Abschlussprüfung Sommer 2025

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Digitalisierung Anmeldeformular

Optimierung des Anmeldeprozesses für neue Mitglieder des IT-Klub Mainz & Rheinhessen e.V. durch die Digitalisierung des Anmeldeformulars

Abgabedatum:
Mainz, den 28.04.2025

Prüfungsbewerber:
Lukas Trapp

Prüflingsnummer:
25006

Ausbildungsbetrieb:
Thiele & Klose GmbH
Mombacher Str. 68
55122 Mainz

Inhaltsverzeichnis

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

API……Application Programming Interface

REST……Representational State Transfer

IDE……Integrated Development Environment

GUI……..Graphical User Interface

HTML..... Hypertext Markup Language

CSS……..Cascading Style Sheets

SCSS..... SassyCascading Style Sheets

MVC ..... Model View Controller

SQL……Structured Query Language

ODM……..Object Document Mapper

PAP……...Programmablaufplan

CORS……..Cross-Origin-Resource Sharing

1 Einleitung

1.1 Projektumfeld

Die Thiele & Klose GmbH ist ein mittelständisches Softwareunternehmen mit Standorten in Daxweiler und Mainz. Aktuell arbeiten dort 10 Mitarbeitende, darunter 2 Auszubildende, ein Werkstudent, 2 externe Kräfte und ein Praktikant.
Das Unternehmen entwickelt maßgeschneiderte Webanwendungen, Portale und Schnittstellen und bietet darüber hinaus IT-Dienstleistungen wie Web- und App-Entwicklung sowie IT-Strategieberatung an.
Besonderen Wert legt Thiele & Klose auf Usability, Barrierefreiheit und eine benutzerfreundliche Gestaltung seiner Softwareprodukte.
Für individuelle Kundenlösungen kommt ein modernes, vielseitiges Tech-Stack zum Einsatz, der sich an vorhandenen Systemen und dem Know-how des Teams orientiert.

1.2 Projektziel

Der Auftraggeber/Kunde des Projekts ist der IT Klub Mainz & Rheinhessen e.V. der für den Relaunch seiner Internetseite ein modernes und in sich von der Designsprache stimmiges, sowie funktional gut überlegtes und intuitiv bedienbares Konzept entwickelt und realisiert haben möchte.

Ziel dieses Projekts ist die Konzeption, Umsetzung und Dokumentation eines digitalen Anmeldeprozesses für neue Mitglieder, von der Formularerstellung bis zur zentralen Speicherung der Daten in einer Datenbank. Im Mittelpunkt steht die Entwicklung eines in die Website eingebetteten, benutzerfreundlichen Formulars sowie die Planung und Realisierung der notwendigen Komponenten samt deren Kommunikation. Der Anmeldevorgang soll vollständig digitalisiert werden und dadurch beschleunigt sowie für Nutzer vereinfacht, und intuitiv ablaufen.

1.3 Projektbegründung

Der Anmeldeprozess findet aktuell noch außerhalb der Website statt, was sowohl für die Nutzer als auch für den IT-Klub Nachteile mit sich bringt, insbesondere entsteht durch fehlende Digitalisierung langfristig Mehraufwand und zusätzliche Kosten. Die drei Hauptgründe der Projektdurchführung sind:

1. **Verbesserung der Usability:** Nutzer müssen das Formular nicht mehr extern bearbeiten, ausdrucken oder manuell per Post/Mail versenden. Die Anmeldung erfolgt direkt und bequem auf der Website, was die Nutzerfreundlichkeit deutlich verbessert.
2. **Effizienz und Nachhaltigkeit:** Statt mehrerer Schritte wird der Prozess auf das Ausfüllen und Absenden eines Onlineformulars reduziert, Seitenintern und papierlos.
3. **Ersparnisse für den IT-Klub:** Nach Umsetzung entfallen manuelle Arbeitsschritte bei der Datenverarbeitung. Die Anträge werden automatisiert in der Datenbank gespeichert, was den Verwaltungsaufwand und damit die Kosten deutlich senkt.

1.4 Projektschnittstellen

Das entwickelte Anmeldeformular interagiert mit mehreren Akteuren und Systemkomponenten. Im Zusammenspiel von Frontend und Backend sendet das Formular die vom Benutzer eingegebenen Daten per POST-Request an die REST-API, welche mit Node.js umgesetzt wurde. Die API verarbeitet die Anfragen mit dem Object Document Mapping (ODM) Mongoose, um die Daten in der MongoDB-Datenbank gemäß dem definierten Schema zu speichern. Die Anwendung läuft isoliert in Docker-Containern, dadurch sind Frontend, API und Datenbank voneinander getrennt und unabhängig lauffähig. Eine serverseitige Validierung der Formulardaten erfolgt sowohl durch die im Controller verwendete Funktion validateMember als auch durch die Mongoose-Methode Member.findOne, bevor die Daten in der Datenbank gespeichert werden. Die Hauptnutzer des Formulars sind Personen, die sich als Mitglieder beim IT-Klub anmelden möchten.

Zu Beginn des Projekts wurde der eingesetzte Tech-Stack ermittelt und festgelegt. Die finale Abnahme erfolgte durch eine Vorstellung beim Kunden, vertreten durch die verantwortliche Person des IT-Klub Mainz & Rheinhessen e.V.

1.5 Projektabgrenzung

Das Projekt schließt ausschließlich den Vorgang der Erfassung der Daten von Neumitgliedern im Frontend mittels des Formulars sowie die Weiterreichung und das Handling dieser bis in die Filebasierte Datenbank mit ein.
Ausdrücklich nicht Bestandteil des Projekts ist die neu entstehende Gesamtwebsite, die zukünftig unter https://www.itklub.de erreichbar sein wird. Ebenso gehören der Header und der Footer, die lediglich importiert und eingebunden wurden, nicht zum Projektumfang. Auch die Erstellung, Einrichtung und Anbindung der Docker-Container wurden nicht im Projektrahmen umgesetzt und waren nicht teil dessen. Die Gestaltungsvorlage

für das Mitgliederformular wurde in Absprache mit der Designabteilung extern entworfen und war ebenfalls nicht Teil der Projektaufgaben. Die Abfrage einer Unterschrift wurde in diesem Projekt nicht mit einbezogen und soll zu einem späteren Zeitpunkt in einem nachgelagerten Projekt realisiert werden.

2. Projektplanung

2.1 Projektphasen

Die Projektplanung beschreibt die zeitliche und organisatorische Umsetzung des Projekts „Digitalisierung Anmeldeformular“. Hierbei wurden alle Phasen detailliert geplant und überprüft, um den Fortschritt zu sichern und festzuhalten. Tabelle 1 zeigt die grobe Zeitplanung des Projekts.

Grobe Zeitplanung	
Projektphase	Geplant
Recherche und Planung	10 Stunden
Entwicklung des Frontend	15 Stunden
Entwicklung des Backend	35 Stunden
Testen und Dokumentation	20 Stunden
Gesamt	80 Std

Tabelle 1: Grobe Zeitplanung

Eine ausführlichere und detailliertere Zeitplanung wird in einem Schaudiagramm mit nachfolgender Tabelle in Anhang >*(A1 Detaillierte Zeitplanung)* dargestellt.

2.2 Abweichungen vom Projektantrag

Im Detail betrachtet zeigte sich während des Projektes eine zeitliche Abweichung, ein Mehraufwand in der Phase (Recherche und Planung von +5h), hier waren etwas tiefere Recherchen notwendig. Dieser Mehraufwand konnte jedoch einige Nachrecherchen in der Phase (Entwicklung des Backends -5h) einsparen und somit das geplante Ziel im Bezug auf die Dauer des Projekts einhalten. Dargestellt als Schaubild in Abschnitt (9.1 Soll-Ist-Vergleich).

2.3 Ressourcenplanung

Die eingesetzten Ressourcen innerhalb des Projektes werden in Hardware, Software und Personal unterteilt und nachfolgend aufgelistet. Die Auswahl der Mittel wurde unter Berücksichtigung und Einhaltung firmenspezifischer Standards getroffen.

Hardware	Software	Personal
MacBook Pro/ OS 14 Bildschirm: LG 38BQ85C-W Apple Magic Trackpad Apple Magic Keyboard	Visual Studio Code Dokumentationen: Astro, Node.js, Express, MongoDB, Mongoose, Apple-System Google zum erläutern Compass MongoDB Postman Microsoft Word	Projektleitung und Ausführung (Autor) Designabteilung (Maira. P.) Eventuelle Rückfragen (Ausbilder)

Tabelle 2: Ressourcenplanung

2.4 Entwicklungsprozess

Auf Grund der Komplexität des Projektes (Frontend, Backend mit API und Datenbankanbindung) wurde ein **iteratives Vorgehensmodell** gewählt, es zeichnet sich durch kurze Zyklen und kontinuierliche Überprüfung der Ergebnisse aus. So konnten Fortschritte Funktion und Logik direkt getestet werden.

Merkmale des Vorgangsmodells:

1. **Analyse und Entwurf:** Anforderungen, sowie Inhalte wurden mit dem alten PDF-Formular abgeglichen und für die Erstellung eines neuen Formulars festgehalten. Ein Entwurf wurde mit der Designabteilung abgesprochen und in Auftrag gegeben.
2. **Iterative Entwicklung:** Nach Implementierung der einzelnen Elemente wurden diese entweder erst mittels Browser und der Konsole bzw. später mittels Postman und der Datenbank getestet. Fehler wurden nach jedem funktionalem oder logischem Schritt direkt getestet und gegebenenfalls behoben, bevor weitere Funktionen implementiert wurden.
3. **Testphase:** Nach Fertigstellung des Formulars und der Logik zwecks Validierung wurde alles nochmal auf Fehleingaben und Richtigkeit der Validierung, ebenso wie das Erreichen des Endpunktes in der Datenbank getestet.
4. **Abnahme:** Nach Abschluss der Tests wurde das Formular inklusive seiner Funktionalität und Benutzeroberfläche dem Verantwortlichen des IT-Klubs präsentiert und von diesem Abgenommen.

3. Analysephase

3.1 Ist-Analyse

Der aktuelle Prozess, dem IT-Klub beizutreten ist sehr umständlich gestaltet, da das Anmeldeformular entweder seitenextern, digital in einem PDF-Editor bearbeitet werden oder gar ausgedruckt, ausgefüllt, unterschrieben, wieder digitalisiert (z. B. eingescannt) und anschließend per E-Mail oder Fax an den Klub gesendet werden muss. Alternativ kann das Formular nach dem Druck und der Unterschrift auch per Post gesendet werden, was den Prozess ebenfalls unnötig aufwendig macht. Der Prozess beinhaltet momentan Aspekte, die im Sinne des IT-Klubs und der User beseitigt werden sollen.

Die bisherige Abwicklung des Anmeldeprozesses ist mit erheblichem Mehraufwand verbunden, da die Weiterverarbeitung physisch auf drei Kanälen per E-Mail, per Post oder per Fax erfolgt. Die händische Sortierung und Speicherung der Daten an einem zentralen Ort führt zu einer höheren Nutzung von Hardware und personellen Ressourcen, was wiederum Stromverbrauch als auch Personalkosten erhöht. Zudem fehlt es dem analogen Verfahren an Nachhaltigkeit, da weder Ressourcen wie Papier und Hardware noch Energie effizient genutzt werden. Der Transportprozess ist durch viele Zwischenschritte und teilweise Postversand zusätzlich ineffizient. Durch die Digitalisierung des Prozesses können diese Ressourcen geschont, Energie eingespart und die Effizienz insgesamt deutlich gesteigert werden.

Darüber hinaus ist die bisherige Lösung für Nutzerinnen und Nutzer unkomfortabel in der Anwendung. Die Usability und User Experience lassen zu wünschen übrig, was die Benutzerfreundlichkeit der Seite beeinträchtigt.

3.2 Wirtschaftlichkeitsanalyse

Auf Grund der erhöhten Kosten bei einer Gegenüberstellung analoger Antragsabwicklung gegenüber der digitalen Abwicklung gibt es hinreichende Gründe, die Digitalisierung des Anmeldeprozesses umzusetzen. Nachdem sich das Projekt amortisiert hat, wird ab diesem Zeitpunkt ein gewisser Betrag pro Jahr eingespart. Zusätzlich wird der Sinn von Green-IT durch die Effizienz und Ressourcenschonung aktiv gelebt. Wenn man die Kosten, die durch das händische Abarbeiten und Einpflegen der Neuanträge sowie den zusätzlichen Stromverbrauch während der Bearbeitungszeit entstehen, mit den Kosten vergleicht, die für die digitale Abbildung und Einbindung des Formulars zum Zweck der

Digitalisierung des Anmeldeprozesses anfallen, wird deutlich, dass sich dieser Schritt nicht nur aus Gründen der Usability und Green IT, sondern auch monetär für den Klub lohnt.

3.2.1 Make-or-Buy-Entscheidung

Die Make-or-Buy-Entscheidung entfällt hier aufgrund der Tatsache, dass es sich bei dem Projekt um Individualsoftware handelt, die in ein bestehendes System (die Website) eingebunden wird. Eine fertige, kaufbare Lösung existiert somit nicht, und ein Baukastensystem könnte die Individualität bzw. das Design nicht umfassend genug aufgreifen und abbilden. Aufgeführt könnte nur der Punkt der monetären Unterscheidung im Sinne von Make-or-Buy werden. Da sich der Durchführende zur Zeit des Projekts als Praktikant im Unternehmen befand konnten die Gesamtkosten gering gehalten werden. In Absprache mit dem Kunden wurde die Entscheidung in allen Punkten für das ,,Make“ getroffen.

3.2.2 Projektkosten

Für die Umsetzung des Projekts wurden, abgesehen von vorhandener Hardware (MacBook, Bildschirm, Trackpad, Tastatur) keine weiteren kostenpflichtigen Mittel eingesetzt. Microsoft Office und Visual Studio Code sind bereits durch die Firmenlizenz abgedeckt, daher entstanden nur Personal- und Hardwarekosten. Der Projektdurchführende war während des Projekts als Praktikant im Rahmen einer Umschulung tätig. Für ihn wurde für die Kalkulation ein fiktiver Stundenlohn von 8 € angenommen. Auch für weitere Positionen wurden auf Grundlage firmeninterner Vorgaben fiktive Werte verwendet.

Eine genaue Aufstellung der Kosten findet sich im **Anhang A2 – Kostenaufstellung.**

Kostenrechnung				
Vorgang	**Personal/Hardware**	**Stunden**	**€ / Stunde**	**Gesamt**
Hardware incl. Stromkosten	MacBook, Bildschirm (anteilig)	80 Stunden	1 €	80 €
Entwicklungskosten	1 Praktikant	79 Stunden	8 €	632 €
Test und Fragen	Mitarbeiter	4 Stunden	65 €	260 €
Rücksprache	Ausbilder	2 Stunden	95 €	190 €
Abnahme	Ausbilder	2 Stunden	95 €	190 €
Vorstellung, Abnahme	Praktikant, Kunde	1 Stunde	8 €	8 €
Gesamt				**1360 €**

Tabelle 3: Kostenrechnung

3.2.3 Amortisationsdauer

Das Projekt spart die Zeit der physischen Bearbeitung durch händisches Auslesen der Daten aus Fax, Mail und Post sowie das Einpflegen dieser ein. Somit wird dieser Bereich effizienter im Prozess und spart Personalaufwand und damit Kosten ein.

Beispielrechnung: Annahme für die Beispielrechnung ist das dem IT-Klub 1 – 3 neue Mitglieder pro Woche beitreten. Herunter gebrochen auf einen Tag wären, dass 0,14 bis 0,43 neue Mitglieder täglich. Eine manuelle Bearbeitung eines Mitgliedsantrages (inclusive E-Mail drucken, Unterschrift scannen und einpflegen des neuen Mitgliedes in einer Datenbank) könnte etwa 10-15 Minuten pro Antrag dauern. Das bedeutet es wird in der Rechnung mit dem Mittelwert aus (1,4 und 6,5 Minuten) 3,925 also 4 Minuten pro Tag angenommen. Für die Rechnung wird ein Aufwand von 35 € pro Stunde für die Sachbearbeitung angenommen.

