==
PROJEKTDOKUMENTATION
Serviceauftragsverwaltung - Glauser AG
==

Autor: Blerjon Asllani
Schule: Benedict ZH
Datum: Februar 2026
Projekt: Abschlussprojekt Informatik

==
1. PROJEKTÜBERSICHT
==

1.1 Ausgangslage

Die Glauser AG ist ein Sanitär- und Heizungsunternehmen, das täglich zahlreiche Serviceaufträge bearbeitet. Bisher wurden Aufträge manuell auf Papier erfasst und verwaltet, was zu Informationsverlusten, Verzögerungen und ineffizienten Arbeitsabläufen führte.

1.2 Zielsetzung

Entwicklung einer webbasierten Anwendung zur digitalen Verwaltung von Serviceaufträgen mit komplettem Workflow von der Erfassung bis zur Verrechnung.

1.3 Hauptfunktionen

- Digitale Erfassung von Serviceaufträgen
- Rollenbasierte Zugriffskontrolle (Admin, Bereichsleiter, Mitarbeiter)
- Zuweisung von Aufträgen an Mitarbeiter
- Erstellung von Arbeitsrapporten durch Mitarbeiter
- Freigabe-Workflow durch Bereichsleiter
- Verrechnung mit Rechnungsnummer
- PDF-Export von Auftragsblättern
- Dashboard mit Statistiken und Übersichten

==
2. TECHNOLOGIEN
==

2.1 Frontend

- React 19 - Modern JavaScript Framework
- React Router - Navigation zwischen Seiten
- Bootstrap 5 - Responsive Design Framework
- Axios - HTTP-Client für API-Kommunikation

2.2 Backend

- PHP 8.2 - Server-side Programmierung
- REST API - Schnittstelle zwischen Frontend und Backend
- TCPDF - PDF-Generierung für Auftragsblätter

2.3 Datenbank

- Supabase (PostgreSQL) - Cloud-Datenbank
- PostgreSQL Trigger - Automatische Auftragsnummern-Generierung

2.4 Entwicklungsumgebung

- XAMPP - Lokaler Webserver (Apache + PHP)
- Node.js & npm - JavaScript Runtime und Package Manager
- Git & GitHub - Versionskontrolle

==========
3. BENUTZER & ROLLEN
==========

3.1 Rollendefinition

ADMINISTRATOR (ADMIN)
- Volle Systemrechte
- Erfassung neuer Aufträge
- Disponierung (Zuweisung an Mitarbeiter)
- Freigabe von Aufträgen
- Verrechnung erstellen
- Alle Aufträge einsehen

BEREICHSLEITER (BL)
- Disponierung (Zuweisung an Mitarbeiter)
- Freigabe von Aufträgen zur Verrechnung
- Alle Aufträge einsehen
- Keine Erfassung neuer Aufträge
- Keine Verrechnung

MITARBEITER (MA)
- Nur eigene zugewiesene Aufträge sehen
- Arbeitsrapport erstellen
- Auftragsdetails einsehen
- Keine Verwaltungsfunktionen

3.2 Test-Accounts

Benutzername: admin
Passwort: Test1234!
Rolle: Administrator

Benutzername: mueller.m
Passwort: Test1234!
Rolle: Bereichsleiter

Benutzername: schmidt.p
Passwort: Test1234!
Rolle: Mitarbeiter

==
===============
4. WORKFLOW
==
===============

Der Workflow eines Serviceauftrags durchläuft folgende Phasen:

Phase 1: ERFASSEN

- Rolle: Administrator
- Aktion: Neuen Auftrag mit Kundendaten erfassen
- Input: Kundendaten, Beschreibung, Priorität, gewünschter Termin
- Output: Auftrag mit Status "erfasst" und automatischer Auftragsnummer
- Auftragsnummer-Format: YYYY-00001 (Jahr + laufende Nummer)

