



---

# todo

## Dokumentation der Anwendung

### Full-Stack-Webanwendungen Sommersemester 2023

---

Lucas Schiessl  
lucas.schiessl@hs-augsburg.de  
Informatik

Hannes Ziereis  
hannes.ziereis@hs-augsburg.de  
Informatik

Christoph Herb  
christoph.herb@hs-augsburg.de  
Informatik

14767 Zeichen (mit Leerzeichen)

# Inhaltsverzeichnis

<b>1</b>	<b>Allgemein</b>	<b>2</b>
1.1	Beschreibung . . . . .	2
1.2	Organisation . . . . .	2
1.3	Aufbau . . . . .	2
<b>2</b>	<b>Frontend</b>	<b>3</b>
2.1	Design . . . . .	3
2.2	Funktionen . . . . .	4
<b>3</b>	<b>Backend</b>	<b>5</b>
3.1	Funktionen . . . . .	5
3.2	Tests . . . . .	6
<b>4</b>	<b>Datenbank</b>	<b>7</b>
<b>5</b>	<b>Quellen</b>	<b>9</b>
5.1	Code-Teile von Dritten . . . . .	9
5.1.1	Frontend . . . . .	9
5.1.2	Backend . . . . .	9
5.2	Bibliotheken . . . . .	10
5.2.1	Frontend . . . . .	10
5.2.2	Backend . . . . .	11
<b>6</b>	<b>Anhang</b>	<b>12</b>
6.1	GitLab Issues . . . . .	12
6.2	API Dokumentation . . . . .	14
6.3	Erstellungserklärung . . . . .	34

# 1 Allgemein

## 1.1 Beschreibung

Ziel dieser Arbeit war es, eine Applikation für das Verwalten von Todos zu implementieren. Der Nutzer soll sich in der Webanwendung zuerst registrieren und dann über einen Login anmelden können. Nach der Anmeldung kann der User dann neue Tasks hinzufügen, vorhandene editieren und lösche sowie sich eine Gesamtübersicht aller geplanten Tasks im Kalender anzeigen lassen. Die Daten des Users können zudem in einem eigenen Fenster bearbeitet werden. Weitere Informationen dazu sind im Frontend, Backend und Datenbank-Teil dieser Anwendung beschrieben. Zum Starten und Betreiben (Default Credentials der User, Datenbank-Zugang) der Anwendung sind weitere Informationen in der *README.md* des Projekts enthalten.

## 1.2 Organisation

Für die Organisation und Verwaltung des Quellcodes wurde GitLab benutzt. Dieses bietet neben der Quellcodeverwaltung auch die Möglichkeit noch offene Aufgaben zu verwalten (Issues), CI/CD-Pipelines zu erstellen (CI/CD) und Container-Images zu speichern (Packages & Registry). Zu Beginn des Projektes wurden Issues für die einzelnen Kernfunktionen erstellt und priorisiert. Im Laufe des Projektes wurden dann für Fehler und zusätzliche Funktionen weitere Issues hinzugefügt und nacheinander abgearbeitet. So war zu jeder Zeit sichtbar, welche Aufgaben bereits abgearbeitet wurden und welche noch offen sind. Eine Übersicht aller Issues und deren zugeteilte Person findet man im Anhang 6.1.

## 1.3 Aufbau

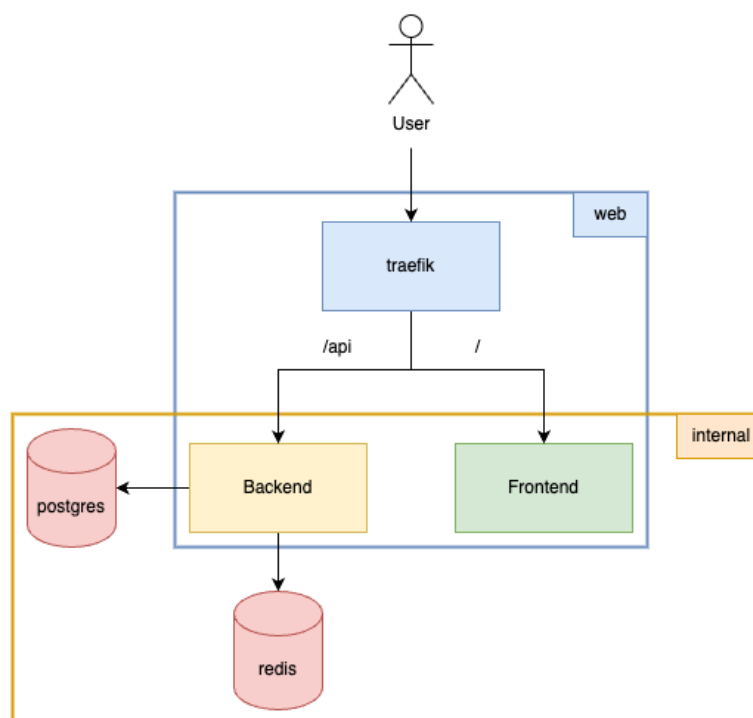
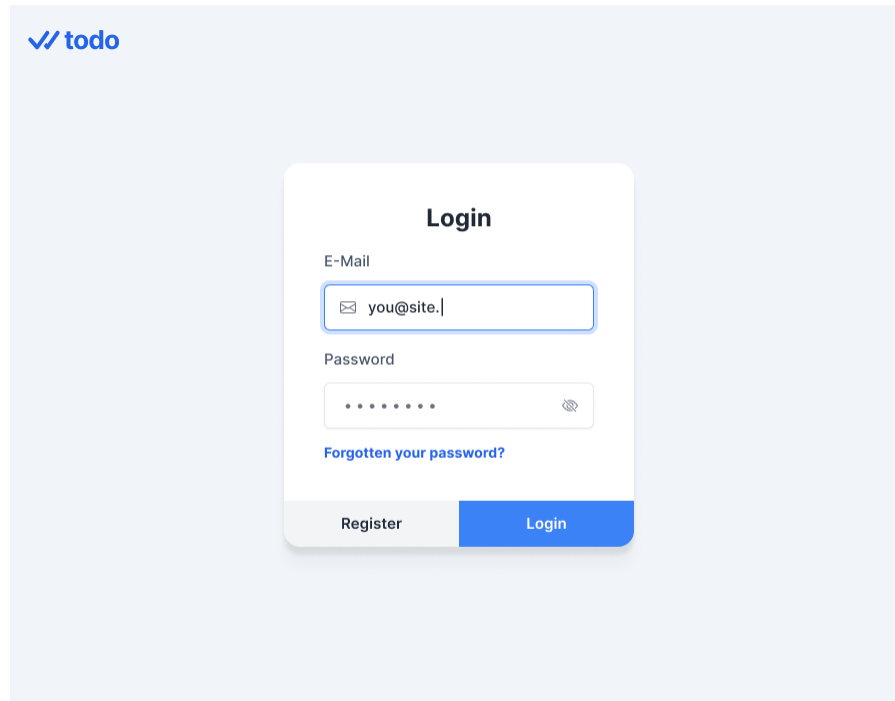


