Architekturdokumentation

Softwareentwicklungsprojekt
SS 2024

Auftraggeber: BitExpert AG

Version: 1.0.0

Datum: 14.05.2024

Autoren:
Johannes Moseler
Zehra-Fikriye Gönenç
Philipp Wäsch

Inhaltsverzeichnis

1 Versionierung

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 Pflichtenheft abgeschlossen ist und nach Prüfung durch das Qualitätsmanagement zur ersten 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**.

1.1 Versionsverzeichnis

Version	Änderung	Kürzel	Datum	freigegeben von	Status
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, dass ab Seite 24 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 in den gegebenen Kapitel sinngemäß erläutert wird.
(Lösungsstrategie, Querschnittliche Konzepte, Risiken und technische Schulden, technischer Kontext und der Qualitätsbaum 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 Prototypen mithilfe von LLMs 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

Qualitätsziel	Beschreibung
Benutzbarkeit	Das System soll für die Endnutzer einfach nutzbar sein. Es muss verständlich und mit einem geringen Aufwand erlernbar und bedienbar sein.
Funktionalität	Die Software soll die definierten und festgestellten Anforderungen erfüllen.
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 funktionalen Spezifikationen einschließen

Qualitätsziel	Beschreibung
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: Qualitätsziele

2.4 Stakeholder

Siehe Projekthandbuch (MAVI KEDI) - Abschnitt 3 Projektbeteiligte

3 Randbedingungen

Die Randbedingungen für das Projekt umfassen sowohl organisatorische als auch externe Anforderungen, darunter zu berücksichtigende Abläufe beim Kunden, Entwicklungsvorgaben wie Prozesse und Programmiersprachen sowie externe Vorgaben wie Regularien, Gesetze und ethische Regeln.

3.1 Verwendung von OpenSource-LLMs

Anforderungs-ID	NFR1
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-LLMs

3.2 Verwendung des Cody-Backends

Anforderungs-ID	NFR2
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.