Phase 2: DISPONIEREN

- Rolle: Administrator oder Bereichsleiter
- Aktion: Mitarbeiter zuweisen und Termin planen
- Input: Mitarbeiter-Auswahl, geplantes Datum, interne Notizen
- Output: Auftrag mit Status "disponiert"
- E-Mail-Benachrichtigung an zugewiesenen Mitarbeiter (optional)

Phase 3: AUSFÜHREN

- Rolle: Mitarbeiter
- Aktion: Arbeit ausführen und Rapport erstellen
- Input: Ausführungsdatum, Arbeitszeit, verwendetes Material, Bemerkungen
- Output: Auftrag mit Status "ausgeführt"
- Rapport enthält: Datum, Arbeitszeit, Material, detaillierte Beschreibung

Phase 4: FREIGEBEN

- Rolle: Bereichsleiter oder Administrator
- Aktion: Rapport prüfen und zur Verrechnung freigeben
- Input: Prüfung des Rapports
- Output: Auftrag mit Status "freigegeben"
- Qualitätskontrolle der durchgeführten Arbeit

Phase 5: VERRECHNEN

- Rolle: Administrator
- Aktion: Rechnung erstellen
- Input: Rechnungsnummer, Betrag, Rechnungsdatum
- Output: Auftrag mit Status "verrechnet"
- Auftrag ist abgeschlossen

5. FUNKTIONSBESCHREIBUNGEN

5.1 Dashboard

Das Dashboard bietet eine Übersicht über:

- Statistiken nach Status
  - Anzahl erfasster Aufträge
  - Anzahl disponierter Aufträge
  - Anzahl ausgeführter Aufträge
  - Anzahl freigegebener Aufträge
  - Anzahl verrechneter Aufträge

- Rollenspezifische Ansichten
  - Admin: Alle Aufträge
  - BL: Alle Aufträge
  - MA: Nur eigene Aufträge

- Quick-Actions
  - Neuer Auftrag (nur Admin)
  - Zu disponierende Aufträge
  - Zu bearbeitende Aufträge

5.2 Auftrags-Liste

Funktionen:
- Tabellarische Darstellung aller Aufträge
- Filterung nach Status (erfasst, disponiert, ausgeführt, freigegeben, verrechnet)
- Volltextsuche über Auftragsnummer, Kunde, Ort, Beschreibung
- Sortierung nach Spalten
- Status-Badges mit Farbcodierung
- Prioritäts-Anzeige (normal, dringend, notfall)
- Klick auf Zeile öffnet Auftragsdetails

5.3 Auftrag erfassen

Formular für neue Aufträge:

Kundendaten (Pflichtfelder):
- Kundenname
- Straße
- PLZ (4-stellig)
- Ort
- Telefon
- E-Mail (optional)

Auftragsdaten:
- Beschreibung (Pflichtfeld, min. 20 Zeichen)
- Priorität (normal, dringend, notfall)
- Gewünschter Termin (optional)

Validierung:
- PLZ muss 4-stellig sein
- Telefonnummer im Format +41 XX XXX XX XX
- E-Mail-Format-Prüfung

Nach Speichern:
- Automatische Auftragsnummer-Generierung
- Weiterleitung zur Auftragsübersicht

5.4 Auftragsdetails

Detailansicht mit allen Informationen:

Basis-Informationen:
- Auftragsnummer
- Status (mit farbigem Badge)
- Priorität
- Erstelldatum
- Zuletzt aktualisiert

Kundendaten:
- Name/Firma
- Vollständige Adresse
- Kontaktdaten (Telefon, E-Mail)

Auftragsdaten:
- Detaillierte Beschreibung
- Gewünschter Termin
- Geplantes Datum (falls disponiert)
- Zugewiesener Mitarbeiter (falls disponiert)
- Interne Notizen (nur für Admin/BL sichtbar)