Abbildung 1: Architektur der deployten Anwendung

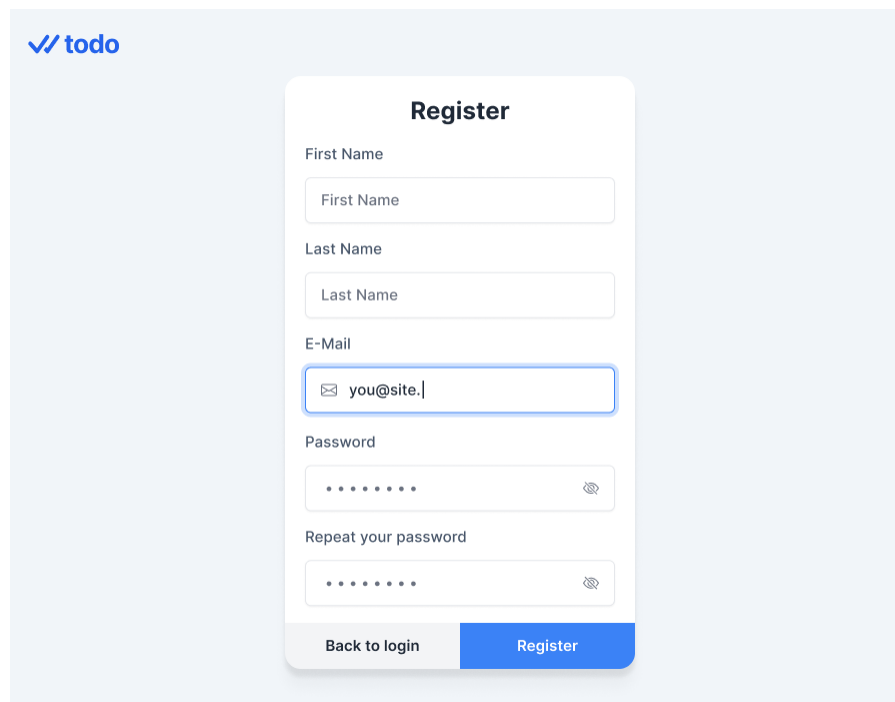
## 2 Frontend

### 2.1 Design



The image shows a Figma design of a login form. In the top left corner, there is a logo consisting of a blue checkmark followed by the text "todo". The login form itself is a white rounded rectangle centered on a light blue background. It has a title "Login" in bold black text. Below the title, there are two input fields: "E-Mail" with a placeholder "you@site." and an email icon, and "Password" with a placeholder of seven dots and an eye icon for toggling visibility. A link "Forgotten your password?" is positioned below the password field. At the bottom of the form, there are two buttons: a grey "Register" button and a blue "Login" button.

Abbildung 2: Figma Design des Logins



The image shows a Figma design of a register form. In the top left corner, there is a logo consisting of a blue checkmark followed by the text "todo". The register form is a white rounded rectangle centered on a light blue background. It has a title "Register" in bold black text. Below the title, there are four input fields: "First Name", "Last Name", "E-Mail" with a placeholder "you@site." and an email icon, and "Password" with a placeholder of seven dots and an eye icon. Below the password field, there is another input field labeled "Repeat your password" with a placeholder of seven dots and an eye icon. At the bottom of the form, there are two buttons: a grey "Back to login" button and a blue "Register" button.

