleinen internen Tools bis hin zu großen Unternehmensanwendungen. Durch seine Flexibilität und starke Community eignet sich React besonders gut für moderne Frontend-Projekte.

(Dieser Text wurde von einer KI generiert)

Projekt Setup und Ordnerstruktur

Der Erste Schritt den ich in meiner Realisierungsphase geplant habe ist das Aufsetzen des Projekts und der Ordnerstruktur, zum Erstellen einer React App kenne ich zwei Möglichkeiten die eine wäre create-react-app und die andere wäre über Vite, ich habe mich für Vite entschieden da das eine bessere Alternative zu create-react-app ist und Vorteile hat wie einen schnelleren Serverstart und Build-Zeit, die Konfiguration ist Flexibler die Entwicklungsserver schnellere Reload zeiten haben was für mein Projekt optimal klingt.

Dazu habe bin ich einem Youtube Guide gefolgt. Das lief folgendermassen ab:

Node --version um zu überprüfen, ob man node.js installiert hat

npm install -g vite um Vite global auf dem Rechner zu installieren

npm create vite@latest um mit dem Beginn der Aufsetzung des Projektes zu beginnen. Dann konnte ich Sachen wie Projektname, Framework und Programmiersprache auswählen:

```
|
|
|◇ Project name:
|  IPA
|
|◇ Package name:
|  ipa
|
|◇ Select a framework:
|  React
|
|◇ Select a variant:
|  TypeScript
|
|◇ Scaffolding project in C:\Users\vladi\Desktop\IPA\IPA...
|
| Done. Now run:
```

Abbildung 5 Projekt Setup

zum Abschluss der Erstellung vom Projekt musste ich noch diese Befehle eingeben, um die Abhängigkeiten zu installieren mit `cd IPA` und `npm install`

als letztes kann ich noch den Server mit `npm run dev` starten.

Komplett sah das ganze bei mir so aus:

```
PS C:\Users\vladi\Desktop\IPA> node --version
v22.13.0
PS C:\Users\vladi\Desktop\IPA> npm install -g vite

added 10 packages in 3s

3 packages are looking for funding
  run `npm fund` for details
PS C:\Users\vladi\Desktop\IPA> npm create vite@latest
Need to install the following packages:
create-vite@6.3.1
Ok to proceed? (y) y

> npx
> cva

|
| ◊ Project name:
|   IPA
|
| ◊ Package name:
|   ipa
|
| ◊ Select a framework:
|   React
|
| ◊ Select a variant:
|   TypeScript
|
| ◊ Scaffolding project in C:\Users\vladi\Desktop\IPA\IPA...
|
| Done. Now run:
|
| cd IPA
| npm install
| npm run dev
|
PS C:\Users\vladi\Desktop\IPA> cd IPA
PS C:\Users\vladi\Desktop\IPA\IPA> npm install

added 181 packages, and audited 182 packages in 9s

43 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Abbildung 6 Projektsetup komplett

Mein nächster Schritt war das Erstellen der Ordnerstruktur, diese wurde so aufgebaut das die es eine logische Trennung und Zuteilung von Seiten, Formularen, wiederverwertbaren Komponenten und CSS gibt. Im Verlauf des Projekts wird noch bei Bedarf die Ordnerstruktur erweitert.

Ordnerstruktur:

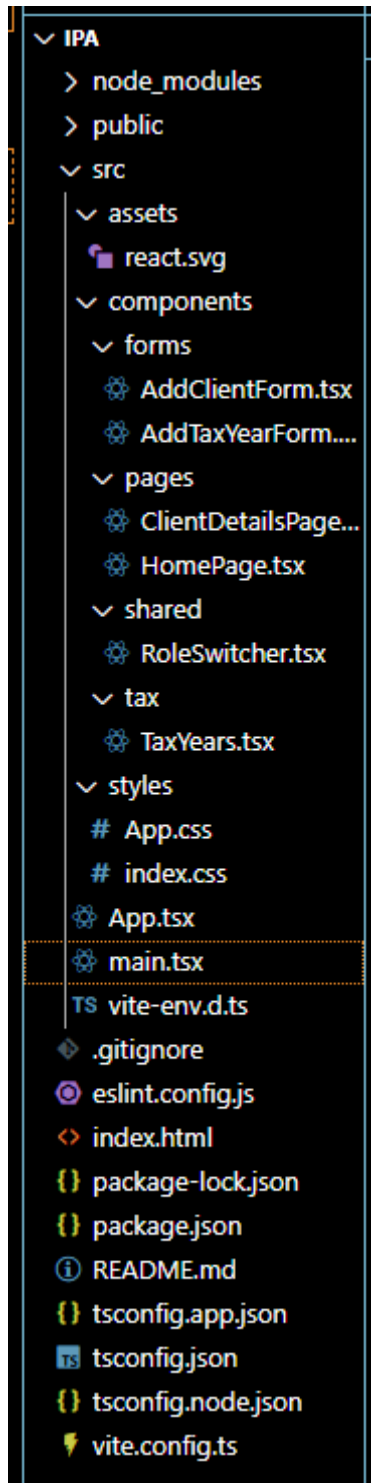


Abbildung 7 Ordnerstruktur Anfang

Mit diesem Setup war ich bereit mit der eigentlichen Umsetzung meines Projekts.

Hier ist noch ein Screenshot wie es am Ende von meinem Projekt ausgesehen hat:

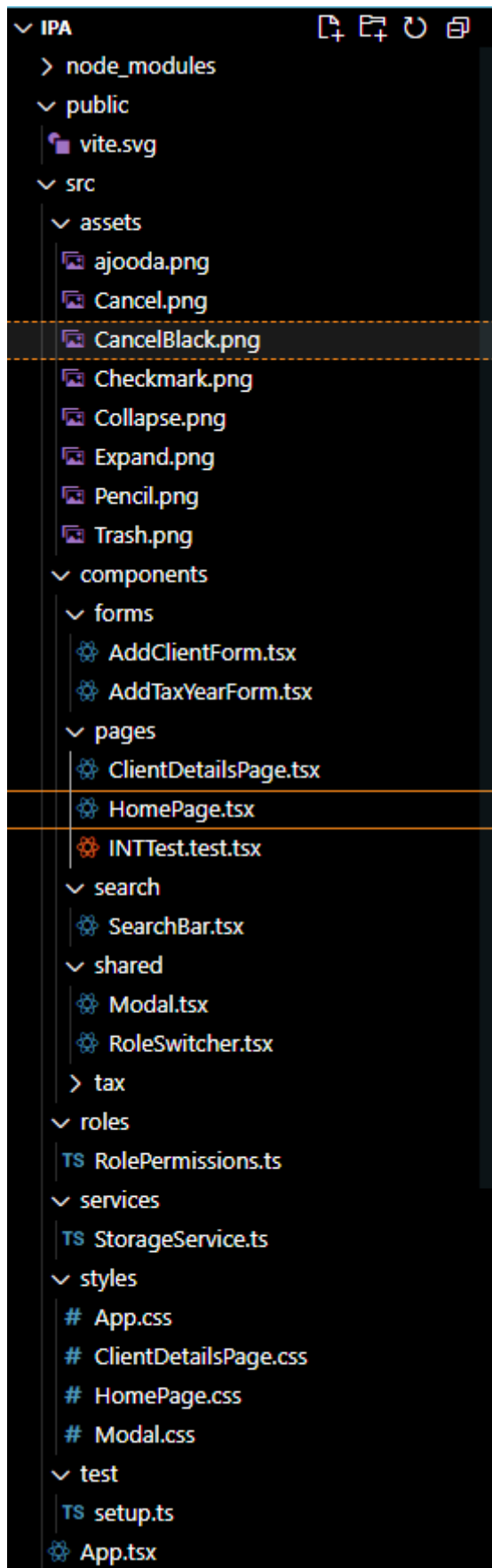


Abbildung 8 Ordnerstruktur Ende 1

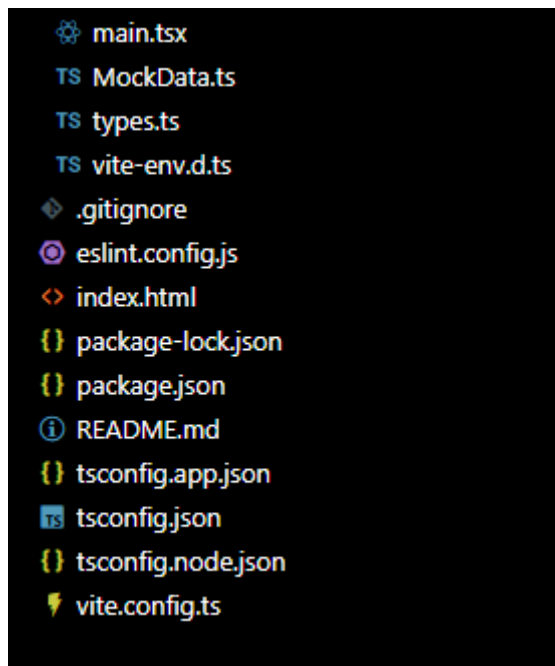


Abbildung 9 Ordnerstruktur Ende 2

Kundenübersicht umsetzen

Die Kundenübersicht ist die Startseite meiner Anwendung und soll nur für Benutzer mit den Rollen Admin oder Advisor sichtbar sein. Diese Seite zeigt eine Liste aller vorhandenen Kunden an, welche im LocalStorage gespeichert sind, falls noch keine Kunden im LocalStorage vorhanden sind, werden die Mock Daten beim ersten Laden in den LocalStorage gespeichert. Diese Seite hat noch 3 weitere wichtige Funktionen, dieser wären: Kunden Hinzufügen, Kunden Löschen und eine Suchfunktion zum Filtern der Kunden.

useEffect

```
useEffect(() => {
  try {
    const parsedData = loadClients();
    if (parsedData.length > 0) {
      setClients(parsedData);
      const storedCounter = localStorage.getItem("idCounter");
      if (!storedCounter) {
        let highestId = 0;
        if (parsedData.length > 0) {
          const ids = parsedData.map((client) => {
            return client.id ? client.id : 0;
          });
          highestId = Math.max(...ids);
        }
        localStorage.setItem("idCounter", (highestId + 1).toString());
      }
    } else {
      setClients(clientData);
      saveClients(clientData);
      localStorage.setItem("idCounter", (clientData.length + 1).toString());
    }
  } catch (error) {
    console.error("Failed to load clients:", error);
    alert("Failed to load client data. Please try again later.");
  }
}, []);
```

Abbildung 10 *useEffect* Funktion

Die *useEffect* Funktion ist ein wichtiger Bestandteil meiner Startseite. Sie übernimmt das Initialisieren der Kundendaten beim ersten Laden der Anwendung. Falls im *localStorage* noch keine Kundendaten vorhanden sind, werden automatisch die im Projekt vorhandenen Mock Daten in den *localStorage* gespeichert. Zusätzlich wird noch ein ID-Counter gesetzt und der State mit den Kundendaten wird aktualisiert. Falls bereits Kundendaten im *localStorage* vorhanden sind, werden stattdessen diese geladen.

Hier hatte ich die Möglichkeit das ganze über *useState* zu lösen, habe mich jedoch für *useEffect* entschieden da die Kundendaten so asynchron beim Laden der Seite geladen werden.

(Der If teil wurde von ChatGPT erstellt)

Kunde hinzufügen (Add Client)

Der Add Client Button dient dazu einen neuen Kunden in die bestehende Liste der Kunden hinzuzufügen und funktioniert wie folgt: beim Klick auf den Button erscheint ein Neues Fenster, in welchem man alle nötigen Kundendaten eintragen kann, beim Absenden des Formulars wird das Ganze auf Korrektheit und Vollständigkeit geprüft wie z. B., ob eine gültige E-Mail-Adresse angegeben wird oder die Telefonnummer nur aus Zahlen und den erlaubten Zeichen besteht. Bei Erfolgreicher Validierung wird diesem Kunden eine eindeutige ID zugewiesen und er wird im localStorage gespeichert und der State wird aktualisiert das der neue Kunde sofort in der Kundenliste angezeigt wird. Bei Erfolgreichem hinzufügen des Kunden wird noch eine Bestätigung des erfolgreichen hinzufügen angezeigt.

handleChange

