Inhaltsverzeichnis

[1. Projektkonzept 2](#_Toc182487090)

[2. Architektur 3](#_Toc182487091)

[3. Technologien 4](#_Toc182487092)

[4. Zeitabschätzung 6](#_Toc182487093)

[5. Anleitung 7](#_Toc182487094)

[*5.1 Einrichtung der Datenbank* 7](#_Toc182487095)

[*5.2 Start der Webseite* 8](#_Toc182487096)

[*5.3 Anmeldung* 8](#_Toc182487097)

[*5.4 Projekt erstellen/bearbeiten* 8](#_Toc182487098)

[*5.6 Die verschiedenen Rollen* 8](#_Toc182487099)

[6. Dokumentation 9](#_Toc182487100)

[Server.js beinhaltet 7 wichtige Funktionalitäten 9](#_Toc182487101)

[taskDetailSCript.js 9](#_Toc182487102)

[taskDeshboardScript.js 10](#_Toc182487103)

[RegistrationScript.js 10](#_Toc182487104)

[projektDashboardUser.js 11](#_Toc182487105)

[projektDasjboardAdmin.js 11](#_Toc182487106)

[loginScript.js 11](#_Toc182487107)

[projectDashboard.js 12](#_Toc182487108)

[projectBoard.js 12](#_Toc182487109)

[creatorScript.js 12](#_Toc182487110)

# Projektkonzept

Es wird eine Website erstellt, die man zum Management von Projekten verwenden kann. Dazu kann man sich individuell registrieren und dann einloggen. Anschließend werden einem die aktuellen Projekte, an denen man arbeitet und deren Fortschritt angezeigt. Wenn man auf ein einzelnes Projekt klickt, kann man das Projekt genauer sehen und auch Änderungen machen, wie beispielsweise Kommentare zu einem Projekt schreiben. Je nach Rolle hat man noch zusätzliche Funktionen, wie das zuweisen von Aufgaben an andere.

Einzelne Funktionen der Website:

* + Registrieren und Anmelden von Benutzern.
  + Rollenbasierte Zugriffskontrolle z. B. Admin oder Mitarbeiter.
  + Möglichkeit zur Zuweisung von Aufgaben an spezifische Benutzer.
  + Erstellen und Verwalten von Projekten.
  + Definieren und Zuweisen von Aufgaben innerhalb eines Projekts.
  + Setzen von Prioritäten und Deadlines für Aufgaben.
  + Fortschrittsverfolgung und Statusanzeigen (z.B. „In Bearbeitung“, „Erledigt“).
  + Möglichkeit zur Kommentierung von Aufgaben durch Benutzer.

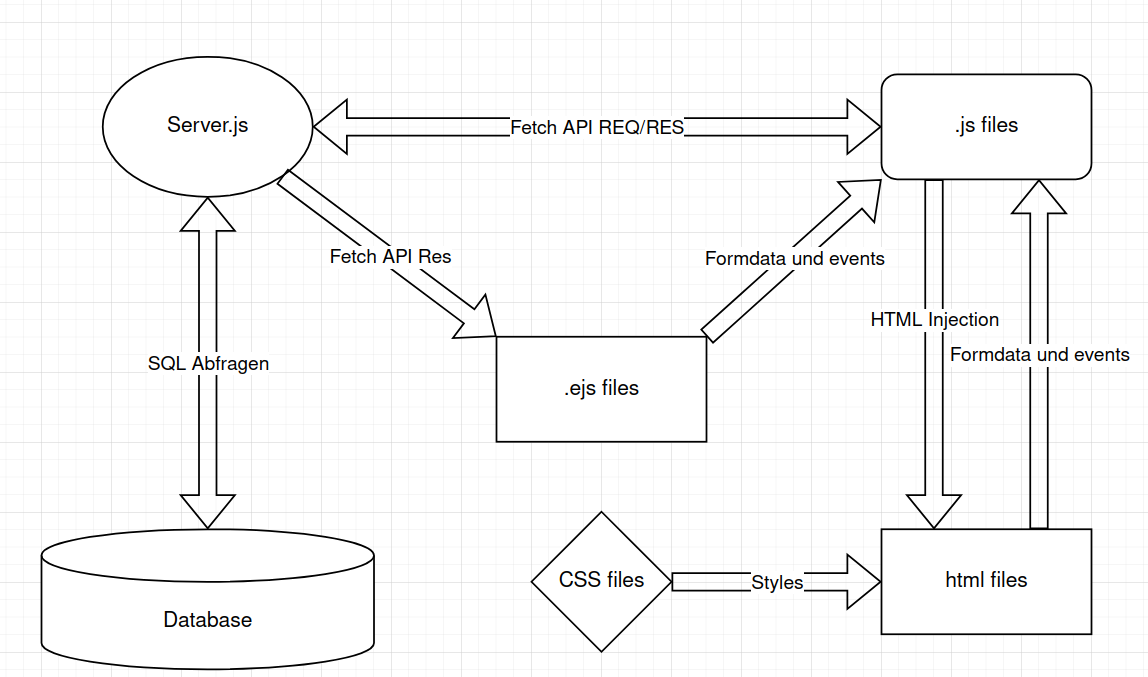
Zusätzliche Funktionen die hinzugefügt werden, sollte die Zeit dafür reichen sind:

