MAVI KEDI

Architekturdokumentation

Softwareentwicklungsprojekt
SS 2024

Auftraggeber:
bitExpert AG

Version: 3.0.0

Datum: 19.06.2024

Autoren:
Johannes Moseler
Zehra-Fikriye Gönenç
Philipp Wäsch

Inhaltsverzeichnis

1 Versionierung 3
1.1 Versionierungsregeln 3
1.2 Versionsverzeichnis 3
2 Einführung und Ziele 4
2.1 Einführung 4
2.2 Aufgabenstellung 5
2.3 Qualitätsziele 6
2.4 Stakeholder 6
3 Randbedingungen 7
3.1 Verwendung von OpenSource-LLMs 7
3.2 Verwendung des Cody-Backends 8
4 Kontextabgrenzung 8
4.1 Fachlicher Kontext 8
4.1.1 Aktivitätsdiagramm 9
4.1.2 Fallbeschreibung 10
4.2 Technischer Kontext 12
5 Bausteinsicht 13
5.1 Ebene 1 13
5.2 Ebene 2 15
6 Laufzeitsicht 19
6.1 Website erstellen lassen 19
6.2 Design-Änderungswunsch 20
7 Verteilungssicht 20
8 Architekturentscheidungen 22
8.1 Webframework für das Back-End 22
8.1.1 Fragestellung 22
8.1.2 Relevante Einflussfaktoren 22
8.1.3 Annahmen 23
8.1.4 Entscheidungskriterien 23
8.1.5 Betrachtete Alternativen 23
8.1.6 Entscheidungen 23
8.2 Lokales LLM 23
8.2.1 Fragestellung 23
8.2.2 Relevante Einflussfaktoren 23
8.2.3 Annahmen 23

8.2.4 Entscheidungskriterien 24
8.2.5 Betrachtete Alternativen 24
8.2.6 Entscheidungen 24
8.3 Skriptsprache 24
8.3.1 Fragestellung 24
8.3.2 Relevante Einflussfaktoren 24
8.3.3 Annahmen 25
8.3.4 Entscheidungskriterien 25
8.3.5 Betrachtete Alternativen 25
8.3.6 Entscheidungen 25
9 Qualitätsanforderungen 26
9.1 Qualitätsbaum 26
9.2 Qualitätsszenarien 26
10 Glossar 27
11 Abbildungsverzeichnis 28
12 Tabellenverzeichnis 28

1 Versionierung

1.1 Versionierungsregeln

Im Folgenden sind die Regeln für die Versionierung aufgeführt.

- Die Versionsnummer besteht aus 3 Zahlen, getrennt durch einen Punkt.
- Die erste Zahl (Hauptversion) wird um eins erhöht, wenn das Architekturdokument abgeschlossen ist und nach Prüfung durch das Qualitätsmanagement zur Abgabe freigegeben wird.
- Die zweite Zahl wird um eins erhöht, wenn neue Abschnitte hinzugefügt werden.
- Die dritte Zahl wird bei kleinen Änderungen oder Fehlerkorrekturen, wie beispielsweise Verbesserung der Rechtschreibung oder der Verbesserung von Formatierungsfehlern um eins erhöht.
- Wenn die erste Zahl erhöht wird, werden die restlichen beiden Zahlen auf null zurückgesetzt.
- Wenn die zweite Zahl erhöht wird, wird die letzte Zahl auf null zurückgesetzt

Die initiale Version ist **0.0.0**.

1.2 Versionsverzeichnis

Version	Änderung	Kürzel	Datum	freigegeben von	Status
3.0.0	Feedback von Prof. Knauber angewendet	zg, pw, jm	18.06.2024	th	✅
2.0.0	Feedback von Prof. Knauber angewendet	zg, pw, jm	04.06.2024	ee	✅
1.2.0	Architekturentscheidungen bearbeitet	zg	31.05.2024	ee	✅
1.1.0	Use-Case Beschreibung bearbeitet, technischer Kontext hinzugefügt, Ollama-Architekturentscheidung bearbeitet, Qualitätsbaum hinzugefügt, Qualitätsszenarien bearbeitet	pw	29.05.2024	ee	✅
1.0.0	Letzte Qualitätssicherung vor Abgabe	zg	14.05.2024	th	✅
0.5.0	Erstellung der Sichten und deren Beschreibungen	jm	13.05.2024	th	✅
0.4.0	Erste grobe Qualitätssicherung durchgeführt	zg	13.05.2024	th	✅

0.3.0	Entwurfsentscheidungen festgelegt und hinzugefügt	zg	10.05.2024	th	✅
0.2.0	Qualitätsziele bearbeitet, Randbedingungen hinzugefügt, fachlicher Kontext hinzugefügt, Qualitätsszenarien hinzugefügt	pw	10.05.2024	th	✅
0.1.1	Aufgabenstellung überarbeitet, Einführung bearbeitet	pw	02.05.2024	th	✅
0.1.0	Grundstruktur festgelegt	zg	29.04.2024	th	✅

Tabelle 1: Versionsverzeichnis

Kürzel	Name Autoren
th	Tobias Heid
ee	Enes Ekincioglu
zg	Zehra-Fikriye Gönenç
pw	Philipp Wäsch
jm	Johannes Moseler
cz	Colin Zenner

Tabelle 2: Legende der Autoren

Symbol	Bedeutung
⚠️	Noch nicht durch das Qualitätsmanagement freigegeben
✅	Von dem Qualitätsmanagement freigegeben

Tabelle 3: Symbol

2 Einführung und Ziele

2.1 Einführung

Die Architekturdokumentation wird im Rahmen des Softwareentwicklungsprojekt 2024 erstellt. Diese umfasst alle Informationen, Entscheidungen und Vorgaben, die hauptsächlich von den Entwicklern getroffen werden. Sie verschafft ein besseres Verständnis über die Architektur unserer Software.

Das *SEP* ist für die *IB*-Studierende an der Hochschule Mannheim im 4. Fachsemester ein größeres Projektsemester. Dabei werden Gruppen von sechs Studierenden gebildet, die ein Softwareprodukt für einen externen Kunden nach seinen Anforderungen über das Semester ausarbeiten.

Dieses Dokument wird im Laufe des Projektes kontinuierlich angepasst, um die Informationen über die Architektur stets auf aktuellen Zustand standzuhalten.

In diesem Dokument werden Verweise auf andere Dokumente verwendet. Sowohl das Pflichtenheft (v. 2.0.0) als auch das Projekthandbuch (v. 2.0.0) sind in der Google Drive gespeichert.

Die kursiv gesetzten Wörter sind in einem Glossar präziser beschrieben, das ab Seite 27-28 zu finden ist.

Das vorliegende Dokument basiert auf dem arc42 Template (*https://arc42.org*, Version 8.2 DE, Januar 2023). Es wurde gewählt, da es im Rahmen der Lehrveranstaltung Software-Engineering 2 an der Hochschule behandelt wurde und alle Teammitglieder damit vertraut sind.
Folgende Kapitel haben wir aus Redundanzgründen weggelassen, da der Sachverhalt der jeweiligen Kapitel bereits sinngemäß erläutert wird.
(Lösungsstrategie, Querschnittliche Konzepte, Risiken und technische Schulden, aus arc42 Template)

Der Kunde erhält die unten aufgeführten Sichten zusätzlich auf dem prooph-BOARD mit den entsprechenden Stickies.

2.2 Aufgabenstellung

Unsere Aufgabe ist es, das Benutzerinterface (UI) der von *Cody* erstellten Webseite mithilfe von *LLM*s zu optimieren, um das Produkt ansprechender für unseren Auftraggeber zu gestalten.

Dazu planen wir, die zukünftigen Endnutzer aktiv in den Gestaltungsprozess der Website einzubeziehen. Dies geschieht zunächst durch einen Fragebogen, aus dem die Informationen für einen Prompt generiert werden, um die erste "Grundseite" zu erstellen. Anschließend können weitere spezifische Anpassungen durch einen Chatbot vorgenommen werden.
Auf diese Weise möchten wir eine agile und effiziente Umgebung schaffen, die es uns ermöglicht, flexibel auf die Bedürfnisse unserer Endnutzer einzugehen und innovative Lösungen zu liefern.