```
const handleChange = (
  e: React.ChangeEvent<HTMLInputElement | HTMLSelectElement>
) => {
  const { name, type } = e.target;
  let { value } = e.target;

  if (name === "phone") {
    value = value.replace(/[^0-9+\s]/g, "");
  }

  if (name === "firstName" || name === "lastName") {
    value = value.replace(/[^A-Za-zÄÖÜäöü\s'-]/g, "");
  }

  if (name === "city") {
    value = value.replace(/[^A-Za-zÄÖÜäöüß\s'-]/g, "");
  }

  if (name === "address") {
    value = value.replace(/[^A-Za-z0-9ÄÖÜäöüß\s.,#/\-']/g, "");
  }

  setClient((prev) => ({
    ...prev,
    [name]: type === "number" ? (value ? Number(value) : 0) : value,
  }));
};
```

Abbildung 11 handleChange Funktion

Diese Funktion ist dazu da, um unerlaubte Zeichen bei der Eingabe zu filtern und die Eingabe im client State zu aktualisieren.

(Der Regex für die if abfrage wurde von ChatGPT erstellt.)

handleSubmit

```
const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();

  if (
    !client.salutation ||
    !client.lastName ||
    !client.firstName ||
    !client.email ||
    !client.phone ||
    !client.address ||
    !client.postalCode ||
    !client.city
  ) {
    alert("Please fill in all fields.");
    return;
  }

  if (!validateEmail(client.email)) {
    alert("Please enter a valid email address.");
    return;
  }
  if (!validatePhone(client.phone)) {
    alert("Please enter a valid phone number.");
    return;
  }
  if (!validatePostalCode(client.postalCode)) {
    alert("Please enter a valid postal code.");
    return;
  }

  onClientAdded(client);
  setFeedback("Client added successfully!");
  setTimeout(() => {
    onClose();
  }, 2000);
};
```

Abbildung 12 *handleSubmit* Funktion

Diese Funktion prüft beim Absenden des Formulars, ob die eingegebenen Werte das gewünschte Format haben, falls das der Fall ist, wird `onClientAdded` ausgelöst welches den Kunden in einem State speichert, eine Bestätigungsnachricht angezeigt und das Formularfenster wird nach 2 Sekunden automatisch geschlossen.

handleClientAdded

```
const handleClientAdded = (newClient: Client) => {  
  const savedData = loadClients();  
  const clientWithId = { ...newClient, id: getNextClientId() };  
  const updatedClients = [...savedData, clientWithId];  
  setClients(updatedClients);  
  saveClients(updatedClients);  
};
```

Abbildung 13 *handleClientAdded* Funktion

Diese Funktion wird ausgelöst, wenn das Formular erfolgreich abgesendet wurde. Zunächst wird die bestehende Kundenliste aus dem localStorage geladen, dann wird dem neuen Kunden Objekt eine ID gegeben und zusammen mit der bestehenden Kundenliste in eine neue Liste hinzugefügt. Die aktualisierte Liste wird im State gesetzt und auch im localStorage gespeichert.

Kunde Löschen (Delete)

Der Delete Button ist dafür zuständig einen Kunden zu löschen und funktioniert folgendermassen: beim Klicken auf den Delete Button wird *handleClientDeleted* aufgerufen und die ID des ausgewählten Kunden wird mitgegeben. Diese Funktion ruft dann *deleteClientFromStorage* auf, um den Kunden aus dem localStorage zu löschen. Nachdem wird die Kundenliste im State aktualisiert und ein Feedback wird angezeigt.

handleClientDeleted

```
const handleClientDeleted = (id: number) => {  
  if (role !== "Admin") {  
    alert("You do not have permission to delete clients.");  
    return;  
  }  
  if (window.confirm("Are you sure you want to delete this client?")) {  
    deleteClientFromStorage(id);  
    const updatedClients = clients.filter((client) => client.id !== id);  
    setClients(updatedClients);  
    setFeedback("Client deleted successfully!");  
    setTimeout(() => setFeedback(""), 2000);  
  }  
};
```

Abbildung 14 *handleClientDeleted* Funktion

Diese Funktion wird ausgelöst sobald der Delete Button geklickt wurde. Falls der User kein Admin ist, wird eine Warnung angezeigt und der Löschvorgang wird abgebrochen. Bevor der Löschprozess beginnt, wird ein Bestätigungspopup erscheinen welches fragt, ob man den Kunden wirklich löschen will. Falls das der Fall ist, wird *deleteClientFromStorage* aufgerufen welches dann den Kunden aus der Kundenliste im localStorage löscht. Dann wird die Kundenliste gefiltert und im State aktualisiert, am Schluss erscheint noch ein Feedback.

Suchfunktion (SearchBar)

Die Suchfunktion auf der Startseite ist eine separate Komponente welche dazu da die Kunden aus der Kundenliste zu filtern. Hier kann man nach Vornamen, Nachname und E-Mail suchen. Während der Suche wird die Liste automatisch bei jeder Eingabe aktualisiert. Falls die gesuchte Eingabe nicht existiert, steht in der Liste: No Clients found.

filteredClients

```
const filteredClients = clients.filter((client) =>
  `${client.firstName} ${client.lastName} ${client.email}`
  .toLowerCase()
  .includes(searchTerm.toLowerCase())
);
```

Abbildung 15 Filterlogik

Die Filterlogik vergleicht hier die Eingabe (searchTerm) mit den Vornamen, Nachnamen und E-Mails der Kunden und wandelt diese in Kleinbuchstaben um damit Gross- und Kleinschreibung bei der Suche keine Rolle spielt.

localStorage integrieren

Der localStorage dient in meiner Anwendung als Speichermedium für die Kundendaten. Dieser ermöglicht es mir die CRUD-Operationen (Create, Read, Update, Delete) in meinem Projekt lokal und korrekt auszuführen.

loadClients

```
export const loadClients = (): Client[] => {  
  const data = localStorage.getItem(STORAGE_KEY);  
  return data ? JSON.parse(data) : [];  
};
```

Abbildung 16 localStorage loadClients Funktion

Diese Funktion wird verwendet, um bestehende Kundendaten aus dem localStorage zu laden. Falls Kundendaten vorhanden sind, wird der JSON-String in ein Array mit Client-Objekten umgewandelt und zurückgegeben. Falls keine Kundendaten im localStorage existieren, wird ein leeres Array zurückgegeben.

saveClients

```
export const saveClients = (clients: Client[]) => {  
  localStorage.setItem(STORAGE_KEY, JSON.stringify(clients));  
};
```

Abbildung 17 localStorage saveClients Funktion

Mit dieser Funktion wird die übergebene Kundenliste im localStorage gespeichert. Dazu werden erstmal die übergebenen Client-Objekte mit JSON.stringify() in einen String umgewandelt und unter dem Schlüssel ClientData gespeichert.

deleteClientFromStorage

```
export const deleteClientFromStorage = (clientId: number): void => {  
  const clients = loadClients();  
  const updatedClients = clients.filter((client) => client.id !== clientId)  
  saveClients(updatedClients);  
};
```

Abbildung 18 localStorage deleteClientFromStorage Funktion

Diese Funktion löscht einen Kunden aus dem localStorage, basierend auf der übergebenen ID. Hier wird erstmal die aktuelle Kundenliste aus dem localStorage geladen und die Liste wird gefiltert nach allen Einträgen, die nicht diese ID haben, die aktualisierte Liste wird dann wieder in den localStorage gespeichert.

getNextClientId

```
export const getNextClientId = (): number => {  
  const idCounter = parseInt(localStorage.getItem("idCounter") || "1", 10);  
  localStorage.setItem("idCounter", (idCounter + 1).toString());  
  return idCounter;  
};
```

Abbildung 19 localStorage getNextClientId Funktion

Diese Funktion liest den aktuellen ID-Counter aus dem localStorage und erhöht ihn um eins. Falls kein ID-Counter vorhanden ist, wird einer mit dem Anfangswert 1 gestartet. Der neue Wert wird dann gespeichert und als number zurückgegeben und so wird sichergestellt, dass jeder Kunde eine eindeutige erhält.

updateClientInStorage

```
export const updateClientInStorage = (updatedClient: Client): void => {  
  const clients = loadClients();  
  const index = clients.findIndex((client) => client.id === updatedClient.i  
  if (index !== -1) {  
    clients[index] = updatedClient;  
    saveClients(clients);  
  }  
};
```

Abbildung 20 localStorage updateClientInStorage Funktion

Mit dieser Funktion kann man einen bestehenden Kunden im localStorage aktualisieren. Als aller erstes wird die aktuelle Kundenliste geladen und dann wird mit findIndex die Position des Kunden mit der passenden ID herausgefunden. Falls dieser existiert, wird der aktuelle Eintrag mit dem neuen überschrieben und zum Schluss wird die aktualisierte Liste wieder in den localStorage gespeichert

Bei der Weiterentwicklung wird hier eine richtige Datenbank verwendet.

(Der localStorage abschnitt wurde in Zusammenarbeit mit ChatGPT erstellt)

Rollen und Rechte umsetzen

Die Rollen und Rechte sind ein wichtiger Teil meiner Anwendung mit welchen bestimmten Funktionen und UI-Elemente gesteuert werden. Es gibt drei Rollen: Admin, Advisor und Client, diese haben je nach Rolle verschiedene Berechtigungen wie z.B das Löschen von Kunden ist als Admin erlaubt. Dazu habe ich in einer eigenen Datei (RolePermissions.ts) definiert, welche Rolle welche Rechte hat. Diese Rechte habe ich als boolean-Wert im interface deklariert und in verschiedenen UI-Elementen eingebunden, um diese ein- oder auszublenden.

RolePermissions.ts

```
export const ROLE_PERMISSIONS: Record<Role, Permissions> = {
  Admin: {
    canNavigateBack: true, // Kann zurück navigieren
    canEditProfile: true, // Kann Profil bearbeiten
    canEditTaxYears: true, // Kann Steuerjahre bearbeiten
    canDeleteTaxYears: true, // Kann Steuerjahre löschen
    canAddClient: true, // Kann Kunden hinzufügen
    canEditClient: true, // Kann Kunden bearbeiten
    canDeleteClient: true, // Kann Kunden löschen
  },
  Advisor: {
    canNavigateBack: true,
    canEditProfile: true,
    canEditTaxYears: true,
    canDeleteTaxYears: false, // Darf Steuerjahre nicht löschen
    canAddClient: true,
    canEditClient: true,
    canDeleteClient: false, // Darf Kunden nicht löschen
  },
  Client: {
    canNavigateBack: false, // Darf nicht zurück navigieren
    canEditProfile: true,
    canEditTaxYears: false, // Darf Steuerjahre nicht bearbeiten
    canDeleteTaxYears: false,
    canAddClient: false,
    canEditClient: false,
    canDeleteClient: false,
  },
};
```

Abbildung 21 Rollenrechte

RoleSwitcher

Um zwischen den Rollen nun Wechseln zu können habe ich dafür eine eigene Komponente erstellt, welche es möglich macht zwischen den drei Rollen (Admin, Advisor und Client) zu wechseln. Die ausgewählte Rolle wird in einem State gespeichert und über die Props (role, setRole) and andere Komponenten weitergegeben.

Der RoleSwitcher dient momentan als Übergangslösung, um die Rollen besser testen zu können und um das ganze besser präsentieren zu können, später bei der Weiterentwicklung des Projekts wird hier ein richtiges Login eingebaut.