* + Anzeige von bevorstehenden Deadlines und überfälligen Aufgaben.
  + E-Mail-Benachrichtigungen für neue Aufgaben, Statusänderungen, Kommentare und bevorstehende Deadlines.

# Architektur

Bei der Architektur wird mySQL als Datenbank verwendet, die die Daten der Projekte sowie Benutzer speichert. Mit Hilfe des Apache-Webservers von Xampp wird die Webseite gehostet.

In der untenstehenden Grafik ist die Architektur der Webseite nochmals genauer dargestellt:



# Technologien

In der nachfolgenden Tabelle sind die verschiedenen Technologien, die bei diesem Projekt Verwendung gefunden haben aufgelistet. Zusätzlich gibt es noch eine Erklärung, für was die jeweilige Technologie verwendet wird.

|  |  |
| --- | --- |
| Technologie | Erklärung |
| Node.js | Node.js ist eine serverseitige JavaScript-Laufzeitumgebung, die es ermöglicht, JavaScript-Code außerhalb eines Browsers auszuführen. |
| Xampp Apache Webserver | XAMPP ist ein kostenloses, plattformübergreifendes Paket, das den Apache Webserver umfasst. Es ermöglicht Entwicklern, eine lokale Serverumgebung einzurichten, um Webanwendungen zu testen und zu entwickeln. |
| Xampp-Datenbank | Die XAMPP-Datenbank bezieht sich in der Regel auf MySQL oder MariaDB, die im XAMPP-Paket enthalten sind. Diese relationalen Datenbanken ermöglichen es Entwicklern, Daten zu speichern, abzurufen und zu verwalten, und sind essenziell für dynamische Webanwendungen. |
| CSS | Cascading Style Sheets (CSS) ist eine Stylesheet-Sprache, die verwendet wird, um das Aussehen und Layout von Webseiten zu gestalten. Mit CSS können Farben, Schriftarten, Abstände und andere visuelle Aspekte von HTML-Elementen definiert werden. |
| HTML | Hypertext Markup Language (HTML) ist die Standard-Auszeichnungssprache zur Erstellung von Webseiten. Sie strukturiert den Inhalt im Web, indem sie Text, Bilder, Links und andere Elemente definiert. |
| VSCode | Visual Studio Code (VSCode) ist ein kostenloser, quelloffener Code-Editor von Microsoft, der viele Funktionen wie Syntaxhervorhebung, IntelliSense (Code-Vervollständigung), Debugging und Erweiterungen bietet. Er unterstützt eine Vielzahl von Programmiersprachen und ist besonders beliebt bei Webentwicklern. |
| GitHub | GitHub ist eine webbasierte Plattform zur Versionskontrolle und Zusammenarbeit an Softwareprojekten. Sie basiert auf Git und ermöglicht es Entwicklern, Code zu speichern, zu teilen, Änderungen nachzuverfolgen und gemeinsam an Projekten zu arbeiten. GitHub bietet auch Funktionen wie Issues, Pull Requests und Wikis, um die Projektverwaltung zu erleichtern. |
| .ejs Files | .ejs-Dateien sind Template-Dateien, die mit EJS (Embedded JavaScript) verwendet werden. EJS ermöglicht es Entwicklern, HTML-Seiten dynamisch zu generieren, indem sie JavaScript-Code in HTML einbetten. |

