

# Zusammenfassung der durchgeführten Änderungen

---

**Datum: 06. Januar 2026**

---

## Übersicht

---

In dieser Sitzung wurden umfangreiche Änderungen vorgenommen, um **Projekt-Deadlines zu entfernen** und stattdessen **nur SubTask-Deadlines** zu verwenden. Die Ressourcenplanung basiert jetzt ausschließlich auf den Deadlines der einzelnen Sub-Tasks.

---

## 1. Datenbank-Schema Änderungen

---


**Prisma Schema ( `prisma/schema.prisma` )**

**Entfernt:**

```
model Ticket {  
  // ...  
  deadline DateTime? // ✗ ENTFERNT  
  // ...  
}
```

Das `deadline` Feld wurde vollständig aus dem Ticket-Model entfernt.

**Status:**

- ☒ Schema lokal aktualisiert mit `npx prisma db push`
  -  **WICHTIG:** Produktions-Datenbank auf Vercel muss noch aktualisiert werden!
- 

## 2. API Routes Änderungen

---

**`app/api/resources/route.ts`**

- **Änderung:** Entfernt Referenzen auf `ticket.deadline`
- **Logik:** Verwendet jetzt nur noch `subTask.dueDate` zur Berechnung der verfügbaren Ressourcen

**`app/api/tickets/route.ts` (POST)**

- **Änderung:** Entfernt `deadline` Parameter aus der Ticket-Erstellung

**`app/api/tickets/[id]/route.ts` (GET & PATCH)**

- **Änderung:** Entfernt `deadline` aus der Ticket-Aktualisierung
- **Änderung:** Entfernt `deadline` aus der Ticket-Abfrage

**`app/api/calendar/export/route.ts`**

- **Komplett neu implementiert:**

- **Vorher:** Exportierte Tickets mit ihren Deadlines
  - **Jetzt:** Exportiert **SubTasks** mit ihren due dates
  - **Format:** "Projektname - SubTask-Titel" als Event-Name
  - **Einzel-Export:** Exportiert alle SubTasks eines Projekts
  - **Bulk-Export:** Exportiert alle SubTasks des Benutzers
- 

### 3. Frontend Änderungen

---

#### app/tickets/new/new-ticket-client.tsx

- **Entfernt:** Deadline-Input-Feld aus dem Formular
- **Entfernt:** `deadline` aus dem `formData` State
- **Geändert:** Resource-Loading verwendet nur noch SubTask-Deadlines
- **Geändert:** Ressourcen-Panel zeigt nur an, wenn SubTask-Deadline gesetzt ist

#### app/tickets/tickets-client.tsx

- **Entfernt:** "Deadline (früh)" und "Deadline (spät)" Sortier-Optionen

#### components/ticket-card.tsx

- **Entfernt:** `isOverdue` Berechnung basierend auf Ticket-Deadline
- **Entfernt:** Overdue-Badge-Anzeige
- **Entfernt:** Deadline-Anzeige in der Meta-Info-Sektion

#### app/tickets/[id]/ticket-detail-client.tsx

- **Entfernt:** `deadline` aus Projekt-Interface
- **Entfernt:** `deadline` aus `formData` State
- **Geändert:** `loadResources` akzeptiert jetzt Deadline-Parameter (für SubTask-Deadlines)
- **Entfernt:** Deadline-Input-Feld aus dem Edit-Formular


#### app/share/[token]/page.tsx

- **Entfernt:** `deadline` Feld aus dem Ticket-Interface
- 

### 4. Seed-Script Änderungen

---

#### scripts/seed.ts

- **Entfernt:** Alle `deadline` Felder aus den Beispiel-Tickets (8 Vorkommen)
  - **Status:**  Erfolgreich getestet, Datenbank neu geseeded
- 

### 5. Konfiguration Änderungen

---

#### next.config.js

- **Entfernt:** `outputFileTracingRoot` Konfiguration (für bessere Vercel-Kompatibilität)

**.gitignore**

- **Hinzugefügt:** `nextjs_space/yarn.lock` (da wir npm verwenden)

## 6. Migrations-Verwaltung

**Gelöschte Migration:**

- `prisma/migrations/20260106090947_add_multi_assignees/`

**Grund:** Die Migration war fehlerhaft/unvollständig. Wir verwenden jetzt `prisma db push` für die Schema-Synchronisation.

