planbar - Sicherheits- und Testbericht

Datum: 20. Januar 2026
Version: dev Branch (Commit 995b980)

---

1. Funktionstest-Ergebnisse

✅ Funktionierende Features (Production)

Feature	Status	Anmerkung
Login/Logout	✅ Funktioniert	Session-basiert mit NextAuth
Dashboard	✅ Funktioniert	Statistiken, Tasks, Projekte
Projekte-Liste	✅ Funktioniert	Filter, Suche, Sortierung
Projekt-Details	✅ Funktioniert	Subtasks, Status, Zuweisung
Team-Verwaltung	✅ Funktioniert	Teams erstellen/bearbeiten
Benutzer-Verwaltung	✅ Funktioniert	Rollen: Admin, Mitglied
Kategorien	✅ Funktioniert	Farben, Namen editierbar
Ressourcen-Übersicht	✅ Funktioniert	Auslastung pro User
Share-Links	✅ Funktioniert	Token-basiertes Teilen
Passwort-Reset	✅ Funktioniert	E-Mail mit Token

⚠️ Nicht auf Production (nur dev Branch)

Feature	Status	Anmerkung
Kalenderplanung	🔶 Nur dev	Ferien/Abwesenheiten eintragen
Koordinator-Rolle	🔶 Nur dev	Mittlere Berechtigungsstufe
Tasks-Seite	🔶 Nur dev	Aufgaben-Übersicht
Mitarbeiter-Vergleich	🔶 Nur dev	Workload-Vergleich
Datumsfilter Tasks	🔶 Nur dev	Heute/Woche/Custom

---

2. Sicherheitsanalyse

🔴 KRITISCH

2.1 Offene Registrierung ohne Validierung

Datei: `/app/api/signup/route.ts`

Problem:
- Jeder kann sich registrieren ohne E-Mail-Verifizierung
- Keine Passwort-Mindestanforderungen
- Potenzial für Spam-Accounts

Empfehlung:

```
// Password validation
const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$/;
if (!passwordRegex.test(password)) {
  return NextResponse.json(
    { error: 'Passwort muss min. 8 Zeichen, Groß-/Kleinbuchstaben und Zahl enthal-
ten' },
    { status: 400 }
  );
}

// Implement email verification with token
```

2.2 Fehlende Rate-Limitierung

Problem:
- Keine Begrenzung bei Login-Versuchen
- Brute-Force-Angriffe möglich
- DoS-Anfälligkeit

Empfehlung: Implementiere Middleware mit Upstash/Redis:

```
import { Ratelimit } from "@upstash/ratelimit";
import { Redis } from "@upstash/redis";

const ratelimit = new Ratelimit({
  redis: Redis.fromEnv(),
  limiter: Ratelimit.slidingWindow(5, "1 m"), // 5 Anfragen pro Minute
});
```

🟠 HOCH

2.3 Inkonsistente Admin-Prüfung

Problem: Verschiedene Stellen prüfen Admin-Rolle unterschiedlich:

```
// Version 1 (nur 'admin')
if (currentUser?.role !== 'admin')

// Version 2 (alle Varianten)
if (['admin', 'Administrator', 'ADMIN'].includes(user?.role || ''))
```

Risiko: Bypass wenn Rolle ‘ADMIN’ oder ‘Administrator’ ist

Empfehlung: Zentrale Hilfsfunktion:

```
// lib/auth-helpers.ts
export function isAdmin(role: string | null | undefined): boolean {
  return ['admin', 'Administrator', 'ADMIN'].includes(role?.toLowerCase() || '');
}

export function isKoordinator(role: string | null | undefined): boolean {
  return role?.toLowerCase() === 'koordinator';
}
```

2.4 Fehlende E-Mail-Validierung

Problem: E-Mail-Format wird nicht geprüft

Empfehlung:

```
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
if (!emailRegex.test(email)) {
  return NextResponse.json({ error: 'Ungültige E-Mail-Adresse' }, { status: 400 });
}
```