4.1.1 Use-Case-Diagramm

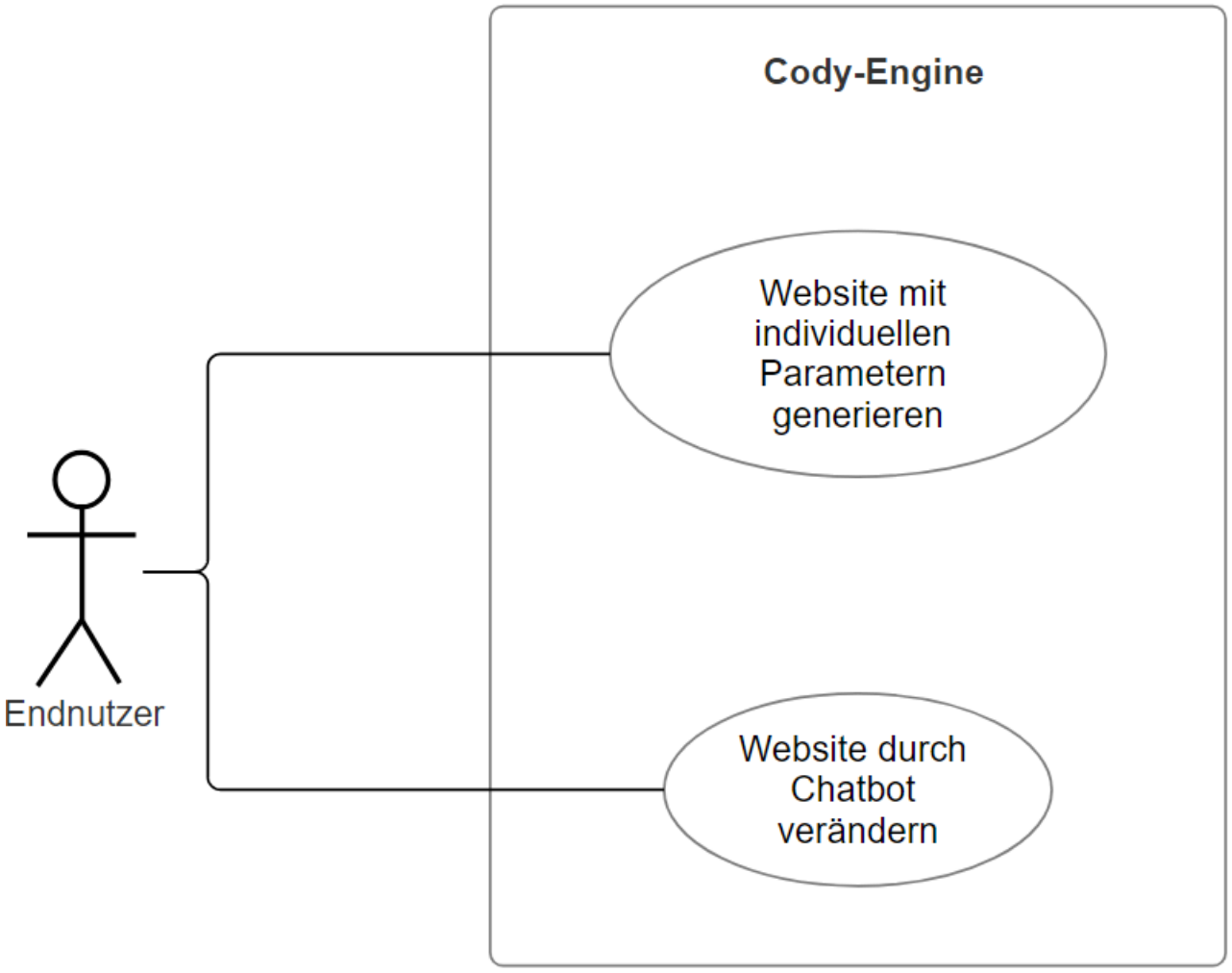


Abbildung 1: Fachlicher Kontext - Use-Case-Diagramm

4.1.2 Use-Case-Beschreibung

Anwendungsfallname	generiereWebsite
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	Cody-Engine wird ausgeführt
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: Use-Case: generiereWebsite

Anwendungsfallname	**verändereWebsitDurchChatbot**
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 und fragt nach optischen Veränderungen 3. Der Endnutzer gibt ein, was er verändern möchte 4. Das System verändert das gewünschte Element durch AI
Anfangsbedingungen	● Website wurde generiert ● Chatbot-Button wird angezeigt
Abschlussbedingungen	Die AI verändert das Element so wie es sich der Endnutzer gewünscht hat
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.

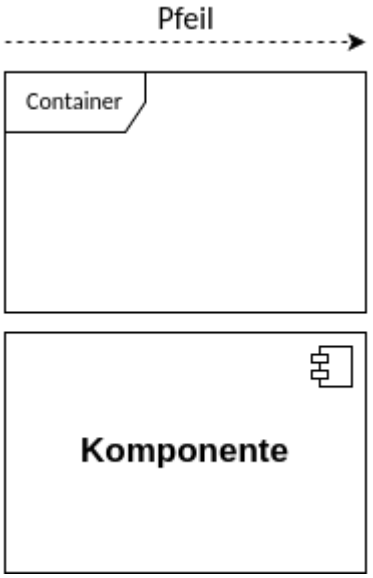
Tabelle 8: Use-Case: verändereWebsiteDurchChatbot