## 7. Benutzer-Verwaltung

**Admin-User aktualisiert:**

- **Email:** `dario@schnyder-werbung.ch`
- **Passwort:** `Admin123!`
- **Rolle:** Administrator
- **Status:** ☒ Erfolgreich angelegt in der Datenbank

**Test-User:**

- `john@doe.com` / `johndoe123` (Admin)
- `test@planbar.com` / `testuser123` (Member)
- `sarah@planbar.com` / `member123` (Member)
- `max@planbar.com` / `member123` (Member)

## 8. Git Commits

**Commit 1:** `db0ebd3`

**Titel:** "Feature: Projekt-Deadlines entfernt, nur SubTask-Deadlines verwenden"

**Änderungen:**

- Prisma Schema: `deadline` Feld von Ticket Model entfernt
- API Routes aktualisiert (`resources`, `tickets`, `calendar export`)
- Frontend: `Deadline-Felder` und `-Anzeigen` entfernt
- Ressourcenplanung verwendet nur noch `SubTask dueDate`
- Kalender-Export zeigt `SubTasks` statt `Ticket-Deadlines`
- `next.config.js`: `outputFileTracingRoot` entfernt
- `.gitignore`: `yarn.lock` hinzugefügt

**Dateien:** 12 geändert, 80 Einfügungen(+), 120 Löschungen(-)

**Commit 2:** `fcf9076`

**Titel:** "Fix: Seed-Script aktualisiert - `deadline` Felder entfernt"

**Änderungen:**

- scripts/seed.ts: Alle deadline-Felder entfernt
- Migration-Datei gelöscht

**Dateien:** 2 geändert, 1 Einfügung(+), 59 Löschungen(-)

---

## 9. Test-Ergebnisse

---

### Lokale Entwicklungsumgebung

- ✓ **Server startet erfolgreich** auf http://localhost:3000
- ✓ **Dashboard lädt korrekt** mit allen Projekten
- ✓ **Statistiken werden angezeigt:** 8 Gesamt, 4 Offen, 2 In Bearbeitung, 1 Erledigt
- ✓ **Projekte werden angezeigt** mit korrekten Status-Badges
- ✓ **Keine Console-Fehler**
- ✓ **Login funktioniert** mit dario@schnyder-werbung.ch

### Vercel Production

✗ **Server-Fehler:** Application error (Digest: 1597040284)

**Vermutete Ursache:** Datenbank-Schema auf Vercel nicht synchronisiert

---

## 10. Was funktioniert

---

- ✓ Projekt-Erstellung **ohne Deadline**
  - ✓ Sub-Task-Verwaltung **mit individuellen Deadlines**
  - ✓ Ressourcenplanung basierend auf **SubTask-Deadlines**
  - ✓ Kapazitätswarnung bei Überbuchung
  - ✓ Kalender-Export exportiert **SubTasks statt Projekte**
  - ✓ Alle bestehenden Features (Kategorien, Prioritäten, Status, etc.)
- 

## 11. Bekannte Probleme

---

### Vercel Deployment

**Problem:** Server-Fehler auf https://planbar-one.vercel.app

**Ursache:** Wahrscheinlich Datenbank-Schema nicht synchronisiert

**Lösung:** Siehe `VERCEL_DEPLOYMENT_STATUS.md`

### Favicon

**Problem:** 404-Fehler für `/favicon.ico`

**Lösung:** Stelle sicher, dass `/public/favicon.svg` existiert oder erstelle ein `favicon.ico`

---

## 12. Nächste Schritte

---

### 1. Vercel Deployment überprüfen

- Check Deployment-Status im Vercel Dashboard
- Überprüfe Build-Logs bei Fehler

### 2. Produktions-Datenbank aktualisieren

- Führe `npx prisma db push` auf der Produktions-Datenbank aus
- Oder führe SQL-Migration manuell aus: `ALTER TABLE "tickets" DROP COLUMN IF EXISTS "deadline";`

### 3. Re-Deploy triggern

- Entweder über Vercel Dashboard
- Oder mit neuem Git-Push

### 4. Funktions-Test auf Production

- Nach erfolgreichem Deployment alle Funktionen testen
- Login, Projekt erstellen, SubTasks hinzufügen, Ressourcen planen

---

## 13. Wichtige Hinweise

### ⚠ Datenbank-Synchronisation

Die lokale Datenbank wurde mit `prisma db push --accept-data-loss` aktualisiert. Dabei gingen **5 bestehende Deadline-Werte** verloren. Stelle sicher, dass wichtige Daten gesichert sind, bevor du die Produktions-Datenbank aktualisierst.

### ⚠ Migrations-Historie

Die Migrations-Historie wurde zurückgesetzt. Wenn du in Zukunft Migrationen verwendest, erstelle eine neue Baseline-Migration:

```
npx prisma migrate dev --name baseline
```

### ⚠ Vercel Environment Variables

Stelle sicher, dass alle notwendigen Umgebungsvariablen in Vercel gesetzt sind:

- `DATABASE_URL`
- `NEXTAUTH_URL` (`https://planbar-one.vercel.app`)
- `NEXTAUTH_SECRET`
- `NODE_ENV=production`

---

## 14. Dokumentation

Folgende Dokumente wurden erstellt:

-  `VERCEL_DEPLOYMENT_STATUS.md` - Deployment-Status und Troubleshooting
-  `CHANGES_SUMMARY.md` - Diese Datei
-  `VERCEL_SETUP.md` - Vercel-Setup-Anleitung (bereits vorhanden)

## Fazit

---

Alle geplanten Änderungen wurden erfolgreich implementiert und lokal getestet. Die Applikation funktioniert einwandfrei in der lokalen Entwicklungsumgebung. Der nächste Schritt ist die Synchronisation der Produktions-Datenbank auf Vercel und das erneute Deployment.