2.3 Qualitätsziele

Als MaviKedi-Team ist es uns wichtig, dass unser Softwareprodukt die folgenden architekturrelevanten Qualitätsziele (siehe Tabelle 4) erfüllt. Diese Quailtätsziele haben wir unter Berücksichtigung der Soll-Anforderungen (siehe *Pflichtenheft Kapitel 3: Soll-Anforderungen*) festgelegt.
Um die Softwarequalität unseres Produkts gewährleisten zu können, orientieren wir uns an den Qualitätsziele aus der Norm *ISO/IEC 9126*. Diese haben wir ausgewählt, da sie unsere Soll-Anforderungen im besten Fall erfüllen.

Qualitätsziel	Beschreibung
Zuverlässigkeit	Das System soll seine Funktionen zuverlässig und fehlerfrei ausführen, um eine konsistente und stabile Benutzererfahrung zu bieten.
Änderbarkeit/Wartbarkeit	Das System soll auf Verbesserungen, Korrekturen und Erweiterungen anpassbar reagieren und dabei die Soll-Anforderungen (siehe: *Pflichtenheft*) einschließen
Effizienz	Die Software sollte nach Benutzereingaben eine maximale Ladezeit von 15 Sekunden haben, um das gewünschte Ergebnis anzuzeigen, und sollte flüssig laufen, ohne dass Fehler auftreten.
Kompatibilität	Das System soll auf den vier meistgenutzten Browsern funktionieren, wobei die Prozentangaben den Marktanteil der Browser (Stand: Mai 2024) repräsentieren: Google Chrome (43,25%), Safari (42,59%), Firefox (7,08%), Edge (6,07%). Dabei sollte sichergestellt werden, dass alle Funktionen und Inhalte korrekt dargestellt und fehlerfrei ausgeführt werden. (*Quelle*)

Tabelle 4: Ausgewählte Qualitätsziele aus *ISO/IEC 9126*

2.4 Stakeholder

Siehe *Projekthandbuch (MAVI KEDI)* - Abschnitt 3 Projektbeteiligte

3 Randbedingungen

Die folgenden Tabellen (Tabelle 5 und Tabelle 6) beschreiben die architekturrelevanten Randbedingungen des Projekts.

3.1 Verwendung von OpenSource-LLMs

Randbedingungs-ID	RB1
Beschreibung	Für das Projekt dürfen ausschließlich OpenSource-LLMs verwendet werden, es sei denn, der Zugriff auf die IONOS-LLM wird gewährt, dann dürfen wir diese ebenfalls nutzen
Rational	Da proprietäre LLMs kostenpflichtig sind, entscheidet sich BitExpert dagegen, sie zur Verfügung zu stellen. BitExpert arbeitet mit IONOS SE zusammen, wodurch sie diese AI zu einem späteren Zeitpunkt zur Verfügung stellen könnten.
Fit-Kriterium	Genutztes LLM ist OpenSource oder die IONOS-AI
Kriterium	Muss-Kriterium
Priorität	Hoch
Konflikte	Keine
Quelle	Kundengespräch
Status	Erfüllt

Tabelle 5: Randbedingung: Verwendung von OpenSource-LLM

3.2 Verwendung des Cody-Backends

Randbedingungs-ID	RB2
Beschreibung	Für das Projekt darf ausschließlich das Backend der Cody-Engine (TypeScript) verwendet werden
Rational	Um die gleiche Funktionalität wie der von der Cody-Engine erstellten Website zu erhalten, muss das Backend der Cody-Engine, welches in TypeScript geschrieben wurde, genutzt werden.
Fit-Kriterium	Verwendung des Cody-Engine-Backends zur Website-Erstellung
Kriterium	Muss-Kriterium
Priorität	Hoch
Konflikte	Keine
Quelle	Kundengespräch 07.05.2024
Status	Erfüllt

Tabelle 6: Randbedingung: Verwendung des Cody-Backends

4 Kontextabgrenzung

4.1 Fachlicher Kontext

Die aktuellen Prototypen der Cody-Engine bieten bereits die gewünschten Funktionalitäten, aber das UI soll durch den Einsatz von Large Language Models (LLMs) und Benutzereingaben verbessert werden.

Der fachliche Kontext wird durch das folgende Aktivitätsdiagramm in Abschnitt 4.1.1 dargestellt (siehe folgende Abbildung 1), das zwei Partitionen umfasst: den Endnutzer und die Cody-Engine. Die im Diagramm enthaltenen Pfeile mit Fallnummern beziehen sich auf die Anwendungsfälle der KEDIAI, die in Abschnitt 4.1.2 in Tabelle 7 und Tabelle 8 detailliert beschrieben werden.