Mitarbeiter bei der Erfassung der Beitrittsanträge:

$$220\,\frac{\text{Tage}}{\text{Jahr}} \cdot\ 4\,\frac{\text{min}}{\text{Tag}} =\ 880\,\frac{\text{min}}{\text{Jahr}} \approx\ 14{,}67\,\frac{\text{h}}{\text{Jahr}}\ \cdot 35\,\frac{€}{\text{Stunde}} = 513{,}45\ €$$

Das Erfassen der Anträge kostet mit den angenommenen Werten 513,45 € pro Jahr.

$$\frac{1360\ €}{513{,}45€/Jahr} = 2{,}65 \approx\ 2{,}7\ Jahre$$

Die Amortisationszeit beträgt also in etwa: 2,7 Jahre = 2 Jahre und 8 Monate. Nach diesem Zeitraum spart der IT-Klub durch die Digitalisierung der Komponente und des Prozesses 513,45 € pro Jahr ein.

3.3 Nutzwertanalyse

Darstellung des nicht-monetären Nutzens (Vorher-/Nachher) (1 schlecht - 10 gut)

	Kriterium	Gewichtung (%)	Alt (Physisch)	Neu (Digital)	Nutzwert Alt	Nutzwert Neu
1	Nutzerfreundlichkeit	25	3	10	0,75	2,5
2	Zeitaufwand	20	4	10	0,8	2
3	Medienbruchfreiheit	15	2	10	0,3	1,5
4	Nachhaltigkeit	15	3	9	0,45	1,35
5	Fehleranfälligkeit	15	5	9	0,75	1,35
6	Barrierefreiheit	10	4	8	0,4	0,8
	Gesamt	**100**	**27**	**56**	**3,45**	**9,5**

Tabelle 4: Nutzwertanalyse

Gewinner der hier angesetzten Nutzwertanalyse ist klar die Digitalisierung (Neu).

3.4 Anwendungsfälle

Der zentrale Anwendungsfall, der mit dem Projekt abgedeckt wird, ist die digitale Anmeldung neuer Mitglieder über ein webbasiertes Formular. Dieser Hauptprozess umfasst mehrere untergeordnete Anwendungsfälle, darunter das Aufrufen des Formulars, die Eingabe der Pflichtdaten, die Zustimmung zu Datenschutzbestimmungen, Satzung und Beitragordnung, das Absenden der Daten sowie die Rückmeldung an den Nutzer durch Weiterleitung auf die Bestätigungsseite (member-approved.astro). Im Hintergrund erfolgt die Speicherung der übermittelten Daten mithilfe von Mongoose in der MongoDB-Datenbank.

Der maßgebliche Akteur ist der Nutzer, der den Anmeldeprozess durchführt. Zur Veranschaulichung der Abläufe finden sich im Anhang ein Programmablaufplan unter (A3) sowie ein Use-Case-Diagramm unter (A4).

3.5 Qualitätsanforderungen

Zu den Qualitätsanforderungen zählen Funktionalität, Zuverlässigkeit, Usability, Effizienz, Wartbarkeit und Portabilität. Die Anwendung muss die korrekte Speicherung der Formulardaten sicherstellen, wobei Pflichtfelder über Regex validiert werden. Fehleingaben dürfen nicht zum Absturz führen, sondern sollen durch Fehlermeldungen aufgezeigt werden.

Die Bedienung soll intuitiv, das Design responsiv und barrierearm, unter anderem durch hohe Kontraste umgesetzt sein. Ladezeiten sollen gering sein, um ein flüssiges Nutzererlebnis zu ermöglichen. Der Code soll nachvollziehbar strukturiert und kommentiert sein, wodurch Wartung und Weiterentwicklung erleichtert werden. Durch die Nutzung von Docker-Containern ist das Projekt leicht auf andere Systeme übertragbar.

4. Entwurfsphase

4.1 Zielplattform

Die Anwendung basiert auf einer Client-Server-Struktur. Das digitale Anmeldeformular wird im Webbrowser (Client) ausgeführt, es sendet Daten an das Server-Backend, welches die Daten verarbeitet und in einer Datenbank speichert. Ein wesentlicher Vorteil ist die Trennung von Benutzeroberfläche und Logik, was durch die getrennte Architektur eine hohe Wartbarkeit und gute Skalierbarkeit ermöglicht, da Zuständigkeiten klar getrennt sind und gut verortet werden können.

Plattformdetails: Die Entwicklung erfolgte lokal auf einem MacBook (macOS) mithilfe von Docker-Containern für eine konsistente Umgebung. Im Frontend kamen HTML, CSS, JavaScript und Astro zum Einsatz, letzteres ermöglicht schnelle Ladezeiten und komponentenbasiertes Arbeiten. Das Backend basiert auf Node.js, einer asynchronen, leichtgewichtigen Plattform, ideal für HTTP-basierte REST-APIs. Als Datenbank wurde MongoDB verwendet sie ist dokumentenbasiert, flexibel und daher ideal für JSON-Daten geeignet. Die Anbindung erfolgte über Mongoose, das einen Schema-basierten Zugriff ermöglicht. Das responsive Design stellt die Kompatibilität mit Desktops, Tablets und Smartphones sicher.

4.2 Architekturdesign

Für die Architektur des Projekts wurde das MVC-Pattern gewählt. Dieses Muster ermöglicht eine klare Trennung der verschiedenen Komponenten im Backend und unterstützt dadurch die Modularität, Wartbarkeit und Skalierbarkeit der Anwendung.
Das **Model** enthält die Formulardaten sowie die zugehörige Struktur und Validierung. Die **View** umfasst die vom Server zurückgegebenen Antworten (z. B. JSON oder Statuscodes). Der **Controller** dient als Vermittler, er verarbeitet Anfragen und leitet die Daten an das Model weiter. Die grafische Benutzeroberfläche im Browser gehört nicht zum MVC und wurde separat im Frontend umgesetzt.

Eine Darstellung der Backend-Architektur ist im Anhang unter A5 zu finden.

4.3 Datenmodell

In diesem Projekt umfasst die Datenbankstruktur eine einzelne Entität, die (Formular) genannt wird. Diese Entität speichert alle Formulardaten, wie Firmenname, Name, E-Mail, Adresse, Telefonnummer und Unternehmensgröße etc. Sie stellt sicher, dass die Benutzerdaten strukturiert und sicher gespeichert werden. Zu sehen im Anhang > *(A6 Schaubild Datenmodell)* und im Detail in Anhang > *(A7 Mongoose Schema und Code)*

4.4 Geschäftslogik

Die fertige Anwendung erfasst mittels des responsiven Onlineformulars, Mitgliedsanfragen für den IT-Klub Mainz & Rheinhessen e.V.. Hierbei wird eine clientseitige Validierung mittels JavaScript durchgeführt. Nach erfolgreicher Prüfung werden die Daten über eine REST-API an das Node.js Backend gesendet, welches die Daten erneut serverseitig validiert und sie anschließend in der verwendeten MongoDB Datenbank speichert. Die wichtigsten Schritte in Reihenfolge sind:

1. **Nutzereingabe:** Der Benutzer füllt das Formular mit den Pflichtfeldern wie Name, Adresse, Telefon, E-Mail etc. aus, setzt den Radiobutton und die Checkboxen.
2. **Clientseitig Validierung:** Die Felder werden im Frontend auf Vollständigkeit und korrektes Format geprüft (z. B. required-Attribute, passende Datentypen sowie reguläre Ausdrücke für Postleitzahl, Telefonnummer und E-Mail-Adresse).
3. **Senden des Formulars:** Anschließend an die erfolgreiche Validierung wird mittels der fetch() Methode ein POST-Request an /membersform gesendet.
4. **Serverseitige Validierung:** Die REST-API prüft die empfangenen Daten erneut mithilfe eines Validator-Moduls, so wie einer Dubletten-Prüfung, bevor diese weiterverarbeitet werden. Sollte hier ein Fehler auftreten wird dies in der Konsole geloggt und der User erhält einen Aussagekräftigen Alert im Browser.
5. **Datenmodellierung:** Mit dem Mongoose-Schema werden die Daten nun in das Format des Datenbankspeichers überführt.
6. **Persistierung:** Die Mitgliedsdaten werden in der MongoDB Datenbank dauerhaft gespeichert.
7. **Antwort an Client:** Nach erfolgreicher Speicherung wird der Nutzer auf die (members-approved.astro) Seite weitergeleitet und erhält dort Rückmeldung über die erfolgreiche Anmeldung zu sehen im Anhang (A8) Members-Approved.

Integration in den Arbeitsfluss des Unternehmens: Im bisherigen Verfahren musste das Beitrittsformular manuell heruntergeladen, ausgedruckt, ausgefüllt und unterschrieben sowie anschließend per E-Mail, Fax oder Post eingereicht werden, ein aufwendiger, fehleranfälliger Prozess für beide Seiten. Die neue, digitale Abhandlung integriert sich nahtlos in den Arbeitsfluss:

1. Neumitglieder füllen das Formular direkt auf der Website aus und senden es ab.
2. Die Daten werden automatisiert, strukturiert und zentral gespeichert.

Der digitale Prozess vereinfacht die Erfassung und Pflege der Mitgliedsdaten, senkt den Aufwand und spart Kosten. Für die Nutzer entsteht ein durchgängig intuitives Erlebnis.

4.5 Maßnahmen zur Qualitätssicherung

Zur Sicherung der Qualität wurden mehrere Maßnahmen getroffen:

1. **Manuelle iterative Tests (Entwicklerseite):** Jede Komponente des Formulars (Inputfelder, Checkboxen, Radiobuttons) wurde auf fehlerhafte Eingabe, Funktion, falsche Zeichen ungültige Längen und/oder Regex-Verstöße getestet. Nach jeder Korrektur wurde die betroffene Stelle erneut getestet.
2. **Fremdtest UX/UI (Projektfremde Person):** Eine Zweite Person hat das Formular auf Benutzerfreundlichkeit (UI/ UX), Barrierefreiheit und Eingabefluss getestet. Dabei wurde sinnvoll geäußerte Kritik, aufgenommen und umgesetzt.
3. **Backend-Test:** Die Rest-API wurde mit dem Tool Postman getestet. Überprüft wurden insbesondere: Struktur der empfangenen Daten, Validierung, Fehlerbehandlung und korrekte Speicherung in der Datenbank.
4. **Unittest der Validierungslogik:** Die Validierungsfunktion in Backend (membersValidator.js) wurde mit unterschiedlichen Beispieldatensätzen auf Richtigkeit geprüft. Tests mit fehlender oder ungültiger E-Mail-Adresse bzw. Postleitzahl führten zu den erwarteten Fehlermeldungen und verhinderten die Speicherung der Daten. Anmeldeversuche mit bereits registrierter E-Mail-Adresse wurden ebenfalls blockiert. Hinweise wurden über einen Alert im Browser als auch per console.log in der Entwicklerkonsole ausgegeben.
5. **Gesamttest (Ende der Entwicklung):** Nach Fertigstellung wurde ein End-to-End-Test durchgeführt. In diesem wurde die GUI/Darstellung des Formulars, die

Datenweitergabe, die Konsistenz und Speicherung der Daten und die Reaktionen auf Fehlbedienung im Ganzen überprüft.

Die mehrstufige Teststrategie stellt sicher, dass das Formular sowohl technisch solide und funktionabel als auch für Endnutzer nachvollziehbar und angenehm nutzbar ist.

5. Implementierungsphase

5.1 Implementierung der Datenstrukturen

Für die Datenstruktur wurde die dokumentenbasierte Datenbank MongoDB gewählt. Die Modellierung erfolgte mit dem ODM Mongoose, wobei ein verschachteltes Schema manuell erstellt wurde, siehe Anhang (A7) Mongoose-Schema und Code. Statt einer klassischen relationalen SQL-Datenbank (z. B. MySQL) fiel die Entscheidung bewusst auf MongoDB aus folgenden Gründen:

1. Die Datenstruktur des Formulars ist dokumentenartig, so dass ein Antrag einem JSON-File entspricht. Das passt ideal zur dokumentenbasierten Speicherung.
2. Es bestehen keine relationalen Abhängigkeiten oder Fremdschlüssel. Ein SQL-System mit starrem Schema würde unnötigen Overhead ohne funktionalen Mehrwert erzeugen.
3. Die Formulardaten können direkt per JSON.stringify() strukturiert übermittelt und gespeichert werden, ohne Konvertierung oder zusätzliche Zwischenschritte.
4. Die Integration mit Node.js und Mongoose ist unkompliziert und ermöglicht dadurch schnellere Entwicklung und Umsetzung von Prototypen.

Aufbau des Datenmodells:

Das Modell erfasst alle für die Registrierung notwendigen Angaben: Unternehmensname, Ansprechpartner, Adresse, Telefonnummer und E-Mail. Die Unternehmensgröße wird über Radiobuttons gewählt, da sie die Beitragshöhe beeinflusst. Zwei Checkboxen bestätigen die Zustimmung zur Satzung, Beitragsordnung und Datenschutzerklärung, alle als verlinkte PDFs einsehbar. Das Datenmodell basiert auf dem in Anhang (A7) dargestellten Mongoose-Schema und ist in logisch getrennte Bereiche gegliedert. Die Felder „contact“ und „address“ sind dabei als verschachtelte Objekte umgesetzt. Der modulare Aufbau ermöglicht eine einfache und effiziente Erweiterung.

5.2 Implementierung der Benutzeroberfläche

Zur Gestaltung der Benutzeroberfläche fand ein Termin mit der Kollegin aus der Designabteilung statt. Dabei wurden das neue Layout und Design des Formulars anhand des alten Mitgliederformulars, der verknüpften PDF-Dokumente sowie des Corporate Designs der neuen Website abgestimmt. Screenshots des finalen Designs finden sich im Anhang unter (A9) Mitgliederformular: Desktop und (A10) Mitgliederformular: Mobil.

5.3 Implementierung der Geschäftslogik

Nach Umsetzung des HTML-Formulars und Designs wurde die dahinterliegende Logik implementiert. Als Entwicklungsumgebung kam Visual Studio Code zum Einsatz. Anschließend folgte die Datenbankanbindung, ein POST-Request nimmt die Formulardaten entgegen und übergibt sie an das Backend, wo sie serverseitig validiert und mittels Mongoose nach definiertem Schema in der DB gespeichert werden. Dabei wird vorab die Datenbankverbindung hergestellt. Eine Skizze zum Entwurfsmuster und dem Gesamtaufbau befindet sich im Anhang (A11). Der Quellcode ist dokumentiert und klar vom Fremdcode abgegrenzt siehe (A12) Quellcodeübersicht, im Anhang. Nach Abschluss der Implementierung, sowie nach erfolgreicher Qualitätssicherung (Abschnitt 4.4) erfolgte die Abnahmephase. Diese umfasste die interne Durchsicht sowie die Vorstellung der GUI und die Übergabe des Benutzerhandbuchs an den Kunden. Auszüge des Handbuchs sind in Anhang (A13) zu sehen.

6. Einführungsphase

Folgend auf die Abnahmephase wurde der Entwicklungs-Branch, auf dem das Formular erstellt wurde, in das Hauptprojekt (abgegrenzt (die Website)) integriert, gemerged. Anschließend war das Formular über die firmeninterne Qualitätssicherungs-URL (QRT) im Gesamtkontext verfügbar und aufrufbar.

7. Dokumentation

Um den Vorgang des Anmeldens für Rückfragen möglichst gut für den User zu gestalten, wurde das erstellte Benutzerhandbuch dem Kunden ausgehändigt. Es ist wie unter Punkt 6 angegeben in Anhang (A13) in Auszügen einsehbar. Hierbei wurde darauf geachtet, dass keine Fachbegriffe verwendet oder diese erläutert wurden.