# Zeitabschätzung

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Aufgabe | Zeit | Tatsächliche Zeit | Saskia | Moritz | | Hamidreza |
| Dokumentation mit Anleitung | 12 | 11 | 10 | 1 | | 1 |
| Benutzerrollen Anmeldung | 12 | 15 | 1 | | 8 | 4 |
| Startseite zum Einloggen | 10 | 14 | 1 | | 8 | 1 |
| Übersicht alle Projekte | 15 | 16 | 2 | | 3 | 10 |
| Details von einem Projekt | 20 | 18 | 7 | | 4 | 9 |
| Abschließender Test | 6 | 8 | 2 | | 2 | 2 |
| Überarbeitung der Doku | 3 | 3 | 1 | | 1 | 1 |
| Abgabe | 1 | 1 | 0 | | 1 | 0 |
| Funktionen zusammenbringen | 15 | 16 | 3 | | 6 | 6 |
| E-Mail-Benachrichtigung | 16 | 12 | 8 | 4 | | 4 |
| Design | 10 | 11 | 6 | 2 | | 2 |
|  | | | | | | |
| Insgesamt | 120 | 125 | 40 | 40 | | 40 |

# Anleitung

## 5.1 Einrichtung der Datenbank

Xampp starten. Dort auch MySQL starten und über Admin auf phpmyAdmin gehen. Eine Datenbank mit den Namen **name\_db** erstellen und **name\_db.sql** file importieren oder unter SQL innerhalb der Datenbank die folgenden Anweisungen eingeben, um die entsprechenden 4 Tabellen erstellen zu lassen.

phpMyAdmin SQL Dump

version 5.2.1

https://www.phpmyadmin.net/

--

Host: 127.0.0.1

Generation Time: Nov 16, 2024 at 12:41 AM

Server version: 11.5.2-MariaDB

PHP Version: 8.2.12

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

Database: `name\_db`

--

--------------------------------------------------------

--

Table structure for table `comments`

--

CREATE TABLE `comments` (

`text` *varchar*(5000) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`user` *varchar*(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`time` *varchar*(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`taskname` *varchar*(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

Dumping data for table `comments`

--

INSERT INTO `comments` (`text`, `user`, `time`, `taskname`) VALUES

('this task is overdue ;((', 'developer', '2024-11-16 00:38:18', 'Example overdue task'),

('Good job Team!', 'developer', '2024-11-16 00:39:43', 'Example normal task');

--------------------------------------------------------

--

Table structure for table `projects`

--

CREATE TABLE `projects` (

`projectName` *varchar*(255) NOT NULL,

`projectDetails` *text* NOT NULL,

`projectProgress` *int*(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

Dumping data for table `projects`

--

INSERT INTO `projects` (`projectName`, `projectDetails`, `projectProgress`) VALUES

('Example Project', 'this is an example project !\nWelcome to our website :)', 0);

--------------------------------------------------------

--

Table structure for table `tasks`

--

CREATE TABLE `tasks` (

`taskname` *varchar*(255) NOT NULL,

`prio` *int*(11) NOT NULL,

`owner` *varchar*(255) DEFAULT NULL,

`assigned` *varchar*(255) DEFAULT NULL,

`description` *text* DEFAULT NULL,

`status` *int*(11) DEFAULT NULL,

`projectName` *varchar*(255) DEFAULT NULL,

`deadline` datetime DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

Dumping data for table `tasks`

--

INSERT INTO `tasks` (`taskname`, `prio`, `owner`, `assigned`, `description`, `status`, `projectName`, `deadline`) VALUES

('Example normal task', 1, 'developer', 'developer', 'This is an example of a normal task that is currently being worked on and the deadline is still open :)', 4, 'Example Project', '2028-01-01 00:00:00'),

('Example overdue task', 2, 'developer', 'developer', 'deadline expired! thats why it\'s red!', 3, 'Example Project', '2020-01-01 00:00:00');