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

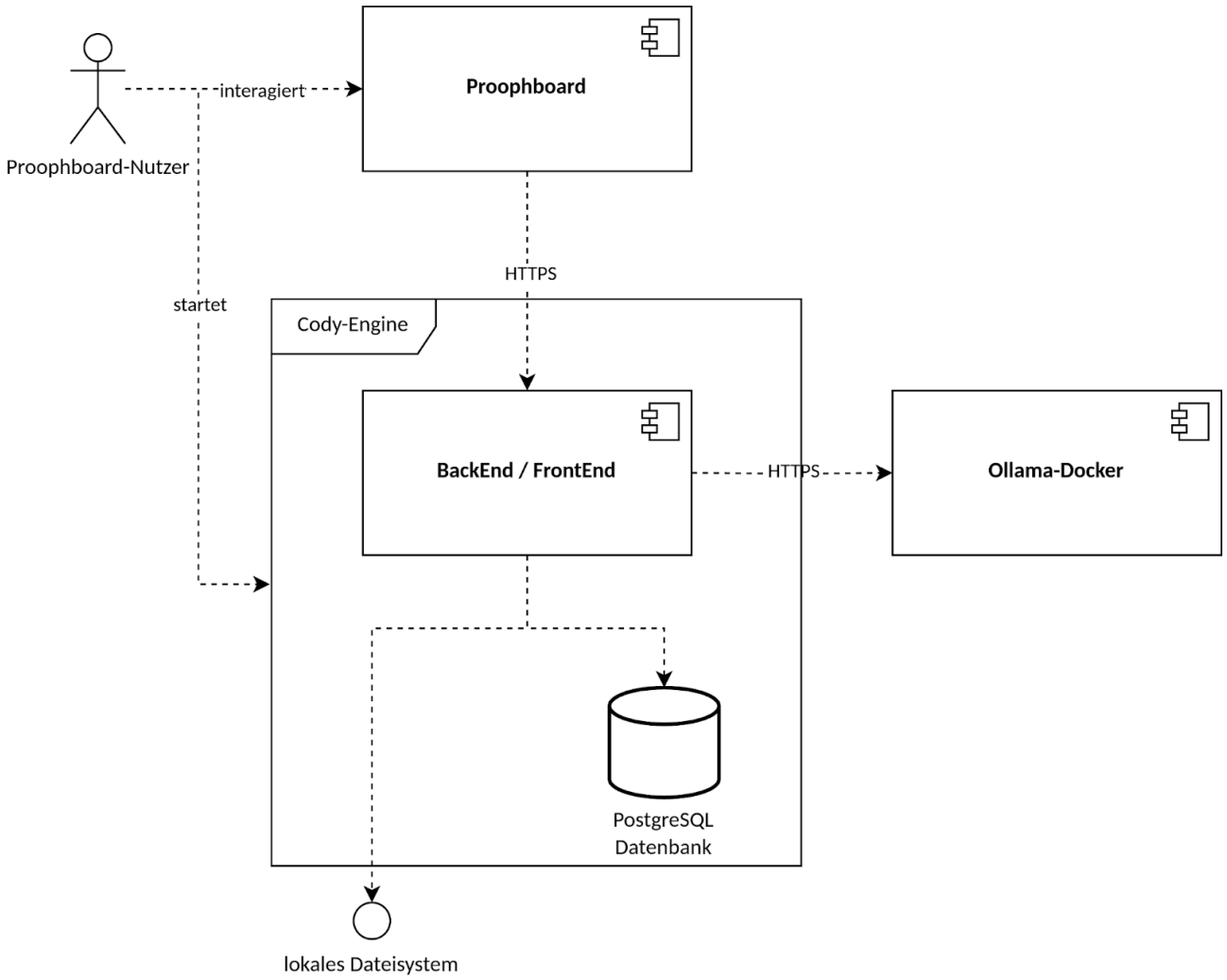


Abbildung 2: Bausteinsicht - Ebene 1

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.
BackEnd/FrontEnd	Dies ist das Back- und Frontend der Cody-Engine.

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
lokales Dateisystem	Unser System ändert Dateien im lokalen Ordner der Cody-Engine, z.B. wird eine prompt.txt erstellt oder die index.html überschrieben
HTTPS (Hypertext Transfer Protocol Secure)	Ist das Netzwerk-Protokoll, über das wir unterschiedliche Dienste ansprechen, z.B. Kommunikation von Back- und Frontend
Ollama-Docker	Wird in der nächsten Ebene beschrieben.