8. Retrospektive / Fazit

8.1 Soll-Ist-Vergleich

Das Projekt wurde erfolgreich durchgeführt und abgeschlossen, die gesteckten Ziele wurden erreicht. Im digitalisierten Mitgliedsantragsformular können die Daten nun bequem seitenintern eingegeben, abgehandelt und automatisiert gespeichert werden. Somit wurde die Zielsetzung in Sachen Usability und Effizienz mit Erfolg erreicht.

Ein Vergleich der geplanten und realisierten Zeiten zeigte jedoch eine leichte Abweichung, was die Zeiten angeht. (Soll-Stunden im Vergleich zu Ist-Stunden)

Projektphase	Soll (Stunden)	Ist (Stunden)	Abweichung (Stunden)
Recherche und Planung	10	11	+1
Entwicklung des Frontends	15	15	
Entwicklung des Backends	35	34	-1
Testen und Dokumentieren	20	20	
Gesamt	**80**	**80**	**0**

Tabelle 5: Soll-/Ist-Vergleich

8.2 Lessons Learned

Durch die Umsetzung des Projekts konnten wertvolle Einblicke in neue Frameworks, Sprachen und Technologien gewonnen werden. Dieses Wissen bildet eine Grundlage für zukünftige Entwicklungen im Berufsleben. Besonders lehrreich war die Erfahrung, dass ein strukturiertes, geplantes Vorgehen deutlich effizienter ist als unkoordiniertes Arbeiten – viele Umwege und Fehler konnten so vermieden werden. Die iterative Projektstruktur erwies sich als vorteilhaft. Fehler wurden früh erkannt, behoben oder sogar verhindert, was unnötige Nacharbeit ersparte. Eine Herausforderung war der Umgang mit mehreren, teils neuen Technologien und Programmiersprachen. Hinzu kam die parallele Arbeit auf zwei Betriebssystemen (Windows und macOS), was Unterschiede in Tastenkombinationen, Zeichensetzung, Shortcuts und Systemverhalten mit sich brachte. Das Lernen durch diese Herausforderungen wird als wertvoller Zuwachs angesehen.

8.3 Ausblick

Erweiterung: Künftig soll die Unterschrift digital erfasst werden. Geplant ist der Einsatz einer externen "Buy" Lösung, die in das bestehende Formular integriert wird. Die Unterschrift kann dann direkt über ein Eingabegerät (z. B. Touchscreen oder Grafiktablett) oder alternativ über einen per QR-Code aufrufbaren Link am Mobilgerät erfolgen.

Quellenverzeichnis

1. Astro Dokumentation

   https://docs.astro.build/en/getting-started/

2. Excalidraw aus Obsidian (Zeichenprogramm)

   https://forum.obsidian.md/t/excalidraw-full-featured-sketching-plugin-in-obsidian/17367

3. Express Dokumentation

   https://expressjs.com/de/

4. Google Suche Programmierung und Methoden:

   https://www.google.com/

5. IHK Logo für das Deckblatt:

   https://www.ihk.de/rheinhessen/

6. IT- Berufe Podcast Seite

   https://it-berufe-podcast.de/vorbereitung-auf-die-ihk-abschlusspruefung-der-it-berufe/

7. MDN Dokumentation

   https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

   https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status#successful_responses

8. Mongo-DB Documentation

   https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/#middleware

9. Mongoose Dokumentation

   https://mongoosejs.com/docs/

10. Thiele und Klose Logo für das Deckblatt und die Seiten:

    https://www.thieleklose.de/

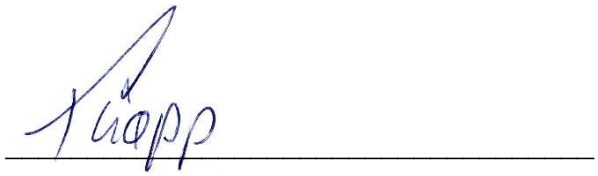
Eidesstattliche Erklärung

Ich, Lukas Trapp, versichere hiermit, dass ich meine Dokumentation zur betrieblichen Projektarbeit mit dem Thema

> *Digitalisierung Anmeldeformular – Optimierung des Anmeldeprozesses für neue Mitglieder des IT-Klub Mainz & Rheinhessen e.V. durch die Digitalisierung des Anmeldeformulars*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Abgabeort: Mainz, den 28.04.2025



Lukas Trapp

Anhang

A1 Detaillierte Zeitplanung

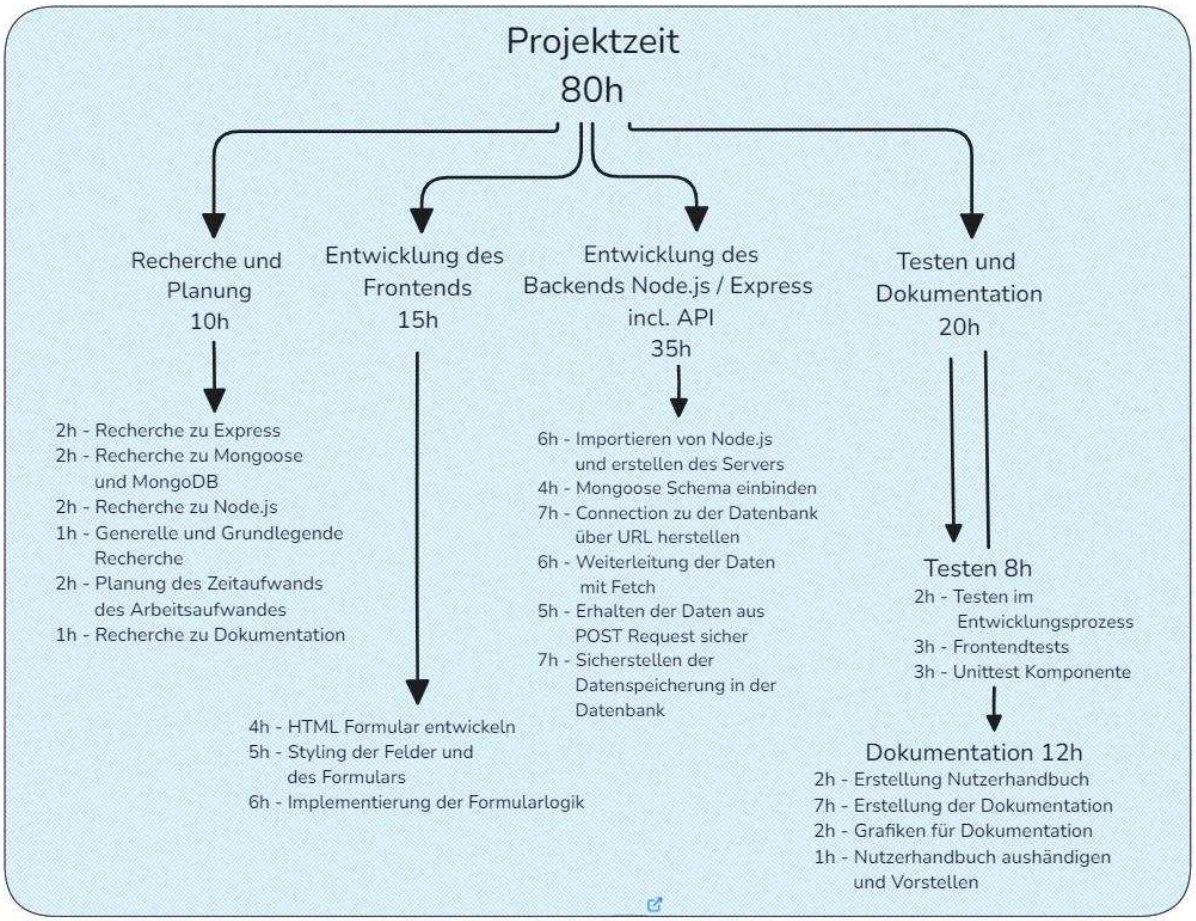


Abbildung 1: Detaillierte Zeitplanung

Recherche und Planung		**10 h**
Recherche zu Express	2 h	
Recherchen in Netz zu Mongoose und Mongo DB	2 h	
Recherche zu Node.js	2 h	
Generelle und Grundlegende Recherchen	1 h	
Planung des Zeitaufwandes des Arbeitsaufwandes und der Schritte	2 h	
Recherche zu Dokumentationen	1 h	

Anhang

Entwicklung des Frontends		**15 h**
Erstellung der Felder und des HML Formulars	4 h	
Stylen der Felder und des HTML Formulars	5 h	
Implementierung der Formularlogik (disabled Submit-Button, required fields…Daten aus dem DOM auslesen und in data Variable speichern)	6 h	
Entwicklung des Backends		**35 h**
Importieren von Node.js und erstellen des Servers	6 h	
Mongoose Schema nach Vorlage des Data-Objekts erstellen und einbinden	4 h	
Connection zu der Datenbank über URL herstellen	7 h	
Weiterleitung der Daten aus dem erstellten data-Objekt aus dem Frontend per Fetch asych gewärleisten.	6 h	
Erhalten der Daten an testendpunkt mittes POST Request mit Postman sicher-stellen.	5 h	
Sicherstellen der Datenspeicherung in die Datenbank. Weiterleiten des Users auf success Seite nach erfolgreichem Submit mit validen Daten gewährleisten.	7 h	
Testen		**8 h**
Zeit für Testen in iterativem Entwicklungsprozess	2 h	
Testen der Validierung im Front wie im Backend	3 h	
Unittest der Komponente an sich	3 h	
Dokumentation		**12 h**
Erstellen des Nutzerhandbuches	2 h	
Erstellen der Projektdokumentation	7 h	
Programmdokumentation und Grafiken erstellen	2 h	
Nutzerhandbuch an den Kunden Aushändigen und vorstellen	1 h	
Gesamtzeit des Projekts		**80 h**

Tabelle 2: Detaillierte Zeitplanung

A2 Kostenaufstellung

Zusammensetzung der Werte für die Beispielrechnung des Projekts (3.2.2 Projektkosten, Tabelle 3 Kostenrechnung):

1. Der Stundenlohn eines erfahrenen externen Mitarbeiters, Ausbilders liegt laut Webrecherchen bei 80-120 €/h daher wird mit einem Mittelwert von 95€ die Stunde gerechnet.

2. Ein realistischer, fiktiverer Stundenlohn für einen Web Entwickler beträgt laut Webrecherche 60-70 €/h, daher wird in der Beispielrechnung von einem Stundenlohn von 65€ ausgegangen.

3. Lohn eines Auszubildenden im zweiten, bzw. dritten Lehrjahr bei je 173 Wochenstunden: 1.338 € / 173 h ≈ 7,73 €/h 1.443 € / 173 h ≈ 8,34 €/h = Mittelwert 8€/h.

Kostenzusammensetzung für MacBook und Strom:

- MacBook Pro (2019), Anschaffungskosten ca. 2.500 €
- LG 38" Curved Monitor, Anschaffungskosten ca. 1.200 €
- Abschreibungszeit: 3 Jahre
- Nutzung: 1.500 Betriebsstunden / Jahr

Rechnung:

$$\frac{3.700€}{3\ \text{Jahre}} = 1.233€ \qquad \frac{1.233\ €}{(1.500\ \text{h/Jahr})} = 0{,}82\ €\ /Stunde$$

Inclusive Strom des MacBooks und des Monitors gehen wir von Kosten von 1 € pro Stunde in der Beispielrechnung aus.