--------------------------------------------------------

--

Table structure for table `users`

--

CREATE TABLE `users` (

`id` int(11) NOT NULL,

`username` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`password` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL,

`rank` int(1) NOT NULL,

`email` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_swedish\_ci;

--

Indexes for dumped tables

--

--

Indexes for table `users`

--

ALTER TABLE `users`

ADD PRIMARY KEY (`id`);

--

AUTO\_INCREMENT for dumped tables

--

--

AUTO\_INCREMENT for table `users`

--

ALTER TABLE `users`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=14;

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

## 5.2 Konfiguration

Das Projekt in VSCode laden. In der File config.json die passenden Daten zur Datenbankverbindung eintragen, wie das Password oder der gewünschte Port, sowie die E-Mail-Adresse und das Passwort der E-Mail, die die E-Mails versendet:

## {

"db": {

"host": "localhost",

"user": "root",

"password": "",

"database": "name\_db"

},

"sessionSecret": "geheimnisvollerSchlüssel",

"sessionMaxAge": 600000,

"port": 3001,

"email": {

"user": "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_",

"pass": "\*\*\*\*\*\*\*\*\*\*\*",

"subject": {

"deadlineNear": "Aufgabe steht kurz vor Ablauf",

"deadlineMissed": "Deadline überschritten"

},

"intervalInMinutes": 1

}

}

## 5.3 Start der Webseite

Um die Website zum Laufen zu bringen muss man zum einen die Datenbank wie in 5.1 beschrieben einrichten und bei Xampp den Apache Webserver und MySQL angeschaltet haben. Im Terminal in VSCode node server.js eingeben. Eventuell gibt es in den terminalen Fehlermeldungen, dass gewisse Module nicht vorhanden sind. Diese mit npm install noch installieren. Und dann den Server über die Ausgabe im Terminal öffnen. So kommt man auf die Seite zum Anmelden/Registrieren. Sollte ein Fehler auftreten, sollte dieser in Terminal dokumentiert werden.

## 5.4 Anmeldung

Wenn man sich auf der Website zu Beginn anmeldet, bekommt der erste User die Rolle des Admins. Die weiteren User werden zu normalen Mitarbeitern. Ist die Registrierung erfolgreich erscheint unter den Eingabefelder eine Benachrichtigung, wenn man sie nicht sieht, dann ist die Bildschirm-zoom zu groß. Nach dem man sich registriert hat, kann man sich anmelden und kommt auf eine Seite, auf der alle bestehenden Projekte angezeigt werden, in denen man mitarbeitet. Sollte man den Rang des Admins ändern (es gibt keinen Admin mehr), dann gibt es auf der Website keine Möglichkeit mehr dies zu ändern. Das muss man dann direkt in der Datenbank machen. Stelle deshalb sicher, dass es immer mindesten einen Admin gibt.

## 5.4 Bedienen der Software

Auf dem Projektdashboard kann man nach dem Erstellen eines Projektes dieses noch bearbeiten. Ebenfalls können hier die verschiedenen Benutzerberechtigungen verwalten werden. Über den Details-Button gelangt man auf das Taskboard, auf dem man neue Task erstellen und alte bearbeiten kann. Task die hierbei Farblich hervorgehoben sind, haben ihre Deadline überschritten.

## 5.6 Die verschiedenen Rollen

Es gibt verschiedene Rollen wie Admin und Manager und User, die manche Funktionen nicht verwenden können. So kann der Admin beispielsweise die Mitarbeiter bearbeiten und deren Rank ändern. Für jede Rollen gibt es im Code ein eigenes Dashboard.

* Der Admin darf alles
* Der Manager der darf keine Benutzer bearbeiten
* Der User kann keine Projekte erstellen und bearbeiten, keine Benutzer bearbeiten und kann ownership und assigned bei Tasks nicht ändern

# Dokumentation

Nachfolgend sind die verschiedenen js-Files erklärt, die ein wichtiger Bestandteil der Funktionalität des Projektes sind.