4.1.1 Aktivitätsdiagramm

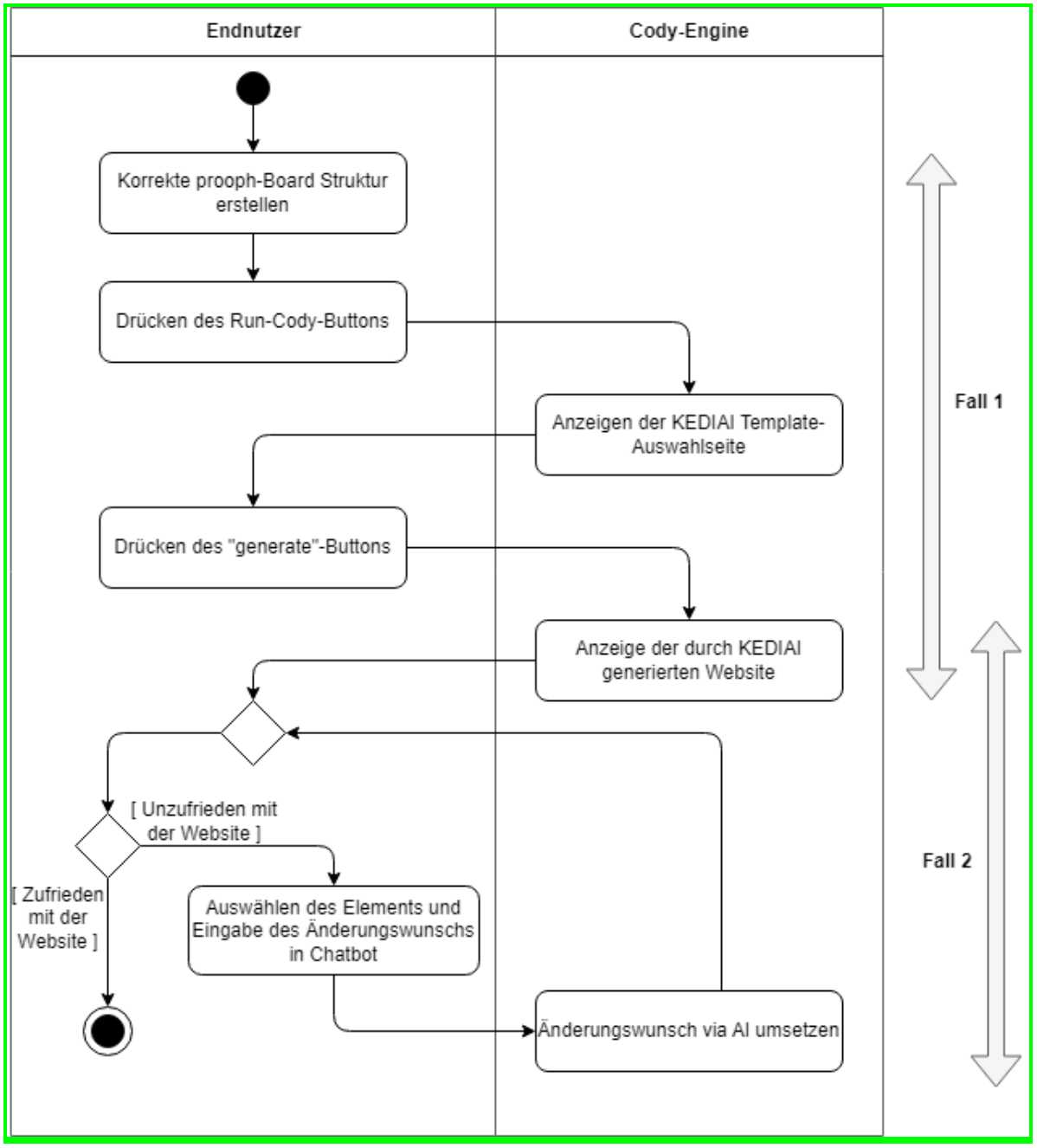


Abbildung 1: Fachlicher Kontext - Aktivitätsdiagramm

4.1.2 Fallbeschreibung

Fall 1	Websitegenerierung
Akteure	Endnutzer
Ziel	Website mit individuellen Parametern generieren
Hauptablauf	1. Der Endnutzer startet die Cody-Engine 2. Das System zeigt einen Fragebogen mit verschiedenen Parametern zur individuellen Website-Erstellung mit auswählbaren Website-Templates an 3. Der Endnutzer füllt den Fragebogen aus und wählt sein favorisiertes Template aus 4. Das System generiert die Website und öffnet sie in einem neuen Tab
Anfangsbedingungen	Technisch korrekte prooph-BOARD-Struktur existiert
Abschlussbedingungen	Der Endnutzer erhält eine nutzerfreundliche, individuell anpassbare Website
Qualitätsanforderungen	Die Seitengenerierung soll mit einer Zeitverzögerung von maximal 15 Sekunden und ohne auftretenden Fehler flüssig laufen. Die benötigten Schritte zur Website-Generierung müssen für den Endnutzer einfach erlernbar und intuitiv sein.

Tabelle 7: Fallbeschreibung Fall 1 - Websitegenerierung

Fall 2	Websiteveränderung durch Chatbot
Akteure	Endnutzer
Ziel	Durch individuelle Endnutzer-Eingaben die Website optisch verändern
Hauptablauf	1. Der Endnutzer klickt auf den Chatbot-Button 2. Das System öffnet den Chatbot, zeigt änderbare Elemente in einer Dropdown-Liste an und fragt nach optischen Veränderungen 3. Der Endnutzer wählt das Element aus, das er verändern will und gibt seinen Änderungswunsch ein. 4. Das System verändert das gewünschte Element durch AI
Anfangsbedingungen	• Website wurde generiert • Chatbot-Button wird angezeigt
Abschlussbedingungen	Die AI hat das Element, so wie es sich der Endnutzer gewünscht hat, verändert.
Qualitätsanforderungen	Das richtige Element muss verändert werden. Die Veränderung des Elements soll mit einer Zeitverzögerung von maximal 15 Sekunden und ohne auftretenden Fehler flüssig geschehen.

Tabelle 8: Fallbeschreibung Fall 2 - Websiteveränderung durch Chatbot

4.2 Technischer Kontext

Der technische Kontext wird im folgenden Verteilungsdiagramm (Abbildung 2) dargestellt. Die Farbgebung zeigt, welche Knoten und Komponenten von wem erstellt wurden. In Tabelle 9 sind die Schnittstellen ausführlich beschrieben.

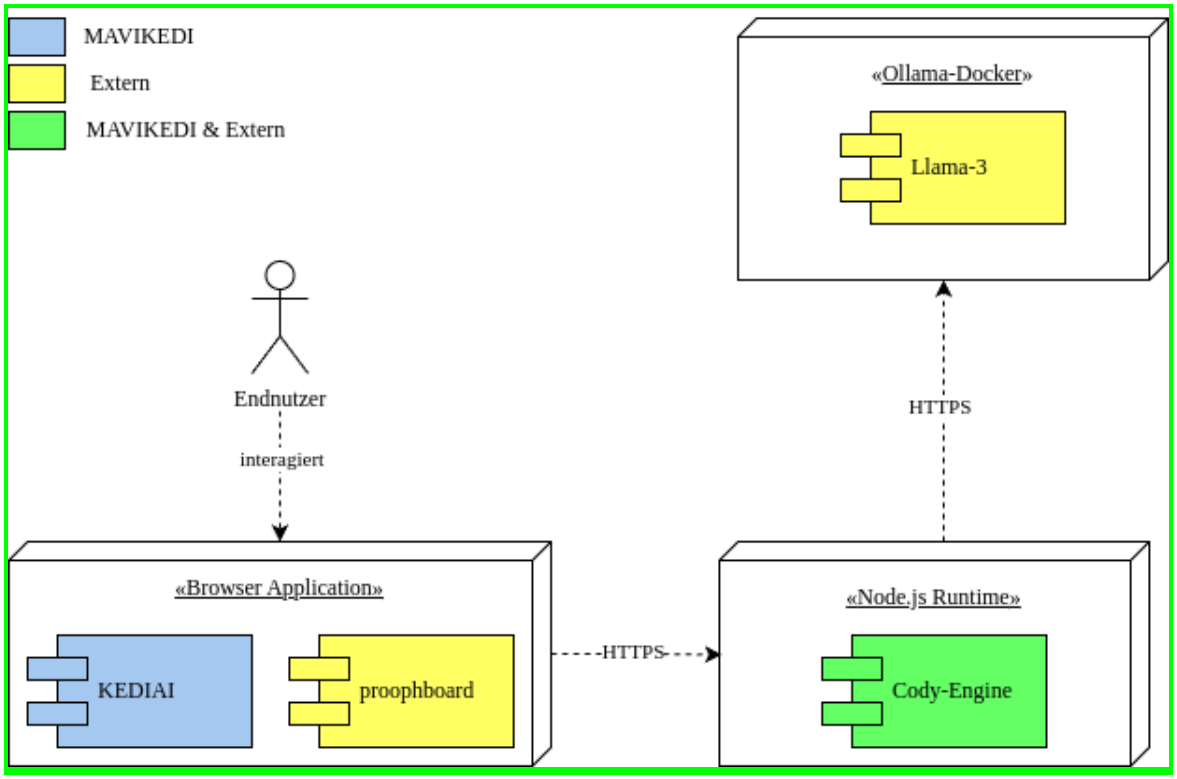


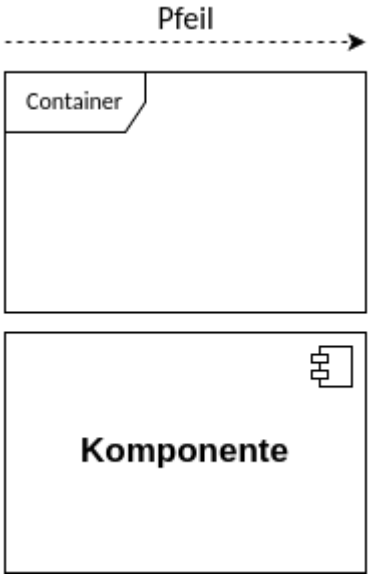
Abbildung 2: Technischer Kontext

Knoten	Schnittstelle	Erklärung
Browser Application	HTTPS	Der Endnutzer interagiert mit der Browser-Anwendung, auf der sowohl KEDI-AI als auch das Proophboard laufen. • Das Proophboard kommuniziert über HTTPS mit der Cody-Engine, um die Proophboard-Struktur in den Prototyp umzusetzen. • KEDI-AI übermittelt ebenfalls über HTTPS die auf der Template-Auswahlseite gewählten Parameter und eingegebenen Prompts an die Cody-Engine.
Ollama-Docker	HTTPS	Die Prompts werden von der Cody-Engine generiert und via HTTPS an den Ollama-Docker gesendet. Wir nutzen HTTPS, um eine sichere, verschlüsselte Verbindung zwischen dem Browser und dem Server zu gewährleisten.

Tabelle 9: Technischer Kontext - Schnittstellen

5 Bausteinsicht

Die Komponente in der Pfeilrichtung wird benutzt und es können auch Daten zurückkommen.