A3 Programmablaufplan

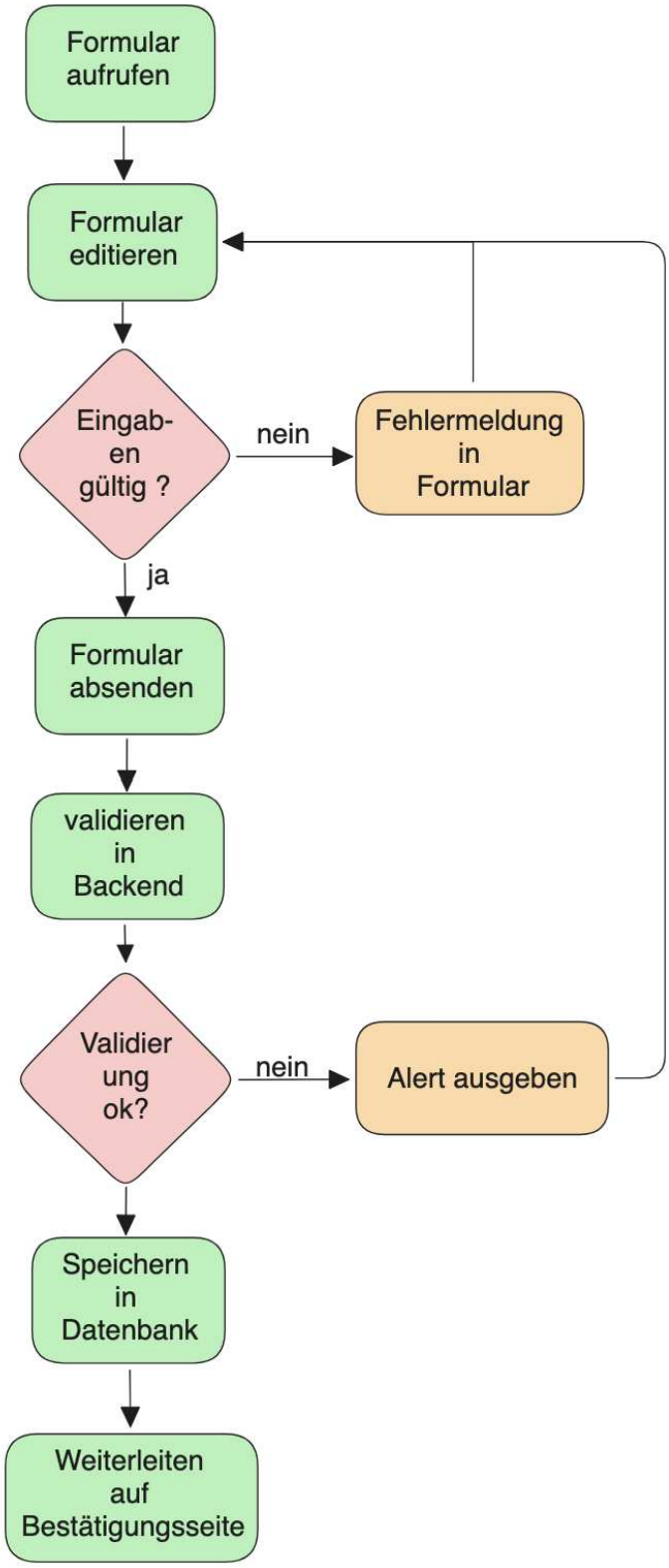


Abbildung 2: Programmablaufplan

A4 Use Case Diagram

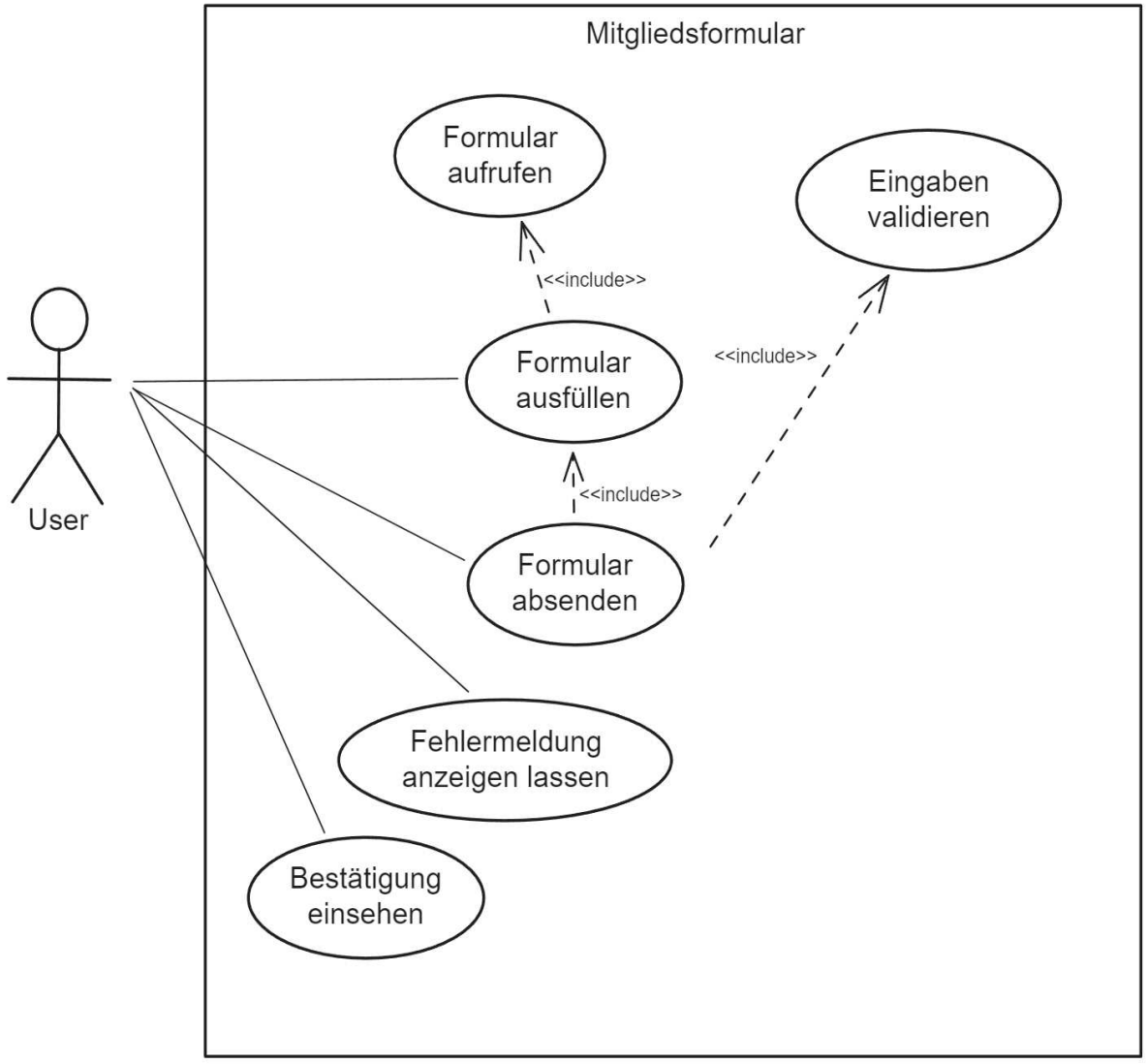


Abbildung 3: Use Case Diagramm

A5 Architektur Design

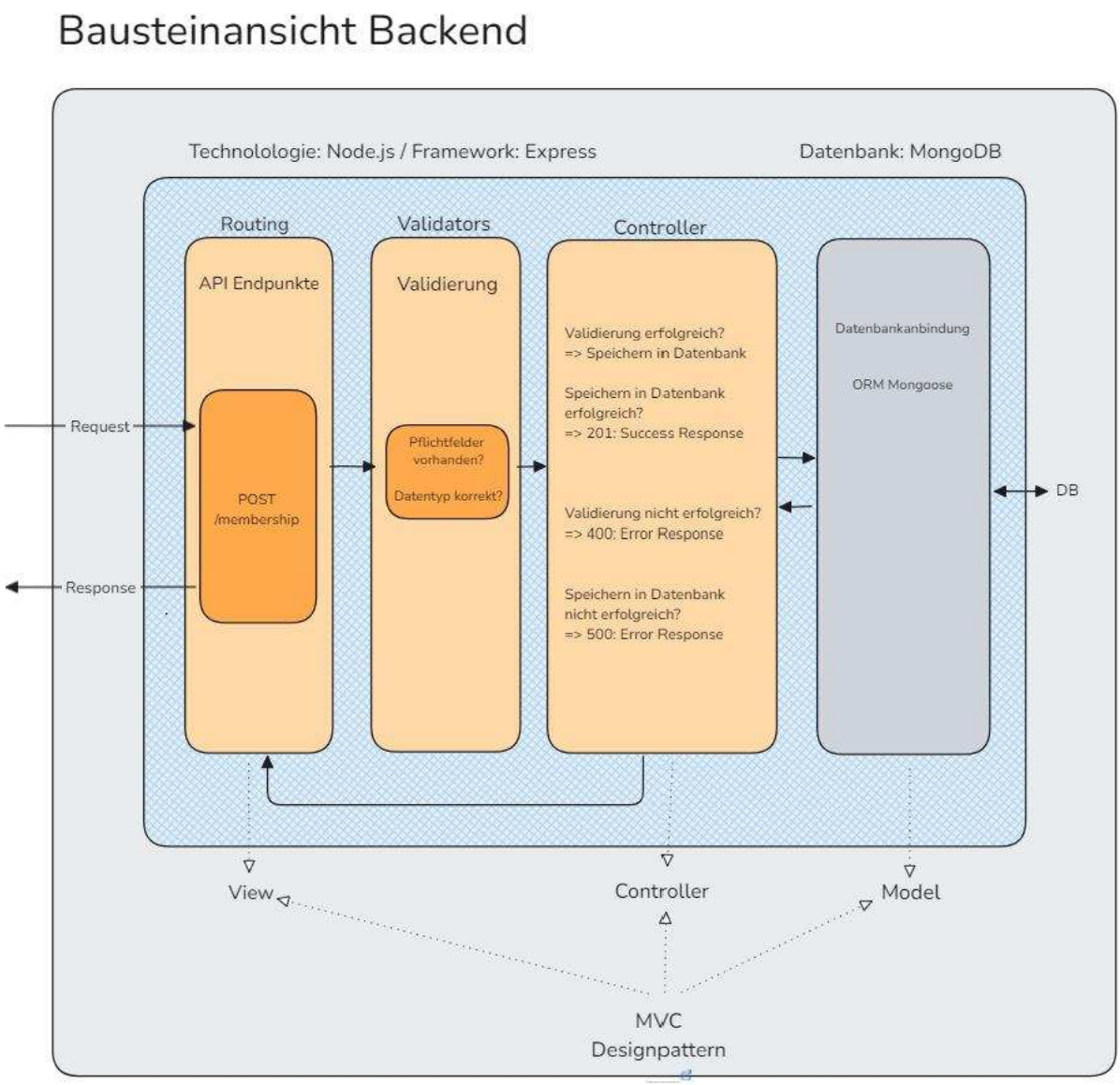


Abbildung 4: Bausteinansicht Backend

A6 Schaubild Datenmodell

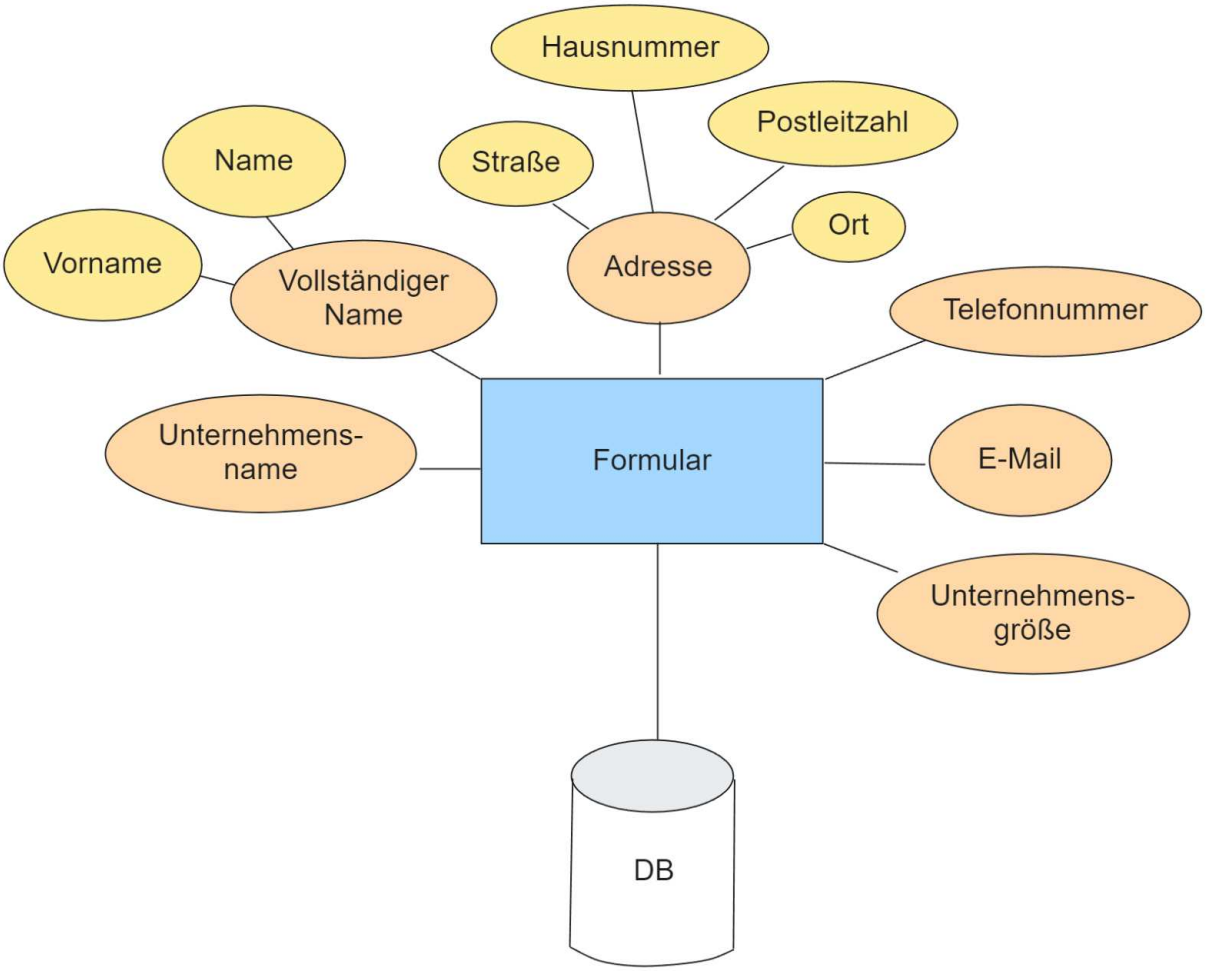


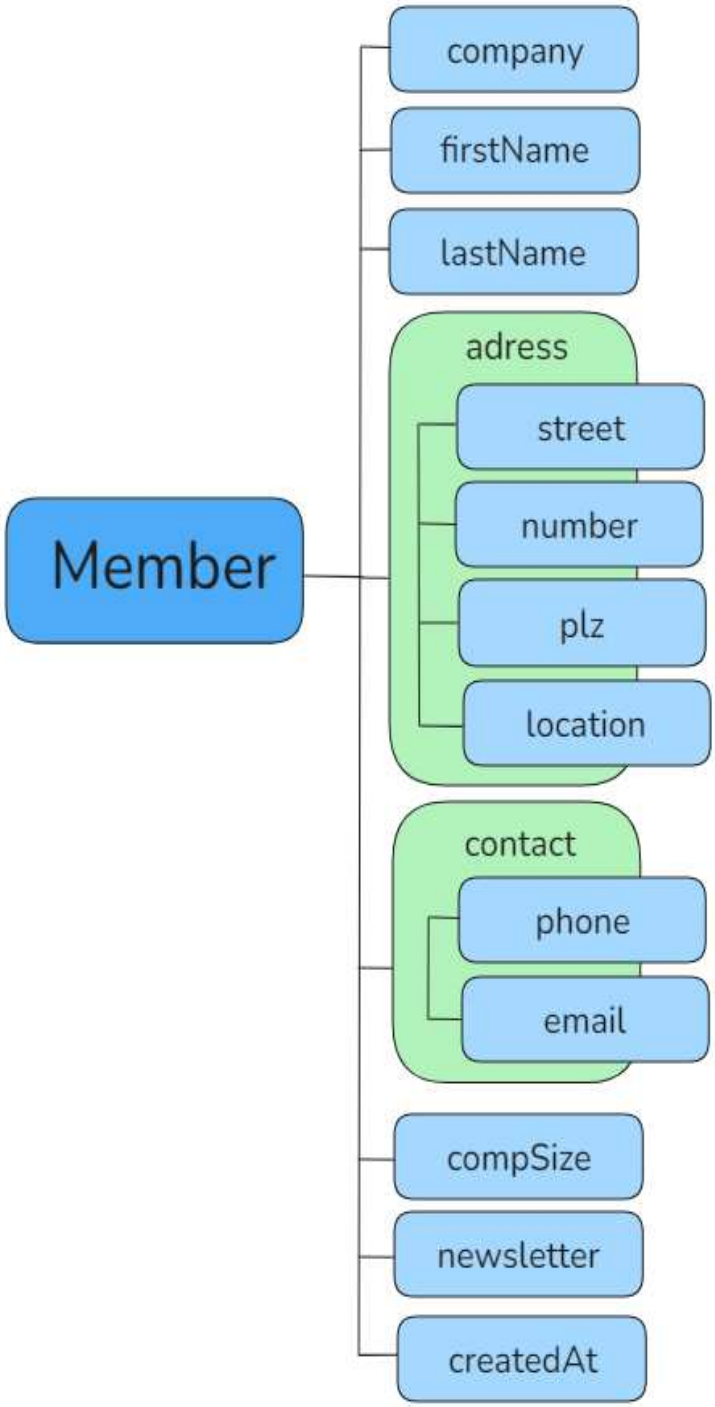
Abbildung 5: Schaubild

A7 Mongoose Schema und Code

project > api > model > JS members.js > ...