Tabelle 9: Bausteinsicht - Ebene 1

5.2 Ebene 2

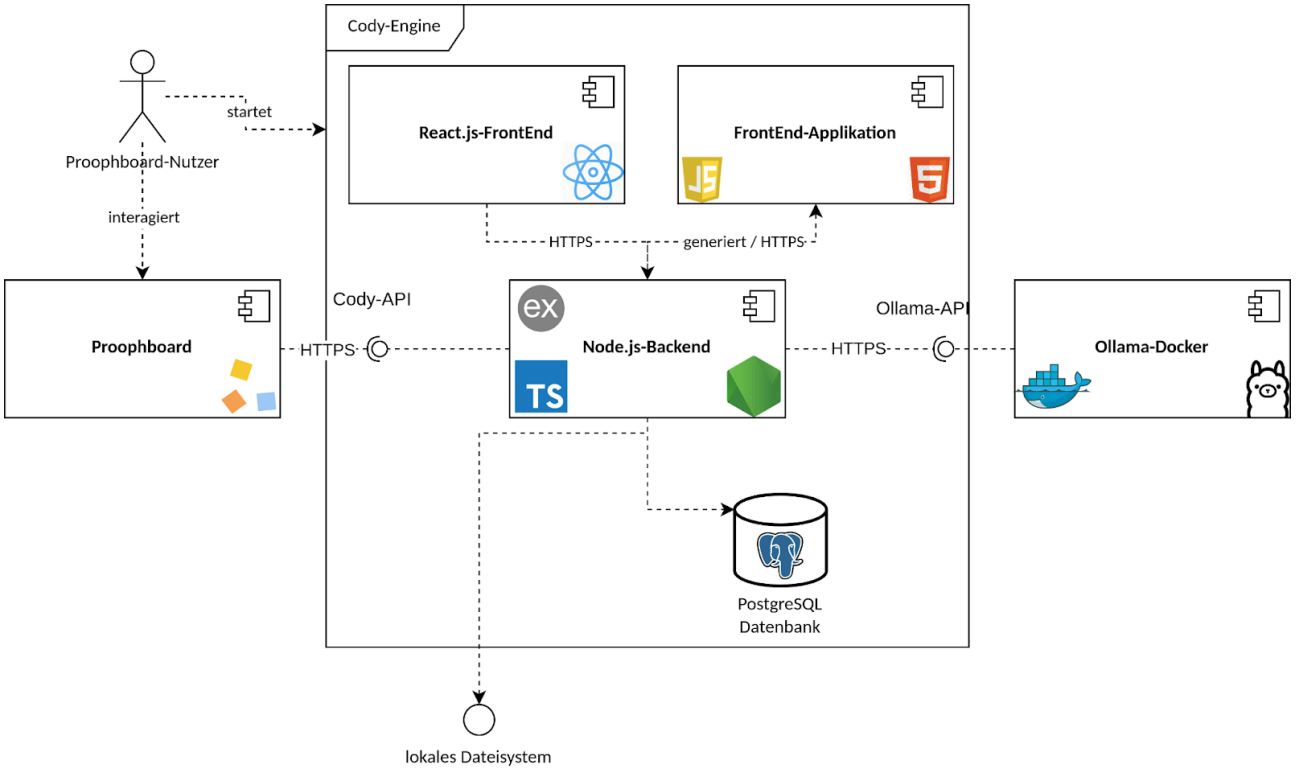


Abbildung 3: Bausteinsicht - Ebene 2

Name	Beschreibung
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. Wir haben unser Backend in das der Cody-Engine eingebunden.
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.
FrontEnd-Applikation	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.
Ollama-Docker	Dies besteht aus einem Ollama Docker, der auf einem Home-Server läuft. Wir benutzen von Ollama bereitgestellte LLM llama3, welche lokal läuft. Wir greifen auf diese über die API von Ollama zu.

Tabelle 10: Bausteinsicht - Ebene 2

5.3 Ebene 3

In dieser Ebene 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.