Ein Container beinhaltet beliebig viele Komponenten und schließt diese zu einem Bund zusammen. Sie sind da, um auf höheren Ebenen Abstraktion zu gewährleisten.

Eine Komponente besitzt mehrere Funktionen und könnte noch genauer gegliedert werden.

5.1 Ebene 1

In dieser Ebene (zu sehen in der folgenden Abbildung 3) stellen wir die von uns entwickelte Architektur vor. Außerdem zeigt sie den groben Aufbau unseres Produktes und das Zusammenwirken der einzelnen Komponenten. In Tabelle 10 werden die einzelnen Komponenten genauer beschrieben.

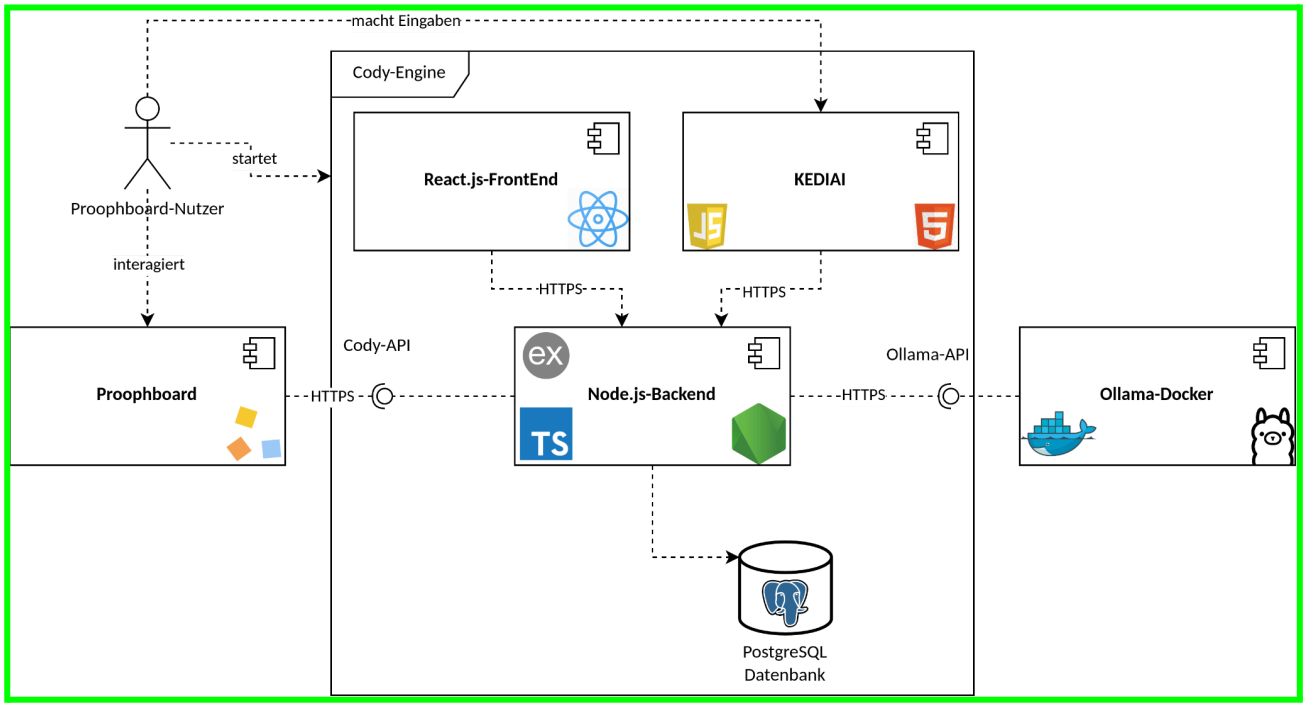


Abbildung 3: Bausteinsicht - Ebene 1

Name	Beschreibung
Proophboard	Das Proophboard ist eine Webseite für Event-Storming. Es ermöglicht die Entwicklung von ereignisgesteuerten Systemen.
Cody-Engine	Die Cody-Engine entwickelt das entsprechende Back- und Frontend basierend auf den auf dem Proophboard erstellten Systemen.
Proophboard-Nutzer	Dies ist der Nutzer, der auf dem Proophboard Systeme erstellt
PostgreSQL Datenbank	Ist die Datenbank die für die erstellten Systeme der Cody-Engine genutzt wird, falls diese eine brauchen
HTTPS (Hypertext Transfer Protocol Secure)	Ist das Netzwerk-Protokoll, über das wir unterschiedliche Dienste ansprechen, z.B. Kommunikation von Back- und Frontend
Node.js-Backend	Das Backend von der Cody-Engine ist eine Node.js Applikation, die in TypeScript geschrieben ist. Es wird auch das Express.js Framework verwendet, um die Entwicklung für API's zu ermöglichen. Da in der Cody-Engine bereits ein Node.js Backend vorhanden ist, hat der Kunde uns vorgeschlagen, dieses mit unseren Funktionen zu erweitern. Außerdem hatten wir bereits Erfahrung mit Javascript, was die Arbeit mit Typescript erleichtert.(siehe auch 3.2 und 8.1)
React.js-Frontend	Dies ist das Frontend, welches von der Cody-Engine erzeugt wird. Hier werden die Funktionalitäten der Systeme, die auf dem Proophboard erstellt werden, gezeigt und können getestet werden.
KEDIAI	Dieses Frontend ist von uns erstellt, wir haben uns für die Arbeit mit HTML und Javascript entschieden, um der AI HTML-Dateien geben zu können, die von dieser bearbeitet werden. Da wir bereits mit HTML und Javascript gearbeitet hatten, haben wir uns dazu

	entschieden, unser Frontend damit aufzubauen. HTML bietet zudem viele Möglichkeiten der Gestaltung, was für unser Produkt von großer Bedeutung ist.(siehe auch 8.3)
Ollama-Docker	Dies besteht aus einem Ollama Docker, der auf einem Server läuft, welcher uns vom Kunden bereitgestellt wurde. Wir benutzen die von Ollama bereitgestellte LLM llama3, welche lokal läuft. Wir greifen auf diese über die API von Ollama zu. Da Ollama viele Open Source Möglichkeiten für LLMs bietet, haben wir uns dazu entschieden, diese zu benutzen da wir dadurch sehr flexibel sind und außerdem kein Kostenaufwand entsteht.(siehe auch 3.1 und 8.2)

Tabelle 10: Bausteinsicht - Ebene 1

5.2 Ebene 2

In der folgenden Ebene (Abbildung 4) sind nur die von uns genutzten Komponenten und Container aufgefasst. Deswegen entfallen die Datenbank und das React.js-Frontend, da diese von unseren Funktionen nicht beeinflusst werden. In Tabelle 11 werden die einzelnen Komponenten und Container genauer beschrieben.