Abbildung 3: Figma Design des Registrierungsformulars

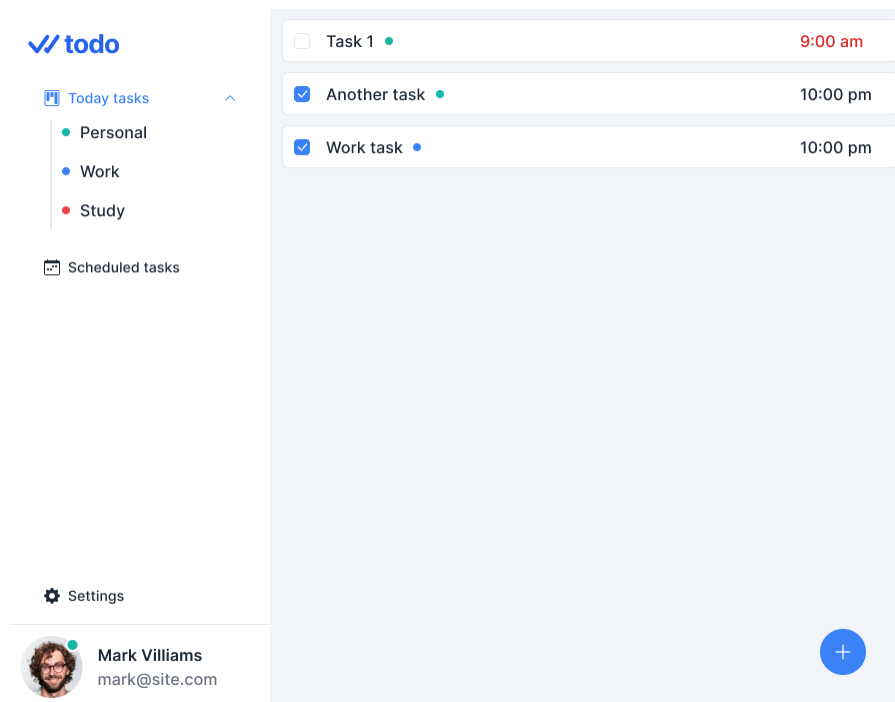


Abbildung 4: Figma Design der Hauptseite

## 2.2 Funktionen

Das Frontend kann untergliedert werden in folgende Teilbereiche:

- Login und Registrierung
- Taskübersicht
- Navigationsleiste

Für das Login wurde mit Pinia ein Auth Store und ein User Store angelegt. Der Auth Store wird dabei für Login und Logout, sowie das Speichern und Refreshen des Authentication Tokens verwendet. Der User Store wird verwendet um den aktuellen Benutzer abzufragen, einen neuen Nutzer anzulegen oder zu updaten.

Über die Login Seite hat der Nutzer die Möglichkeit sich in der Anwendung anzumelden. Zudem kann durch Klick auf Registration ein neuer Nutzer angelegt werden. Ist der Nutzer nicht authentifiziert wird er immer zur Login Seite redirected. Das Passwortfeld in der Login- und Registrierungsmaske wurde um einen Button ergänzt, der es dem Nutzer erlaubt das Passwort im Klartext anzuzeigen, falls gewünscht. Ist der Login erfolgreich, wird der Nutzer auf die Home View weitergeleitet und ein Authentication Token im lokal Storage gespeichert.

Ein neuer Benutzer kann über die Registrierungsseite angelegt werden. Dabei müssen Vor- und Nachname, Email Adresse und Passwort angegeben, sowie das Passwort bestätigt werden. Ist ein Feld leer oder nicht korrekt ausgefüllt wird eine entsprechende Fehlermeldung darunter angezeigt. Wird der Nutzer erfolgreich angelegt, wird der Nutzer auf eine Registration Success Seite weitergeleitet. Anschließend kann man sich einloggen.

Auf der Main Page befindet sich links die Navigationsleiste und rechts die Tasks des aktuell eingeloggten Nutzers. Auf der Navigationsleiste kann der Nutzer zwischen einer Listen und einer Kalender Ansicht der Tasks wählen. Zudem können die Tasks nach den ihnen zugewiesenen Tags gefiltert werden. Am unteren Rand der Navbar wird der Name, die Email Adresse und die Initialen

des aktuell eingeloggten Nutzers angezeigt. Die Initialen könnten in einer zukünftigen Version durch ein Profilbild ersetzt werden. Über dieser Anzeige hat der Nutzer die Möglichkeit sich wieder auszuloggen, oder die Einstellungen zu ändern.

Die Settings Seite bietet dem Benutzer die Möglichkeit seinen Namen und sein Passwort, sowie die Sprache der App zu ändern. Die Texte der App wurden mit dem `i18n` Plugin für Vue implementiert, so dass zusätzliche Sprachen einfach hinzugefügt werden können. Aktuell unterstützt werden Deutsch und Englisch.

Auf der rechten Seite der Main Page befinden sich die Tasks in einer Liste. In der Liste werden der Titel und falls vorhanden eine verkürzte Beschreibung, der Tag und das Fälligkeitsdatum der Aufgabe angezeigt. Klickt man auf einen Task, öffnet sich eine Detail Ansicht als Modal Fenster. In dieser werden alle Attribute des Tasks angezeigt. Zudem hat der Nutzer die Möglichkeit den Task zu löschen, oder zu bearbeiten.

Wenn das Modal während des Bearbeiten geschlossen wird, entweder durch den X-Button oder durch klicken außerhalb des Modals, bleiben die vorgenommen Änderungen erhalten und können bei erneutem öffnen des Tasks weiter bearbeitet werden. Die Persistierung der Änderungen findet im `localStorage` des Browsers statt.

Ein neuer Task kann durch das unterste Item der Task Liste angelegt werden. Dafür wird mindestens ein Titel benötigt, alle anderen Attribute können durch das Ausklappen des Items eingetragen werden.

Klickt man auf `Scheduled Tasks` wird die Task Liste durch eine Kalender Ansicht ersetzt. Hier werden allerdings nur die Tasks angezeigt, die ein Fälligkeitsdatum als Attribut besitzen. Falls der Task zudem einen zugewiesenen Tag hat, wird der Task in der entsprechenden Farbe angezeigt. Auch in der Kalender Ansicht kann durch Klick auf den Task das Task Detail Modal geöffnet werden.

## 3 Backend

### 3.1 Funktionen

Das Backend ist in TypeScript implementiert und verwendet das Prisma ORM für den Datenbankzugriff.