```
const RoleSwitcher: React.FC<RoleSwitcherProps> = ({ role, setRole }) => {
  return (
    <div className="role-switcher-container">
      <label htmlFor="role-switcher">Switch Role:</label>
      <select
        id="role-switcher"
        value={role}
        onChange={(event) => setRole(event.target.value as Role)}
      >
        {ROLES.map(
          (
            roleOption
          ) => (
            <option key={roleOption} value={roleOption}>
              {roleOption}
            </option>
          )
        )}
      </select>
    </div>
  );
};
```

Abbildung 22 RoleSwitcher

Um das ganze nun bei UI-Elementen nutzen zu können brauche ich diesen code, der die Rolle abfragt.

```
{ROLE_PERMISSIONS[role].canDeleteClient ? (
```

Abbildung 23 Rollenrechte Abfrage im Code

(die Rollen und Rechte wurde in Zusammenarbeit mit ChatGPT erstellt.)

Kundendetails umsetzen

Die Kundendetails Seite ist die zweite Seite in meiner Anwendung und ist für alle drei Rollen (Admin, Advisor, Client) zugänglich. Auf diese Seite werden die vollständigen Informationen eines einzelnen Kunden angezeigt.

Die Seite besteht aus zwei Hauptbereichen:

- Im Linken Bereich werden die Kundendaten angezeigt und können von allen Rollen bearbeitet werden (der Client kann ausschliesslich nur seine eigenen Daten bearbeiten). Zusätzlich hat der Admin noch die Möglichkeit den Kunden von dieser Seite aus zu löschen.
- Im rechten Bereich der Seite befindet sich die TaxYears Komponente diese zeigt die Steuerjahre des Kunden an und je nach Rolle hat man auch die Möglichkeit diese zu bearbeiten oder zu löschen.

Im oberen Bereich noch ein Zurück-Button, der nur für Admin und Advisor sichtbar ist und zur Startseite führt. Ausserdem ist hier noch der RoleSwitcher integriert und das Wechseln der Rollen zu ermöglichen. Hier wurden ähnliche Funktionen wie auf der Kundenübersicht benutzt.

renderField

```
const renderField = (
  label: string,
  value: string | number,
  field: keyof Client
) => {
  return (
    <div className="field-display">
      <span className="field-label">{label}</span>
      {isEditing ? (
        <input
          className="form-input"
          type="text"
          value={value}
          onChange={(e) => {
            const newVal =
              typeof value === "number"
                ? Number(e.target.value)
                : e.target.value;
            handleEditCustomerField(field, newVal.toString());
          }}
        />
      ) : (
        <span className="field-value">{value}</span>
      )}
    </div>
  );
};
```

Abbildung 24 renderField Funktion

Die renderField Funktion wurde erstellt, um ein einzelnes Feld der Kundendaten dynamisch darzustellen.

Diese hat drei Parameter:

- label: Das ist der Sichtbare name des Feldes z. B. Vorname
- value: Das ist der aktuelle wert des Feldes, welcher entweder ein string oder eine number ist.
- field: Das ist der Key im Client Objekt also z. B. firstName oder E-Mail

Jetzt wird je nach Zustand (isEditing) das Feld entweder als einfacher Text oder ein bearbeitbares Input-Feld dargestellt. Wenn das Feld bearbeitet wird, wird das Input-Feld mit dem aktuellen wert von value dargestellt. Sobald man etwas in das Input-Feld eingibt, wird geprüft ob der ursprüngliche Wert eine number ist, falls das der Fall ist, wird es in eine Zahl umgewandelt, ansonsten wird es einfach als string übernommen. Anschliessend wird die Funktion handleEditCustomerField aufgerufen, um das entsprechende Feld in Client Objekt mit dem neuen Wert zu aktualisieren. Falls der Zustand isEditing false steht, dann wird der Wert einfach als normaler Text angezeigt.

(die renderField Funktion wurde in Zusammenarbeit mit ChatGPT erstellt.)

Kunde bearbeiten (Edit)

Der Edit Button ist dafür zuständig, den Benutzer in den Bearbeitungsmodus zu versetzen. Wenn dieser jetzt geklickt wird, werden die Textfelder mit Hilfe der renderField funktion als Input-Feld angezeigt und mit handleEditCustomerField wird die Bearbeitung der Felder verwaltet. Zusätzlich erscheinen noch neue Buttons, mit welchen man die Änderungen speichern oder die Bearbeitung abbrechen kann.

handleEditCustomerField

```
const handleEditCustomerField: HandleEditCustomerField = (
  field,
  newValue
) => {
  setClient((prev) => ({
    ...prev,
    [field]: typeof prev[field] === "number" ? Number(newValue) : newValue,
  }));
};
```

Abbildung 25 handleEditCustomerField Funktion

Die Funktion handleEditCustomerField ist dazu da, um ein einzelnes Feld im Client Objekt zu aktualisieren. Diese arbeitet mit der renderField Funktion zusammen und ist darauf aufgebaut. Sobald in renderField ein Feld bearbeitet wird, wird hier die handleEditCustomerField aufgerufen und sie erhält den Feldnamen sowie den neuen Wert und speichert die aktualisierte Kopie des Client Objekt im State.

handleSave

```
const handleSave = () => {  
  updateClientInStorage(client);  
  setClient({ ...client });  
  setIsEditing(false);  
  setFeedback("Changes saved successfully!");  
  setTimeout(() => setFeedback(""), 2000);  
};
```

Abbildung 26 handleSave Funktion

Diese Funktion wird aufgerufen, sobald der Save Button geklickt wurde, welcher sich im Bearbeitungsmodus der Kundendaten befindet. Als erstes wird die aktualisierte Kundenliste in localStorage mit updateClientInStorage gespeichert und mit setClient im State aktualisiert, um die aktuelle Liste anzuzeigen. Dann wird der Bearbeitungsmodus geschlossen und eine Feedback Nachricht für 2 Sekunden angezeigt, als letztes wird die Feedback Nachricht noch zurückgesetzt.

HandleCancelEdit

```
const handleCancelEdit = () => {  
  setClient(originalClient);  
  setIsEditing(false);  
};
```

Abbildung 27 handleCancelEdit Funktion

Diese Funktion wurde implementiert, um den Bearbeitungsmodus abubrechen. Wenn der Benutzer jetzt auf den Cancel Button im Bearbeitungsmodus klickt, wird der ursprüngliche State der Kundendaten(originalClient) wiederhergestellt. Dadurch werden alle Änderungen, welche bei der Bearbeitung vorgenommen wurden, verworfen. Der State originalClient wird beim Öffnen des Bearbeitungsmodus gesetzt. Als Letztes wird noch der Bearbeitungsmodus mit setIsEditing(false) geschlossen.

Kunde Löschen (Delete)

handleDelete

```
const handleDelete = () => {  
  if (role !== "Admin") {  
    alert("You do not have permission to delete clients.");  
    return;  
  }  
  if (window.confirm("Are you sure you want to delete this client?")) {  
    deleteClientFromStorage(client.id!);  
    setFeedback("Client deleted successfully!");  
    setDeleted(true);  
    setTimeout(() => {  
      navigate("/");  
    }, 2000);  
  }  
};
```

Abbildung 28 handleDelete Funktion

Die handleDelete Funktion wurde eigentlich genau wie bei der Startseite umgesetzt ausser das man hier noch zur Startseite weitergeleitet wird, wenn der Kunde gelöscht ist.

Steuerjahre Integrieren

Die Steuerjahre sind eine eigene Komponente, welche dafür zuständig ist, die Steuerjahre eines Kunden anzuzeigen und auch die Möglichkeit bietet diese zu verwalten. Jedes Steuerjahr hat Informationen wie z. B das Jahr, die Bundes- und die Staatssteuern. Admin und Advisor haben noch die Möglichkeit über ein Formular ein neues Steuerjahr hinzuzufügen, je nach Rolle kann man auch noch die Steuerjahre bearbeiten oder löschen. Zusätzlich werden noch einige Werte automatisch berechnet und dargestellt.

formatValue

```
const formatValue = (value: string | number | undefined): string => {  
  if (!value) return "CHF 0";  
  const numericValue = typeof value === "string" ? parseFloat(value) : value;  
  return `CHF ${numericValue.toLocaleString()}`;  
};
```

Abbildung 29 formatValue Funktion

Die formatValue Funktion ist dazu da um Zahlen als Schweizer Franken (CHF) zu formatieren und mit zwei Dezimalstellen darzustellen.

(die formatValue Funktion wurde in Zusammenarbeit mit ChatGPT erstellt.)

updateClient

```
const updateClient = (updatedTaxYears: TaxYear[]) => {  
  const updatedClient = { ...client, taxYears: updatedTaxYears };  
  setClient(updatedClient);  
  updateClientInStorage(updatedClient);  
};
```

Abbildung 30 updateClient Funktion

Die Funktion updateClient befindet sich zwar auf der Kundendetails Seite wird aber vom Steuerjahr teil benutzt, diese wird über Props and die Steuerjahr Komponente übergeben. Sie wird verwendet, um die geänderten Steuerjahre im Kunden Objekt zu speichern dabei wird der State aktualisiert, um alles korrekt anzuzeigen und die Daten im localStorage gespeichert.

Steuerjahr hinzufügen (Add New Tax Year)

Das hinzufügen eines neuen Steuerjahres erfolgt über eine separate Komponente ähnlich gelöst wie beim Hinzufügen eines neuen Kunden und funktioniert wie folgt: Wenn der Benutzer den Button zum Hinzufügen eines Steuerjahres klickt, erscheint ein Neues Fenster, in welchem man die Daten des Steuerjahres eintragen kann, aber nur diese welche nicht automatisch berechnet werden. Bei der Eingabe wird sichergestellt, dass bei number keine Zeichen wie: E, Plus, Minus und Punkt eingegeben werden können. Wenn man jetzt das Formular absendet, wird der State aktualisiert, um die Steuerjahre korrekt anzuzeigen und im localStorage gespeichert, anschliessend wird noch eine Feedback Nachricht angezeigt das das ganze erfolgreich war.

handleChange

```
const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {  
  const { name, value } = e.target;  
  
  setTaxYear((prev) => ({  
    ...prev,  
    [name]: value ? Number(value) : 0,  
  }));  
};
```

Abbildung 31 handleChange Funktion

Die handleChange Funktion ist dafür da, um den taxYear State bei jeder Eingabe zu aktualisieren und sie wandelt den eingegebenen Wert in eine Zahl um.

handleSubmit

