🧠 **app.js – Zeile für Zeile erklärt**

const { createApp, reactive } = Vue;

* Importiert die createApp- und reactive-Funktion aus Vue 3. Damit starten wir eine neue App und machen Objekte reaktiv.

const app = createApp({

setup() {

* Erstellt die App und definiert das setup()-Feld. Das ist der Hauptlogikbereich in der Vue 3 Composition API.

🔧 Reaktive Zustände definieren:

const state = reactive({

columns: {

backlog: [],

todo: [],

inprogress: [],

review: [],

done: []

},

* state ist ein reaktives Objekt. Es enthält die Spalten des Kanban-Boards. Jede Spalte ist ein Array mit Aufgaben.

users: [],

newTask: "",

newUser: "",

selectedUser: "",

selectedDate: "",

selectedRole: ""

});

* Weitere Zustände für neue Aufgaben, Nutzername, ausgewählten Nutzer, Deadline und Rolle.

🧩 Hilfsfunktionen:

const addTask = (column) => {

* Diese Funktion fügt eine neue Aufgabe zur angegebenen Spalte hinzu.

if (state.newTask.trim() && state.selectedUser && state.selectedDate) {

* Überprüft, ob alle Eingabefelder ausgefüllt sind.

state.columns[column].push({

text: state.newTask,

user: state.selectedUser,

deadline: state.selectedDate

});

* Erstellt ein neues Aufgabenobjekt mit Text, Nutzer und Deadline und fügt es zur Spalte hinzu.

state.newTask = "";

state.selectedUser = "";

state.selectedDate = "";

}

};

* Leert die Eingabefelder nach dem Hinzufügen.

const moveTask = (fromColumn, toColumn, taskIndex) => {

const task = state.columns[fromColumn][taskIndex];

state.columns[toColumn].push(task);

state.columns[fromColumn].splice(taskIndex, 1);

};

* Bewegt eine Aufgabe von einer Spalte in eine andere.

const deleteTask = (column, taskIndex) => {

state.columns[column].splice(taskIndex, 1);

};

* Löscht eine Aufgabe aus der Spalte anhand ihres Index.

const addUser = () => {

* Funktion zum Hinzufügen eines Nutzers.

if (state.newUser.trim() && state.selectedRole) {

state.users.push({ name: state.newUser, role: state.selectedRole });

state.newUser = "";

state.selectedRole = "";

}

};

* Erstellt ein neues Nutzerobjekt (mit Name und Rolle) und fügt es zur Nutzerliste hinzu. Danach werden die Eingabefelder geleert.

const getRoleColor = (role) => {

switch (role) {

case "Product-Owner": return "#ff6666";

case "Entwickler": return "#66b3ff";

case "Scrum-Master": return "#66ff99";

default: return "#ccc";

}

};

* Gibt je nach Rolle eine Farbe zurück, z. B. für eine farbige Markierung neben dem Namen.

return {

state,

addTask,

moveTask,

deleteTask,

addUser,

getRoleColor

};

* Gibt alle reaktiven Zustände und Funktionen an die Template-Ebene zurück.

}

});

* Schließt das setup() und das App-Objekt.

app.mount("#app");

* Bindet die Vue-App an das HTML-Element mit der ID app.