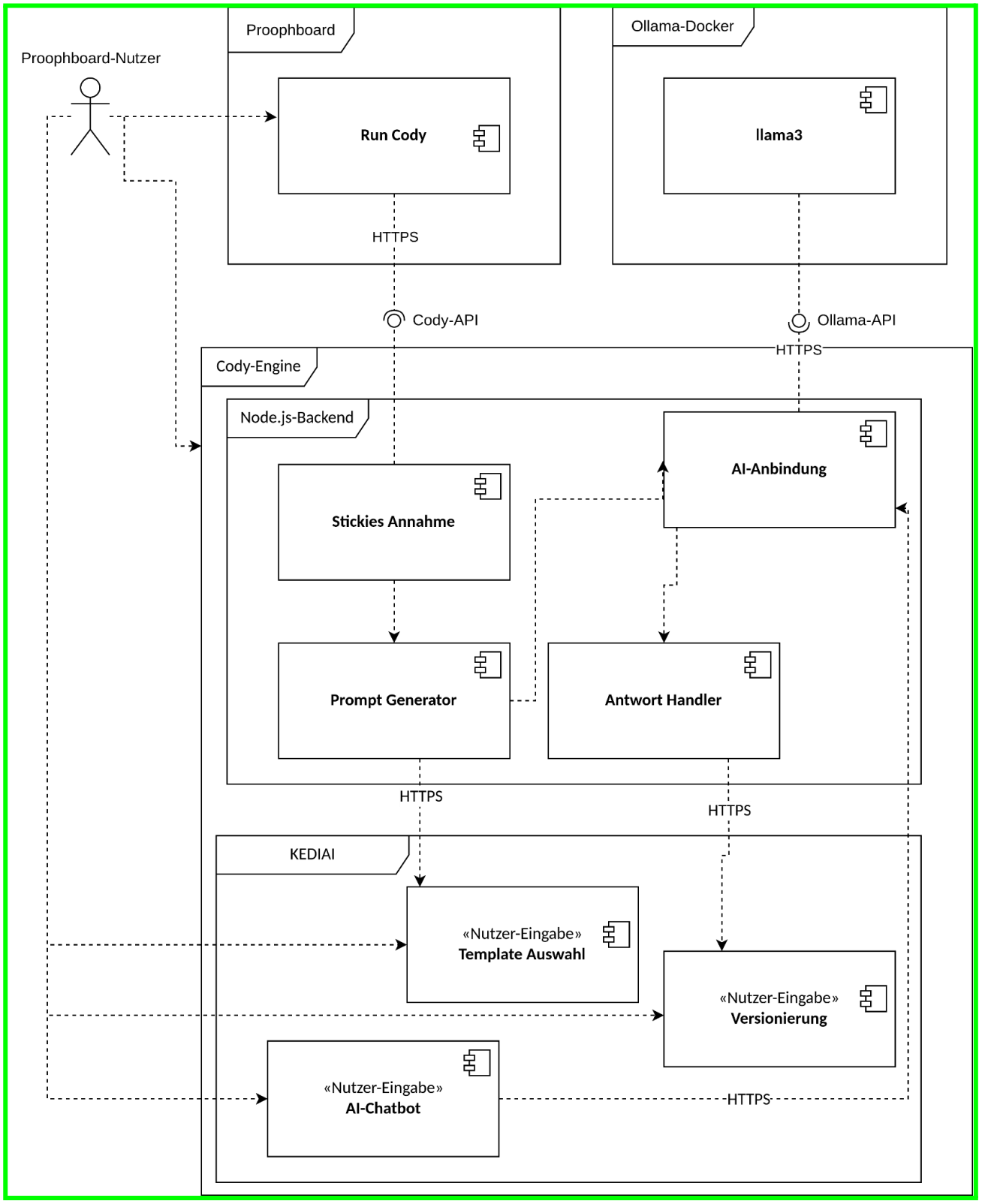


Abbildung 4: Bausteinsicht - Ebene 2

Name	Beschreibung
Stickies Annahme	Diese Komponente nimmt die Stickies, die auf dem Proophboard stehen, entgegen. Die angenommenen Stickies werden in einer .json-Datei gespeichert. Die Stickies, die vom Proophboard aus ans Backend gesendet werden, ermöglichen es uns, die Funktion der auf dem Proophboard erstellten Logik auszulesen.
Prompt Generator	Es wird die Template Auswahl für den Nutzer geöffnet. Hier werden die Stickies analysiert und zu einer Prompt zusammengefasst. Nachdem der Nutzer die Template Auswahl abgeschlossen hat, wird die Auswahl empfangen und das Template wird ausgelesen. Dann werden das Template und der Prompt in eine Textdatei geschrieben(Dies ist erstmal nur zum Testen der Prompts). Als letztes wird der endgültige Prompt an die AI-Anbindung übergeben. Der Prompt Generator ermöglicht es uns, die Funktionalität ganz leicht mit den ausgewählten Templates zu verbinden. Es ist auch möglich, die übergebene Prompt an die AI zu ändern, um unterschiedliche Prompts zu testen und die beste Prompt zu ermitteln.
AI-Anbindung	Nimmt die Prompt an und schickt diese an die API von Ollama. Dann wartet sie auf die Rückgabe der LLM. Die Rückgabe ist im .json Format und muss deswegen erstmal in einen String umgewandelt werden. Der String wird dann an den Antwort Handler weitergegeben.
Antwort Handler	Hier wird der String angenommen. Dann wird der String mit einer Regular Expression durchlaufen, der die HTML Datei ausliest. Dann wird in diese HTML Datei über ein Skript unser Chatbot eingefügt. Als letztes wird noch die erstellte HTML Datei geöffnet. Dies ermöglicht es, eine vollständige HTML Seite ins Frontend zu schicken, wo diese dann genutzt wird.
Template Auswahl	In der Template Auswahl hat der Nutzer die Möglichkeit ein vorgefertigtes Template

	auszuwählen und noch Änderungen an der Font oder der Farbe vorzunehmen. Die Auswahl wird dann per HTTPS an das Backend geschickt. Da wir flexible Gestaltung versprechen, haben wir uns dazu entschieden, eine Template Auswahl Funktion zu implementieren, damit der Nutzer schon mal mehrere grobe Vorschläge hat, die er dann noch weiter verfeinern kann.(siehe auch 9.2)
AI-Chatbot	Der Nutzer hat die Möglichkeit, noch Änderungen an der erstellten Seite vorzunehmen. Dafür kann er einfach seinen Wunsch in ein Chatbox schreiben. Dieser Wunsch wird dann mit der HTML der Seite an die AI-Anbindung geschickt. Diese Funktion ermöglicht, flexible Veränderungen an den bereitgestellten Templates vorzunehmen, ohne Kentnisse in HTML oder CSS zu haben.(siehe auch 9.2)
Versionierung	Der Nutzer kann zwischen den von dem Chatbot erstellten Versionen auswählen, welcher Stand der aktuelle sein soll. Damit ermöglichen wir, Veränderungen rückgängig machen zu können und von alten Ständen neue Veränderungen vornehmen zu können.(siehe auch 9.2)
llama3	Dies ist das Modell, das von uns für dieses Projekt benutzt wird, da wir durch Vergleich mit anderen LLMs bei diesem Modell zu den besten Ergebnissen gekommen sind.

Tabelle 11: Bausteinsicht-Ebene 2

6 Laufzeitsicht

In diesem Kapitel stellen wir Laufzeitsichten vor, die in der Form von Sequenzdiagrammen dargestellt werden. Hierbei wird der Durchlauf genauer erläutert und der Nutzen der einzelnen Komponenten besser erklärt.