You, 4 weeks ago | 1 author (You)

```
1   import mongoose from 'mongoose';
2   const { Schema, model } = mongoose;
3
4   const memberSchema = new Schema({
5       company: {
6           type: String,
7           required: true,
8       },
9       firstName: {
10          type: String,
11          required: true,
12      },
13      lastName: {
14          type: String,
15          required: true,
16      },
17      address: {
18          street: {
19              type: String,
20              required: true,
21          },
22          number: {
23              type: String,
24              required: true,
25          },
26          plz: {
27              type: String,
28              required: true,
29          },
30          location: {
31              type: String,
32              required: true,
33          },
34      },
35      contact: {
36          phone: {
37              type: String,
38              required: true,
39          },
40          email: {
41              type: String,
42              required: true,
43          },
44      },
45      compSize: {
46          type: String,
47          required: true,
48          enum: ['bis10', 'bis50', 'ab50', '
49      },
50      newsletter: {
51          type: Boolean,
52          default: false,
53      },
54      createdAt: {
55          type: Date,
56          default: Date.now,
57      },
58  });
59
60  const Member = model('Member', memberSchema);
61  export default Member;
```



A8 Member-Approved

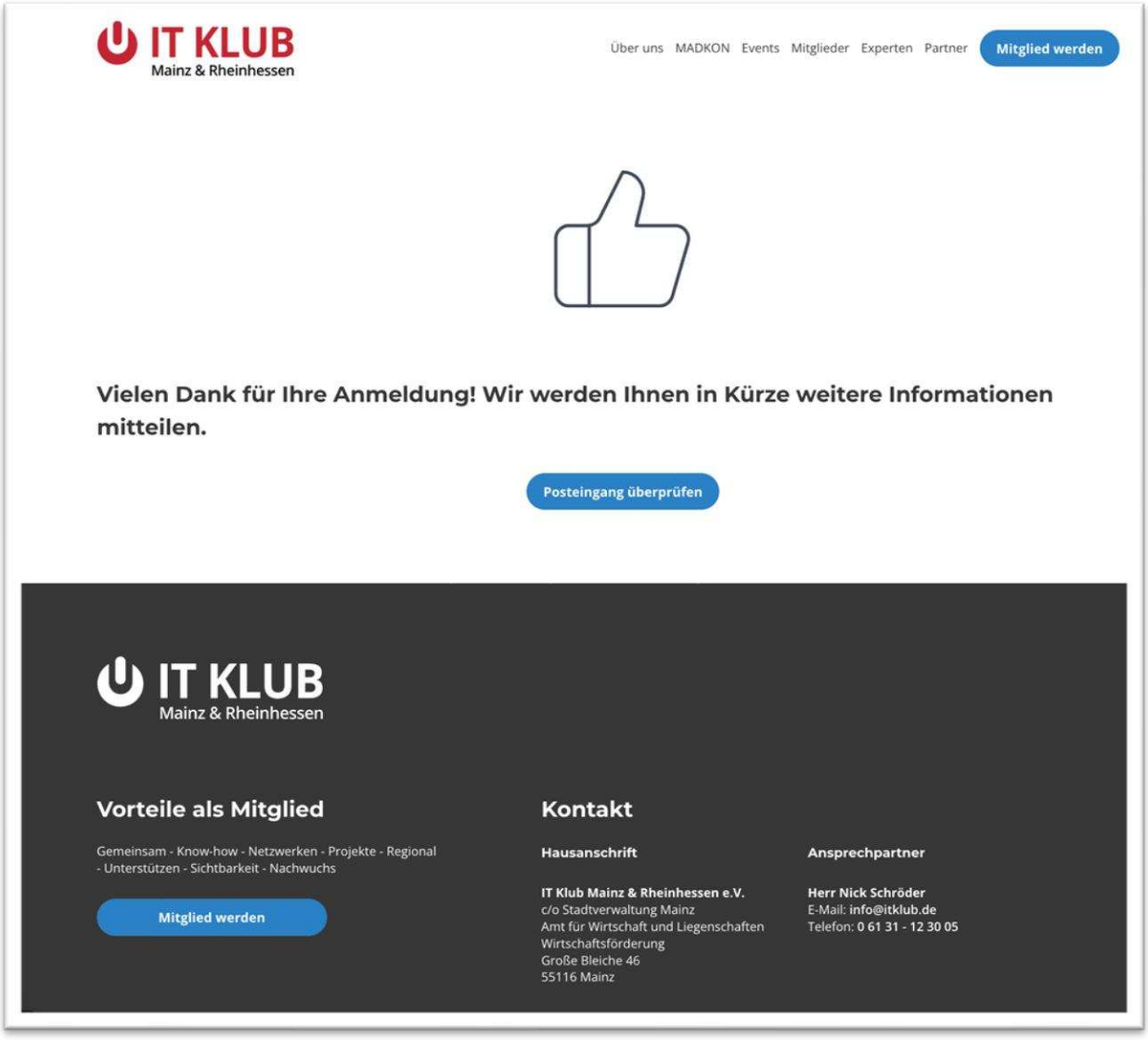


Abbildung 7: member-approved.astro

A9 Mitgliederformular: Desktop

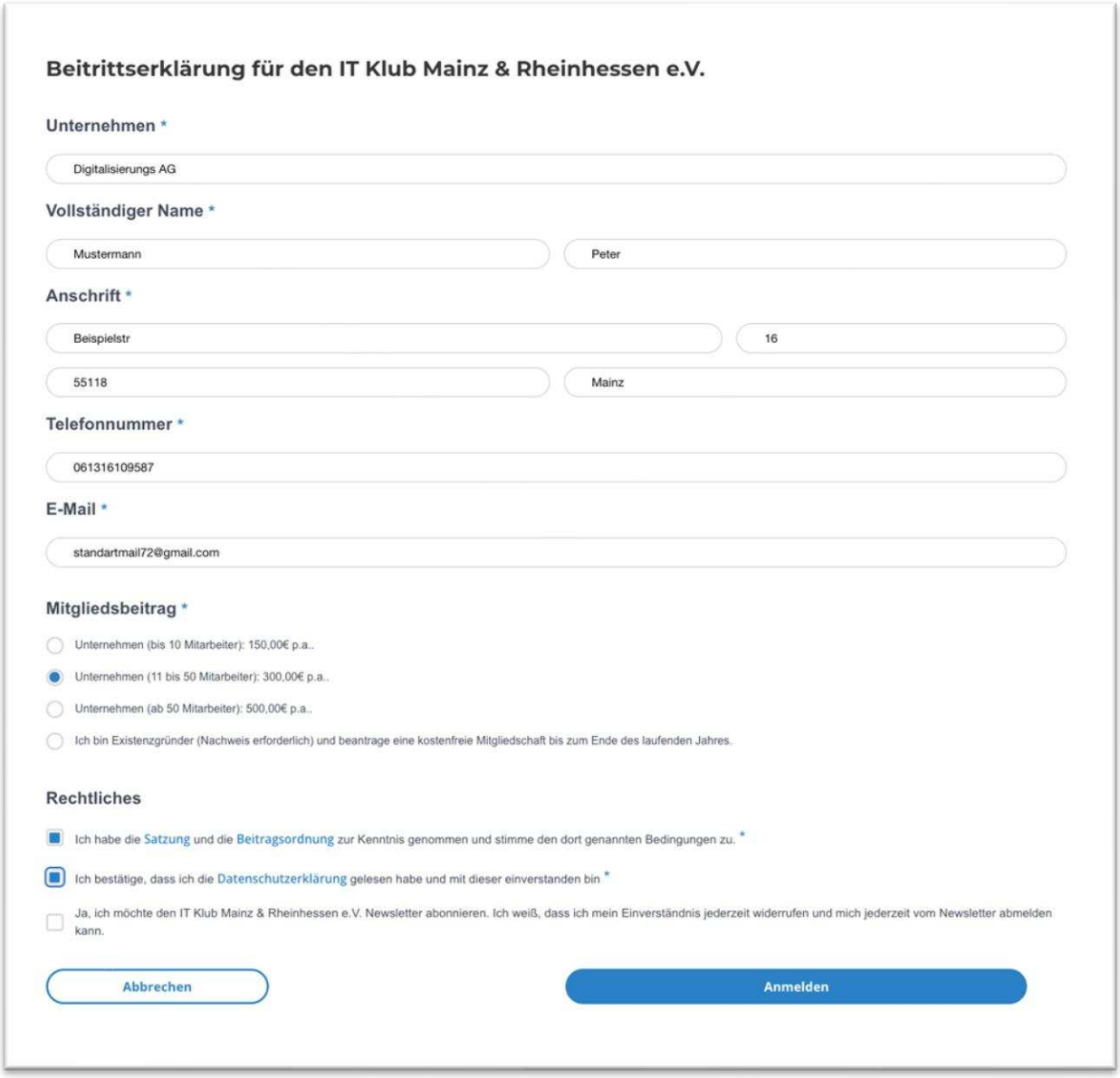


Abbildung 8: Mitgliederformular

A10 Mitgliederformular: Mobil

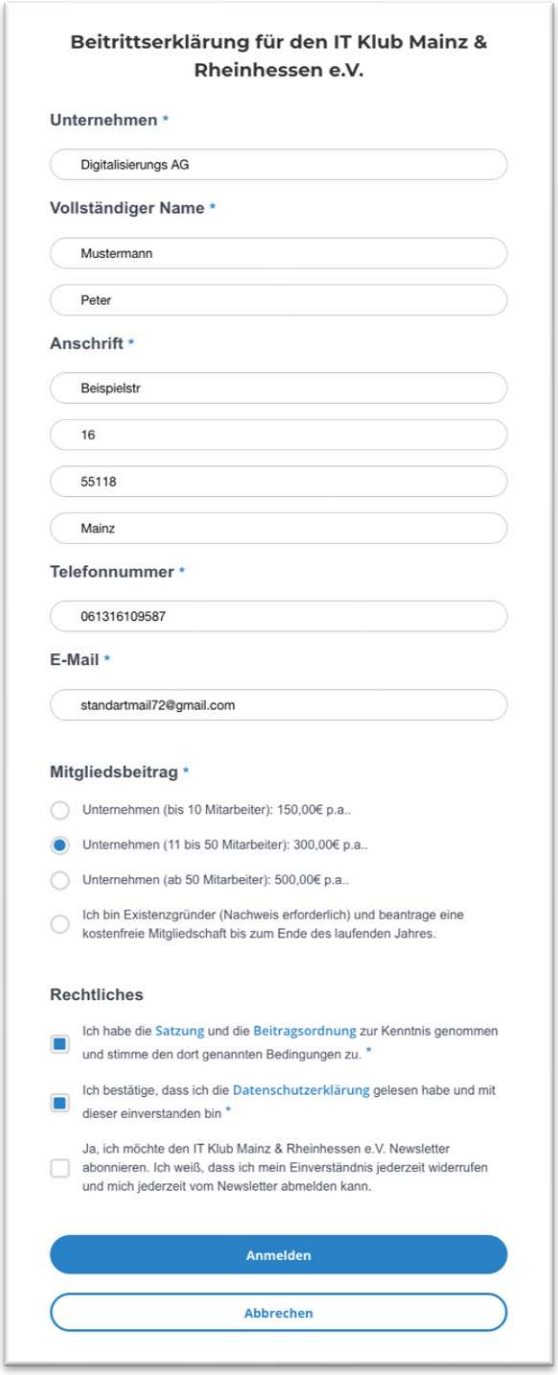


Abbildung 9: Mitgliederformular

A11 Skizzen zum Aufbau

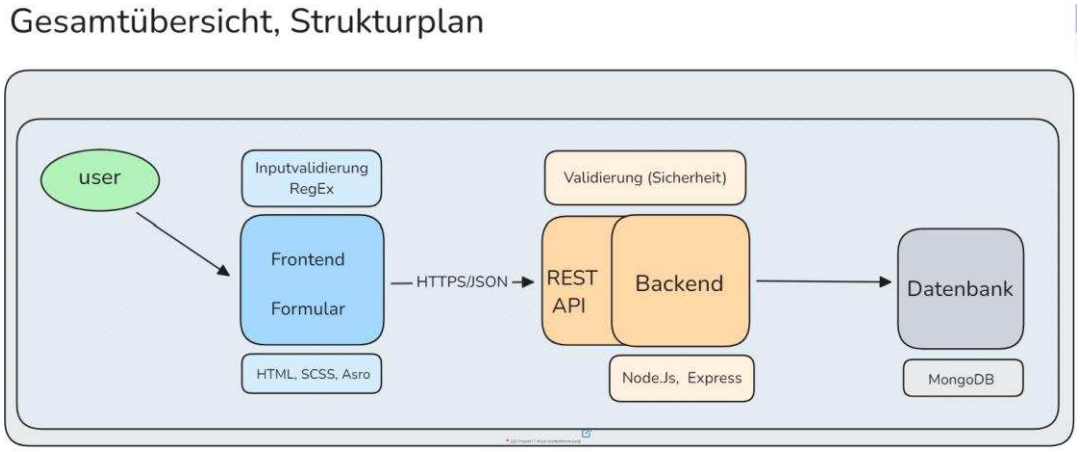


Abbildung 10: Gesamtübersicht

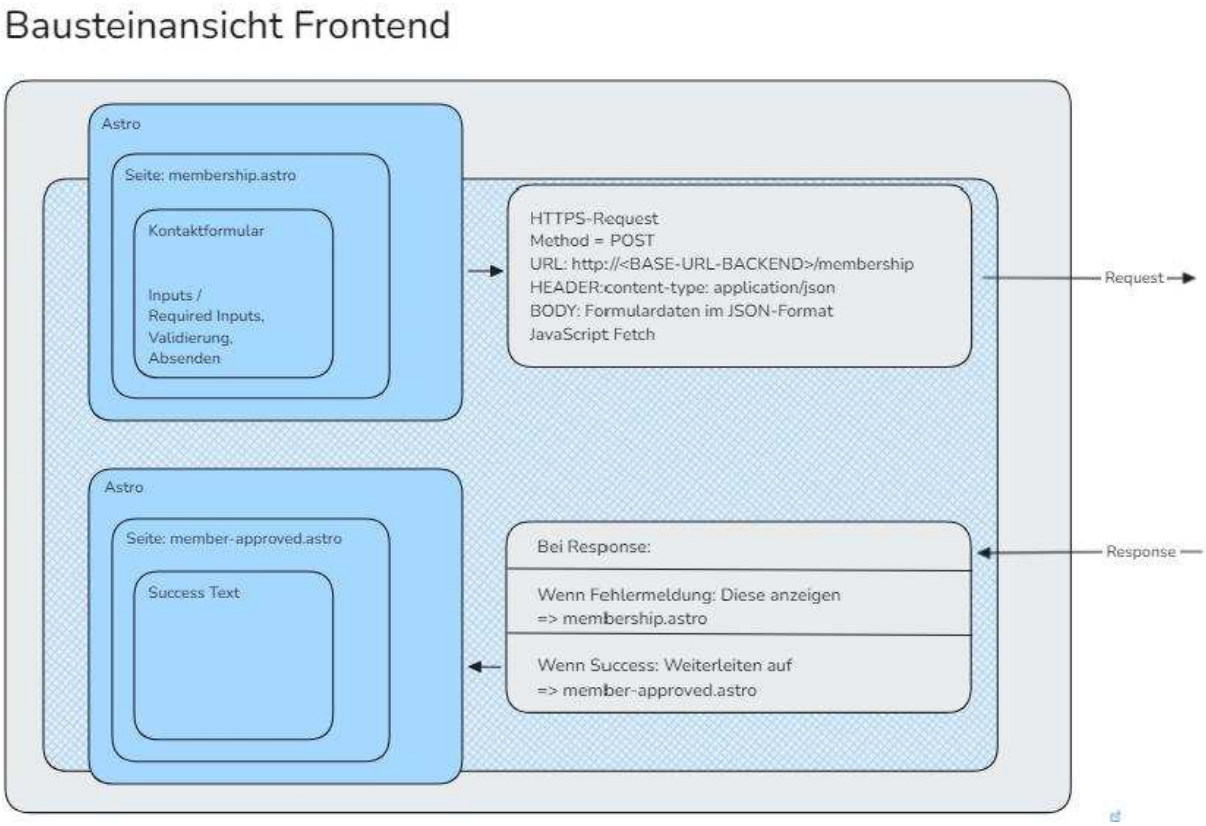


Abbildung 11: Bausteinübersicht Frontend

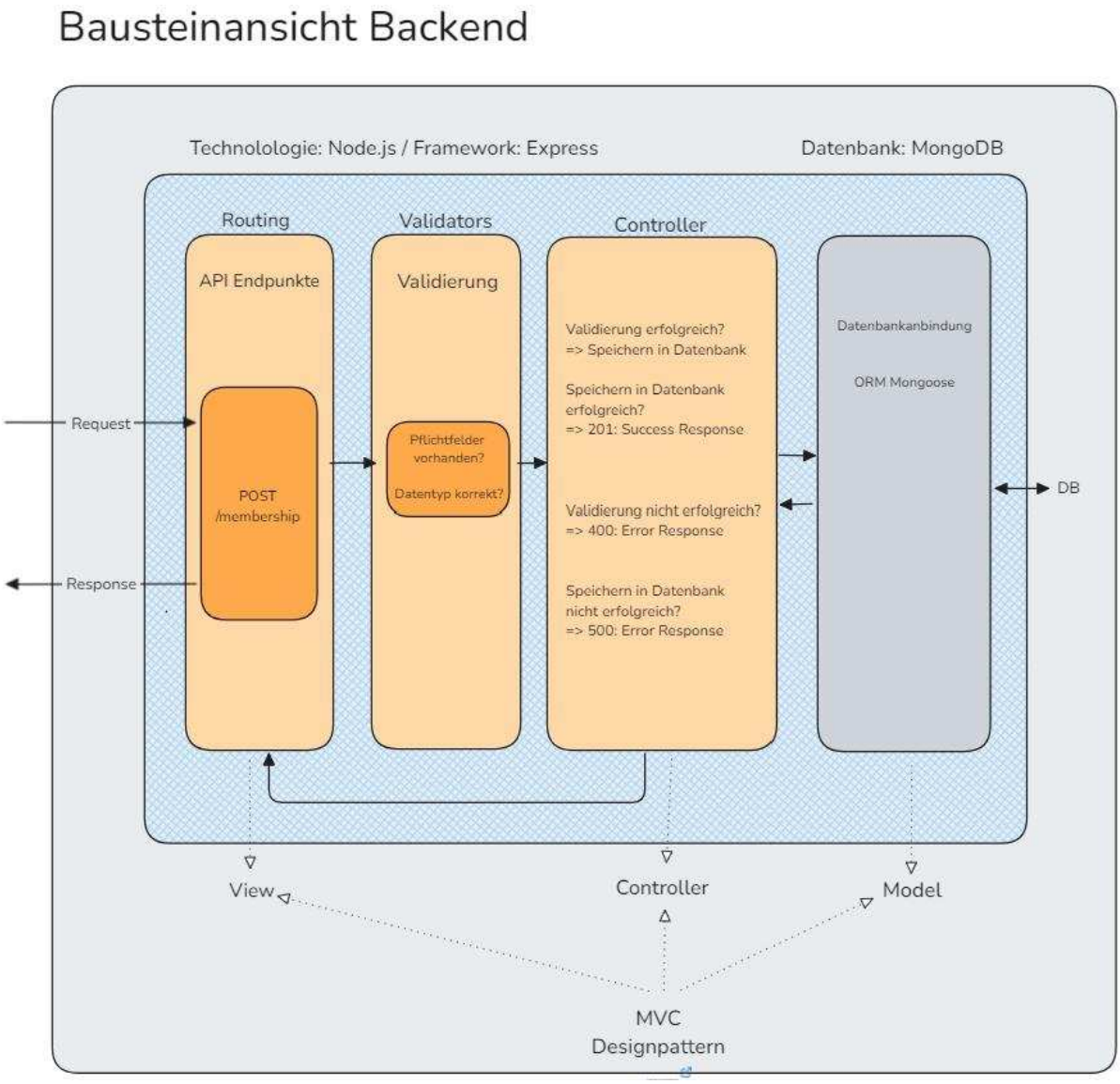


Abbildung 12: Bausteinübersicht Backend

A12 Quellcodeübersicht

Die folgenden Codebeispiele wurden als Screenshots mit Syntax-Highlighting in hellem Design eingebunden, einerseits zur Verbesserung der allgemeinen Lesbarkeit, andererseits zur klaren Abgrenzung zwischen Eigen- und Fremdcode. Eigene Implementierungen sind ausführlich kommentiert, Fremdcode wurde farblich hervorgehoben oder beschrieben.

A12.1 Button-Komponente: Fremdcode (blau hinterlegt, existierte schon), dient der Veranschaulichung der im Formular importierten Komponente Button.

project > web > src > components > elements > Button.astro > ...

You, 2 weeks ago | 4 authors (Abdelhafid Tarazzit and others)

```astro
---
import { ArrowLeft, ArrowRight, Check } from 'lucide-astro';

You, last month | 3 authors (You and others)
interface Props {
    id?: string;
    disabled?: boolean;
    variant:
        | 'default'
        | 'defaultOutline'
        | 'check'
        | 'checkIcon'
        | 'checkOutline'
        | 'checkIconOutline'
        | 'next'
        | 'nextIcon'
        | 'nextIconGrey'
        | 'nextOutline'
        | 'nextIconOutline'
        | 'previous'
        | 'previousIcon'
        | 'previousIconGrey'
        | 'previousOutline'
        | 'previousIconOutline'
        | 'submit'
        | 'submitOutline'
        | 'cancel'
        | 'cancelOutline';
    className?: string;
    ariaLabel?: string;
    href?: string;
    tabIndex?: number;
    type?: 'button' | 'submit' | 'reset';
}

const { variant, disabled, className = '', id, ariaLabel = '', href, tabIndex = '', type = 'button' } = Astro.props;
// href loaded linked in Layout.astro
---

{
    variant === 'default' ? (
        <button
            id={id}
            class={`check-button is-default-button ${className}`}
            aria-label={`${ariaLabel}`}
            disabled={disabled}
            tabindex={`${tabIndex}`}
            data-href={href}
            type={type}
        >
            <span class="text">
                <slot />
            </span>
        </button>
        // .........weiter Buttons

```

A12.2 Layout-Komponente: Fremdcode (blau hinterlegt, existierte schon). Dient der Veranschaulichung der im Formular importierten Komponente Layout. Im Slot des Layouts wurde der gesamte Formularcode an markierter Stelle verankert.

project > web > src > layouts > Layout.astro > ...

You, 2 weeks ago | 5 authors (You and others)

```
1   ---      Tamara Glöckner, 5 months ago • build: initial commit (ITK-88)
    You, 4 months ago | 2 authors (Tamara Glöckner and one other)
2   interface Props {
3       title: string;
4   }
5   import 'bulma/css/bulma.css';
6   import '../../public/fonts/font.css';
7   import '../styles/style.scss';
8   import Navigation from '../components/Navigation.astro';
9   import Footer from './Footer.astro';
10
11  const { title } = Astro.props;
12  ---
13
14  <script>
15      import '../scripts/header';
16      document.addEventListener('DOMContentLoaded', () => {
17          const allButtons = document.querySelectorAll('button[data-href]');
18          // Handle href link see at Button.astro
19          allButtons.forEach((btn) => {
20              const href = btn.getAttribute('data-href');
21              if (href && typeof href === 'string') {
22                  btn.addEventListener('click', () => {
23                      window.location.href = href;
24                  });
25              }
26          });
27      });
28  </script>
29
30  <html lang="de" data-theme="light">
31      <head>
32          <meta charset="UTF-8" />
33          <meta name="description" content="Astro description" />
34          <meta name="viewport" content="width=device-width" />
35          <link rel="icon" type="image/png" href="/assets/images/favicon/favicon-96x96.png" sizes="96x96" />
36          <link rel="icon" type="image/svg+xml" href="/assets/images/favicon/favicon.svg" />
37          <link rel="shortcut icon" href="/assets/images/favicon/favicon.ico" />
38          <link rel="apple-touch-icon" sizes="180x180" href="/assets/images/favicon/apple-touch-icon.png" />
39          <meta name="apple-mobile-web-app-title" content="IT-Klub" />
40          <link rel="manifest" href="/assets/images/favicon/site.webmanifest" crossorigin="use-credentials" />
41          <meta name="generator" content={Astro.generator} />
42          <title>{title}</title>
43      </head>
44      <body>
45          <header class="navbar-sticky" aria-label="Kopfzeile">
46              <Navigation />
47          </header>
48          <main aria-label="Main Inhalt">
49              <div class="main-container">
50                  <slot />
51              </div>
52          </main>
53          <Footer />
54      </body>
55  </html>
```

Anhang

A12.3 Auszug Formular (membership.astro): Die importierten Komponenten Layout und Button (blau markierte Zeilen) waren wie erwähnt bereits vorhanden, sind Fremdcode auf das Projekt bezogen. Die Layout Komponente und der Button wurden hier lediglich eingebunden und angewandt. Das Layout diente wie oben gezeigt als Slot, in dem der Code des Formulars implementiert wurde. (Ende von (</Layout>) auf Grund des Ausschnittes nicht zu sehen.

project > web > src > pages > membership.astro > ...

You, 2 weeks ago | 1 author (You)

```
1  ---
2  import Layout from '../layouts/Layout.astro';      You, 4 months ago • feat: global Layout implemented (itk-116)
3  import Button from '../components/elements/Button.astro';
4  import Checkbox from '../components/elements/Checkbox.astro';
5  import Radio from '../components/elements/Radiobutton.astro';
6  import Alert from '../components/elements/Alert.astro';
7  ---
8
9  <Layout title="Membership">
10   <form class="beitrittsformular">
11     <h1 class="primary-heading first-headline">Beitrittserklärung für den IT Klub Mainz & Rheinhessen e.V.</h1>
12
13     <!-- Eingabefelder -->
14     <div class="adress">
15       <label class="second-headline sub-head" for="unternehmen"
16         >Unternehmen<span class="requires second-headline"> *</span></label
17       >
18       <input
19         type="text"
20         class="box-size primary-text"
21         id="unternehmen"
22         name="unternehmen"
23         placeholder="Unternehmensname"
24         required
25       />
26       <Alert>Geben Sie bitte den Unternehmensnamen an!</Alert>
27
28       <label class="second-headline margin-small" for="name"
29         >Vollständiger Name<span class="requires second-headline"> *</span></label
30       >
31       <div class="fixed-grid has-12-cols mb-0">
32         <div class="grid grid-container">
33           <div class="name-container cell is-col-span-12-mobile is-col-span-12-tablet is-col-span-6-desktop">
34             <div class="alert-container">
35               <input
36                 class="primary-text"
37                 type="text"
38                 id="lastName"
39                 name="lastName"