```
const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault();

  if (
    !taxYear.year ||
    !taxYear.stateTax ||
    !taxYear.federalTax ||
    !taxYear.withholdingTaxPaid ||
    !taxYear.savings
  ) {
    alert("Please fill in all fields.");
    return;
  }

  if (taxYear.year < 2020 || taxYear.year > new Date().getFullYear() + 1) {
    alert("Please enter a valid year (between 2020 and next year).");
    return;
  }

  if (
    taxYear.stateTax < 0 ||
    taxYear.federalTax < 0 ||
    taxYear.withholdingTaxPaid < 0 ||
    taxYear.savings < 0
  ) {
    alert("All tax values must be positive numbers.");
    return;
  }

  const updatedTaxYear = {
    ...taxYear,
    totalTax: taxYear.stateTax + taxYear.federalTax,
    difference:
      taxYear.stateTax + taxYear.federalTax - taxYear.withholdingTaxPaid,
  };

  onTaxYearAdded(updatedTaxYear);
  setFeedback("Tax year added successfully!");
  setTimeout(() => {
    onClose();
    setFeedback("");
  }, 2000);
};
```

Abbildung 32 handleSubmit Funktion

Die `handleSubmit` Funktion ist dafür da, ein neues Objekt mit den eingegebenen Steuerdaten zu erstellen. Hier findet auch die Validierung der Eingabe statt, bevor man das Formular absendet, wie z. B. das alle Felder einen Wert haben müssen und das Jahr nicht unter dem Jahr 2020 und dem nächsten Jahr sein darf, zusätzlich wird hier `totalTax` und `difference` berechnet. Wenn alle Bedingungen erfüllt sind, wird das Objekt an `onTaxYearAdded` weitergegeben, um das Aktualisieren des States und das Speichern im `localStorage` kümmert sich anschliessend `updateClient`. Als letztes wird noch eine Feedback Nachricht zurückgegeben und das Fenster wird nach zwei Sekunden geschlossen.

Steuerjahr bearbeiten (Edit)

Der Edit Button der Steuerjahre ist dafür zuständig, den Benutzer in den Bearbeitungsmodus zu bringen. Sobald dieser geklickt wird, werden die Steuerdaten als Input-Field angezeigt und man hat die Möglichkeit diese zu bearbeiten. Die Bearbeitung der Felder wird mit der `handleEditTaxYearField` Funktion verwaltet, zusätzlich wenn z. B. `stateTax` oder `federalTax` bearbeitet wird, wird `totalTax` automatisch gerechnet. Das gleiche passiert auch bei `difference`. Wenn man im Bearbeitungsmodus ist, erscheinen zusätzliche Buttons zum Speichern der Änderungen oder um die Bearbeitung abzubrechen.

handleEditTaxYearField

```
const handleEditTaxYearField: HandleEditTaxYearField = (
  year,
  field,
  newValue
) => {
  setUpdatedTaxYears((prev) => {
    prev.map((taxYear, i) => {
      if (i !== year) return taxYear;

      const updatedTaxYear = { ...taxYear, [field]: newValue };

      if (field === "stateTax" || field === "federalTax") {
        updatedTaxYear.totalTax =
          updatedTaxYear.stateTax + updatedTaxYear.federalTax;
      }

      updatedTaxYear.difference =
        updatedTaxYear.totalTax - updatedTaxYear.withholdingTaxPaid;

      return updatedTaxYear;
    })
  });
};
```

Abbildung 33 *handleEditTaxYearField* Funktion

Die Funktion `handleEditTaxYearField` ist dafür da, ein einzelnes Feld der eines Steuerjahres zu aktualisieren. Sobald ein Feld bearbeitet wird, erhält die Funktion den Feldnamen und auch den Neuen Wert, die aktualisierte Kopie davon wird dann im State aktualisiert. Wenn `stateTax` oder `federalTax` bearbeitet wird, wird zusätzlich `totalTax` automatisch berechnet, die automatische Berechnung passiert auch bei `difference`.

handleSave

```
const handleSave = () => {  
  updateClient(updatedTaxYears);  
  setEditingIndex(null);  
  setFeedback("Changes saved successfully!");  
  setTimeout(() => setFeedback(""), 2000);  
};
```

Abbildung 34 *handleSave* Funktion

Die `handleSave` Funktion ist dafür zuständig die bearbeiteten Steuerjahre zu speichern. Sie wird aufgerufen, sobald der Save Button gedrückt wurde und gibt das aktualisierte Steuerjahr an `updateClient` weiter, beendet den Bearbeitungsmodus und zeigt den Benutzer eine Feedback Nachricht, welche nach zwei Sekunden verschwindet.

handleCancel

```
const handleCancel = () => {  
  if (editingIndex !== null && originalTaxYear) {  
    setUpdatedTaxYears((prev) => {  
      const newList = [...prev];  
      newList[editingIndex] = originalTaxYear;  
      return newList;  
    });  
  }  
  setEditingIndex(null);  
  setOriginalTaxYear(null);  
};
```

Abbildung 35 *handleCancel* Funktion

Die `handleCancel` Funktion ist dafür zuständig den Bearbeitungsmodus abubrechen. Sie ersetzt zusätzlich noch das bearbeitete Steuerjahr in den ursprünglichen State aus `originalTaxYear` zurück.

Steuerjahr löschen (Delete)

handleDelete

```
const handleDelete = (year: number) => {  
  if (  
    window.confirm(`Are you sure you want to delete the tax year ${year}?`)  
  ) {  
    setEditingIndex(null);  
    const updatedList = updatedTaxYears.filter((ty) => ty.year !== year);  
    setUpdatedTaxYears(updatedList);  
    updateClient(updatedList);  
    setFeedback(`Tax year ${year} deleted successfully!`);  
    setTimeout(() => setFeedback(""), 2000);  
  }  
};
```

Abbildung 36 handleDelete Funktion

Wenn man auf den Delete Button der Steuerjahre klickt, wird die handleDelete Funktion ausgelöst. Diese fragt erstmal nach einer Bestätigung zum Löschen des Steuerjahres, falls dies der Fall ist, wird der Bearbeitungsmodus geschlossen. Die Steuerjahrliste wird dann so gefiltert, dass nur noch die Steuerjahre dabei sind, welche nicht dem gelöschten Jahr entsprechen. Die neue Liste wird dann im State gespeichert und eine Feedback Nachricht erscheint, welche nach 2 Sekunden verschwindet.

Feedback und Bestätigung

In diesem Teil wurden Feedback und Bestätigung als auch Validierung erstellt.

```
const validateEmail = (email: string): boolean => {  
  const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;  
  return regex.test(email);  
};  
  
const validatePhone = (phone: string): boolean => {  
  const regex = /^\\+?[0-9\\s-]+$/;  
  return regex.test(phone);  
};  
  
const validatePostalCode = (postalCode: number): boolean => {  
  return postalCode > 0 && postalCode.toString().length <= 10;  
};
```

Abbildung 37 Beispiel Validation

(Der Regex wurde von ChatGPT generiert)

```
if (  
  !client.salutation ||  
  !client.lastName ||  
  !client.firstName ||  
  !client.email ||  
  !client.phone ||  
  !client.address ||  
  !client.postalCode ||  
  !client.city  
) {  
  alert("Please fill in all fields.");  
  return;  
}
```

Abbildung 38 Beispiel Validation 2

Die Validierung wurde wie auf der oberen Abbildung zu sehen, durch regex oder if abfragen gelöst und an den benötigten Stellen integriert.

Zur Bestätigung habe ich ein `window.confirm` benutzt welches bei wichtigen Teilen der Anwendung wie z. B. dem Löschen eines Kunden aufpoppt.

```
if (window.confirm("Are you sure you want to delete this client?")) {
```

Abbildung 39 Beispiel Bestätigung

Bei Funktionen wie beim Hinzufügen, bearbeiten und löschen eines Kunden oder eines Steuerjahres habe ich eine Feedback Nachricht hinzugefügt, welche zur visuellen Bestätigung dienen soll, dass die Aktion durchgeführt wurde.