Rapport-Daten (falls ausgeführt):
- Ausführungsdatum
- Arbeitszeit (von-bis) oder Gesamtstunden
- Verwendetes Material
- Bemerkungen des Mitarbeiters

Verrechnungs-Daten (falls verrechnet):
- Rechnungsnummer
- Betrag in CHF
- Rechnungsdatum

Aktionen (statusabhängig):
- Disponieren (erfasst → disponiert)
- Rapport erstellen (disponiert → ausgeführt)
- Freigeben (ausgeführt → freigegeben)
- Verrechnen (freigegeben → verrechnet)
- PDF herunterladen (alle Status)

5.5 PDF-Export

Generiert professionelles Auftragsblatt mit:∂

Header:
- Firmenlogo (Glauser AG)

- Auftragsnummer
- Erstelldatum

Kundendaten:
- Name/Firma
- Vollständige Adresse
- Kontaktdaten

Auftragsinformationen:
- Status mit Farbcodierung
- Priorität
- Beschreibung
- Termine

Rapport (falls vorhanden):
- Ausführungsdatum
- Arbeitszeit
- Material
- Bemerkungen

Verrechnung (falls vorhanden):
- Rechnungsnummer
- Betrag
- Datum

Footer:
- Seitennummer
- Erstellungszeitpunkt
- Firmenkontakt

==========
==========
6. DATENBANK-SCHEMA
==========
==========

6.1 Tabelle: benutzer

Speichert alle Systembenutzer mit Rollen und Authentifizierung.

Spalten:
- benutzerid (INT, PRIMARY KEY, AUTO_INCREMENT)
  Eindeutige Benutzer-ID

- benutzername (VARCHAR(50), UNIQUE, NOT NULL)
  Login-Name, muss eindeutig sein

- passwordhash (VARCHAR(255), NOT NULL)
  Verschlüsseltes Passwort (bcrypt, Kostenfaktor 10)

- vorname (VARCHAR(50), NOT NULL)
  Vorname des Benutzers

- nachname (VARCHAR(50), NOT NULL)
  Nachname des Benutzers

- email (VARCHAR(100), UNIQUE, NOT NULL)
  E-Mail-Adresse, muss eindeutig sein

- rolle (VARCHAR(10), NOT NULL)
  Rolle: 'ADMIN', 'BL', oder 'MA'

- aktiv (BOOLEAN, DEFAULT TRUE)
  Account-Status (aktiv/deaktiviert)

- erstelltam (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
  Zeitpunkt der Account-Erstellung

6.2 Tabelle: auftrag

Kern-Tabelle mit allen Serviceaufträgen.

Spalten:
- auftragid (INT, PRIMARY KEY, AUTO_INCREMENT)
  Eindeutige Auftrags-ID

- auftragsnummer (VARCHAR(20), UNIQUE, NOT NULL)
  Automatisch generiert: YYYY-00001

- kundename (VARCHAR(100), NOT NULL)
  Name oder Firma des Kunden

- kundestrasse (VARCHAR(100), NOT NULL)
  Straße und Hausnummer

- kundeplz (CHAR(4), NOT NULL)
  Postleitzahl (4-stellig, Schweiz)

- kundeort (VARCHAR(50), NOT NULL)
  Ort

- kundetelefon (VARCHAR(20), NOT NULL)
  Telefonnummer

- kundeemail (VARCHAR(100))
  E-Mail-Adresse (optional)

- beschreibung (TEXT, NOT NULL)
  Detaillierte Auftragsbeschreibung

- prioritaet (VARCHAR(10), DEFAULT 'normal')
  Werte: 'normal', 'dringend', 'notfall'

- status (VARCHAR(15), DEFAULT 'erfasst')
  Werte: 'erfasst', 'disponiert', 'ausgeführt', 'freigegeben', 'verrechnet'

- gewuenschtertermin (DATE)
  Vom Kunden gewünschter Termin (optional)