## Server.js beinhaltet 7 wichtige Funktionalitäten

1. Serverkonfiguration und Middleware: Der Server wird mit Express.js auf Port 3001 konfiguriert und verwendet Middleware, um URL-kodierte Formulardaten zu parsen sowie statische Dateien aus dem `public`-Verzeichnis bereitzustellen. Die Verbindung zur MySQL-Datenbank wird eingerichtet, und Sessions werden verwaltet, um die Benutzerauthentifizierung und den Zugriffsschutz zu ermöglichen.

2. Benutzerregistrierung: Die Benutzerregistrierungsfunktion überprüft, ob der Benutzername bereits existiert, und weist neuen Benutzern einen Rang zu, bevor sie in der Datenbank gespeichert werden. Bei erfolgreicher Registrierung erhält der Benutzer eine Bestätigung.

3. Benutzerauthentifizierung: Die Login-Funktion vergleicht die eingegebenen Anmeldedaten mit denen in der Datenbank und erstellt bei erfolgreicher Authentifizierung eine Session für den Benutzer. Bei ungültigen Anmeldedaten wird eine entsprechende Fehlermeldung ausgegeben.

4. Taskverwaltung: Die Funktion zum Hinzufügen von Aufgaben prüft, ob eine Aufgabe mit demselben Namen bereits existiert, bevor sie die neue Aufgabe in die Datenbank einfügt. Es gibt auch Routen zum Abrufen und Aktualisieren von Aufgaben, die nach ihrem Status organisiert werden.

5. Kommentarverwaltung: Die Kommentarabruf-Funktion ermöglicht es Benutzern, Kommentare zu Aufgaben anzuzeigen, wobei die neuesten Kommentare zuerst angezeigt werden. Eine HTML-Darstellung der Kommentare wird erstellt und zurückgegeben, einschließlich einer Nachricht, wenn keine Kommentare vorhanden sind.

6. Projektverwaltung: Die Projektverwaltungsfunktionen ermöglichen das Hinzufügen, Aktualisieren und Abrufen von Projekten, wobei Duplikate durch Überprüfungen vermieden werden. Alle Projekte werden in einer JSON-Antwort zurückgegeben, um eine einfache Verarbeitung auf der Client-Seite zu ermöglichen.

7. Serverstart: Der Server wird am Ende der Datei gestartet und gibt eine Bestätigung in der Konsole aus, dass er läuft. Dies ermöglicht die Reaktion auf eingehende Anfragen und die Bereitstellung aller definierten Routen und Funktionen.

## taskDetailSCript.js

1. Laden von Kommentaren: Die Funktion `loadComments` ist dafür zuständig, Kommentare für eine spezifische Aufgabe abzurufen. Dazu wird das Formular mit den Aufgabendaten verwendet. Die Kommentare werden als HTML-Inhalt in den vorgesehenen Abschnitt der Seite eingefügt. Dies fördert eine flüssige Benutzererfahrung, da die Informationen dynamisch aktualisiert werden.

2. Aktualisieren einer Aufgabe: Die Funktion `updateTask` ermöglicht es Benutzern, die Details einer Aufgabe zu ändern. Das Formular wird ausgelesen, und die aktuellen Werte werden gesammelt, um sie an den Server zu senden. Nach dem erfolgreichen Update zeigt die Anwendung eine Bestätigungsmeldung an.

3. Benutzer laden: Eine weitere wichtige Funktion besteht darin, eine Liste von Benutzern abzurufen, die in der Anwendung verfügbar sind. Diese Benutzer werden in Dropdown-Menüs eingefügt, die es den Nutzern ermöglichen, einen Besitzer oder einen zugewiesenen Benutzer für die Aufgabe auszuwählen. Dies wird beim Laden der Seite automatisch durchgeführt, sodass die Benutzer sofort die verfügbaren Optionen sehen.

4. Fehlerbehandlung: Der Code enthält auch Mechanismen zur Fehlerbehandlung. Bei Problemen, wie etwa fehlgeschlagenen Netzwerkaufrufen, werden Fehlermeldungen in der Konsole protokolliert.

## taskDeshboardScript.js

1. URL-Parameter auslesen: Zu Beginn der Funktion wird der Projektname aus den URL-Parametern extrahiert. Der Projektname ist entscheidend, um die Aufgaben für das gewählte Projekt zu laden.