```
setFeedback("Client deleted successfully!");  
setTimeout(() => setFeedback(""), 2000);
```

Abbildung 40 Beispiel Feedback

Styling (CSS)

In diesem Teil habe ich für meine beiden Seiten der Anwendung CSS geschrieben, diese habe ich in Drei Dateien aufgeteilt App.css für allgemeines, HomePage.css für die Startseite und ClientDetailsPage.css für die Kundendetailseite, da es hier etwas langweilig wäre den CSS zu beschreiben zeige ich einfach einen vorher nachher vergleich.

HomePage vorher:

Switch Role: Advisor

Client List


Add Client

Add Client

Salutation	Last Name	First Name	Email	Phone	Address	Postal Code	City
Add	Cancel						
ID	Last Name	First Name	Email	Phone	Address	Tax Years	Actions
1	Bauer	Michael12412	michael.bauer@example.com	0123456789	Hauptstraße 12	2020, 2021, 2023, 2024	
2	Meier	Laura	laura.meier@example.com	0987654321	Schulstraße 8	2019, 2021	
3	Schneider	Sophia	sophia.schneider@example.com	0176543210	Bahnhofstraße 15	2022, 3124	
6	qwe	qwr	1@2.ch	1231	124		
7	fsaf	fasf	1@2.ch	1231	124	2022	

Abbildung 41 Startseite vorher

HomePage nachher:



Switch Role: Admin

Client List

Add Client

ID	LAST NAME	FIRST NAME	EMAIL	PHONE	ADDRESS	TAX YEARS	ACTIONS
1	Bauer	Michael12412	michael.bauer@example.com	0123456789	Hauptstraße 12	2020, 2021, 2023, 2024	
2	Meier	Laura	laura.meier@example.com	0987654321	Schulstraße 8	2019, 2021	
3	Schneider	Sophia	sophia.schneider@example.com	0176543210	Bahnhofstraße 15	2022, 3124	
6	qwe	qwr	1@2.ch	1231	124		
7	fsaf	fasf	1@2.ch	1231	124	2022	
8	fwefwfwfwef	dwedwfwef	vladi.ajooda@outlook.com	12312412512	124		

Abbildung 42 Startseite nachher

ClientDetailsPage vorher:

Back

Switch Role: Advisor

Client Details

Salutation:Mr.
Last Name:Bauer
First Name:Michael12412
Email:michael.bauer@example.com
Phone Number:0123456789
Address:Hauptstraße 12
Postal Code:8001
City:Zurich213

Edit

Tax Years

Add Tax Year

Year	State Tax	Federal Tax	Total Tax	Withholding Tax Paid	Difference	Potential savings	Actions
2020	CHF 2,800	CHF 15,000	CHF 17,800	CHF 17,000	CHF 800 remaining	CHF 500	Edit
2021	CHF 3,000	CHF 16,000	CHF 19,000	CHF 18,500	CHF 500 remaining	CHF 400	Edit
2023	CHF 3,200	CHF 17,000	CHF 20,200	CHF 19,500	CHF 700 remaining	CHF 600	Edit
2024	CHF 124	CHF 124	CHF 248	CHF 124,124	CHF 123,876 refund	CHF 124	Edit

Abbildung 43 Kundendetailsseite vorher

ClientDetailsPage nachher:

Back

Switch Role: Admin

Client Details

Salutation:Mr.
Last Name:Bauer
First Name:Michael12412
Email:michael.bauer@example.com
Phone Number:0123456789
Address:Hauptstraße 12
Postal Code:8001
City:Zurich213

Add Tax Year

YEAR	STATE TAX	FEDERAL TAX	TOTAL TAX	WITHHOLDING TAX PAID	DIFFERENCE	POTENTIAL SAVINGS	ACTIONS
2020	CHF 2,800	CHF 15,000	CHF 17,800	CHF 17,000	CHF 800 remaining	CHF 500	<div>Tax returnProvisional calculationShow commentsShow savings</div>
2021	CHF 3,000	CHF 16,000	CHF 19,000	CHF 18,500	CHF 500 remaining	CHF 400	<div>Tax returnProvisional calculationShow commentsShow savings</div>
2023	CHF 3,200	CHF 17,000	CHF 20,200	CHF 19,500	CHF 700 remaining	CHF 600	<div>Tax returnProvisional calculationShow commentsShow savings</div>
2024	CHF 124	CHF 124	CHF 248	CHF 124,124	CHF 123,876 refund	CHF 124	<div>Tax returnProvisional calculationShow commentsShow savings</div>

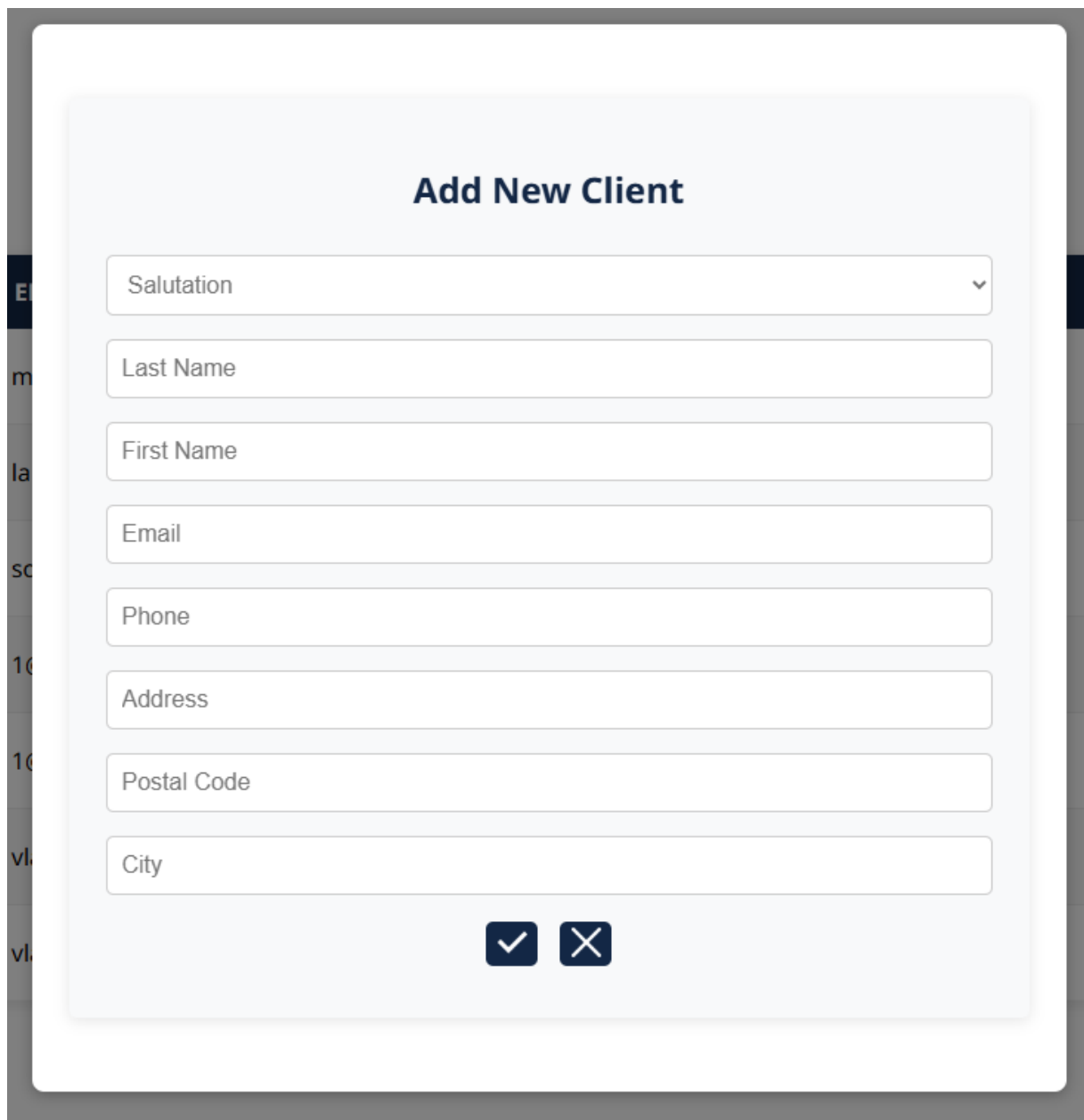
Abbildung 44 Kundendetailsseite nachher

Zusatzfunktionen und Features

In Diesem Abschnitt beschreibe ich noch kurz zusätzliche Funktionen und Features, die ich eingebaut habe.

Modal

Damit die Formulare auf meinen Seiten beim Öffnen und Schliessen nicht immer das Layout der Seite verschieben habe ich mich entschieden diese in einem Modal Fenster wiederzugeben, das ganze sieht dann so aus:



Add New Client

Salutation ▼

Last Name

First Name

Email

Phone

Address

Postal Code

City

✓ ✕

Abbildung 45 Modal Fenster

(Das Model Fenster wurde mit Hilfe von ChatGPT erstellt.)

Zu- und Aufklappen der Kundendetails

Damit auf der Kundendetailseite die Möglichkeit besteht die Kundendetail zu- und aufzuklappen habe ich die Kundendetails und die Steuerjahre in je ein column gepackt und noch ein zusätzlich ein drittes column erstellt welches je nach dem State von collapsed den Column mit den Kundendetail einblendet oder den Column mit dem Button, um das ganze wieder zu wechseln.

toggleCollapse

```
const toggleCollapse = () => {  
  |   setCollapsed((prev) => !prev);  
};
```

Abbildung 46 toggleCollapse Funktion

Die toggleCollapse Funktion wechselt beim Klick vom Collapse/Expand Button den State von Collapsed zwischen true oder false.

Schnelles Löschen in der Suchleiste

Um die Eingabe bei der Suchleiste schnell löschen zu können habe ich eine kleine Funktion erstellt, welche den State von searchTerm löscht.

handleClearSearch

```
const handleClearSearch = () => {  
  |   setSearchTerm("");  
};
```

Abbildung 47 handleClearSearch Funktion

Die handleClearSearch Funktion wird beim Klick auf einen Button getriggert, dieser Button wurde innerhalb der Suchleiste platziert.

Kontrollieren

Das Kontrollieren ist die fünfte Phase der IPERKA-Methode

In dieser Phase habe ich durch automatisierte Integrationstests mit dem Vitest Framework sichergestellt, dass die wichtigsten Funktionen meiner Anwendung wie erwartet funktionieren. Falls Fehler auftreten, werden diese sofort behoben. Die Tests befinden sich in einer Testdatei innerhalb meiner Projektverzeichnisses und wurden jeweils einzeln getestet.

(Der Testcode wurde von ChatGPT generiert.)

Testprotokoll

Testfall	T-1 INT – Kunde hinzufügen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird getestet, ob ein Kunde nach dem Ausfüllen des Formulars korrekt in die Kundenliste übernommen wird.
Testdaten (Eingabe)	Salutation: Mr. Vorname: Max Nachname: Muster E-Mail: max@test.ch Telefon: +41791234567 Adresse: Musterstrasse 1 PLZ: 8000 Stadt: Zürich
Testschritte	1. Auf „Add Client“ klicken 2. Formular ganz ausfüllen 3. Auf Bestätigen klicken
Erwartetes Ergebnis	Die Kundendaten werden in die Liste hinzugefügt.
Tatsächliches Ergebnis	Der Kunde wurde wie erwartet in der Liste angezeigt
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 1007ms
✓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden 1005ms
↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 48 Test1 Erfolgreich

Testfall	T-2 INT – Kunde löschen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Beim Klicken des Lösch-Buttons wird der Kunde aus der Liste gelöscht.
Testdaten (Eingabe)	Bereits vorhandener Kunde: z. B. Nachname: Schneider (MockData)
Testschritte	1. Startseite öffnen (Rolle: Admin) 2. Lösch-Button klicken 3. Prüfen, ob der Kunde aus der Liste verschwindet
Erwartetes Ergebnis	Der Kunde wird aus der Kundenliste entfernt
Tatsächliches Ergebnis	Der Kunde (Schneider) ist nicht mehr sichtbar.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 182ms
  ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ✓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt 181ms
  ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 49 Test2 Erfolgreich

Testfall	T-3 INT – Kundendaten bearbeiten
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob geänderte Kundendaten korrekt gespeichert und angezeigt werden.
Testdaten (Eingabe)	Kunde mit ID 3 aus MockData: Vorname „Sophia“ wird zu „Sophie“ geändert.
Testschritte	1. Detailseite von Kunde 3 öffnen (initialEntries /details/3) 2. Auf „Bearbeiten“ klicken 3. Vorname zu „Sophie“ ändern 4. Speichern (Checkmark klicken) 5. Prüfen, ob „Sophie“ angezeigt wird
Erwartetes Ergebnis	Der neue Vorname „Sophie“ wird korrekt gespeichert und angezeigt.
Tatsächliches Ergebnis	Der neue Vorname „Sophie“ wurde korrekt gespeichert und angezeigt.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 568ms
  ↳ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↳ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ✓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt 566ms
  ↳ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↳ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↳ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↳ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↳ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↳ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↳ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↳ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↳ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 50 Test3 Erfolgreich

Testfall	T-4 INT – Steuerjahr löschen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob ein Steuerjahr nach dem Klick auf den Lösch-Button korrekt entfernt wird und nicht mehr angezeigt wird.
Testdaten (Eingabe)	Kunde mit ID 3 aus den MockData, Steuerjahr 2022 soll gelöscht werden
Testschritte	<ol style="list-style-type: none"> 1. Detailseite von Kunde 3 öffnen (initialEntries /details/3) 2. Auf „Steuerjahr löschen“ neben dem Jahr 2022 klicken 3. Prüfen, ob Jahr 2022 nicht mehr in der Liste erscheint
Erwartetes Ergebnis	Das Steuerjahr 2022 wird nicht mehr angezeigt
Tatsächliches Ergebnis	Das Steuerjahr 2022 wurde erfolgreich gelöscht und war nicht mehr sichtbar.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 317ms
  ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ✓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt 316ms
  ↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 51 Test4 Erfolgreich

Testfall	T-5 INT – Steuerjahr bearbeiten
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob die Bearbeitung eines bestehenden Steuerjahres korrekt gespeichert wird und anschließend korrekt angezeigt wird.
Testdaten (Eingabe)	Kunde mit ID 3 aus den MockData Steuerjahr 2022 soll bearbeitet werden Neuer StateTax Wert: 3500 CHF
Testschritte	<ol style="list-style-type: none"> 1. Detailseite von Kunde 3 öffnen (initialEntries=['/details/3']) 2. Auf „Steuerjahr bearbeiten“ neben dem Jahr 2022 klicken 3. StateTax-Wert auf 3500 ändern 4. Auf „Speichern“ (Checkmark-Button) klicken 5. Prüfen, ob der neue Wert 3500 für StateTax angezeigt wird
Erwartetes Ergebnis	Der neue Steuerwert wird korrekt gespeichert und angezeigt.
Tatsächliches Ergebnis	Der neue Wert 9999 CHF wurde erfolgreich gespeichert und korrekt angezeigt.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 443ms
  ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ✓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt 441ms
  ↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 52 Test5 Erfolgreich

Testfall	T-6 INT – Steuerjahr hinzufügen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Teststart	Integrationstest
Beschreibung	Es wird geprüft, ob ein neues Steuerjahr korrekt über das Formular hinzugefügt, gespeichert und in der Liste angezeigt wird.
Testdaten (Eingabe)	Kunde mit ID 3 aus den MockData. Folgendes Steuerjahr soll hinzugefügt werden: <ul style="list-style-type: none"> • Jahr: 2024 • State Tax: 3500 • Federal Tax: 18000 • Withholding Tax Paid: 20000 • Savings: 400
Testschritte	<ol style="list-style-type: none"> 1. Detailseite von Kunde 3 öffnen (initialEntries /details/3) 2. Auf „Steuerjahr hinzufügen“ klicken 3. Formular vollständig ausfüllen 4. Auf Speichern (Checkmark) klicken 5. Prüfen, ob Jahr 2024 in der Liste erscheint
Erwartetes Ergebnis	Das neue Steuerjahr 2024 wird korrekt in der Liste angezeigt.
Tatsächliches Ergebnis	Das Steuerjahr 2024 wurde erfolgreich hinzugefügt und angezeigt.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.
<pre> ✓ src/components/pages/INTTest.test.tsx (12 tests 11 skipped) 948ms ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden ↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt ↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt ↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt ✓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt 947ms ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden ↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt ↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr ↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt ↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar Test Files 1 passed (1) Tests 1 passed 11 skipped (12) </pre>	

Abbildung 53 Test6 Erfolgreich

Testfall	T-7 INT – Kunde hinzufügen und in Details aufrufen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob ein neu hinzugefügter Kunde korrekt gespeichert wird und durch Klicken in der Kundenliste aufgerufen werden kann, sodass die Detailseite angezeigt wird.
Testdaten (Eingabe)	Salutation: Mr. Vorname: Testmann Nachname: Testo E-Mail: testo@example.com Telefon: +41791234567 Adresse: Testgasse 1 PLZ: 8000 Stadt: Zürich
Testschritte	1. Startseite öffnen 2. Auf „Add Client“ klicken 3. Formular mit Kundendaten ausfüllen 4. Formular bestätigen (Checkmark klicken) 5. In der Tabelle auf „Müller“ klicken 6. Prüfen, ob auf der Detailseite „Testo“ angezeigt wird
Erwartetes Ergebnis	Der neue Kunde wird korrekt hinzugefügt und beim Klick auf den Eintrag wird die Detailseite mit den Kundendaten angezeigt.
Tatsächliches Ergebnis	Die Kundendaten wurden erfolgreich übernommen. Der Name „Testo“ war auf der Detailseite sichtbar.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 1149ms
  ↳ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↳ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↳ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↳ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↳ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↳ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ✓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet 1148ms
  ↳ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↳ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↳ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↳ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↳ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files  1 passed (1)
Tests       1 passed | 11 skipped (12)