- geplantesdatum (DATE)
  Von BL geplanter Ausführungstermin

- internenotizen (TEXT)
  Interne Notizen, nicht für MA sichtbar

- zugewiesenerma (INT, FOREIGN KEY → benutzer.benutzerid)
  Zugewiesener Mitarbeiter

- erstelltvon (INT, FOREIGN KEY → benutzer.benutzerid, NOT NULL)
  Ersteller des Auftrags

- erstelltam (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
  Erstellungszeitpunkt

- aktualisiertam (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
  Letzte Aktualisierung (automatisch via Trigger)

6.3 Tabelle: rapport

Arbeitsrapport nach Auftragsausführung.

Spalten:
- rapportid (INT, PRIMARY KEY, AUTO_INCREMENT)
  Eindeutige Rapport-ID

- auftragid (INT, FOREIGN KEY → auftrag.auftragid, UNIQUE, NOT NULL)
  Zugehöriger Auftrag (1:1 Beziehung)

- ausfuehrungsdatum (DATE, NOT NULL)
  Datum der Arbeitsausführung

- arbeitszeitvon (TIME)
  Startzeit der Arbeit (optional)

- arbeitszeitbis (TIME)
  Endzeit der Arbeit (optional)

- arbeitsstunden (DECIMAL(5,2))
  Gesamte Arbeitsstunden (alternative Eingabe)

- verwendetesmaterial (TEXT)
  Liste des verwendeten Materials

- bemerkungen (TEXT, NOT NULL)
  Detaillierte Arbeitsbeschreibung (min. 20 Zeichen)

- erstelltam (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
  Erstellungszeitpunkt

6.4 Tabelle: verrechnung

Rechnungsdaten für abgeschlossene Aufträge.

Spalten:
- verrechnungid (INT, PRIMARY KEY, AUTO_INCREMENT)
  Eindeutige Verrechnungs-ID

- auftragid (INT, FOREIGN KEY → auftrag.auftragid, UNIQUE, NOT NULL)
  Zugehöriger Auftrag (1:1 Beziehung)

- rechnungsnummer (VARCHAR(50), UNIQUE, NOT NULL)
  Eindeutige Rechnungsnummer

- betrag (DECIMAL(10,2), NOT NULL)
  Rechnungsbetrag in CHF

- rechnungsdatum (DATE, NOT NULL)
  Datum der Rechnungsstellung

- erstelltam (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
  Erstellungszeitpunkt

6.5 Beziehungen
---

benutzer → auftrag:
- 1:N - Ein Benutzer kann viele Aufträge erstellen
- 1:N - Ein Benutzer kann viele Aufträge zugewiesen bekommen

auftrag → rapport:
- 1:1 - Ein Auftrag hat maximal einen Rapport

auftrag → verrechnung:
- 1:1 - Ein Auftrag hat maximal eine Verrechnung

6.6 Trigger

Trigger: before_auftrag_insert
Zweck: Automatische Generierung der Auftragsnummer
Format: YYYY-00001 (Jahr + 5-stellige laufende Nummer)
Beispiel: 2026-00001, 2026-00002, etc.

Trigger: update_auftrag_timestamp
Zweck: Automatische Aktualisierung von 'aktualisiertam'
Zeitpunkt: Bei jedem UPDATE auf auftrag-Tabelle

==========
7. API-ENDPUNKTE
==========

7.1 Authentifizierung

POST /api/auth/login.php
- Beschreibung: Benutzer-Login
- Input: { benutzername, passwort }
- Output: { success, user: { userId, benutzername, vorname, nachname, rolle } }
- Session: Startet PHP-Session

POST /api/auth/logout.php
- Beschreibung: Benutzer-Logout
- Output: { success }
- Session: Zerstört PHP-Session

GET /api/auth/check.php
- Beschreibung: Session-Validierung
- Output: { authenticated, user: {...} }
- Verwendung: Beim Laden der App