Im Backend gibt es verschiedene Service-Module wie `user.service.ts` und `task.service.ts`, die die Logik für die Benutzerverwaltung und die Aufgabenverwaltung enthalten. Der `user.service.ts` ermöglicht das Erstellen, Lesen, Aktualisieren und Löschen von Benutzerdaten. Der `task.service.ts` ermöglicht das Erstellen, Lesen, Aktualisieren und Löschen von Aufgaben. Beide Services arbeiten eng mit der Prisma-Datenbank zusammen, um die Daten persistent zu speichern.

Die Routenmodule wie `user.route.ts` und `task.route.ts` definieren die API-Endpunkte für die Benutzerverwaltung und die Aufgabenverwaltung. Diese Endpunkte sind durch Sicherheitsmechanismen wie JWT-Authentifizierung geschützt. Die Routenmodule rufen die entsprechenden Funktionen aus den Service-Modulen auf und geben die Ergebnisse als JSON an die Client-Anwendung zurück.

Das Backend verwendet auch verschiedene Hilfsmodule wie `helpers.ts`, um Validierungen und andere allgemeine Aufgaben zu unterstützen. Es gibt auch Exception-Module, die spezifische Fehlerklassen wie `NotFoundError` oder `ValidationError` enthalten, um Fehler in der API-Behandlung zu verwalten und entsprechende HTTP-Statuscodes und Fehlerantworten zu generieren. Die komplette API des Backends wurde außerdem mithilfe von Swagger dokumentiert und kann nach dem Start im DEV Modus über die URL `http://localhost:3000/api-docs` aufgerufen werden. Außerdem

wurde die Dokumentation als PDF exportiert und kann im Anhang 6.1 dieser Dokumentation gefunden werden.

## 3.2 Tests

Die Verwendung von Cucumber.js ermöglicht es, Tests in einer natürlichen Sprache zu schreiben, die für alle Projektbeteiligten leicht verständlich ist. Die Testszenarien werden in sogenannten Feature-Dateien definiert, während die Schritte in den Step-Definitionen implementiert werden.

Die Feature-Datei (`task.feature`) enthält beschreibende Szenarien, die die verschiedenen Anwendungsfälle für Tasks abdecken. Jedes Szenario besteht aus einem Titel, einer Beschreibung und einer Liste von Schritten. Die Schritte beschreiben den Zustand, die Aktion und die erwartete Überprüfung für jeden Schritt.

Die Step-Definitionen (`api_steps.ts`) enthalten die Implementierung der Schritte aus den Feature-Dateien. Hier wird die Testlogik für jeden Schritt definiert, einschließlich der Interaktion mit dem zu testenden System (z.B. HTTP-Anfragen senden) und der Überprüfung des erwarteten Verhaltens (z. B. Überprüfen des Statuscodes und des Antwortformats).

Die `ApiSteps`-Klasse enthält Methoden, die mit den Annotationen `@given`, `@when` und `@then` markiert sind, um die entsprechenden Schritte abzudecken. In den Methoden werden die Aktionen und Überprüfungen ausgeführt, um den Testfall zu validieren.

Die Task-Tests folgen einem BDD-Ansatz (Behavior-Driven Development), bei dem die Testszenarien aus der Sicht des Endbenutzers formuliert werden. Die Szenarien beschreiben typische Abläufe wie das Erstellen, Aktualisieren, Löschen und Abrufen von Tasks. Die Step-Definitionen stellen sicher, dass diese Szenarien automatisch getestet werden können.

In `user.feature` und `auth.feature` sind, ähnlich wie in `task.feature` die Tests für den User- und Authentifizierungsteil des Backends definiert.

Während der Ausführung der Tests wird Cucumber.js die Feature-Dateien und die zugehörigen Step-Definitionen analysieren. Für jeden Schritt in einem Szenario wird die entsprechende Methode in den Step-Definitionen aufgerufen, um die Aktion auszuführen und die Überprüfungen durchzuführen. Die Testergebnisse werden dann zusammengefasst und ausgegeben, um zu zeigen, welche Szenarien erfolgreich waren und welche fehlgeschlagen sind. Die Tests können mit folgendem Befehl gestartet werden.

```
# Start dependencies
docker-compose up -d

# Execute tests
yarn test:integration
```

```

@auth
Feature: Authentication
  Login user, verify user, refresh token

Scenario: Login user (correct email/password)
  Given the Content-Type is 'application/json'
  When I send a POST request to "http://localhost:8000/api/v1/auth/login" with json:
    """
    {
      "email": "admin@todo.com",
      "password": "admin"
    }
    """
  Then the response code should be 200
  And the response body should be json:
    """
    {
      "tokenType": "Bearer",
      "accessToken": String
    }
    """

```

Abbildung 5: Beispiel eines Cucumber Integration Tests

## 4 Datenbank

*Als Datenbank wurde eine Postgresql DB verwendet. Da für den Datenbankzugriff Prisma ORM verwendet wurde, sind die Tabellen in der backend/prisma/schema.prisma Datei definiert. Um Testdaten in die Datenbank zu schreiben wird die Datei backend/prisma/seed.ts einmalig beim Hochfahren der Datenbank ausgeführt. Die Tabellen und ihre Relationen können in der folgenden Grafik eingesehen werden:*