```

Abbildung 54 Test7 Erfolgreich

Testfall	T-8 INT – Kunde bearbeiten und Suchfunktion nutzen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob ein bestehender Kunde bearbeitet und anschließend über die Suchfunktion gefunden werden kann.
Testdaten (Eingabe)	Vorhandener Kunde aus den MockData: Vorname: Sophia Bearbeitet zu: Vorname: SofieSuchtest
Testschritte	<ol style="list-style-type: none"> 1. Startseite öffnen 2. Auf Eintrag „Sophia“ klicken 3. Bearbeiten-Button klicken 4. Vorname ändern zu „SofieSuchtest“ 5. Speichern (Checkmark klicken) 6. Zurück zur Startseite 7. In der Suchleiste „SofieSuchtest“ eingeben 8. Prüfen, ob der Kunde erscheint
Erwartetes Ergebnis	Der geänderte Kunde mit dem neuen Namen SofieSuchtest wird über die Suchfunktion korrekt gefunden.
Tatsächliches Ergebnis	Kunde wurde erfolgreich bearbeitet. Der Name SofieSuchtest war nach der Suche sichtbar.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Funktion arbeitet korrekt, keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 982ms
  ↳ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↳ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↳ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↳ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↳ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↳ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↳ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ✓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden 980ms
  ↳ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↳ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↳ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↳ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 55 Test8 Erfolgreich

Testfall	T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob der Löschbutton abhängig von der Benutzerrolle korrekt angezeigt oder verborgen wird. Admins sollen ihn sehen können, Advisors nicht.
Testdaten (Eingabe)	Vorhandener Kunde: Nachname „Schneider“ (MockData) Getestete Rollen: Admin & Advisor
Testschritte	1. Startseite mit Rolle Admin öffnen 2. Kunden „Schneider“ in der Tabelle finden 3. Prüfen, ob der Löschbutton sichtbar ist 4. Startseite mit Rolle Advisor neu laden 5. Prüfen, ob kein Löschbutton mehr sichtbar ist
Erwartetes Ergebnis	Bei Rolle Admin ist der Button sichtbar, bei Rolle Advisor nicht.
Tatsächliches Ergebnis	Verhalten wie erwartet: Der Button war nur bei Admin sichtbar, bei Advisor nicht.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Rollenabhängige Rechte funktionieren korrekt. Keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 56ms
  ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ✓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt 55ms
  ↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 56 Test9 Erfolgreich

Testfall	T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob ein neuer Kunde korrekt hinzugefügt und anschließend ein Steuerjahr für diesen Kunden gespeichert und angezeigt werden kann.
Testdaten (Eingabe)	Kunde: Salutation: Mr. Vorname: Klaus Nachname: Steuerlich E-Mail: klaus@steuer.ch Telefon: 0777777777 Adresse: Teststrasse PLZ: 9999 Ort: Zug Steuerjahr: Jahr: 2025 State Tax: 4000 Federal Tax: 12000 Withholding Tax Paid: 15000 Potential Savings: 300
Testschritte	<ol style="list-style-type: none"> 1. Auf „Add Client“ klicken 2. Kundenformular ausfüllen und bestätigen (Checkmark) 3. In der Tabelle den neuen Kunden „Steuerlich“ anklicken 4. Auf „Add Tax Year“ klicken 5. Steuerjahr-Formular ausfüllen 6. Formular abschicken (Checkmark) 7. Prüfen, ob das Jahr 2025 erscheint
Erwartetes Ergebnis	Der Kunde Steuerlich wird hinzugefügt und das Steuerjahr 2025 korrekt gespeichert und angezeigt.
Tatsächliches Ergebnis	Der Kunde wurde korrekt hinzugefügt, das Steuerjahr 2025 erschien wie erwartet in der Liste.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Rollenabhängige Rechte funktionieren korrekt. Keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 1938ms
  ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ✓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr 1937ms
  ↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 57 Test10 Erfolgreich

Testfall	T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob ein bearbeitetes Steuerjahr korrekt gespeichert und anschließend auf der Startseite beim entsprechenden Kunden angezeigt wird.
Testdaten (Eingabe)	Kunde: Sophia (aus den MockData) Bearbeitung: Steuerjahr 2022 → geändert auf 2026
Testschritte	<ol style="list-style-type: none"> 1. Startseite öffnen 2. Auf „Sophia“ klicken, um zur Detailseite zu gelangen 3. Steuerjahr 2022 bearbeiten 4. Jahr auf 2026 ändern 5. Speichern (Checkmark klicken) 6. Zurück zur Startseite navigieren 7. Prüfen, ob in Sophias Zeile 2026 angezeigt wird
Erwartetes Ergebnis	Das bearbeitete Steuerjahr (2026) wird auf der Startseite korrekt in der Zeile des Kunden Sophia angezeigt.
Tatsächliches Ergebnis	Das Steuerjahr wurde erfolgreich bearbeitet. Auf der Startseite stand in der Zeile von Sophia das Jahr 2026.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Rollenabhängige Rechte funktionieren korrekt. Keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 532ms
  ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ✓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt 531ms
  ↓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar

Test Files 1 passed (1)
Tests 1 passed | 11 skipped (12)

```

Abbildung 58 Test11 Erfolgreich

Testfall	T-12 INT – Kunde löschen und erneuten Zugriff prüfen
Testzeitpunkt	Projekttag 9, 11.04.2025
Testperson	Vladyslav Astashyn
Testart	Integrationstest
Beschreibung	Es wird geprüft, ob ein gelöschter Kunde nach dem Löschen nicht mehr über einen Direktlink aufrufbar ist.
Testdaten (Eingabe)	Vorhandener Kunde aus den MockData: Nachname: Schneider
Testschritte	<ol style="list-style-type: none"> 1. Startseite öffnen 2. Kunde „Schneider“ lokalisieren 3. Kundennummer (ID) merken 4. Kunde löschen (Bestätigung „OK“) 5. Direktlink zur Detailseite mit details/:id aufrufen 6. Prüfen, ob die Detailseite nicht geladen wird
Erwartetes Ergebnis	Der gelöschte Kunde ist nicht mehr sichtbar und die Detailseite wird nicht mehr geladen.
Tatsächliches Ergebnis	Der Kunde „Schneider“ wurde erfolgreich gelöscht. Beim direkten Aufruf der Detailseite erschien kein Inhalt mehr.
Mängelstufe	0 – mängelfrei
Fazit / Empfehlung	Test erfolgreich. Rollenabhängige Rechte funktionieren korrekt. Keine Anpassung nötig.

```

✓ src/components/pages/INTTest.test.tsx (12 tests | 11 skipped) 171ms
  ↓ T-1 INT – Kunde hinzufügen > Nach dem Ausfüllen des Formulars soll der Kunde hinzugefügt werden
  ↓ T-2 INT – Kunde löschen > Beim Klicken des Lösch-Buttons wird der Kunde entfernt
  ↓ T-3 INT – Kundendaten bearbeiten > Änderung der Kundendaten wird korrekt gespeichert und angezeigt
  ↓ T-4 INT – Steuerjahr löschen > Ein Steuerjahr wird korrekt aus der Liste entfernt
  ↓ T-5 INT – Steuerjahr bearbeiten > Geänderte Steuerjahr-Daten werden korrekt gespeichert und angezeigt
  ↓ T-6 INT – Steuerjahr hinzufügen > Nach dem Ausfüllen des Formulars wird das Steuerjahr korrekt hinzugefügt
  ↓ T-7 INT – Kunde hinzufügen und in Details aufrufen > Ein neuer Kunde wird korrekt zur Detailseite weitergeleitet
  ↓ T-8 INT – Kunde bearbeiten und Suchfunktion nutzen > Ein Kunde wird bearbeitet und über die Suchleiste gefunden
  ↓ T-9 INT – Rolle beeinflusst Benutzeroberfläche und Rechte > Der Löschbutton wird je nach Rolle korrekt angezeigt oder versteckt
  ↓ T-10 INT – Kunde hinzufügen und Steuerjahr hinzufügen > Ein Kunde wird hinzugefügt und danach ein Steuerjahr
  ↓ T-11 INT – Steuerjahr bearbeiten und korrekt auf Startseite anzeigen > Bearbeitetes Steuerjahr wird korrekt auf der Startseite angezeigt
  ✓ T-12 INT – Kunde löschen und erneuten Zugriff prüfen > Gelöschter Kunde ist über Direktlink nicht mehr aufrufbar 170ms