2. Formulardaten: Anschließend werden die Formulardaten für die Anfrage vorbereitet. Der Projektname wird hinzugefügt, um sicherzustellen, dass die Anfrage die richtigen Daten anfordert. Die Funktion verwendet die Fetch API, um eine POST-Anfrage an den Endpunkt /getTasks zu senden.

3. Antwort: Nach einer erfolgreichen Anfrage wird die Antwort als JSON verarbeitet. Bei einem Fehler wird eine Ausnahme ausgelöst, und die Aufgaben werden extrahiert, um bereit zur Anzeige zu sein.

4. Inhalte zurücksetzen: Vorhandene Inhalte in den Fortschrittsspalten der Benutzeroberfläche werden gelöscht, um sicherzustellen, dass keine veralteten Daten angezeigt werden. Jede Spalte wird zudem mit einer Überschrift versehen.

5. Aufgabenanzeigen: Für jede Aufgabe wird ein neues div-Element mit der Klasse task-card generiert. Die Deadline wird formatiert und, falls sie in der Vergangenheit liegt, rot hinterlegt. Wesentliche Informationen zur werden in die Karte eingefügt. Ein Button ermöglicht den Zugriff auf die Detailseite der Aufgabe.

6. Laden der Aufgaben: Die Funktion loadTasks wird automatisch beim Laden der Seite aufgerufen, um sicherzustellen, dass die Aufgaben sofort angezeigt werden.

## RegistrationScript.js

1.Ereignislistener hinzufügen: Der Code fügt einen Listener für das submit-Ereignis des Formulars mit der ID registerForm hinzu. Dadurch wird eine Funktion aktiviert, wenn der Benutzer das Formular absendet.

2. Standardverhalten verhindern: Durch event.preventDefault(); wird das Standardverhalten des Formulars unterdrückt, sodass die Seite nicht neu geladen wird. Dies ermöglicht eine asynchrone Verarbeitung der Formulardaten.

3. Formulardaten sammeln und senden: Die Formulardaten werden in ein URL-kodiertes Format umgewandelt und mit der Fetch API an den Server gesendet. Die Anfrage erfolgt an den Endpunkt /register mit der Methode POST und dem entsprechenden Inhaltstyp.

4. Verarbeiten der Serverantwort: Der Code verarbeitet die Antwort des Servers. Bei erfolgreicher Rückmeldung wird das Ergebnis im einem HTML-Element angezeigt. Im Falle eines Fehlers wird eine Fehlermeldung angezeigt und der Fehler in der Konsole protokolliert.

## projektDashboardUser.js

1. Zurück zur Projektübersicht: Der Code ermöglicht es Benutzern, zur Projektübersicht zurückzukehren, indem er beim Klicken auf den entsprechenden Link bestimmte Container ein- oder ausblendet. Der neue Projektbereich und der Bearbeitungsbereich werden verborgen, während der Hauptcontainer sichtbar bleibt.

2. Ereignisdelegierung: Der Code verwendet Ereignisdelegierung, um Klicks auf dynamische Elemente zu verwalten. Wenn der Benutzer auf "Details anzeigen" klickt, wird die Projekt-ID und der Name ermittelt, um zur Dashboard-Seite weiterzuleiten. Bei einem Klick auf "Dieses Projekt bearbeiten" werden die entsprechenden Bereiche aktualisiert, um den Bearbeitungsbereich anzuzeigen.

3. Laden der Projekte: Bei der Initialisierung der Seite wird eine Fetch-Anfrage an den Server gesendet, um die Projektinformationen abzurufen. Die empfangenen Daten werden in einer Tabelle angezeigt, wobei jede Projektzeile die Projekt-ID als Datenattribut enthält. Fehler bei der Datenabfrage werden in der Konsole protokolliert.

## projektDasjboardAdmin.js

1. Dashboard-Navigation: Der Code enthält Links, um zwischen verschiedenen Bereichen zu navigieren, darunter die Projektübersicht, das Erstellen neuer Projekte, das Bearbeiten von Projekten und das Bearbeiten von Benutzern.

2. Formulare: Der Admin kann neue Projekte erstellen, bearbeiten oder auch Benutzer bearbeiten, indem er die entsprechenden Details eingibt.