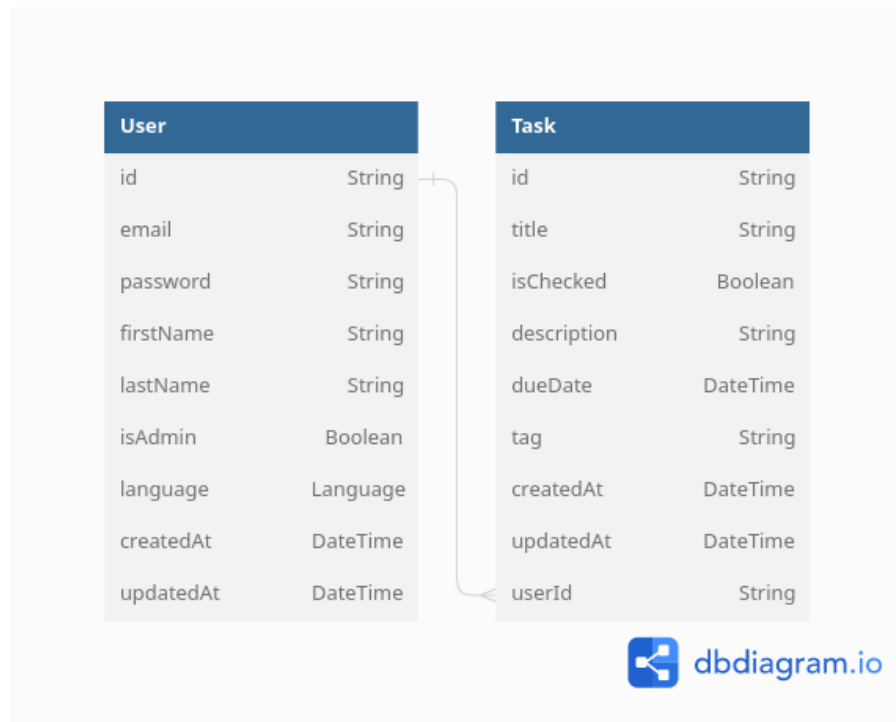


Abbildung 6: Datenbank Diagramm

## 5 Quellen

### 5.1 Code-Teile von Dritten

#### 5.1.1 Frontend

- *dockerfile for production*: <https://medium.com/bb-tutorials-and-thoughts/how-to-serve-vue-js-application-with-nginx-and-docker-d8a872a02ea8>
- *nginx config*: <https://www.appsyoda.com/blog/deploying-vuejs-app-using-nginx/>
- *combobox styled*: <https://headlessui.com/react/combobox>

#### 5.1.2 Backend

- <https://github.com/domideimel/error-middleware> (not maintained anymore)
- *help from here*: <https://www.elliотdenolf.com/blog/cucumberjs-with-typescript>
- *redis singleton client*: <https://stackoverflow.com/questions/54240635/how-to-make-express-js-app-connect-redis-only-1-time-when-the-app-start-without>
- *how to emulate object enums*: <https://stackoverflow.com/questions/41179474/use-object-literal-as-typescript-enum-values>
- *prisma middleware for hashing passwords*: <https://stackoverflow.com/questions/69233726/cannot-hash-the-users-password-with-prisma-middleware-in-nestjs-on-create-user>

## 5.2 Bibliotheken

### 5.2.1 Frontend

```
1  "dependencies": {
2    "@headlessui/tailwindcss": "^0.1.3",
3    "@headlessui/vue": "^1.7.14",
4    "@heroicons/vue": "^2.0.17",
5    "@preline/overlay": "^1.4.0",
6    "@vuepic/vue-datepicker": "^5.2.0",
7    "@vueuse/core": "^10.1.2",
8    "autoprefixer": "^10.4.14",
9    "axios": "^1.3.4",
10   "pinia": "^2.0.33",
11   "postcss": "^8.4.21",
12   "preline": "^1.7.0",
13   "storejs": "^2.0.5",
14   "tailwindcss": "^3.2.7",
15   "vue": "^3.2.47",
16   "vue-final-modal": "^4.4.2",
17   "vue-i18n": "9",
18   "vue-router": "^4.1.6",
19   "vue-simple-calendar": "^6.3.1"
20 },
21 "devDependencies": {
22   "@rushstack/eslint-patch": "^1.2.0",
23   "@types/jsdom": "^21.1.0",
24   "@types/node": "^18.14.2",
25   "@vitejs/plugin-vue": "^4.0.0",
26   "@vue/eslint-config-typescript": "^11.0.2",
27   "@vue/test-utils": "^2.3.0",
28   "@vue/tsconfig": "^0.1.3",
29   "autoprefixer": "^10.4.14",
30   "cypress": "^12.7.0",
31   "eslint": "^8.34.0",
32   "eslint-plugin-cypress": "^2.12.1",
33   "eslint-plugin-vue": "^9.9.0",
34   "jsdom": "^21.1.0",
35   "npm-run-all": "^4.1.5",
36   "postcss": "^8.4.21",
37   "start-server-and-test": "^2.0.0",
38   "tailwindcss": "^3.2.7",
39   "typescript": "^4.8.4",
40   "vite": "^4.1.4",
41   "vitest": "^0.29.1",
42   "vue-tsc": "^1.2.0"
43 }
```