Test Files  1 passed (1)
Tests       1 passed | 11 skipped (12)

```

Abbildung 59 Test12 Erfolgreich

Testbericht

Die Wichtigsten Funktionen der Anwendung wurde in einer Lokalen Testumgebung mit Mockdaten getestet. Alle Testfälle wurden ausgeführt und jeder Test hat mängelfrei bestanden. Die Anwendung verhält sich wie erwartet und die Funktionen funktionieren wie geplant. Es sind keine Abweichungen aufgetreten, somit sind keine Änderungen nötig

Auswerten

Das Auswerten ist die Letzte Phase von IPERKA.

In dieser Phase habe ich mein Projekt reflektiert und geprüft, ob ich alle Ziele gemäss der Aufgabenstellung erreicht habe und eine Reflexion geschrieben. Zusätzlich habe ich noch eine kurze Bedienungsanleitung erstellt.

Reflexion

Ich hatte einen Holprigen Start in meine IPA, jedoch habe ich schnell den Faden gefunden und konnte dank meines Zeitplans strukturiert vorgehen, dabei hat die Projektmanagement Methode IPERKA sehr geholfen.

Das Informieren lief gut, die Aufgabenstellung war klar, nur bei einzelnen Punkten musste ich mich an den Fachvorgesetzten wenden.

Das Planen ein nicht so Starker Punkt in meiner IPA, ich hatte zwar einen klaren Zeitplan erstellt musste diesen jedoch im Verlauf der Arbeit anpassen oder auch gewisse Arbeiten den anderen vorziehen. Hier habe ich auch die wichtige Entscheidung getroffen mehr Zeit in den Zeitplan einzubauen ohne welche es sonst sehr knapp geworden wäre.

Bei der Realisierungsphase bin ich gut vorwärtsgekommen, wenn ich nicht weiterwusste, habe ich im Internet nach einer Lösung gesucht. Ich konnte mich auch gut an die Änderungen im Plan anpassen und habe diese Phase gut gemeistert. Was mir hier Schwierigkeiten bereitete war das Dokumentieren meiner Arbeit da ich hier sich ständig Sachen ändern musste.

Das Kontrollieren verlief auch sehr gut, ich hatte zu Beginn des Testings Schwierigkeiten, da ich noch nie Tests in diesem Stil gemacht habe, jedoch verlief das Testing selbst ausgezeichnet und ich war sehr zufrieden als alle Tests keinen einzigen Fehler zurückgaben.

Während der Umsetzung meiner IPA habe ich gelegentlich AI-Hilfsmittel wie ChatGPT benutzt, um bei bestimmten Problemen mit Code oder beim Testen Unterstützung zu erhalten

Was ich unterschätzt habe, ist die Dokumentation, da ich hier nicht immer die Passenden Worte gefunden habe und vieles nachträglich korrigieren musste, hat mich das viel Zeit gekostet. Dank der Extrazeit hat das Ganze dann doch einigermaßen gut funktioniert.

Was ich mir fürs nächste Projekt mitnehme, ist, dass ich mir mehr Zeit für die Dokumentation nehme, werde und die Planung des Projekts besser durchdenke.

Das ist meine erste richtige grosse Arbeit in diesem Stil gewesen, welche ich innerhalb von zehn Tagen eigenständig planen realisieren und dokumentieren musste. Insgesamt bin ich sehr zufrieden mit dem Endresultat und dem gesamten Ablauf meines Projekts.

Danksagung

Ich bedanke mich hiermit herzlich bei Pantos Dejan für das Durchlesen meiner Dokumentation und das anschliessende Feedback, das ich bekam, natürlich bedanke ich mich auch bei Meinem Fachvorgesetzten Veselin Lakic der mir zur Seite stand und sich die Zeit nahm, meine offenen Fragen zu klären.

Anhang

Glossar

Name	Beschreibung
CRUD-Operationen	Erstellen, Lesen, Aktualisieren und Löschen von Daten
DOM	Document Object Model – Struktur der HTML-Seite im Browser
Hooks	React Funktionen für Logik wie z. B. State
Integrationstest	Test, um Zusammenspiel von mehreren Komponenten zu prüfen
IPERKA	Projektmanagementmethode
Jsx	Syntaxerweiterung für JavaScript
JSON-String	Textformat, um Daten strukturiert zu speichern oder zu übertragen
localStorage	Speicher im Browser
Modal	Popup Fenster
Mock Daten	Fake Daten zum testen
Mockups	Entwurf des Designs
Prettier	VSCode Plugin zum formatieren
Props	Daten die von einer Elternkomponente zu einer Kindkomponente in React übergeben werden
React	Framework für moderne Web-Apps
Realisierungskonzept	Technischer Umsetzungsplan eines Projekts
Regex	Muster zur Erkennung oder Überprüfung von Zeichenfolgen
State	Zustand einer Komponente
States	Verwaltung von mehreren Zuständen in einer Komponente
Typen	Deklariert die Art von Daten z. B. String oder Number
useEffect	React Hook der auf Änderungen in React reagiert
useState	Speichert Werte im Komponenten State
Usecase diagramm	Visualisierung, wer was im System machen kann
Vietuelles DOM	Schnelle DOM-Kopie für Updates
Vitest	Modernes Test-Framework für Vite/Frontend
Vite	Build Tool für React und andere Frameworks

Quellenverzeichnis

Datum: 01.04.2025 IPERKA: https://www.ict-berufsbildung-bern.ch/resources/lperka_OdA_200617.pdf

Datum: 02.04.2025 Testkonzept: <https://www.peterjohann-consulting.de/testpyramide/>

Datum: 03.04.2025 Projektsetup: https://www.youtube.com/watch?v=17Yw-P-hfB4&ab_channel=CodingCrashkurse

Datum: 04.04.2025: useState im Projekt: https://www.w3schools.com/react/react_usestate.asp

Datum: 04.04.2025: Props im Projekt: https://www.w3schools.com/react/react_props.asp

Datum: 04.04.2025: Events im Projekt: https://www.w3schools.com/react/react_events.asp

Datum: 04.04.2025 wurde bei den Formularen Kunden Hinzufügen und Steuerjahr hinzufügen benutzt: <https://de.legacy.reactjs.org/docs/forms.html>

Datum: 05.04.2025: useEffect Funktion: https://www.w3schools.com/react/react_useeffect.asp

Datum: 09.04.2025: CSS: <https://www.w3schools.com/css/>

Abbildungsverzeichnis

Abbildung 1 Zeitplan	12
Abbildung 2 Usecase Diagramm	31
Abbildung 3 Mockups Startseite	32
Abbildung 4 Mockups Kundendetailsseite	32
Abbildung 5 Projekt Setup	40
Abbildung 6 Projektsetup komplett	41
Abbildung 7 Ordnerstruktur Anfang	42
Abbildung 8 Ordnerstruktur Ende 1	43
Abbildung 9 Ordnerstruktur Ende 2	44
Abbildung 10 useEffect Funktion	45
Abbildung 11 handleChange Funktion	46
Abbildung 12 handleSubmit Funktion	47
Abbildung 13 handleClientAdded Funktion	48
Abbildung 14 handleClientDeleted Funktion	48
Abbildung 15 Filterlogik	49
Abbildung 16 localStorage loadClients Funktion	50
Abbildung 17 localStorage saveClients Funktion	50
Abbildung 18 localStorage deleteClientFromStorage Funktion	50
Abbildung 19 localStorage getNextClientId Funktion	50
Abbildung 20 localStorage updateClientInStorage Funktion	51
Abbildung 21 Rollenrechte	52
Abbildung 22 RoleSwitcher	53
Abbildung 23 Rollenrechte Abfrage im Code	53
Abbildung 24 renderField Funktion	54
Abbildung 25 handleEditCustomerField Funktion	55
Abbildung 26 handleSave Funktion	56
Abbildung 27 handleCancelEdit Funktion	56
Abbildung 28 handleDelete Funktion	57
Abbildung 29 formatValue Funktion	58
Abbildung 30 updateClient Funktion	58
Abbildung 31 handleChange Funktion	59
Abbildung 32 handleSubmit Funktion	60
Abbildung 33 handleEditTaxYearField Funktion	61
Abbildung 34 handleSave Funktion	62
Abbildung 35 handleCancel Funktion	62
Abbildung 36 handleDelete Funktion	63
Abbildung 37 Beispiel Validation	64
Abbildung 38 Beispiel Validation 2	64
Abbildung 39 Beispiel Bestätigung	64
Abbildung 40 Beispiel Feedback	65
Abbildung 41 Startseite vorher	66
Abbildung 42 Startseite nachher	66
Abbildung 43 Kundendetailsseite vorher	67
Abbildung 44 Kundendetailsseite nachher	67

Abbildung 45 Modal Fenster.....	68
Abbildung 46 toggleCollapse Funktion	69
Abbildung 47 handleClearSearch Funktion.....	69
Abbildung 48 Test1 Erfolgreich	70