40                 placeholder="Name"
41                 required
42               />
43               <Alert> Geben Sie bitte Ihren vollständigen Namen an! </Alert>
44             </div>
45           </div>
46           <div class="name-container cell is-col-span-12-mobile is-col-span-12-tablet is-col-span-6-desktop">
47             <div class="alert-container">
48               <input
49                 class="primary-text"
50                 type="text"
51                 id="firstName"
52                 name="firstName"
53                 placeholder="Vorname"
54                 required
55               />
56               <Alert> Geben Sie bitte Ihren vollständigen Namen an! </Alert>
57             </div>
58           </div>
59         </div>
60       </div>
```

Anhang

A12.4 Auszug Formular, Radiobuttons:

```
<div class="radiobutton-container">      You, last month • fix: fields tel and email add in form. (itk-227…
    <label class="second-headline sub-head" for="mitgliedsbeitrag"
        >Mitgliedsbeitrag<span class="requires second-headline"> *</span></label
    >
    <Radio name="membership" id="bis10" value="bis10" required>
        Unternehmen (bis 10 Mitarbeiter): 150,00€ p.a..
    </Radio>
    <Radio name="membership" id="bis50" value="bis50">Unternehmen (11 bis 50 Mitarbeiter): 300,00€ p.a..</Radio>
    <Radio name="membership" id="ab50" value="ab50">Unternehmen (ab 50 Mitarbeiter): 500,00€ p.a..</Radio>
    <Radio name="membership" id="existenzgründer" value="ja"
        >Ich bin Existenzgründer (Nachweis erforderlich) und beantrage eine kostenfreie Mitgliedschaft bis zum
        Ende des laufenden Jahres.</Radio
    >
    <Alert>Wählen Sie bitte eine Option aus!</Alert>
</div>
```

A12.5 Auszug Formular, Checkboxen:

```
<div class="checkbox-container">      You, last month • fix: fields tel and email add in form. (itk-227…
    <label class="second-headline sub-heading-m" for="rechtliches">Rechtliches</label>

    <Checkbox name="Satzung" required>
        Ich habe die
        <a class="neutral-text-link" href="/assets/pdf-files/IT-Klub_Satzung_3-2-2_20170705_FB.pdf">Satzung</a> und
        die
        <a class="neutral-text-link" href="/assets/pdf-files/ITKlub_Beitragsordnung_2-0-0_20180424_MH.pdf"
            >Beitragsordnung</a
        > zur Kenntnis genommen und stimme den dort genannten Bedingungen zu.
    </Checkbox>
    <Alert>Diese Feld ist erforderlich!</Alert>

    <Checkbox name="dataProtection" required
        >Ich bestätige, dass ich die
        <a class="neutral-text-link" href="/assets/pdf-files/Datenschutzerklaerung_IT_Klub.pdf"
            >Datenschutzerklärung</a
        > gelesen habe und mit dieser einverstanden bin</Checkbox
    >
    <Alert>Diese Feld ist erforderlich!</Alert>

    <Checkbox name="newsletter">
        Ja, ich möchte den IT Klub Mainz & Rheinhessen e.V. Newsletter abonnieren. Ich weiß, dass ich mein
        Einverständnis jederzeit widerrufen und mich jederzeit vom Newsletter abmelden kann.
    </Checkbox>
</div>
```

Anhang

A12.6 Radiobutton-Komponente, (ausgelagert):

project > web > src > components > elements > Radiobutton.astro > Props

```
You, 3 weeks ago | 1 author (You)
interface Props {
    value: string;
    id: string;
    name: string;
    required?: boolean;
}       You, last month • feat: a view steps to mobile design (itk-227)

const { value, required = false, id, name } = Astro.props;
---

<div class="options">
    <input type="radio" id={id} name={name} value={value} required />
    <span><slot /></span>
</div>

<style lang="scss">
    You, last month | 1 author (You)
    input[type='radio'] {
        appearance: none;
        min-width: 25px;
        width: 25px;
        height: 25px;
        border: 3px solid var(--secondary-border-grey);
        border-radius: 50%; /* Für radio-Button */
        display: inline-block;
        position: relative;
        cursor: pointer;
        vertical-align: middle; /* Stellt sicher, dass es mit dem Text auf einer Linie bleibt */
    }
    You, last month | 1 author (You)
    input[type='radio']:checked::after {
        content: '';
        position: absolute;
        top: 50%;
        left: 50%;
        width: 15px;
        height: 15px;
        background-color: var(--highlight-blue);
        border-radius: 50%;
        transform: translate(-50%, -50%);
    }
    You, last month | 1 author (You)
    .options {
        margin: 20px 0 0 0;
        display: flex;
        align-items: center;
        gap: 1rem;
    }
</style>
```

Anhang

A12.7 Checkbox-Komponente, (ausgelagert):

project > web > src > components > elements > Checkbox.astro > Props

```
You, last month | 1 author (You)
1   ---
    You, last month | 1 author (You)
2   interface Props {      You, last month • feat: a view steps to mobile design (itk-227)
3       name: string;
4       required?: boolean;
5   }
6
7   const { name, required = false } = Astro.props;
8   ---
9
10  <div class="primary-text options">
11      <input type="checkbox" name={name} required={required} role="checkbox" tabindex="0" aria-checked="false" />
12      <span>
13          <slot />
14          {required && <span class="requires second-headline"> *</span>}
15      </span>
16  </div>
17
18  <style lang="scss">
        You, last month | 1 author (You)
19      input[type='checkbox'] {
20          appearance: none;
21          min-width: 25px;
22          width: 25px;
23          height: 25px;
24          border: 3px solid var(--secondary-border-grey);
25          border-radius: 5px;
26          display: inline-block;
27          position: relative;
28          cursor: pointer;
29          vertical-align: middle;
30      }
        You, last month | 1 author (You)
31      input[type='checkbox']:checked::after {
32          content: '';
33          position: absolute;
34          top: 50%;
35          left: 50%;
36          width: 15px;
37          height: 15px;
38          background-color: var(--highlight-blue);
39          border-radius: 2px;
40          transform: translate(-50%, -50%);
41      }
        You, last month | 1 author (You)
42      .options {
43          margin: 20px 0 0 0;
44          display: flex;
45          align-items: center;
46          gap: 1rem;
47      }
        You, last month | 1 author (You)
48      .requires {
49          color: var(--highlight-blue);
50      }
51  </style>
```

A12.8 Daten auslesen und bündeln: Auslesen der Werte aus dem DOM bzw. den Formularfeldern und Zwischenspeicherung in einzelnen Konstanten. Anschließend Zusammenführung dieser Werte in einem konstanten Objekt (data) zur strukturierten Weitergabe an das Backend.

```
// generate data object from inputs
const companyElement: HTMLInputElement | null = document.querySelector('input[name="unternehmen"]');
const companyValue = companyElement?.value;

const firstNameElement: HTMLInputElement | null = document.querySelector('input[name="firstName"]');
const firstNameValue = firstNameElement?.value;

const lastNameElement: HTMLInputElement | null = document.querySelector('input[name="lastName"]');
const lastNameValue = lastNameElement?.value;

const streetElement: HTMLInputElement | null = document.querySelector('input[name="strasse"]');
const streetValue = streetElement?.value;

const numberElement: HTMLInputElement | null = document.querySelector('input[name="hausnr"]');
const numberValue = numberElement?.value;

const plzElement: HTMLInputElement | null = document.querySelector('input[name="plz"]');
const plzValue = plzElement?.value;

const locationElement: HTMLInputElement | null = document.querySelector('input[name="ort"]');
const locationValue = locationElement?.value;

const phoneElement: HTMLInputElement | null = document.querySelector('input[name="telefon"]');
const phoneValue = phoneElement?.value;

const emailElement: HTMLInputElement | null = document.querySelector('input[name="email"]');
const emailValue = emailElement?.value;

const compSizeElement: HTMLInputElement | null = document.querySelector('input[name="membership"]:checked');
const compSizeValue = compSizeElement?.value;

const data = {
    company: companyValue,
    firstName: firstNameValue,
    lastName: lastNameValue,
    street: streetValue,
    number: numberValue,
    plz: plzValue,
    location: locationValue,
    phone: phoneValue,
    email: emailValue,
    compSize: compSizeValue,
};
```

A12.9 Daten fetchen in sendData(): Die asynchrone Funktion sendData() versucht im try-Block, mithilfe der fetch()-Funktion die im Objekt data gespeicherten Formulardaten – nach dem Parsen mit der .stringify()-Methode – per POST-Request im Body an den in der Konstanten url hinterlegten API-Endpunkt (http://localhost:3003/membersform) zu senden. Dem Request wird ein Header mitgegeben, der den Inhalt des Bodys als Datentyp application/json ausweist. Dadurch erkennt der Server, dass es sich bei der Anfrage an die API um JSON-Daten handelt.