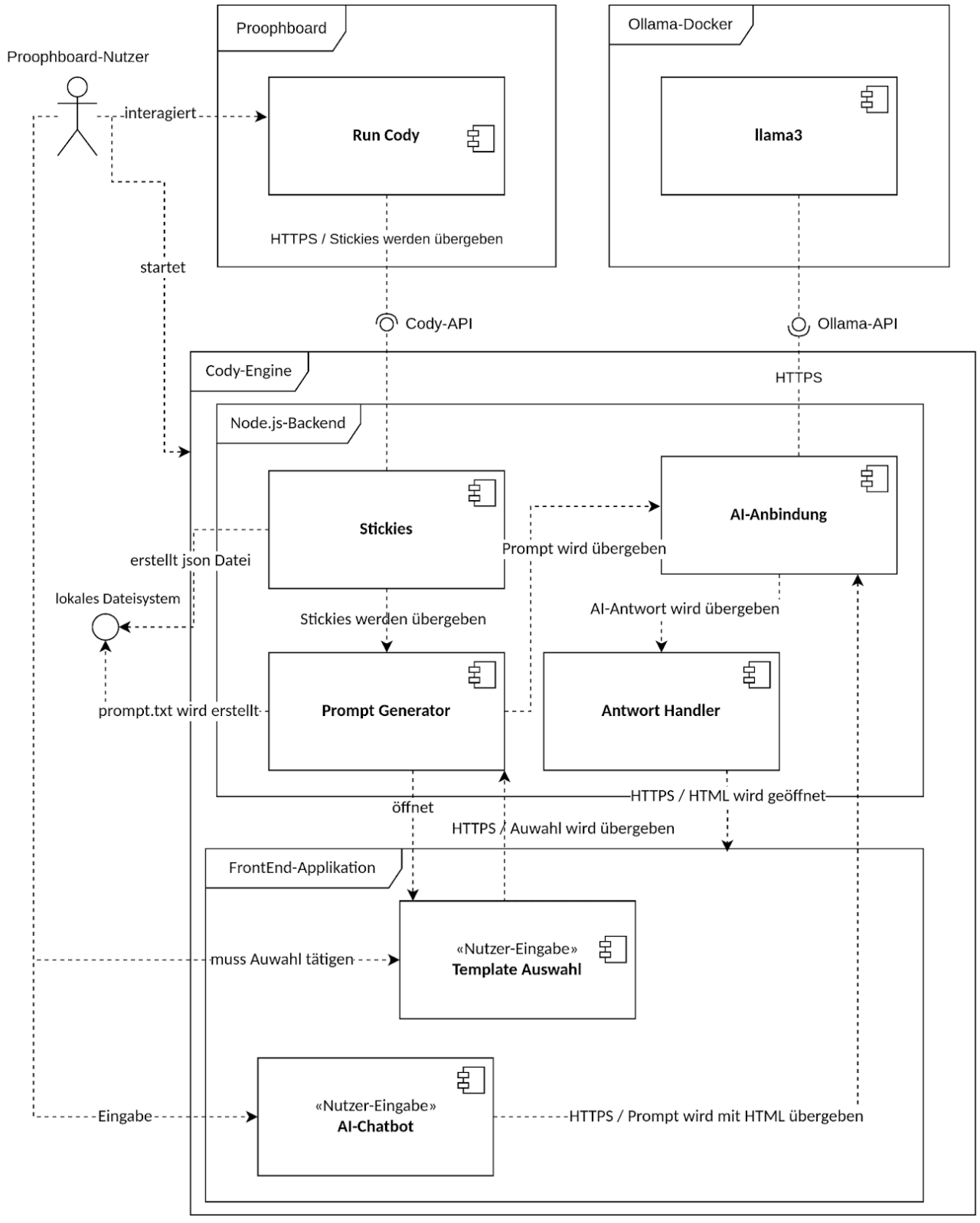


Abbildung 4: Bausteinsicht - Ebene 3

Name	Beschreibung
Stickies	Diese Komponente nimmt die Stickies, die auf dem Proophboard stehen, entgegen. Die angenommenen Stickies werden in einer .json-Datei gespeichert.
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.
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 die Ausgabe zu HTML Komponente weitergegeben.
Datenmanager	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.
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.
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.
llama3	Dies ist das Modell, das von uns für dieses Projekt benutzt wird.