Abbildung 49 Test2 Erfolgreich	71
Abbildung 50 Test3 Erfolgreich	72
Abbildung 51 Test4 Erfolgreich	73
Abbildung 52 Test5 Erfolgreich	74
Abbildung 53 Test6 Erfolgreich	75
Abbildung 54 Test7 Erfolgreich	76
Abbildung 55 Test8 Erfolgreich	77
Abbildung 56 Test9 Erfolgreich	78
Abbildung 57 Test10 Erfolgreich	79
Abbildung 58 Test11 Erfolgreich	80
Abbildung 59 Test12 Erfolgreich	81

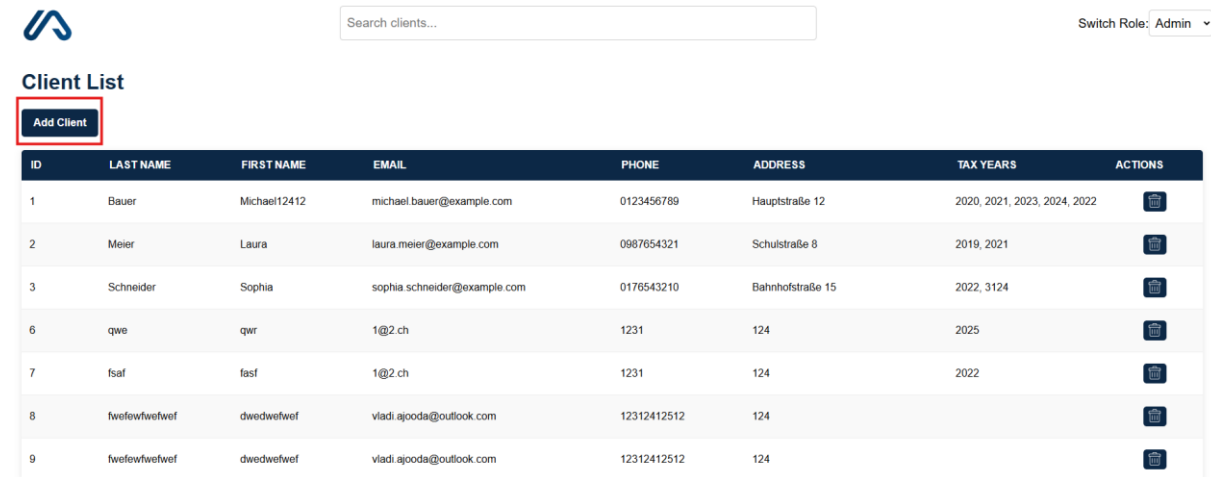
Anleitung Steuerverwaltungsplattform

In dieser Anleitung werden die Wichtigsten Funktionen der Steuerverwaltungsplattform erklärt.

Die Anwendung wird durch Eingabe von `npm run dev` im Projektverzeichnis gestartet.

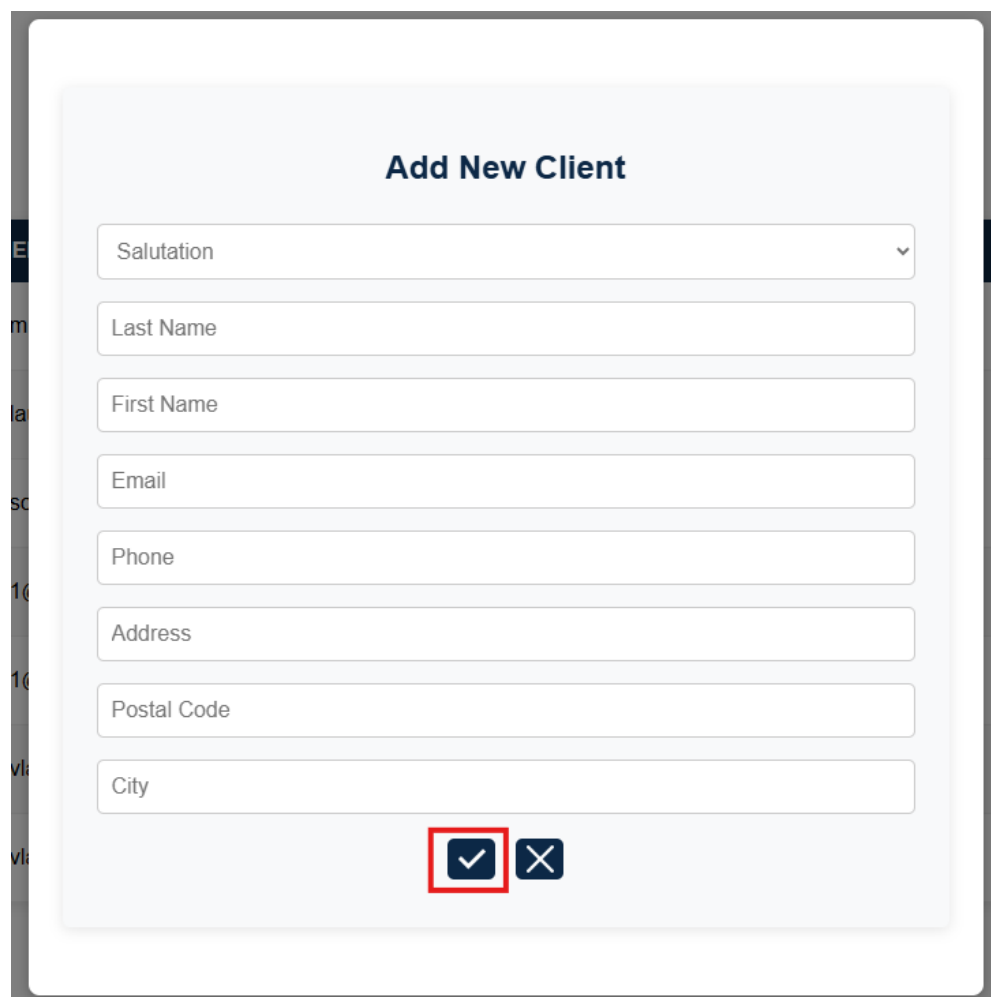
Kunde hinzufügen

1. Auf Add Client klicken.



ID	LAST NAME	FIRST NAME	EMAIL	PHONE	ADDRESS	TAX YEARS	ACTIONS
1	Bauer	Michael12412	michael.bauer@example.com	0123456789	Hauptstraße 12	2020, 2021, 2023, 2024, 2022	
2	Meier	Laura	laura.meier@example.com	0987654321	Schulstraße 8	2019, 2021	
3	Schneider	Sophia	sophia.schneider@example.com	0176543210	Bahnhofstraße 15	2022, 3124	
6	qwe	qwr	1@2.ch	1231	124	2025	
7	fsaf	fasf	1@2.ch	1231	124	2022	
8	fwefwfwefwef	dwedwefwef	vladi.ajooda@outlook.com	12312412512	124		
9	fwefwfwefwef	dwedwefwef	vladi.ajooda@outlook.com	12312412512	124		

2. Formular ausfüllen und absenden.



Add New Client

Salutation

Last Name

First Name



Email

Phone

Address

Postal Code

City



Kunde löschen

1. In der Letzten Spalte der Liste auf den Papierkorb klicken.



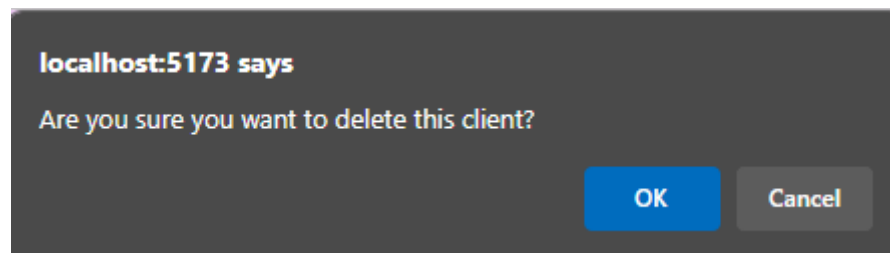
Switch Role: Admin ▾

Client List

[Add Client](#)


ID	LAST NAME	FIRST NAME	EMAIL	PHONE	ADDRESS	TAX YEARS	ACTIONS
1	Bauer	Michael12412	michael.bauer@example.com	0123456789	Hauptstraße 12	2020, 2021, 2023, 2024, 2022	
2	Meier	Laura	laura.meier@example.com	0987654321	Schulstraße 8	2019, 2021	
3	Schneider	Sophia	sophia.schneider@example.com	0176543210	Bahnhofstraße 15	2022, 3124	
6	qwe	qwr	1@2.ch	1231	124	2025	
7	fsaf	fasf	1@2.ch	1231	124	2022	
8	fwefwfwefwef	dwedwefwef	vladi.ajpoda@outlook.com	12312412512	124		
9	fwefwfwefwef	dwedwefwef	vladi.ajpoda@outlook.com	12312412512	124		

2. Bestätigungsfenster akzeptieren.





Kunde Bearbeiten

1. Die Zelle des zu bearbeitenden Kunden anklicken

2	Meier	Laura	laura.meier@example.com	0987654321	Schulstraße 8	2019, 2021	
---	-------	-------	-------------------------	------------	---------------	------------	---

2. Im Kundendetails Bereich den Stift zum Bearbeiten klicken.

 [Back](#)

Client Details 

Salutation:
Mrs.

Last Name:
Meier

First Name:
Laura



Email:
laura.meier@example.com


Phone Number:
0987654321

Address:
Schulstraße 8



Postal Code:
4001

City:
Basel


 

Switch Role: Admin 

Tax Years
[Add Tax Year](#)

YEAR	STATE TAX	FEDERAL TAX	TOTAL TAX	WITHHOLDING TAX PAID	DIFFERENCE	POTENTIAL SAVINGS	ACTIONS
2019	CHF 2,500	CHF 14,000	CHF 16,500	CHF 16,000	CHF 500 remaining	CHF 300	Tax return Provisional calculation Show comments Show savings 
2021	CHF 2,900	CHF 15,500	CHF 18,400	CHF 17,500	CHF 900 remaining	CHF 450	Tax return Provisional calculation Show comments Show savings 

3. Kunde bearbeiten und bestätigen.

Client Details 

Salutation:

Last Name:

First Name:



Email:

Phone Number:

Address:

Postal Code:

City:

Rolle wechseln

1. In der oberen Rechten Ecke den Rollenwechsler finden.

Switch Role: Admin ▼


2. Gewünschte Rolle auswählen.

Switch Role: Admin ▼

- Admin
- Advisor
- Client

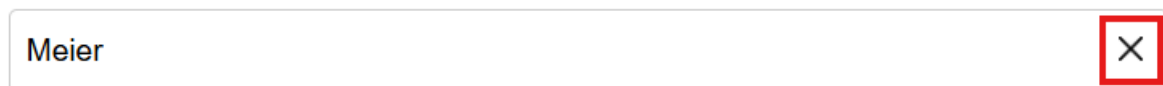
Suche

1. In der Suchleiste gesuchten begriff eingeben.



ID	LAST NAME	FIRST NAME	EMAIL	PHONE	ADDRESS	TAX YEARS	ACTIONS
2	Meier	Laura	laura.meier@example.com	0987654321	Schulstraße 8	2019, 2021	

2. Nach der Suche, Suchbegriff mit dem X löscht



Navigation

Um zur Kundendetails Seite zu navigieren, auf die Zeile des Kunden klicken.

ID	LAST NAME	FIRST NAME	EMAIL	PHONE	ADDRESS	TAX YEARS	ACTIONS
1	Bauer	Michael	michael.bauer@example.com	0123456789	Hauptstraße 12	2020, 2021, 2023, 2024, 2022	
2	Meier	Laura	laura.meier@example.com	0987654321	Schulstraße 8	2019, 2021	
3	Schneider	Sophia	sophia.schneider@example.com	0176543210	Bahnhofstraße 15	2022, 3124	
6	qwe	qwr	1@2.ch	1231	124	2025	

Um zurück auf die Startseite zu navigieren, den Back Button der sich oben Links befindet.



Client Details **Tax Years**

Salutation: Mrs.

Add Tax Year

Steuerjahr hinzufügen

1. Add Tax Year Button auf der Kundendetail Seite klicken.

The screenshot shows the 'Client Details' page for a client named Laura Meier. On the right, the 'Tax Years' section is visible, containing a table with tax data for 2019 and 2021. The 'Add Tax Year' button is highlighted with a red box.


YEAR	STATE TAX	FEDERAL TAX	TOTAL TAX	WITHHOLDING TAX PAID	DIFFERENCE	POTENTIAL SAVINGS	ACTIONS
2019	CHF 2,500	CHF 14,000	CHF 16,500	CHF 16,000	CHF 500 remaining	CHF 300	Tax return Provisional calculation Show comments Show savings
2021	CHF 2,900	CHF 15,500	CHF 18,400	CHF 17,500	CHF 900 remaining	CHF 450	Tax return Provisional calculation Show comments Show savings

2. Formular ausfüllen und absenden.

The screenshot shows the 'Add New Tax Year' form. It contains five input fields: Year, State Tax, Federal Tax, Withholding Tax Paid, and Potential savings. At the bottom, there are two buttons: a checkmark button (highlighted with a red box) and a close button (X).

Steuerjahr bearbeiten

1. Auf der Kundendetails Seite im Steuerjahr Bereich den Bearbeitungsbutton klicken.



Back



Client Details

Salutation: Mrs.
Last Name: Meier
First Name: Laura
Email: laura.meier@example.com
Phone Number: 0987654321
Address: Schulstraße 8
Postal Code: 4001
City: Basel

Switch Role: Admin

Tax Years





Add Tax Year

YEAR	STATE TAX	FEDERAL TAX	TOTAL TAX	WITHHOLDING TAX PAID	DIFFERENCE	POTENTIAL SAVINGS	ACTIONS
2019	CHF 2,500 Tax return	CHF 14,000 Provisional calculation	CHF 16,500 Show comments	CHF 16,000	CHF 500 remaining	CHF 300 Show savings	
2021	CHF 2,900 Tax return	CHF 15,500 Provisional calculation	CHF 18,400 Show comments	CHF 17,500	CHF 900 remaining	CHF 450 Show savings	

2. Steuerjahr bearbeiten und bestätigen.


Tax Years

Add Tax Year

YEAR	STATE TAX	FEDERAL TAX	TOTAL TAX	WITHHOLDING TAX PAID	DIFFERENCE	POTENTIAL SAVINGS	ACTIONS
2019	2500	14000	16500	18000	1500	300	  
2021	CHF 2,900 Tax return	CHF 15,500 Provisional calculation	CHF 18,400 Show comments	CHF 17,500	CHF 900 remaining	CHF 450 Show savings	



Steuerjahr löschen

1. Steuerjahr bearbeiten.





 [Back](#)

Client Details <

Salutation:
Mrs.
Last Name:
Meier
First Name:
Laura
Email:
laura.meier@example.com
Phone Number:
0987654321
Address:
Schulstraße 8
Postal Code:
4001
City:
Basel

Tax Years
[Add Tax Year](#)

YEAR	STATE TAX	FEDERAL TAX	TOTAL TAX	WITHHOLDING TAX PAID	DIFFERENCE	POTENTIAL SAVINGS	ACTIONS
2019	CHF 2,500 Tax return	CHF 14,000 Provisional calculation	CHF 16,500 Show comments	CHF 16,000	CHF 500 remaining	CHF 300 Show savings	 
2021	CHF 2,900 Tax return	CHF 15,500 Provisional calculation	CHF 18,400 Show comments	CHF 17,500	CHF 900 remaining	CHF 450 Show savings	 

Switch Role: Admin

2. Im Bearbeitungsmodus vom Steuerjahr den Papierkorb klicken.

YEAR	STATE TAX	FEDERAL TAX	TOTAL TAX	WITHHOLDING TAX PAID	DIFFERENCE	POTENTIAL SAVINGS	ACTIONS
2019	2500	14000	16500	16000	500	300	  

3. Löschen bestätigen

localhost:5173 says
Are you sure you want to delete the tax year 2019?
[OK](#) [Cancel](#)