🟡 MITTEL

2.5 Share-Links ohne Ablaufdatum

Datei: `/app/api/share/route.ts`

Problem: Share-Tokens laufen nie ab

Empfehlung: Ablaufdatum hinzufügen:

```
// In Prisma Schema
shareExpires DateTime?

// Bei Erstellung
shareExpires: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000) // 7 Tage
```

2.6 Fehlerhafte Error-Responses

Problem: Manche Fehlermeldungen geben zu viele Infos preis

Empfehlung: Generische Fehlermeldungen für Produktion

🟢 NIEDRIG

2.7 Console.error in Produktion

Empfehlung: Logging-Service wie Sentry nutzen

---

3. Verbesserungsvorschläge

3.1 Performance

1. **Pagination für Projekte/Tasks**
   - Aktuell: Alle Einträge werden geladen
   - Besser: Lazy Loading mit Infinite Scroll

2. **Optimistic Updates**
   - Task-Status sofort ändern, Server im Hintergrund

3. **Caching**
   - SWR oder React Query für API-Calls
   - Redis für häufige Abfragen

3.2 UX-Verbesserungen

1. **Keyboard Shortcuts**
   - `Ctrl+N` für neues Projekt
   - `Escape` zum Schließen von Modals

2. **Drag & Drop**
   - Subtasks per Drag sortieren
   - Kanban-Board für Projekte

3. **Dark Mode**
   - Schon vorbereitet in Tailwind (`darkMode: 'class'`)

4. **Benachrichtigungen**
   - In-App Notifications
   - Push Notifications (PWA)

3.3 Funktionale Erweiterungen

1. **Zeiterfassung**
   - Timer für Tasks starten/stoppen
   - Automatische Stundenberechnung

2. **Kommentare**
   - Kommentare an Tickets/Tasks
   - @Mentions für User

3. **Dateien**
   - Anhänge an Projekte
   - Vorschau von Bildern/PDFs

4. **Berichte**
   - Export als CSV/PDF
   - Wöchentliche Zusammenfassung per E-Mail

5. **Integration**
   - iCal-Export für Kalender
   - Slack/Teams Notifications

---

4. Deployment-Status

Production (main Branch)

- **URL:** https://planbar-one.vercel.app
- **Status:** ✅ Funktioniert
- **Version:** Ohne neue Features (Kalender, Koordinator, etc.)

Preview (dev Branch)

- **Commit:** 995b980
- **Features:** Alle neuen Features
- **Problem:** Abwesenheiten speichern funktioniert erst nach prisma db push

Empfohlene Aktion

Dev-Branch nach Main mergen:

```
git checkout main
git merge dev
git push origin main
```

---

5. Code-Qualität

✅ Positiv

- TypeScript durchgehend
- Konsistente Fehlerbehandlung
- Gute Trennung von Komponenten
- Responsive Design mit Tailwind
- Framer Motion für Animationen

⚠️ Verbesserungspotenzial

- Mehr Unit Tests nötig
- API-Dokumentation fehlt
- Einige `any` Types sollten typisiert werden
- Wiederholter Code könnte in Hooks ausgelagert werden

---

6. Zusammenfassung

Gesamtbewertung: 🟡 Gut mit Verbesserungspotenzial

Bereich	Bewertung
Funktionalität	⭐⭐⭐⭐☆
Sicherheit	⭐⭐⭐☆☆
Performance	⭐⭐⭐⭐☆
Code-Qualität	⭐⭐⭐⭐☆
UX/Design	⭐⭐⭐⭐⭐

Priorität der Maßnahmen:

1. 🔴 Rate Limiting implementieren
2. 🔴 E-Mail-Verifizierung für Registrierung
3. 🟠 Admin-Prüfung vereinheitlichen
4. 🟠 Passwort-Anforderungen
5. 🟡 Share-Link Ablauf

Bericht erstellt am 20.01.2026 durch DeepAgent