Bevor die Response aus dem Promise fetch() geprüft werden kann, wird durch das await der Programmfluss angehalten, bis sich das Promise aufgelöst hat. So wird sichergestellt, dass bei etwaiger Latenz die Response auch wirklich vorhanden ist, bevor sie weiterverarbeitet wird.

Sollte der Server mit einem 200er-Code antworten (response.ok), wird der Benutzer automatisch zur Bestätigungsseite (const href = "http://localhost:4321/member-approved") weitergeleitet.

Falls ein Fehler auftritt – etwa bei einem 404-Fehler (Server nicht erreichbar) oder einem 500-Fehler (ungültige Antwort) – wird eine neue Instanz vom Typ Error erzeugt und geworfen.

Der catch-Block fängt diesen Fehler ab und verarbeitet ihn weiter, sodass sowohl eine Fehlermeldung in der Konsole (z. B. in den Developer Tools unter "Console") als auch ein alert-Fenster im Browser (Popup) angezeigt wird. Dieser visuelle Hinweis ist wichtig, damit der Nutzer unmittelbar über den Fehler informiert wird und kein für ihn stummer Fehler entsteht, er nicht in einen nicht reagierenden Zustand gerät.

Anhang

```
const href = defUrl + '/member-approved';
const url = apiUrl + '/membersform';

async function sendData(url: string, data: any, href: string) {
    try {
        const response = await fetch(url, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(data),
        });

        if (!response.ok) {
            const errorData = await response.json();

            if (response.status === 400) {
                alert(errorData.message || 'Ungültige Eingaben.');
            }

            if (response.status === 409) {
                alert(errorData.error || 'Ein Mitglied mit dieser E-Mail existiert bereits');
                return;
            }

            if (response.status === 500) {
                alert(
                    errorData.error ||
                        'Unerwarteter Serverfehler. Bitte versuchen sie es zu einem späterem Zeitpunkt erneut.'
                );
            }
            throw new Error(`Fehler: ${response.status} - ${response.statusText}`);
        }
        window.location.href = href;
    } catch (error) {
        console.error('Fehler beim Senden der Daten:', error);
        alert('Daten konnten nicht gesendet werden. Bitte versuche es erneut.');
    }
}
```

Aufruf von sendData() mit Übergabeparametern.

```
    sendData(url, data, href);
```

Um den fetch abhandeln zu können muss natürlich der Server laufen, welcher folgend beschrieben wird.

A12.10 Beschreibung des Servers Index.js unter Node.js: Nach den Importen, darunter auch Fremdcode des Middleware-Frameworks express sowie des ODM mongoose (blau hervorgehoben), folgt der Initialisierungsteil. Dort werden Konstanten wie port und app festgelegt. Die Konstante app ist dabei eine Instanz des Express-Frameworks. Direkt danach wird connectToDatabase() aus der Datei connection.js aufgerufen, um eine Verbindung zur Datenbank herzustellen. Da diese Funktion asynchron ist, wartet der Code, bis das Promise (aus mongoose.connect) erfüllt ist, erst danach wird der Programmfluss fortgesetzt.

Anschließend wird app, basierend auf zuvor importierten Modulen mit der Middleware cors() (aus dem gleichnamigen Paket cors) und express.json() (aus dem Framework express) erweitert. Dadurch werden Cross-Origin-Requests erlaubt und JSON-Daten können im Request-Body automatisch verarbeitet werden.

Es folgen zwei definierte Endpunkte: /api dient als einfacher Test-Endpunkt (z. B. zur Überprüfung mit Postman) und /membersform als produktiver POST-Endpunkt, der die Funktion postMembersRequest aus der Datei membersController.js ausführt.

Zum Schluss wird mit app.listen() der Server auf dem angegebenen Port gestartet. Bei erfolgreichem Start erscheint eine Bestätigungsmeldung in der Konsole.

Kurz gesagt: Wenn alle Importe korrekt geladen wurden, die Initialisierung abgeschlossen wurde, die Datenbankverbindung hergestellt und der Server gestartet ist, kann ein POST-Request an /membersform verarbeitet werden. Der Request wird dabei validiert und nach dem vorgegebenen Schema in der Datenbank gespeichert. Bei erfolgreichem Ablauf erhält der Benutzer eine Response und wird auf die Success-Seite weitergeleitet.

Ein Auszug des Codes (index.js) folgt auf der nächsten Seite.

Anhang

project > api > JS index.js > ...

You, 7 days ago | 2 authors (You and one other)

```
1  import cors from 'cors';        You, last month • fieat: stand 12.03.2025 (itk-227)
2  import 'dotenv/config';
3  import express from 'express';
4  import membersController from './controller/membersController.js';
5  import connectToDatabase from './model/connection.js';
6
7  // Set port and initialize express app instance / object
8  const port = 3003;
9  const app = express();
10
11 // Initialize database connection (async, with awaite promise)
12 connectToDatabase();
13
14 // Enable Cross-Origin Resource Sharing / CORS for external requests (z.B. from frontend)
15 app.use(cors());
16
17 // Parse incoming JSON requests
18 app.use(express.json());
19
20 // Endpoint to check if server is running, connection workes (z.B. via Postman)
21 app.get('/api', (req, res) => {
22     res.send('API running');
23 });
24
25 // POST endpoint to handle incoming member form data and create a new database entry
26 app.post('/membersform', membersController.postMembersRequest);
27
28 // Start server and listen on port
29 app.listen(port, () => {
30     console.log(`API running on port ${port}`);
31 });
```

A12.11 Aufbau der Verbindung zur Datenbank: In der Datei connection.js wird zunächst das Paket mongoose importiert (Fremdcode, blau hinterlegt). Anschließend wird eine Konstante namens connectToDatabase deklariert. Diese enthält eine anonyme, asynchrone Funktion, die mithilfe von await über die URL, die in der Umgebungsvariable gespeichert ist, versucht, eine Verbindung zur MongoDB-Datenbank herzustellen. Ist die Verbindung erfolgreich, wird in der Konsole die Meldung "connected to database" ausgegeben, tritt beim Verbindungsaufbau ein Fehler auf, wird dieser im catch-Block abgefangen und ebenfalls in der Konsole ausgegeben. Am Ende wird die Funktion mit export default exportiert, wodurch sie

in anderen Dateien importiert und verwendet werden kann. Da es sich um einen Default-Export handelt und nicht um einen Named Export, kann die Funktion beim Import unter einem beliebigen Namen eingebunden werden.

A12.12 Die Funktion postMembersRequest: Die in der Konstante membersController gespeicherte asynchrone Funktion postMembersRequest wird getriggert, wenn eine Anfrage über den in der Datei index.js angegebenen POST-Endpunkt /membersform gesendet wird. Anschließend wird die Response über eine anonyme Funktion gesteuert. Diese speichert die aus dem Request empfangenen Daten (req.body) in der Konstante data. Das Ergebnis der Funktion validateMember(data) wird in der Konstante validated gespeichert. Wenn bei der Prüfung des anschließenden if-Blocks validated den Wert false hat, wird eine Response mit dem Statuscode 400 und einer entsprechenden Message im JSON-Format zurückgegeben. Ist dies nicht der Fall, läuft der Code weiter, und eine Instanz des Member-Objekts wird erzeugt und der Konstante newMember zugewiesen. Dabei werden die empfangenen und validierten Daten in die vom Schema wie im folgenden Screenshot zu sehen vorgegebene Datenstruktur eingefügt bzw. zugewiesen.

Daraufhin wird mit dem await-Schlüsselwort auf das Promise für das Speichern des newMember-Objekts gewartet. Hat dieser Vorgang funktioniert, wird über die Response der Statuscode 201 (success) sowie das entsprechende JSON, wie im Code zu sehen, zurückgegeben. Sollte beim Speichern ein Fehler auftreten, wird dieser vom catch-Block abgefangen und mit console.error('Fehler beim Speichern des Mitglieds', error) in der Konsole ausgegeben. Ebenso wird eine Response mit dem Statuscode 500 sowie einem JSON mit den entsprechenden Informationen zurückgegeben.

Diese Response wird an den Aufrufer, die Funktion sendData() zurückgegeben, um im Frontend in der GUI das Ergebnis des Vorgangs darzustellen. Im Erfolgsfall gelangt der User auf die Bestätigungsseite members-approved.astro. Ist die Response nicht erfolgreich, wird dem Benutzer in der GUI ein Fehler-Alert angezeigt. Der Fehler wird außerdem wie in der sendData-Funktion in der Konsole geloggt.

Im dargestellten membersController wurden zur Umsetzung externe Frameworks verwendet. So stammen Funktionen wie findOne() und save() aus dem Mongoose-Framework.

Anhang

Codeauszug: postMembersRequest:

> api > controller > JS membersController.js > ...

```
You, 35 minutes ago | 1 author (You)
import Member from '../model/members.js';
import validateMember from '../validation/membersValidator.js';

const membersController = {
    // Receive data and validate it; return 400 if invalid.
    postMembersRequest: async (req, res) => {
        try {
            const data = req.body;
            const validated = validateMember(data);
            if (!validated) {
                return res.status(400).json({
                    message: 'invalide member attributes',
                });
            }

            // Check if a member with this email already exists; return 409 Conflict if found.
            data.email = data.email.toLowerCase().trim();
            const existMember = await Member.findOne({ 'contact.email': data.email });
            if (existMember) {
                return res.status(409).json({ error: 'Mitglied mit dieser E-Mail existiert bereits.' });
            }

            // Create new member object based on validated data.
            const newMember = new Member({
                company: data.company,
                firstName: data.firstName,
                lastName: data.lastName,
                address: {
                    street: data.street,
                    number: data.number,
                    plz: data.plz,
                    location: data.location,
                },
                contact: {
                    phone: data.phone,
                    email: data.email,
                },
                compSize: data.compSize,
                newsletter: data.newsletter || false,
            });

            await newMember.save();

            // succes
            res.status(201).json({
                success: true,
                message: 'Mitgliedschaft erfolgreich gespeichert',
                data: newMember,
            });
        } catch (error) {
            console.error('Fehler beim Speichern des Mitglieds:', error);
            res.status(500).json({
                success: false,
                message: 'Fehler beim Speichern der Daten',
                error: error.message,
            });
        }
    },
};

export default membersController;
```

A12.13 Mongoose-Schema: Erstellung einer Schema-Instanz durch Mongoose-Konstruktor.

project > api > model > JS members.js > ...

You, 4 weeks ago | 1 author (You)

```
1   import mongoose from 'mongoose';
2   const { Schema, model } = mongoose;
3
4   const memberSchema = new Schema({
5       company: {
6           type: String,
7           required: true,
8       },
9       firstName: {
10          type: String,
11          required: true,
12      },
13      lastName: {
14          type: String,
15          required: true,
16      },
17      address: {
18          street: {
19              type: String,
20              required: true,
21          },
22          number: {
23              type: String,
24              required: true,
25          },
26          plz: {
27              type: String,
28              required: true,
29          },
30          location: {
31              type: String,
32              required: true,
33          },
34      },
35      contact: {
36          phone: {
37              type: String,
38              required: true,
39          },
40          email: {
41              type: String,
42              required: true,
43          },
44      },
45      compSize: {
46          type: String,
47          required: true,
48          enum: ['bis10', 'bis50', 'ab50', 'ja'],
49      },
50      newsletter: {
51          type: Boolean,
52          default: false,
53      },
54      createdAt: {
55          type: Date,
56          default: Date.now,
57      },
58  });
59
60  const Member = model('Member', memberSchema);
61  export default Member;
```

A12.14 Validierung der Eingabedaten mit membersValidator.js

project > api > validation > JS membersValidator.js > ...

You, 3 hours ago | 1 author (You)

```
const CompanySize = {      You, 4 weeks ago • feat: functionabitity given (itk-227) …
    BIS_10: 'bis10',
    BIS_50: 'bis50',
    UEBER_50: 'ab50',
    GRUENDER: 'ja',
};

const allowedSizes = Object.values(CompanySize);

const validateMember = (data) => {
    if (typeof data.company !== 'string' || data.company.length < 1) {
        return false;
    }
    if (typeof data.firstName !== 'string' || data.firstName.length < 1) {
        return false;
    }
    if (typeof data.lastName !== 'string' || data.lastName.length < 1) {
        return false;
    }
    if (typeof data.street !== 'string' || data.street.length < 1) {
        return false;
    }
    if (typeof data.number !== 'string' || data.number.length < 1 || data.number.length > 5) {
        return false;
    }
    if (typeof data.plz !== 'string' || !/^\d{5}$/.test(data.plz)) {
        return false;
    }
    if (typeof data.location !== 'string' || data.location.length < 1) {
        return false;
    }
    if (typeof data.phone !== 'string' || data.phone.length < 5) {
        return false;
    }
    if (typeof data.email !== 'string' || !/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/.test(data.email)) {
        return false;
    }
    if (typeof data.compSize !== 'string' || !allowedSizes.includes(data.compSize)) {
        return false;
    }
    return true;
};

export default validateMember;
```

A13 Benutzerhandbuch (Auszug)

Benutzerhandbuch – Digitales Anmeldeformular

1. Ziel:

Mit diesem Formular können sich neue Mitglieder des IT-Klub Mainz & Rheinhessen e.V. bequem digital anmelden.

2. Zugriff:

Sie können das Formular über 2 Wege erreichen. Einmal über den im Navigationsmenü rechts oben angezeigten Button, ,,Mitglied werden".

Screenshot 1

2.1 Mobile Version

Sollten sie sich in der mobilen Version unserer Seite befinden öffnen sie rechts oben das Burgermenü (Drei Striche, in Screenshot 2 zu sehen). Dort finden sie wieder den Menüpunkt, Mitglied werden. (Screenshot 3)

Screenshot 2

Screenshot3

Anhang

2.2 Über den Footer am Ende jeder Seite

Die zweite Möglichkeit finden sie im Footer (Der graue Bereich ganz unten auf der Seite. Diesen finden sie ebenso auf jeder unserer Unterseiten), auch hier ist der Button wählbar ,,Mitglied werden''.

Screenshot 4

Alternativ ist das Formular zusätzlich über folgenden Link erreichbar:

https://www.itklub.de/membership

3. Systemanforderungen:

- Ein aktueller Webbrowser (z. B. Chrome, Firefox, Safari)
- Eine Internetverbindung

4. Bedienung:

4.1 Öffnen Sie die Seite des Mitgliederantrages wie in Punkt ,,2. Zugriff '' beschrieben.