6.1 Website erstellen lassen

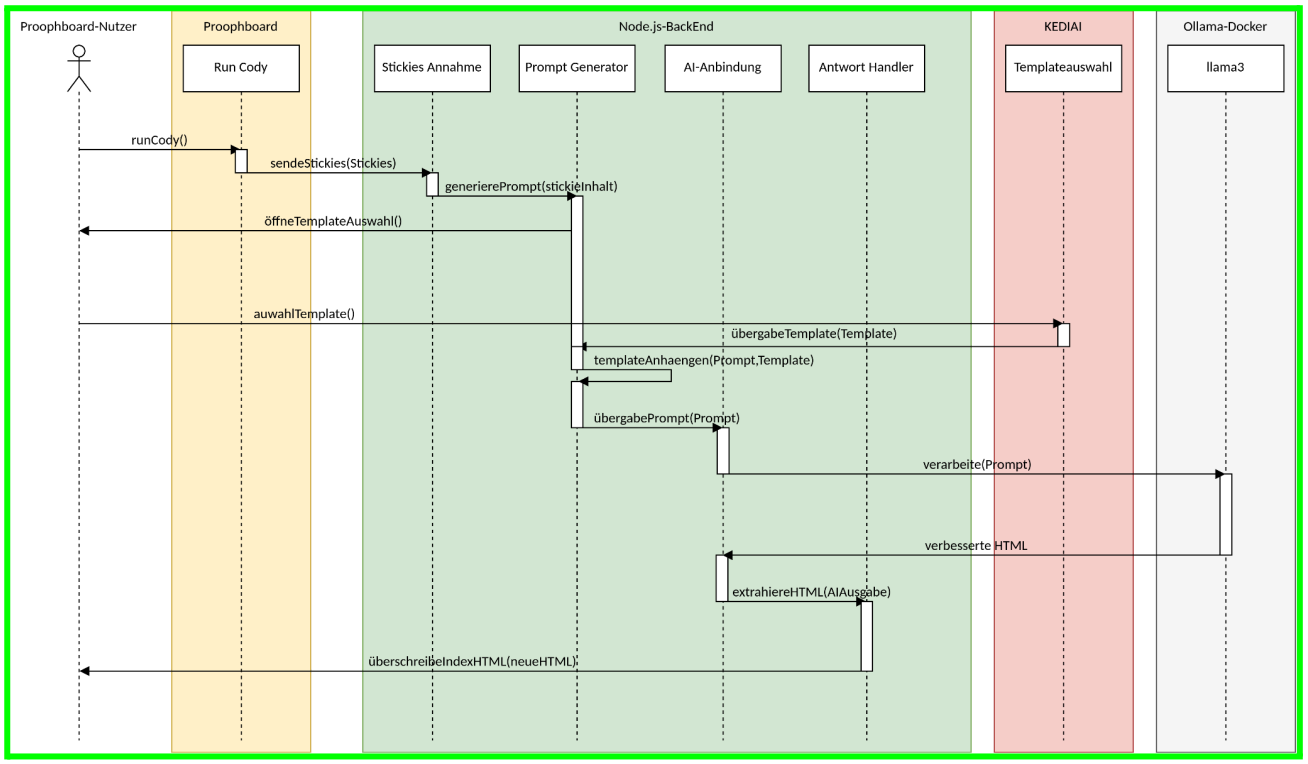


Abbildung 5: Laufzeitsicht- Website erstellen lassen

In diesem Sequenzdiagramm (Abbildung 5) wird gezeigt, wie die Website von der AI generiert wird. Es beginnt damit, dass der Nutzer die Cody-Engine startet. Danach wird dem Backend die Sticky Information übergeben und daraus eine Prompt generiert .Dann muss der Nutzer noch ein Template auswählen, das passiert asynchron. Die Funktionalität vom Proophboard wird nun in das Template geschrieben und dem Nutzer angezeigt.

6.2 Design-Änderungswunsch

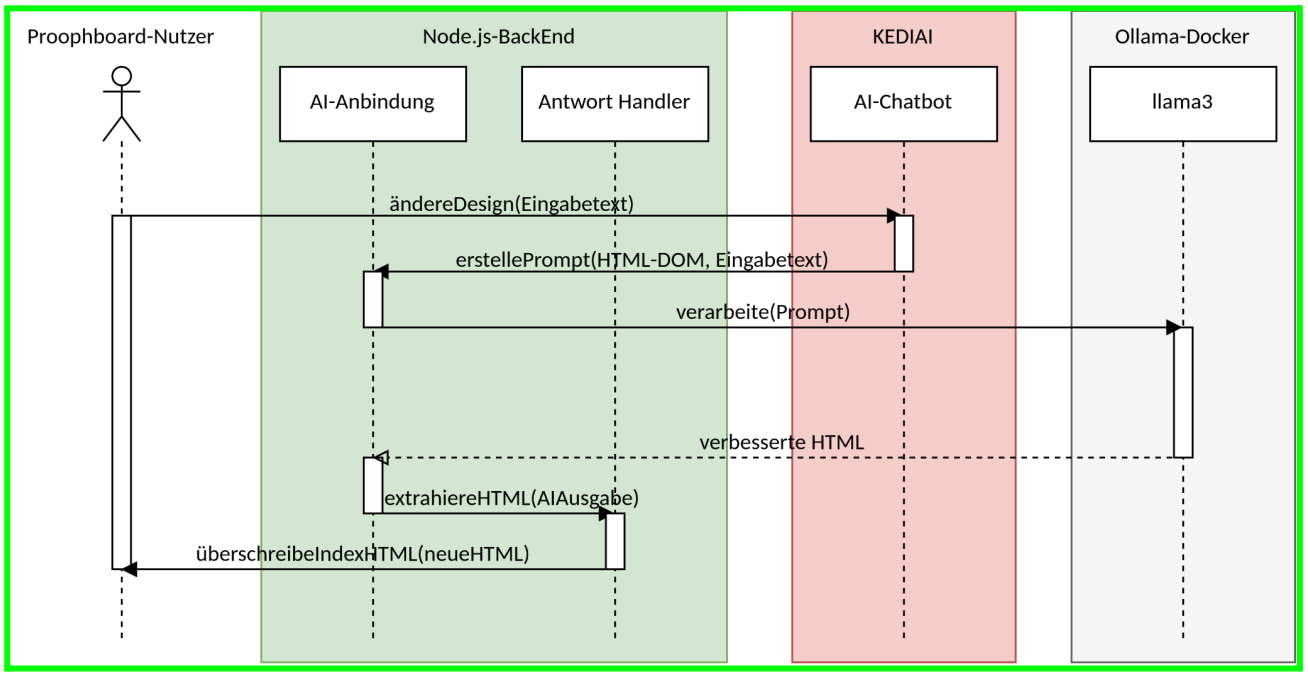


Abbildung 6: Laufzeitsicht- Design-Änderungswunsch

Das hier dargestellte Sequenzdiagramm (Abbildung 6) zeigt einen Nutzer des Proophboardes, der seine erstellte Seite über eine Eingabe im Chatbot verändern will.

7 Verteilungssicht

Unsere Verteilungssicht (zu sehen in der folgenden Abbildung 7) besteht aus drei Teilen:

- Proophboard Server
- Server vom Kunden auf dem der Ollama-Docker läuft
- Client-PC

Wir haben diese Verteilung gewählt, da die Cody-Engine lokal läuft und der Kunde uns einen Server mit API-Anschluss an einen Ollama-Docker anbietet. Da wir die Proophboard-Website nutzen, ist auch dieser Server ein Teil unseres Systems.
In Tabelle 12 werden die einzelnen Teile der Verteilungssicht genauer beschrieben.