3. Benutzerverwaltung: Der Abschnitt zur Benutzerbearbeitung zeigt eine Tabelle mit Benutzerdaten, einschließlich Benutzer-ID, Benutzernamen und Rang, sowie Schaltflächen zum Bearbeiten von Benutzern.

4. Aktionen: Für jede Benutzer- und Projektzeile gibt es Aktionen, die es dem Administrator ermöglichen, spezifische Änderungen vorzunehmen, wie das Aktualisieren von Benutzerinformationen oder Projektdetails.

## loginScript.js

1. Ereignislistener für das Anmeldeformular: Der Code fügt einen submit-Ereignislistener für das Anmeldeformular mit der ID loginForm hinzu. Wenn das Formular abgeschickt wird, wird die Standardaktion (Seitenneuladen) durch event.preventDefault(); unterdrückt.

2. Formulardaten sammeln und senden: Die Formulardaten werden erfasst und in ein URL-kodiertes Format umgewandelt. Diese Daten werden dann über die Fetch API an den Server gesendet, indem eine POST-Anfrage an den Endpunkt /login gesendet wird.

3. Verarbeiten der Serverantwort: Nach dem Senden der Anfrage wird die Antwort als JSON verarbeitet. Wenn die Anmeldung erfolgreich, wird der Benutzer je nach Rang zu unterschiedlichen Dashboard-Seiten weitergeleitet. Bei einem Fehler wird eine Fehlermeldung angezeigt.

## projectDashboard.js

1. Formularumschaltung: Klicks auf die Links für "Neues Projekt", "Projektübersicht" und "Projekt bearbeiten" steuern die Sichtbarkeit der entsprechenden Container. Das "Neue Projekt"-Formular wird angezeigt.

2. Wiederherstellen der "Details anzeigen"-Schaltflächen: Die Funktion restoreDetailsButtons() setzt den Text der Schaltflächen für die Detailansicht zurück, wenn zur Projektübersicht gewechselt wird.

3. Ereignisdelegierung: Ein Ereignislistener behandelt Klicks auf die "Details anzeigen"- und "Dieses Projekt bearbeiten"-Schaltflächen. Bei Klicks werden die Projekt-ID und der Name ermittelt, um zur entsprechenden Dashboard-Seite weiterzuleiten oder das Bearbeitungsformular anzuzeigen.

4. Datenabfrage: Beim Laden der Seite wird eine Fetch-Anfrage an den Server gesendet, um Projektinformationen abzurufen und in einer Tabelle anzuzeigen. Die Daten werden dynamisch in das tbody-Element eingefügt.

## projectBoard.js

1. Formularumschaltung: Beim Klick auf den Link für "Neues Projekt" wird das entsprechende Formular angezeigt, während andere Bereiche ausgeblendet werden.

2. Zurück zur Projektübersicht: Der Link für die Projektübersicht blendet das "Neue Projekt"-Formular aus und zeigt den Hauptbereich an, während die Bearbeitungsbereiche verborgen bleiben.

3. Projektbearbeitung: Ein Klick auf den Link für "Projekt bearbeiten" zeigt den Bearbeitungsbereich an und blendet andere Bereiche aus.

4. Ereignisdelegierung: Klicks auf spezifische Schaltflächen werden behandelt, um die jeweiligen Bereiche auszublenden oder anzuzeigen, je nach Benutzeraktion.

## creatorScript.js

1. Formularübermittlung: Ein Event-Listener für das `submit`-Ereignis verhindert das Neuladen der Seite und sammelt die Eingabewerte.

2. Eingabewerte validieren: Überprüft, ob alle erforderlichen Felder ausgefüllt sind. Bei fehlenden Angaben wird eine Warnung angezeigt.

3. Daten an den Server senden: Formulardaten werden an den Endpunkt `/addTask` gesendet. Bei Erfolg wird eine Nachricht angezeigt, bei Fehlern erfolgt eine Fehlermeldung.

4. Benutzer und Projekte laden: Nach dem Laden der Seite werden die Funktionen `loadUsers` und `loadProjects` aufgerufen, um die entsprechenden Dropdown-Menüs mit Daten zu befüllen.