4.2 Füllen Sie alle Pflichtfelder (mit * gekennzeichnet) aus:

- Unternehmen*
- Vollständiger Name*
- Anschrift*
- Telefonnummer*
- E-Mail*
- Mitgliedbeitrag* (hier kann nur eine Auswahl getroffen werden.)
- In der Rubrik Rechtliches muss bei der Satzung/ Beitragsordnung und bei Datenschutz die Auswahl angeharkt werden, nach dem diese zur kenntniss genommen wurden. Das Abonnieren des Newsletters ist freiwillig und kann später jederzeit widerrufen werden.

4.3 Überprüfen Sie Ihre Angaben.

4.4 Klicken Sie auf „Absenden".

4.5 Bei Erfolg erscheint eine Bestätigung auf der Seite, auf dieser könne sie auch direkt ihre Bestätigungsmail über einen Link einsehen. Diese sollte nach wenigen Minuten in Ihrem Postfach erscheinen.

Hinweise:

- Die Daten werden verschlüsselt über HTTPS übertragen.
- Pflichtfelder müssen ausgefüllt werden, ansonsten erscheint eine Fehlermeldung.
- Bei Fragen oder technischen Problemen wenden Sie sich bitte per mail an: info@itklub.de

Datenschutz:

Die übermittelten Daten werden ausschließlich zur Vereinsverwaltung verwendet und nicht an Dritte weitergegeben.

IHK Rheinhessen

Bei Abgabe der Dokumentation für IT- Berufe sind diese Erklärungen <u>jeder Dokumentation</u> anzuhängen:

1. Erklärung des Prüfungsteilnehmers / der Prüfungsteilnehmerin

Eidesstattliche Erklärung des Prüfungsteilnehmers / der Prüfungsteilnehmerin:

Den in der Themenstellung beschriebenen Auftrag/Teilauftrag habe ich eigenständig in der beschriebenen Art und Weise durchgeführt. Die Dokumentation wurde ohne fremde Hilfe verfasst. Ich habe mich keiner anderen als der von mir angegebenen Hilfsmittel bedient.

Mainz 24.4.2025 .....
Ort und Datum

.....
Unterschrift des Prüfungsteilnehmers

2. Erklärung des Ausbildungsbetriebes / Praktikumbetriebes

Erklärung des Prüfungsbetriebes:

Hiermit wird bestätigt, dass der Auftrag/Teilauftrag der Themenstellung so durchgeführt wurde, wie er in der Dokumentation beschrieben wurde.

Mainz 24.04.2025 .....
Ort und Datum

THIELE KLOSE
Thiele & Klose GmbH | Ingolheimer Straße 13 | 55442 Dexheim
Telefon: +49 6724 2063702 | E-Mail: info@thieleundklose.de
.....
Stempel und Unterschrift des Prüfungsbetriebes

Hinweise/Checkliste zur Dokumentation

Dokumentationsarten	Projektdokumentation unterschieden nach: 1. Projektdokumentation 2. Projektbericht 3. Kundendokumentation **ist zwingend erforderlich** und muss Zielgruppen-gerecht sein. 4. Betriebliche Dokumentationen 5. Weitere Anlagen	X
Anlagen	Auf Anlagen verwiesen zu welchen in der Doku. Bezug genommen wurde	X
Anlagen	Erklärende Unterlagen im Anhang beigefügt	X
Anlagen (Anwendungsentwicklung)	Quellcode im Anhang beigefügt (Nur für die Fachrichtung Anwendungsentwicklung)	X
Anlagen	Anlagen welche nicht durch mich erstellt wurden sind gekennzeichnet	X
Anlagen	Projektantrag inklusive dem Genehmigungsvermerk ist der Dokumentation beigefügt	X
Zeitplanung	Tatsächliche benötigter Zeitaufwand im Vergleich zum Projektantrag	X
Quellenangaben	Quellenverweise sind anzugeben	X
Layout	Hinweise zum Layout beachtet	X
Datenschutz	Betriebliche Daten / Kundendaten sollten anonymisiert werden.	X
Lesbarkeit	Alle Unterlagen/Grafiken/Sreenshots sind klar und lesbar	X
Fachbegriffe	Fachbegriffe sind ausgeschrieben oder einmalig beschrieben	X
Rechtschreibung	Alle Dokumentationen auf deutsche Rechtschreibung und Grammatik geprüft	X
Erklärung	Eidesstattliche Erklärung / Checkliste unterzeichnet von Prüfling und Projektbetreuer und beigefügt	X

Unterschrift Prüfling

Unterschrift Projektbetreuer

Die Hinweise/Checkliste zur Dokumentation ist der Projektdokumentation zwingend beizufügen

Beachten Sie, dass Ihre endgültige Dokumentation, welche Sie ins Online-Portal hochladen, eine Größe von 4 MB nicht überschreiten darf.

IHK Rheinhessen

Sommerprüfung 2025

Ausbildungsberuf
Fachinformatiker/Fachinformatikerin (VO 2020) Fachrichtung: Anwendungsentwicklung

Prüfungsbezirk
Fachinf. Anwendungsentwicklung 2020 (AP T2V1)

Lukas Trapp
Identnummer: 185724-92

Ausbildungsbetrieb: DAA Deutsche Angestelltenakademie // Thiele und Klose Mainz
Projektbetreuer: Rüdiger Klose
E-Mail: ruediger.klose@thieleundklose.de, Telefon: +49 1577 1520736

Thema der Projektarbeit
Optimierung des Anmeldeprozesses für neue Mitglieder des IT-Klub Mainz & Rheinhessen e.V. durch die Digitalisierung des Anmeldeformulars.

1 Thema der Projektarbeit

Optimierung des Anmeldeprozesses für neue Mitglieder des IT-Klub Mainz & Rheinhessen e.V. durch die Digitalisierung des Anmeldeformulars.

2 Geplanter Bearbeitungszeitraum

Beginn: 01.03.2025
Ende: 30.04.2025

3 Ausgangssituation

Im Rahmen meiner Ausbildung und meines Praktikums bei der Thiele & Klose GmbH entwickeln wir eine neue Version der Website „IT-Klub Mainz & Rheinhessen e.V.“. Aktuell leidet die Performance der Website des IT-Klubs unter veralteten, nicht mehr zeitgemäßen Technologien, die weder barrierefreie noch responsive Kriterien erfüllen. Zudem ist der aktuelle Prozess, dem IT-Klub beizutreten, sehr umständlich gestaltet, da das Anmeldeformular erst heruntergeladen, ausgedruckt, ausgefüllt, unterschrieben, wieder digitalisiert (z. B. eingescannt) und anschließend per E-Mail oder Fax an den Klub gesendet werden muss. Alternativ kann das Formular nach dem Druck und der Unterschrift auch per Post gesendet werden, was den Prozess ebenfalls unnötig aufwendig macht. Ein weiterer Aspekt, der nicht gegeben ist, ist die Nachhaltigkeit, bedingt durch einen ressourcenschonenden und effizienten Transportprozess.

4 Projektziel

Im Zuge meiner Projektarbeit werde ich, losgelöst vom Gesamtprojekt und dem zu meinem Projekt nachgelagerten Prozess der Unterschriftseinholung, was über eine separate Softwarelösung geschieht, den Anmeldeprozess in digitaler Form abbilden und benutzerfreundlicher sowie effizienter gestalten. Dazu entwickle ich eine Komponente mit integriertem Formular, welches direkt auf der Website vom Neumitglied ausgefüllt und abgesendet werden kann. Die Daten aus dem Formular werden bei der Eingabe validiert und automatisch über eine Schnittstelle nach nochmaliger Validierung in einer Datenbank abgelegt. Nun können die Daten zentral verwaltet und verarbeitet werden. Auch die Hürde, sich anzumelden, wird deutlich reduziert und gleichzeitig wird eine bessere Benutzererfahrung geschaffen, da der Nutzer die Website nicht verlassen muss. Ebenso entfällt die Verwaltung im physischen Posteingang des IT-Klubs.

Zuerst werde ich das Frontend implementieren, also eine Oberfläche, die das Formular inklusive Validierung der Eingabedaten beinhaltet. Hier wird sichergestellt und geregelt, welche Felder auf jeden Fall ausgefüllt werden müssen und was in welchem Feld hinsichtlich der Zeichen erlaubt ist (z. B. Zahlen, E-Mail, Buchstaben und Sonderzeichen). Danach werde ich die Schnittstelle erzeugen und einbinden, welche mit dem Backend in Verbindung stehen wird. Die API nimmt eine erneute Validierung der weitergegebenen Daten vor und speichert sie. In

diesem Zuge und für diesen Schritt wird auch die Datenbank aufgesetzt und an das Backend angeschlossen. Bei der Prüfung der Daten wird auch geprüft, ob ein Mitglied bzw. die E-Mail-Adresse bereits registriert wurde, bevor das Mitglied angelegt wird. Darüber hinaus gewährleistet die Schnittstelle eine sichere Übertragung der Informationen.

Nach Beendigung der Programmieraufgabe wird alles – Benutzeroberfläche, Weitergabe, Speicherung und Konsistenz der Daten ausführlich getestet. Miteingeschlossen ist ein Unittest der API. Im Anschluss wird das Formular und der Prozess von einer zweiten Person geprüft.

Meine Herausforderung ist die begrenzte Erfahrung mit den verwendeten Technologien, die ich durch gezielte Recherche und Tutorials bewältigen werde.

Das Frontend der Seite wurde mit HTML und Astro realisiert. Das Framework Astro zeichnet sich durch Modularität und die dadurch resultierende hohe Geschwindigkeit aus, die durch partielles Laden gewährleistet wird. Außerdem ermöglicht es die einfache Integration von verschiedenen JavaScript-Frameworks sowie von HTML und CSS. Diese Eigenschaften ermöglichen eine flexible und effiziente Entwicklung. Das Backend des IT-Klubs beruht auf Node.js. Node ist eine serverseitige Plattform, die auf JavaScript basiert. Sie ermöglicht somit eine konsistente Entwicklungsumgebung zwischen Frontend und Backend. Zur Kommunikation zwischen Frontend und Datenbank habe ich mich für eine RESTful API entschieden. REST wird aufgrund der Einfachheit, Flexibilität und breiten Unterstützung gewählt.

5 Zeitplanung

Die Umsetzung des Projekts wird auf eine Gesamtdauer von 80 Stunden angesetzt und setzt sich wie folgt zusammen:

Recherche und Planung des Projekts (10 Stunden):

Recherche im Internet, in Foren sowie in Dokumentationen zu den einzelnen Technologien und Frameworks. Planung des Zeitaufwands und der Arbeitsschritte basierend auf den Projektzielen.

Entwicklung des Frontends (15 Stunden):

Erstellung des Formulars (Logik und Oberfläche mit Astro, HTML und CSS).

Entwicklung des Backends inklusive API (35 Stunden):

Programmieren der Datenbank und der Übergabe der Daten inklusive Validierung über eine API

an das Backend (Node.js).

Testen und Dokumentation (20 Stunden):

Testen (8 Stunden)

Testen der einzelnen Bereiche und des Formulars auf potenzielle Fehler wie falsche Zeichen, unzulässige Eingabelängen oder andere Richtlinien (z. B. reguläre Ausdrücke). Fehler müssen korrigiert und die betroffenen Passagen erneut getestet werden. Anschließend wird das Formular von einer projektfremden Person getestet, wobei Kritik hinsichtlich Intuitivität und Benutzerfreundlichkeit (UI/UX) einbezogen wird, falls sie objektiv sinnvoll ist.

Dokumentation (12 Stunden)

Die Dokumentation des Projekts, der Planung, der technische Umsetzung sowie der durchgeführten Tests. Unter anderem werden Fehler, die während des Testens auftreten behoben. Dies wird ebenfalls festgehalten.

Abnahme zuerst durch den Ausbilder und anschließend durch den Kunden, den verantwortlichen Mitarbeiter des IT-Klubs.

6 Anlagen

siehe Anlage 1

7 Präsentationsmittel

MacBook
HDMI Kabel
Presenter

8 Hinweis!

Ich bestätige, dass der Projektantrag dem Ausbildungsbetrieb vorgelegt und vom Ausbildenden genehmigt wurde. Der Projektantrag enthält keine Betriebsgeheimnisse. Soweit diese für die Antragstellung notwendig sind, wurden nach Rücksprache mit dem Ausbildenden die entsprechenden Stellen unkenntlich gemacht.

Mit dem Absenden des Projektantrages bestätige ich weiterhin, dass der Antrag eigenständig von mir angefertigt wurde. Ferner sichere ich zu, dass im Projektantrag personenbezogene

Daten (d. h. Daten über die eine Person identifizierbar oder bestimmbar ist) nur verwendet werden, wenn die betroffene Person hierin eingewilligt hat.

Bei meiner ersten Anmeldung im Online-Portal wurde ich darauf hingewiesen, dass meine Arbeit bei Täuschungshandlungen bzw. Ordnungsverstößen mit „null“ Punkten bewertet werden kann. Ich bin weiter darüber aufgeklärt worden, dass dies auch dann gilt, wenn festgestellt wird, dass meine Arbeit im Ganzen oder zu Teilen mit der eines anderen Prüfungsteilnehmers übereinstimmt. Es ist mir bewusst, dass Kontrollen durchgeführt werden.

Anlage 1

Entscheidungshilfe zum Projektantrag

Wenn Sie die nachfolgenden Punkte berücksichtigen, steigt die Wahrscheinlichkeit, dass der Projektantrag genehmigt wird und Sie zeitnah mit der Umsetzung Ihres Projektes beginnen können.

Projektbeschreibung	Inhalt des Projekts Realisierung und Anpassung eines komplexen Systems. (Die berufliche Handlungsfähigkeit, muss sowohl fachlich als auch methodisch dargestellt werden. Siehe Seite 1)	✓
Projektinhalt	Aufgaben- bzw. Problemstellung Ziele und Nutzen des Kunden klar herausgestellt	✓
Ausgangslage	Darstellung von Ist- Zustand, Zielgruppe bzw. Auftraggeber (externer Kunde oder eigenes Unternehmen)	✓
Fremdleistungen	Fremdleistungen und Schnittstellen sind vollständig gekennzeichnet (sofern vorhanden)	✓
Zeitbedarf	Tätigkeiten außerhalb der Prüfungszeit sind klar abgegrenzt (sofern vorhanden)	✓
Rechtschreibung	Antrag auf deutsche Rechtschreibung und Grammatik geprüft	✓
Fachlicher Bezug	Identifikation der Kernaufgabe des Projekts mit Bezug zur Verordnung (40 oder 80 Stunden)	✓
Projektphasen	Mindestens 3 Projektphasen vorhanden	✓
Projektphasen	Projektphasen Aussagekräftig bezeichnet - Darstellung der wesentlichen Arbeitsschritte - Kennzeichnung der Kernaufgaben des Projektes - Kennzeichnung der prüfungsrelevanten Aufgaben - Darstellung der zeitlichen Abhängigkeit	✓
Zeitumfang	Zeitbedarf gem. Verordnung eingehalten (40 oder 80 Stunden)	✓
Zeit der Dokumentation	Zeitbedarf der Dokumentation kleiner 15%	✓
Anwendungsbereich	Anwendungsbereich Durchführung für... / bei... Ansprechpartner Geben Sie an, in welchem Umfeld Sie Ihr Projekt bearbeiten. Hilfreich sind bspw. Informationen zum Auftraggeber, weitere Größenordnungen (z.B. Anzahl der Mitarbeiter, Anzahl der Systeme), Schnittstellen etc.	✓
Fachbegriffe	Unternehmensinterne Abkürzungen und Fachbegriffe sind zu erläutern.	✓
Checkliste	Checkliste dem Antrag beigefügt	✓

Unterschrift Prüfling

Unterschrift Projektbetreuer

Die Entscheidungshilfe ist dem Projektantrag zwingend beizufügen