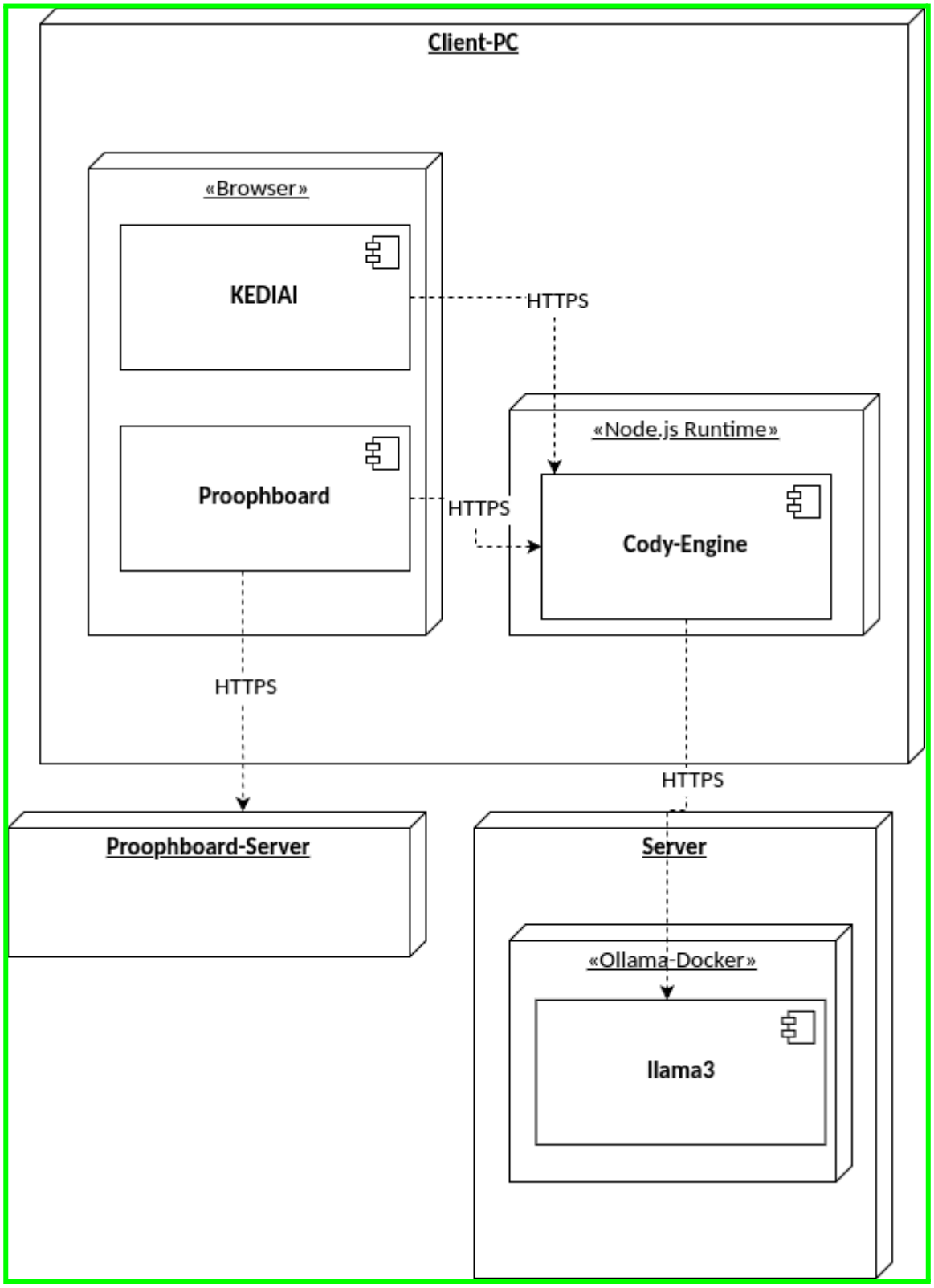


Abbildung 7: Verteilungssicht

Name	Beschreibung
Client-PC	Dies ist der Rechner, auf dem das System ausgeführt wird.
Browser	Der Browser bietet eine Runtime Umgebung sowohl für unseren Frontend als auch für das Proophboard.
KEDIAI	Hiermit ist unser Frontend gemeint.(siehe auch 5.1 KEDIAI)
Node.js Runtime	Node.js ermöglicht es uns, eine API zu erstellen, über die unser Frontend mit dem Backend kommuniziert und auf die Cody-Engine von Proophboard zugreifen kann.
Proophboard-Server	Ist der Server, auf dem die Proophboard Webseite läuft. Auf diesen greifen wir über die Proophboard Webseite zu. Wie genau dieser Server arbeitet, wissen wir nicht, deswegen hat dieser keine genauere Komponente.
Docker	Ein Ollama-Docker, der auf einem vom Kunden bereitgestellten Server läuft. Ollama bietet eine API. Über diese API schicken wir Ollama Prompts die von llama3 verarbeitet werden. Die Antwort wird dann zurückgeschickt.

Tabelle 12: Verteilungssicht

8 Architekturentscheidungen

8.1 Webframework für das Back-End

8.1.1 Fragestellung

Welches Back-End Webframework setzen wir ein?

8.1.2 Relevante Einflussfaktoren

- ➔ Siehe Qualitätsziel: *Änderbarkeit/Wartbarkeit*
- ➔ Siehe Randbedingung: *Verwendung des Cody-Backends*
- ➔ Siehe Qualitätsziel: *Kompatibilität*

8.1.3 Annahmen

Wir nehmen an, dass node.js die Entwicklung vereinfacht und beschleunigt, da node.js sowohl für das Front- als auch für das Back-End dieselbe Skriptsprache JavaScript verwendet.

8.1.4 Entscheidungskriterien

Ein entscheidendes Kriterium ist unsere Randbedingung *Verwendung des Cody-Backends*. Diese Randbedingung muss definitiv berücksichtigt werden.
Sie soll ohne eine komplizierte Lernkurve schnell anwendbar und anpassbar sein (siehe: *Änderbarkeit/Wartbarkeit*), da wir für das Projekt ein Semester zur Verfügung haben.

8.1.5 Betrachtete Alternativen

Python-Flask ist ein kleines und schlankes Python-Webframework mit nützlichen Tools und Funktionen, die das Erstellen von Webanwendungen in Python erleichtern.
(*Informationsquelle*)

Node.js hat einen einzigartigen Vorteil, da Entwickler, die JavaScript für den Browser schreiben, jetzt den serverseitigen Code zusätzlich zum Clientseitigen Code schreiben können, ohne dass sie eine völlig neue Sprache anwenden müssen. Des Weiteren ist Node.js hoch skalierbar.
(*Informationsquelle*)

8.1.6 Entscheidungen

Für unser Back-End wählen wir Node.js als Laufzeitumgebung, da diese unsere Kriterien am Besten erfüllen.
Unser Softwareentwicklungsprojekt läuft ein Semester lang, deshalb ist uns ein schneller und leichter Einstieg ins Programmierparadigma wichtig. Außerdem weist die Laufzeitumgebung eine hohe Wartbarkeit auf, die ebenfalls unseren Kriterien entspricht. *Siehe 8.1.2 Relevante Einflussfaktoren.*

8.2 Lokales LLM

8.2.1 Fragestellung

Welches Tool verwenden wir, um ein *LLM* lokal laufen zu lassen?

8.2.2 Relevante Einflussfaktoren

- ➔ Siehe Qualitätsziel: *Änderbarkeit/Wartbarkeit*
- ➔ Siehe Randbedingung: *OpenSource-LLMs*

8.2.3 Annahmen

Wir nehmen an, dass die LLM-Modelle der betrachteten Tools ähnlich gute Ergebnisse liefern.

8.2.4 Entscheidungskriterien

Das ausgewählte LLM-Tool muss Open-Source sein, da der Kunde es so vorgegeben hat. Des Weiteren sollte das LLM-Modell einfach änderbar sein.

8.2.5 Betrachtete Alternativen

- LocalAI
- IONOS-AI

8.2.6 Entscheidungen

Wir haben uns für Ollama entschieden, weil es Open-Source ist und einen unkomplizierten Zugriff auf öffentliche LLMs über ein Repository ermöglicht, das die Download-URLs verwaltet. Das Ändern des LLM-Modells ist über eine API-Anfrage möglich, dabei muss lediglich der LLM-Modellname geändert werden.
Außerdem empfiehlt unser Tutor, der viel Erfahrung mit LLMs/AIs hat, die Verwendung von Ollama.

LocalAI ist komplizierter zu nutzen, da man beispielsweise beim Ändern des Modells eine detaillierte Einrichtung mit spezifischen Anpassungen der Backend-Bibliothek vornehmen muss.

Der Kunde kann die IONOS-AI nicht im gegebenen Zeitraum liefern. Somit entfällt diese Möglichkeit.
(*siehe: 8.2.2 Relevante Einflussfaktoren*).

8.3 Skriptsprache

8.3.1 Fragestellung

Welche Skriptsprache wollen wir für unsere Software nutzen?

8.3.2 Relevante Einflussfaktoren

- ➔ Siehe Randbedingung: *Verwendung des Cody Backends*
- ➔ Siehe Architekturentscheidung 8.1.2 Relevante Einflussfaktoren: *Die Cody-Engine baut auf Node.js auf*
- ➔ Siehe Qualitätsziel: *Änderbarkeit/Wartbarkeit*
- ➔ Siehe Qualitätsziel: *Kompatibilität*
- ➔ Siehe Qualitätsziel: *Zuverlässigkeit*

8.3.3 Annahmen

Wir erwarten, dass unser Entwicklungsprozess durch die Nutzung von JavaScript schneller ist als Python, weil im Team Vorwissen zu JavaScript existiert und der Aufwand in Personenstunden für das Erlernen einer neuen Skriptsprache dadurch erspart wird.

8.3.4 Entscheidungskriterien

Siehe *8.3.2 Relevante Einflussfaktoren*.