## 5.2.2 Backend

```
1  "dependencies": {
2    "@prisma/client": "^4.11.0",
3    "class-validator": "^0.14.0",
4    "cookie-parser": "^1.4.6",
5    "datejs": "^1.0.0-rc3",
6    "dayjs": "^1.11.8",
7    "dotenv-cli": "^7.1.0",
8    "express": "^4.18.2",
9    "express-actuator": "^1.8.4",
10   "express-async-handler": "^1.2.0",
11   "express-jsdoc-swagger": "^1.8.0",
12   "express-promise-router": "^4.1.1",
13   "jsonwebtoken": "^9.0.0",
14   "ms": "^2.1.3",
15   "prisma": "^4.11.0",
16   "redis": "^4.6.6",
17   "tslog": "^4.8.2"
18 },
19 "devDependencies": {
20   "@cucumber/cucumber": "^9.0.1",
21   "@types/chai": "^4.3.5",
22   "@types/chai-json-pattern": "^1.1.0",
23   "@types/cookie-parser": "^1.4.3",
24   "@types/cucumber": "^7.0.0",
25   "@types/datejs": "^0.0.33",
26   "@types/express": "^4.17.17",
27   "@types/express-actuator": "^1.8.0",
28   "@types/jsonwebtoken": "^9.0.1",
29   "@types/ms": "^0.7.31",
30   "@types/node": "^18.15.5",
31   "@types/swagger-ui-express": "^4.1.3",
32   "axios": "^1.3.4",
33   "chai": "^4.3.7",
34   "chai-json-pattern": "^1.1.0",
35   "cucumber-html-reporter": "^6.0.0",
36   "cucumber-tsflow": "^4.0.0-rc.11",
37   "nodemon": "^2.0.21",
38   "ts-node": "^10.9.1",
39   "typescript": "^5.0.2"
40 }
```

## 6 Anhang

### 6.1 GitLab Issues

<i>Title</i>	<i>Assignee</i>
<i>FE Install Tailwindcss</i>	<i>Lucas Schießl</i>
<i>BE Create blueprint for backend</i>	<i>Christoph Herb</i>
<i>BE Add postgresql</i>	<i>Lucas Schießl</i>
<i>BE Add refresh endpoint</i>	<i>Christoph Herb</i>
<i>BE Add getting started documentation</i>	<i>Christoph Herb</i>
<i>DOC Create documentation</i>	<i>Christoph Herb</i>
<i>FE Create Figma Design</i>	<i>Christoph Herb</i>
<i>BE Create Database model</i>	<i>Hannes Ziereis</i>
<i>BE Implement swagger docs or create REST-API diagram manually</i>	<i>Lucas Schießl</i>
<i>FE Implement basic design</i>	<i>Lucas Schießl</i>
<i>FE First interaction with BE</i>	<i>Christoph Herb</i>
<i>FE Implement authentication with BE</i>	<i>Lucas Schießl</i>
<i>FE Cleanup frontend</i>	<i>Christoph Herb</i>
<i>FE Implement Nav Bar and design Login Form</i>	<i>Lucas Schießl</i>
<i>BE Endpoints for tasks</i>	<i>Hannes Ziereis</i>
<i>FE Interaction with tasks</i>	<i>Hannes Ziereis</i>
<i>BE Implement better logger</i>	<i>Christoph Herb</i>
<i>BE Hash password before saving into DB</i>	<i>Christoph Herb</i>
<i>FE Implement login in frontend</i>	<i>Lucas Schießl</i>
<i>FE style login form</i>	<i>Lucas Schießl</i>
<i>FE create Registration Form</i>	<i>Lucas Schießl</i>
<i>BE Add logout endpoint</i>	<i>Christoph Herb</i>
<i>FE Add error handling to login and registration form</i>	<i>Lucas Schießl</i>
<i>BE Fix tasks endpoint, swagger docs and cucumber tests</i>	<i>Christoph Herb</i>
<i>BE, FE Add docker support</i>	<i>Christoph Herb</i>
<i>FE Call Logout and refresh endpoints from frontend</i>	<i>Lucas Schießl</i>
<i>BE Fix bugs in backend</i>	<i>Christoph Herb</i>
<i>DOC Add postman collection</i>	<i>Christoph Herb</i>
<i>FE Add more functionality to task bar</i>	<i>Lucas Schießl</i>
<i>FE Add functionality to settings page</i>	<i>Lucas Schießl</i>
<i>FE Refactoring</i>	<i>Lucas Schießl</i>
<i>FE Refresh does not work as expected</i>	<i>Lucas Schießl</i>
<i>BE, FE Fix docker prod environment</i>	<i>Christoph Herb</i>
<i>BE Error messages for frontend</i>	<i>Christoph Herb</i>
<i>BE Sorting of tasks with date</i>	<i>Christoph Herb</i>
<i>BE Add filtering for categories</i>	<i>Christoph Herb</i>
<i>FE Add support for multiple languages</i>	<i>Lucas Schießl</i>
<i>FE Show taskdetails in task list</i>	<i>Hannes Ziereis</i>
<i>FE Refactor AppTask into its components</i>	<i>Hannes Ziereis</i>

*Continued on next page*

<i>Title</i>	<i>Assignee</i>
<i>BE Move password hashing to postgres</i>	<i>Christoph Herb</i>
<i>FE Cleanup</i>	<i>Christoph Herb</i>
<i>BE Generate secrets instead of hard coding</i>	<i>Christoph Herb</i>
<i>BE Add optional Tags to Tasks</i>	<i>Lucas Schießl</i>
<i>FE Fix token refresh endless loop</i>	<i>Christoph Herb</i>
<i>FE Add calendar for scheduled tasks</i>	<i>Christoph Herb</i>
<i>FE Show Tag in overview and details modal</i>	<i>Lucas Schießl</i>
<i>FE BUG - Modals opens two times</i>	<i>Christoph Herb</i>
<i>FE Detailed View - Done status doesn't update</i>	<i>Christoph Herb</i>
<i>FE Fix styling of task list</i>	<i>Christoph Herb</i>
<i>FE Edit Task in Detail modal</i>	<i>Hannes Ziereis</i>
<i>FE Improve modal</i>	<i>Christoph Herb</i>
<i>DOC Write documentation for backend</i>	<i>Hannes Ziereis</i>
<i>DOC Write documentation for frontend</i>	<i>Lucas Schießl</i>
<i>DOC Write documentation for database</i>	<i>Lucas Schießl</i>
<i>DOC Write general documentation</i>	<i>Christoph Herb</i>
<i>DOC Screencast</i>	<i>Lucas Schießl</i>
<i>FE, BE Finalize application</i>	<i>Christoph Herb</i>
<i>BUG Line breaks in description are not rendered</i>	<i>Christoph Herb</i>
<i>FE update tags</i>	<i>Christoph Herb</i>
<i>FE delete tasks</i>	<i>Hannes Ziereis</i>
<i>FE remove forgot Password from Login</i>	<i>Lucas Schießl</i>
<i>DOC Update postman collection</i>	<i>Christoph Herb</i>
<i>DOC Add infos about sources and code parts of others</i>	<i>Christoph Herb</i>