Tabelle 11: Bausteinsicht-Ebene 3

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 Eingabe in den Chatbot

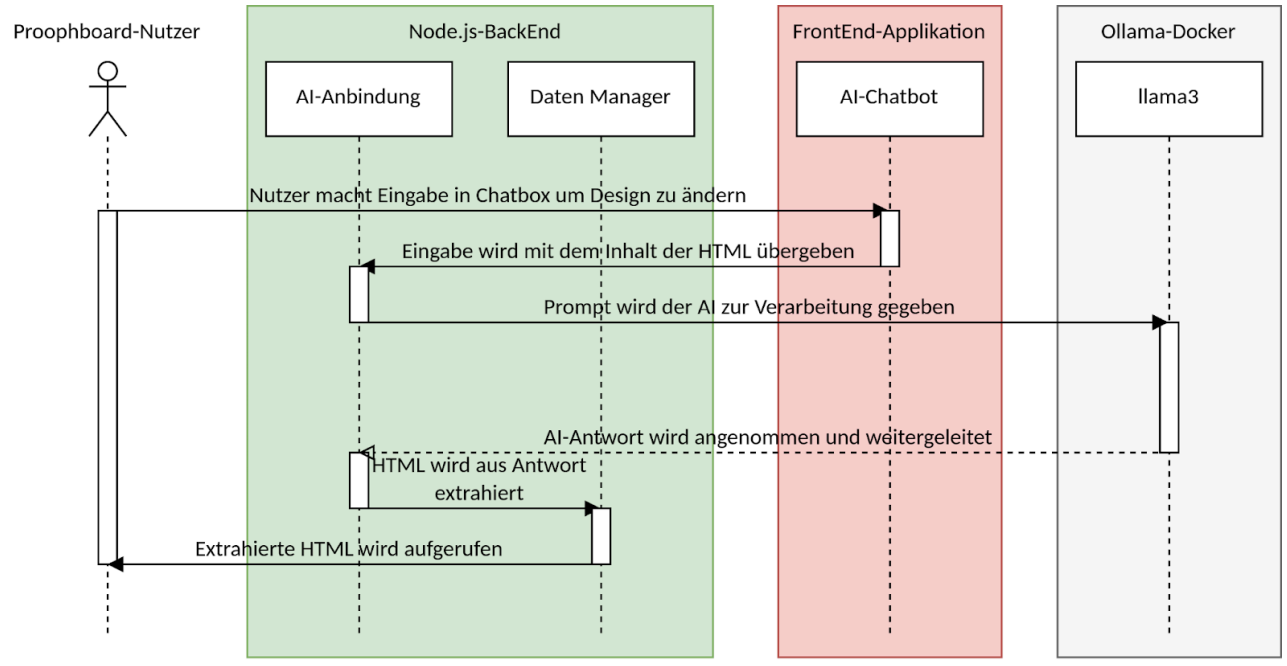


Abbildung 5: Laufzeitsicht- Eingabe in den Chatbot

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

6.2 Durchlauf des Systems

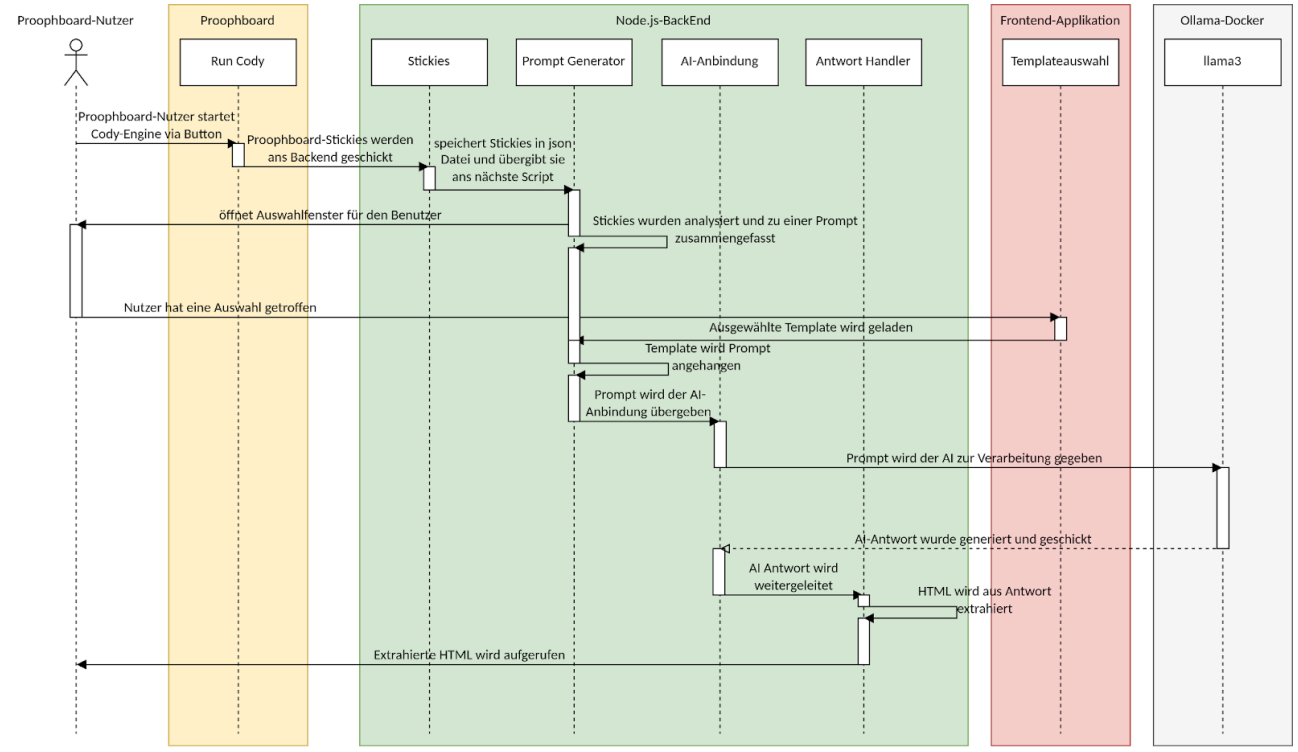


Abbildung 6: Laufzeitsicht- Durchlauf des Systems

In diesem Sequenzdiagramm wird der Ablauf des Systems simuliert. Es beginnt damit, dass der Nutzer die Cody-Engine startet, dann muss der Nutzer noch ein Template auswählen und dann wird ihm nach kurzer Zeit die generierte Seite gezeigt.

7 Verteilungsschicht

Unsere Verteilungsschicht besteht aus drei Teilen:

- Proophboard Server
- Homeserver auf dem die AI läuft
- Client-PC

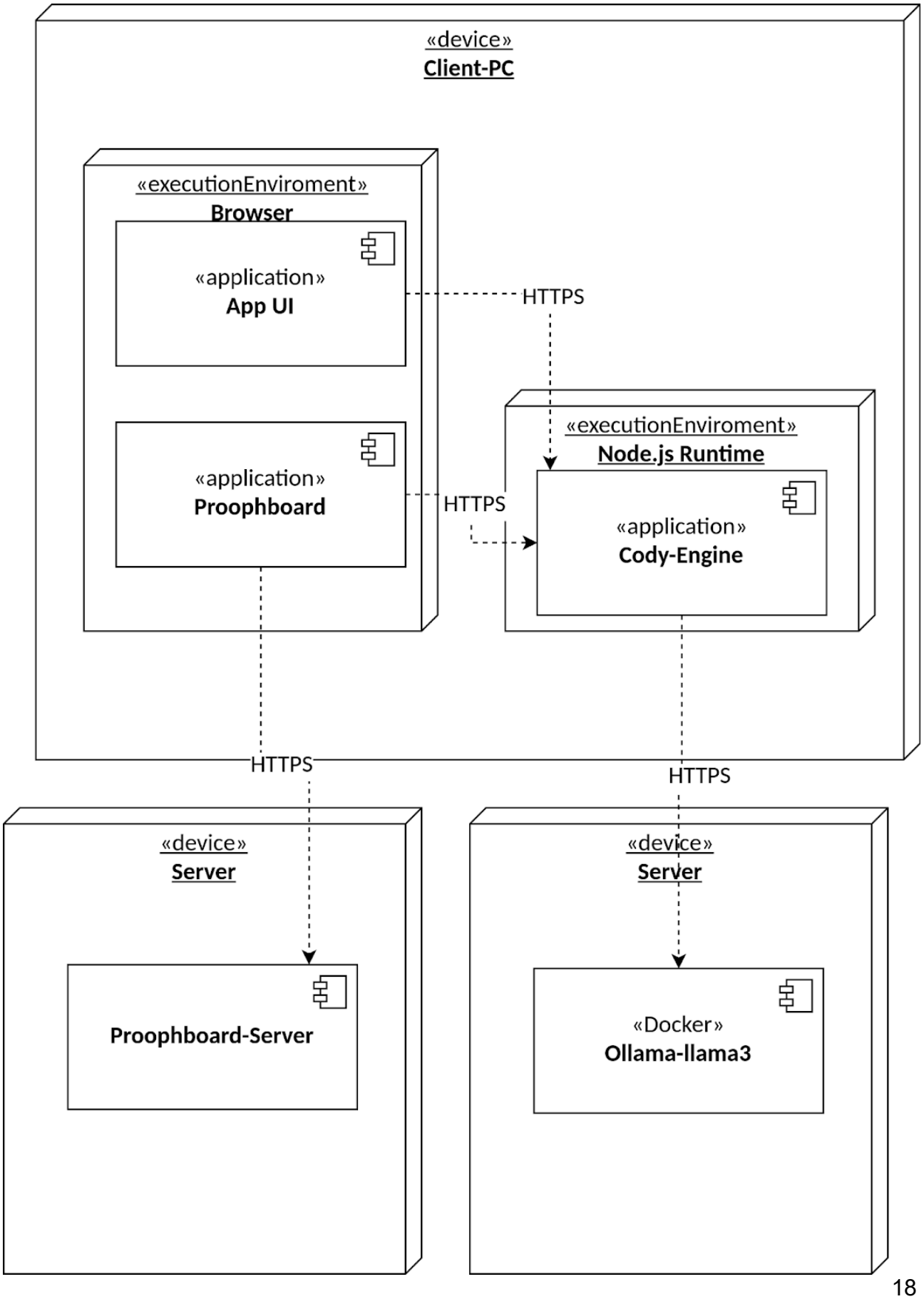

«device»
Client-PC
«executionEnviroment»
Browser
«application»
App UI
HTTPS
«application»
Proophboard
HTTPS
«executionEnviroment»
Node.js Runtime
«application»
Cody-Engine
HTTPS
HTTPS
«device»
Server
Proophboard-Server
«device»
Server
«Docker»
Ollama-llama3

Abbildung 7: Verteilungsschicht

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.
App UI	Hiermit ist unser Frontend gemeint.
Express.js Runtime	Express.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.
Docker	Ein Ollama-Docker, der auf einem 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 Back-End node.js

8.1.1 Fragestellung

Welches Back-End Webframework setzen wir ein?

8.1.2 Relevante Einflussfaktoren

- ➔ Vor allem Qualitätsziel: Änderbarkeit
- ➔ siehe: Randbedingungen
- ➔ siehe: Qualitätsziele

8.1.3 Annahmen

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

8.1.4 Entscheidungskriterien

Das benutzte Framework sollte für uns simpel, aber dennoch leistungsstark sein.
Hohe Skalierbarkeit und eine schnelle Verarbeitung der Daten gehören zu unseren schwerwiegenden Entscheidungskriterien. Sie soll ohne eine komplizierte Lernkurve schnell anwendbar und anpassbar sein, da wir für das Projekt ein Semester zur Verfügung haben.

8.1.5 Betrachtete Alternativen

Python 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 Skalierbarkeit auf, die ebenfalls unseren Kriterien entspricht.

8.2 Ollama

8.2.1 Fragestellung

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

8.2.2 Relevante Einflussfaktoren

➔ siehe: Randbedingungen
➔ siehe: Qualitätsziele

8.2.3 Annahmen

Es ist angenommen, dass die Dokumentation ausreichend ist, um mit dem Tool zu arbeiten.

8.2.4 Entscheidungskriterien

Das ausgewählte LLM-Tool sollte Open Source sein, benutzerfreundlich sein und Selbsthosting unterstützen.

8.2.5 Betrachtete Alternativen

- LocalAI
- IONOS-AI

8.2.6 Entscheidungen

Wir haben uns für Ollama entschieden, weil es einfach zu nutzen ist und einen unkomplizierten Zugriff auf öffentliche LLMs über ein Repository ermöglicht, das die Download-URLs verwaltet. 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. Im Gegensatz dazu erfordert Ollama lediglich eine Änderung der Variable.

Wir wollten auch nicht bis zur Bereitstellung der AI von bitExpert bzw. IONOS abwarten, da wir keine Gewissheit über diese AI besitzen.