8.3.5 Betrachtete Alternativen

Python ist eine plattformübergreifende Sprache. Sie unterstützt funktionale, objektorientierte und prozedurale Programmierparadigmen. Sie wird oft als Skriptsprache verwendet.
(*Informationsquelle*)

JavaScript ist eine dynamisch typisierte Skriptsprache, die eine schnelle Implementierung ohne Angaben von genauen Typen ermöglicht. Die Einfachheit macht JavaScript berühmt. JS verfügt über umfangreiche Ressourcen wie Bibliotheken, Dokumentationen, die Entwicklern dabei helfen, effizient zu arbeiten und Probleme zu lösen.
(*Informationsquellen*)

8.3.6 Entscheidungen

Die Entscheidung fällt auf JavaScript, da unsere Entwickler Vorwissen in JavaScript besitzen und da unsere Präferenzen auf einer ausführlichen und umfangreichen Dokumentation basieren, die uns bei Fehlern ermöglicht, eine effiziente Lösung durch die Dokumentation und der zahlreichen Communitys zu finden. Zumal auch die Entwickler der Cody-Engine das Back End in TypeScript programmiert haben und JavaScript eine ähnliche Skriptsprache ist, im Gegensatz zu Python.
(*siehe: 8.3.2 Relevante Einflussfaktoren*).

9 Qualitätsanforderungen

9.1 Qualitätsbaum

In der folgenden Abbildung (Abbildung 8) ist unser Qualitätsbaum dargestellt.

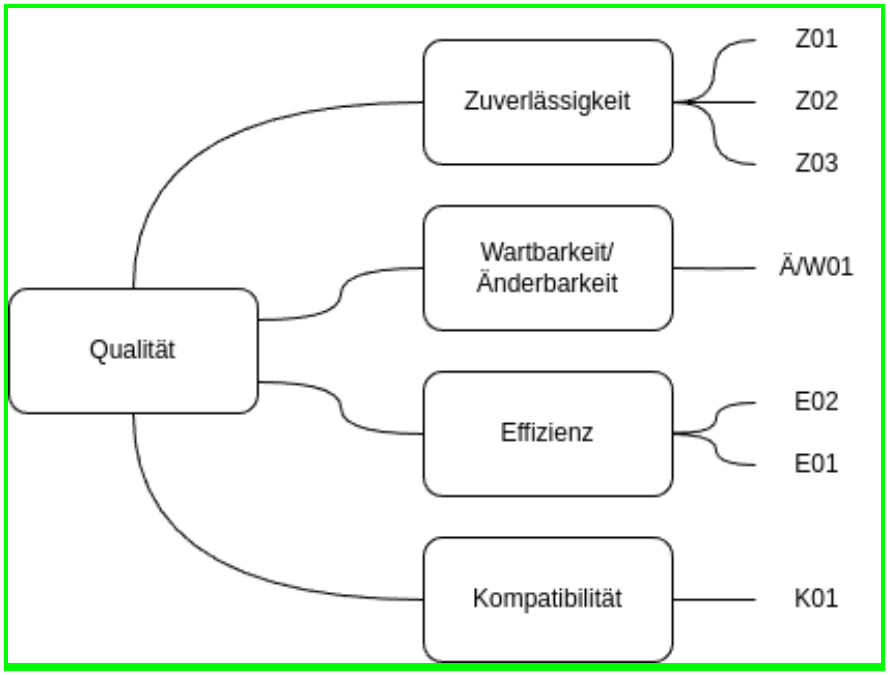


Abbildung 8: Qualitätsbaum

9.2 Qualitätsszenarien

Die Anfangsbuchstaben der IDs in der folgenden Tabelle stehen jeweils für das übergeordnete Qualitätsmerkmal, B beispielsweise für Benutzbarkeit.

ID	Szenario
Z01	Der Endnutzer drückt nach dem Fragebogen auf den “generiere Website”-Button. Eine Website nach seinen Wünschen und Parametern wird erstellt.
Z02	Der Endnutzer möchte via Chatbot das Aussehen der Website ändern. Der Chatbot ändert nur das Aussehen der Seite und nicht deren Funktionen.
Z03	Der Endnutzer möchte seine letzten 5 durch den Chatbot realisierten Website-Änderungen zurücksetzen. Durch das fünfmalige Klicken auf den “undo”-Button gelangt er zu der gewünschten Website-Version.
Ä/W 01	Das Modell Llama-3 ist nicht mehr erreichbar. Durch das Ändern der LLM-Modellvariable ist ein schneller Wechsel des Modells innerhalb einer Minute möglich.
E01	Der Endnutzer nutzt den Chatbot, um die Hintergrundfarbe zu ändern. Nach maximal 15 Sekunden soll die Farbe geändert sein.
E02	Der Endnutzer möchte seine letzte Änderung durch den Chatbot rückgängig machen. Er

	klickt auf den “undo”-Button. Das System setzt die Seite spätestens nach zwei Sekunden zurück.
K01	Der Endnutzer nutzt Google Chrome, um die Seite generieren zu lassen. Die generierte Seite wird korrekt und fehlerfrei angezeigt.

Tabelle 13: Qualitätsszenarien

10 Glossar

Begriff	Definition
Cody	Cody ist ein Bot oder Agent, der in der Lage ist, ein Event Map Design in funktionierende Software zu übersetzen.(*quelle*)
Event-Storming	Event-Storming ist ein Hilfsmittel, um Fach- und Implementierungsexperten über die abzubildende Domäne diskutieren zu lassen, mit dem Ziel, die wichtigsten Domänenobjekte und deren Wechselwirkungen herauszuarbeiten.
IB	Studiengang: Allgemeine Informatik (an der Hochschule Mannheim)
LLM	LLM steht für "Large Language Models" und sind leistungsstarke künstliche Intelligenzen, die natürliche Sprache verstehen und generieren können.
MAVI KEDI	Softwareentwicklungsteam vom Sommersemester 2024. MAVI KEDI heißt ,,Blaue Katze” auf türkisch.
nginx	Nginx, ausgesprochen "engine-ex", ist ein beliebter Open-Source-Webserver.
prooph-BOARD	prooph-BOARD ist ein *Event-Storming* und Modelling-Tool um Fach- und Implementierungsexperten über die abzubildene Domäne diskutieren zu lassen, mit dem Ziel die wichtigsten Domänenobjekte herauszufinden und Relationen schnell zu erkennen.
SEP	Softwareentwicklungsprojekt
TEW	TEW (Teamentwicklungs-Workshop) ist eine Veranstaltung, die parallel zum Projekt belegt wird, dabei wird im Team auf sozial-pädagogische Maßnahmen ergriffen.

Tabelle 14: Glossar

11 Abbildungsverzeichnis

Abbildung 1: Fachlicher Kontext - Aktivitätsdiagramm 9
Abbildung 2: Technischer Kontext 12
Abbildung 3: Bausteinsicht - Ebene 1 13
Abbildung 4: Bausteinsicht - Ebene 2 16
Abbildung 5: Laufzeitsicht - Website erstellen lassen 19
Abbildung 6: Laufzeitsicht - Design-Änderungswunsch 20
Abbildung 7: Verteilungssicht 21
Abbildung 8: Qualitätsbaum 26

12 Tabellenverzeichnis

Tabelle 1: Versionsverzeichnis 3-4
Tabelle 2: Legende der Autoren 4
Tabelle 3: Symbol 4
Tabelle 4: Ausgewählte Qualitätsziele aus ISO/IEC 9126 6
Tabelle 5: Randbedingung: Verwendung von OpenSource-LLMs 7
Tabelle 6: Randbedingung: Verwendung des Cody-Backends 8
Tabelle 7: Fallbeschreibung Fall 1 - Websitegenerierung 10
Tabelle 8: Fallbeschreibung Fall 2 - Websiteveränderung durch Chabot 11
Tabelle 9: Technischer Kontext - Schnittstellen 12
Tabelle 10: Bausteinsicht-Ebene 1 14-15
Tabelle 11: Bausteinsicht-Ebene 2 17-18
Tabelle 12: Verteilungssicht 22
Tabelle 13: Qualitätsszenarien 26-27
Tabelle 14: Glossar 27-28