## 6.2 API Dokumentation

# Tasks API

## Overview

Backend API for the Tasks service.

MIT

## Tags

### Actuator

Infos about system

### Auth

Bearer authentication

### Me

Me endpoint

### Tasks

Task endpoint

### User

User endpoint

## Security

Type	Name	Scopes
apiKey	BearerAuth	

## Paths

### **POST** /api/v1/auth/login Login

#### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links

Code	Description	Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

## **GET /api/v1/auth/verify** Verify logged in user

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
<b>apiKey</b>	BearerAuth	

## **GET /api/v1/auth/refresh** Refresh token after expiration of your access token

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links



Code	Description	Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

## **GET /api/v1/auth/logout** Logout user

### Responses

Code	Description	Links
204	success response	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## **GET /api/v1/me** Get infos about current logged in user

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links

Code	Description	Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
<b>apiKey</b>	BearerAuth	

## **GET /api/v1/me/tasks** Get tasks of current logged in user

### Parameters

Type	Name	Description	Schema
<b>query</b>	<b>orderBy</b> <i>optional</i>	orderBy	enum ( asc,desc)
<b>query</b>	<b>sortBy</b> <i>optional</i>	sortBy	enum ( dueDate,isChecked,createdAt,updatedAt)
<b>query</b>	<b>tag</b> <i>optional</i>	tag	string
<b>query</b>	<b>isChecked</b> <i>optional</i>	isChecked	enum ( false,true)

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
<b>apiKey</b>	BearerAuth	

## **POST /api/v1/me/tasks** Create a new Task with current user

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
400	Validation error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
<b>apiKey</b>	BearerAuth	

## **GET /api/v1/me/tags** Get Tags of tasks of current logged in user

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
<b>apiKey</b>	BearerAuth	

## GET /api/v1/tasks Get all Tasks

### Parameters

Type	Name	Description	Schema
query	<b>orderBy</b> <i>optional</i>	orderBy	enum ( asc,desc)
query	<b>sortBy</b> <i>optional</i>	sortBy	enum ( dueDate,isChecked,createdAt,updatedAt)
query	<b>tag</b> <i>optional</i>	tag	string
query	<b>isChecked</b> <i>optional</i>	isChecked	enum ( false,true)

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## POST /api/v1/tasks Create a new Task

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
400	Validation error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## **GET** /api/v1/tasks/{taskId} Get specific task

### Parameters

Type	Name	Description	Schema
path	taskId <i>optional</i>	Task ID	string

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
404	NotFound error  <i>Content</i> <b>application/json</b>	No Links

Code	Description	Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
<b>apiKey</b>	BearerAuth	

## **PUT** **/api/v1/tasks/{taskId}** Update a Task

### Parameters

Type	Name	Description	Schema
<b>path</b>	<b>taskId</b> <i>optional</i>	Task ID	string

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
400	Bad Request  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
404	NotFound error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## ***DELETE*** **/api/v1/tasks/{taskId}** Delete a Task

### Parameters

Type	Name	Description	Schema
path	<b>taskId</b> <i>optional</i>	Task ID	string

### Responses

Code	Description	Links
204	success response	No Links
400	Bad Request  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
404	NotFound error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## **PATCH** /api/v1/tasks/{taskId}/toggle Toggle the isChecked field

### Parameters

Type	Name	Description	Schema
path	taskId <i>optional</i>	Task ID	string

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
400	Bad Request  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
404	NotFound error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## **GET** /api/v1/users Get all users

### Responses



Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## **POST** /api/v1/users Create a new user

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
400	Validation error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

## **GET** /api/v1/users/{userId} Get user by id

### Parameters

Type	Name	Description	Schema
path	<b>userId</b> <i>optional</i>	userId	string

#### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
404	Not Found error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## **PUT** /api/v1/users/{userId} Update user by id

#### Parameters

Type	Name	Description	Schema
path	<b>userId</b> <i>optional</i>	userId	string

#### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
404	Not Found error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
<b>apiKey</b>	BearerAuth	

## **DELETE** **/api/v1/users/{userId}** Delete user by id

### Parameters

Type	Name	Description	Schema
<b>path</b>	<b>userId</b> <i>optional</i>	userId	string

### Responses

Code	Description	Links
204	success response	No Links
401	Unauthorized error  <i>Content</i> <b>application/json</b>	No Links

Code	Description	Links
403	Forbidden error  <i>Content</i> <b>application/json</b>	No Links
404	Not Found error  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

Type	Name	Scopes
apiKey	BearerAuth	

## **GET /actuator/health** Check health status

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

## **GET /actuator/info** Get infos about backend

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

## **GET /actuator/metrics** Get metrics about backend

### Responses

Code	Description	Links
200	success response  <i>Content</i> <b>application/json</b>	No Links
500	Internal Server error  <i>Content</i> <b>application/json</b>	No Links