8.3 JavaScript und TypeScript

8.3.1 Fragestellung

Welche Skriptsprache wollen wir für unser Softwareprojekt in Kraft setzen?

8.3.2 Relevante Einflussfaktoren

➔ siehe: Verwendung des Cody Backends
➔ Die Cody-Engine baut auf Node.js auf
➔ siehe: Qualitätsziele
➔ siehe: Randbedingungen

8.3.3 Annahmen

Die gewählten Frameworks (siehe: *Bausteinsicht -> Ebene 2*) sind mit beiden Skriptsprachen kompatibel.

8.3.4 Entscheidungskriterien

Wichtig für uns ist dabei die Leichtigkeit in der Implementierung und eine vielfältige Dokumentation, um schneller auf Fehler aufmerksam zu werden. Die Integration in bestehende

Webseiten und Anwendungen soll problemlos laufen, ohne Einsatz zusätzlicher Compiler oder Konfigurationen.

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*)

TypeScript ist eine stark typisierte Sprache, die leistungsstarke Typüberprüfungsfunktionen bietet, durch die es möglich ist, frühzeitig auf Fehler aufmerksam zu werden. Daher können größtenteils keine Laufzeitfehler erzeugt werden. Außerdem ist die Cody-Engine ausschließlich in TypeScript implementiert.
(*Informationsquellen*)

8.3.6 Entscheidungen

Die Entscheidung fällt auf JavaScript und TypeScript, 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. Und da die Entwickler der Cody-Engine das Back End in TypeScript programmiert haben.

8.4 Apache Webserver oder nginx Webserver

8.4.1 Fragestellung

Welchen Webserver wird bei uns angewandt?

8.4.2 Relevante Einflussfaktoren

- ➔ siehe: Randbedingungen
- ➔ siehe: Qualitätsziele

8.4.3 Annahmen

Nginx und *Apache* sind die zwei beliebtesten Webserver, die in Deutschland eingesetzt werden(*quelle*).

8.4.4 Entscheidungskriterium

Die Wichtigkeit einer umfassenden Dokumentation ist für unser Projekt von großer Bedeutung.

8.4.5 Betrachtete Alternativen

Die Popularität steigt mit der Zeit für nginx , vor allem macht sie sich durch die ressourcenschonende und effiziente Performance von Verbindungen beliebt, die durch die eventgesteuerte Architektur garantiert wird (*Informationsquelle*).

Der Apache Webserver ist vor allem bekannt durch seine breite Unterstützung für Skriptsprachen. Im Gegensatz zu nginx basiert Apache auf einer prozessbasierten Architektur, die sie nachteilig macht. Nichtsdestotrotz besitzt Apache durch die jahrelange Erfahrung und Anwendung eine umfangreiche Dokumentation. (*Informationsquelle*)

8.4.6 Entscheidungen

Da unsere Prämisse auf einer zahlreichen und umfassenden Dokumentation des Webservers beruht, fällt unsere Entscheidung rasch auf den Apache-Server.

9 Qualitätsanforderungen

9.1 Qualitätsszenarien

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

ID	Szenario
B01	Der Endnutzer füllt den Fragebogen zur Website-Generierung aus. Er versteht intuitiv, was mit den jeweiligen Punkten gemeint ist.
B02	Der Endnutzer möchte eine Überschrift ändern. Er sieht nach spätestens 5 Sekunden den AI-Chatbot-Button und weiss direkt, was er dort eingeben muss.
F01	Der Endnutzer möchte bei dem Buchliste-Prototyp ein Buch hinzufügen. Die Funktionalität bleibt nach Website-Generierung erhalten.
Z01	Der Endnutzer drückt nach dem Fragebogen auf den “generiere Website”-Button. Eine Website nach seinen Wünschen und Parametern wird erstellt.
Ä/W 01	Der Entwickler möchte einen Bug fixen. Durch eine ausführliche Code-Dokumentation gemäß den Code-Richtlinien können Fehler schneller gefunden und verbessert werden.
E01	Der Endnutzer nutzt den Chatbot, um die Hintergrundfarbe zu ändern. Nach maximal 15

	Sekunden soll die Farbe geändert sein.
K01	Der Endnutzer nutzt Internet Explorer, um die Seite generieren zu lassen. Die generierte Seite wird korrekt und fehlerfrei angezeigt.

Tabelle 13: Qualitätsszenarien

10 Glossar

Begriff	Definition
Apache	Apache ist der meistverbreitetste Open-Source-Webserver im Internet.
Cody	Cody ist ein Bot oder Agent, der in der Lage ist, ein Event Map Design in funktionierende Software zu übersetzen.(*quelle*)
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

12 Tabellenverzeichnis