## Components

### Schemas

#### BaseError

BaseError

#### Properties

Name	Description	Schema
errorCode <i>optional</i>	ErrorCode	number
errorMessage <i>optional</i>	ErrorMessage	string
details <i>optional</i>	Details	< <a href="#">ErrorSchema</a> > array

#### LoginSchema

LoginSchema

#### Properties

Name	Description	Schema
email <i>required</i>	Email	string
password <i>optional</i>	Password	string

## AuthLoginSchema

AuthLoginSchema

### Properties

Name	Description	Schema
tokenType <i>optional</i>	Type of token	string
accessToken <i>optional</i>	Access token	string

## JwtPayloadSchema

JwtPayloadSchema

### Properties

Name	Description	Schema
userId <i>optional</i>	UserId	string
email <i>optional</i>	Email	string
isAdmin <i>optional</i>	IsAdmin	boolean

## ErrorSchema

ErrorSchema

### Properties

Name	Description	Schema
field <i>optional</i>	Field	string
value <i>optional</i>	Value <i>nullable</i>	string
replyCode <i>optional</i>	ReplyCode <i>nullable</i>	number
replyMessage <i>optional</i>	ReplyMessage <i>nullable</i>	string

## CreateTaskSchema

CreateTaskSchema

### Properties

Name	Description	Schema
------	-------------	--------

Name	Description	Schema
title <i>required</i>	Title	string
userId <i>required</i>	UserId	string
description <i>optional</i>	Description	string
dueDate <i>optional</i>	Due date	string (date-time)
tag <i>optional</i>	Tag	string

## CreateTaskMeSchema

CreateTaskMeSchema

*Properties*

Name	Description	Schema
title <i>required</i>	Title	string
description <i>optional</i>	Description	string
dueDate <i>optional</i>	Due date	string (date-time)
tag <i>optional</i>	Tag	string

## UpdateTaskSchema

UpdateTaskSchema

*Properties*

Name	Description	Schema
title <i>optional</i>	Title <i>nullable</i>	string
description <i>optional</i>	Description <i>nullable</i>	string
dueDate <i>optional</i>	Due date <i>nullable</i>	string (date-time)
isChecked <i>optional</i>	Is completed? <i>nullable</i>	boolean
tag <i>optional</i>	Tag <i>nullable</i>	string

## ReadTaskSchema

ReadTaskSchema

### Properties

Name	Description	Schema
id <i>optional</i>	ID	string
userId <i>optional</i>	UserId	string
title <i>optional</i>	Title	string
description <i>optional</i>	Description	string
dueDate <i>optional</i>	Due date	string (date-time)
isChecked <i>optional</i>	Is Checked	boolean
tag <i>optional</i>	Tag	string

## GetTasksWithSpecifiedTagSchema

GetTasksWithSpecifiedTagSchema

### Properties

Name	Description	Schema
tag <i>optional</i>	Tag	string

## CreateUserSchema

CreateUserSchema

### Properties

Name	Description	Schema
email <i>required</i>	Email	string
firstName <i>required</i>	First Name	string
lastName <i>required</i>	Last Name	string
password <i>required</i>	Pasword	string

## UpdateUserSchema

UpdateUserSchema



#### Properties

Name	Description	Schema
email <i>optional</i>	Email	string
firstName <i>optional</i>	First Name	string
lastName <i>optional</i>	Last Name	string
password <i>optional</i>	Password	string

### ReadUserSchema

ReadUserSchema

#### Properties

Name	Description	Schema
id <i>optional</i>	Id	string
email <i>optional</i>	Email	string
firstName <i>optional</i>	First Name	string
lastName <i>optional</i>	Last Name	string
isAdmin <i>optional</i>	IsAdmin	boolean

### ActuatorHealthSchema

ActuatorHealthSchema

#### Properties

Name	Description	Schema
status <i>optional</i>	Status	string
details <i>optional</i>	Details	object

### ActuatorInfoBuildSchema

ActuatorInfoBuildSchema

#### Properties

Name	Description	Schema
description <i>optional</i>	Description	string

Name	Description	Schema
name <i>optional</i>	Name	string
version <i>optional</i>	Version	string

## ActuatorInfoSchema

ActuatorInfoSchema

*Properties*

Name	Description	Schema
build <i>optional</i>		<a href="#">ActuatorInfoBuildSchema</a>

## ActuatorMetricsSchema

ActuatorMetricsSchema

*Properties*

Name	Description	Schema
mem <i>optional</i>	Memory	object
uptime <i>optional</i>	Uptime	string

### 6.3 Erstellungserklärung

#### Erklärung zur Abschlussarbeit

Hiermit versichere ich, die eingereichte Abschlussarbeit selbständig verfasst und keine andere als die von mir angegebenen Quellen und Hilfsmittel benutzt zu haben. Wörtlich oder inhaltlich verwendete Quellen wurden entsprechend den anerkannten Regeln wissenschaftlichen Arbeitens zitiert. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht anderweitig als Abschlussarbeit eingereicht wurde.

Das Merkblatt zum Täuschungsverbot im Prüfungsverfahren der Hochschule Augsburg habe ich gelesen und zur Kenntnis genommen. Ich versichere, dass die von mir abgegebene Arbeit keinerlei Plagiate, Texte oder Bilder umfasst, die durch von mir beauftragte Dritte erstellt wurden.

Augsburg, 06.07.2023  
Ort, Datum

C. Ab. Lucas Schipf, Zsch  
Unterschrift des/der Studierenden