7.2 Aufträge

GET /api/auftraege/list.php
- Beschreibung: Alle Aufträge abrufen
- Parameter: ?status=erfasst&search=keyword
- Output: { success, data: [...] }
- Berechtigung: Alle (MA sieht nur eigene)

GET /api/auftraege/get.php?id=123
- Beschreibung: Einzelnen Auftrag abrufen
- Parameter: id (Auftrags-ID)
- Output: { success, data: {...} }
- Berechtigung: Alle (MA nur eigene)

POST /api/auftraege/create.php
- Beschreibung: Neuen Auftrag erstellen
- Input: Kundendaten, Beschreibung, Priorität, etc.
- Output: { success, auftragId, auftragsnummer }
- Berechtigung: Nur ADMIN

POST /api/auftraege/disponieren.php
- Beschreibung: Auftrag disponieren
- Input: { auftragId, mitarbeiterId, geplantesdatum, internenotizen }
- Output: { success }
- Berechtigung: ADMIN, BL

POST /api/auftraege/rapport.php
- Beschreibung: Rapport erstellen
- Input: { auftragId, ausfuehrungsdatum, arbeitszeit, material, bemerkungen }
- Output: { success }
- Berechtigung: Zugewiesener MA

POST /api/auftraege/freigeben.php
- Beschreibung: Auftrag freigeben
- Input: { auftragId }
- Output: { success }
- Berechtigung: ADMIN, BL

POST /api/auftraege/verrechnen.php
- Beschreibung: Auftrag verrechnen
- Input: { auftragId, rechnungsnummer, betrag, rechnungsdatum }
- Output: { success }
- Berechtigung: Nur ADMIN

GET /api/auftraege/pdf.php?id=123

- Beschreibung: PDF-Auftragsblatt generieren
- Parameter: id (Auftrags-ID)
- Output: PDF-Datei (application/pdf)
- Berechtigung: Alle

GET /api/auftraege/mitarbeiter.php
- Beschreibung: Liste aller Mitarbeiter
- Output: { success, data: [...] }
- Berechtigung: ADMIN, BL

==
===============
8. SICHERHEIT
==
===============

8.1 Authentifizierung

- Passwort-Hashing mit bcrypt (Kostenfaktor 10)
- Session-basierte Authentifizierung
- Session-IDs werden bei jedem Login neu generiert
- Passwort-Validierung:
  - Mindestens 8 Zeichen
  - Mindestens 1 Großbuchstabe
  - Mindestens 1 Kleinbuchstabe
  - Mindestens 1 Ziffer
  - Mindestens 1 Sonderzeichen

8.2 Autorisierung

- Rollenbasierte Zugriffskontrolle (RBAC)
- Session-Prüfung bei jedem API-Request
- Berechtigungsprüfung auf Backend-Ebene
- Mitarbeiter sehen nur zugewiesene Aufträge
- Statusübergänge sind rollenspezifisch geschützt

8.3 Input-Validierung

- Frontend: Formular-Validierung mit Bootstrap
- Backend: Strikte Validierung aller Eingaben
- SQL-Injection-Schutz durch Prepared Statements
- XSS-Schutz durch Output-Encoding
- CSRF-Schutz durch SameSite-Cookies

8.4 API-Sicherheit

- CORS: Nur localhost:3000 erlaubt
- Content-Type: application/json erzwungen
- HTTP-Only Cookies für Session
- Rate-Limiting auf API-Ebene (geplant)

==
===============
9. INSTALLATION & DEPLOYMENT

==
===============

9.1 Voraussetzungen

- XAMPP (Apache 2.4+, PHP 8.2+)
- Node.js 18+
- npm (Node Package Manager)
- Internetverbindung (für Supabase-Datenbank)
- Moderner Webbrowser (Chrome, Firefox, Safari, Edge)

9.2 Backend-Installation
----
1. Projekt nach XAMPP kopieren
   Windows: C:\xampp\htdocs\Projekt
   Mac: /Applications/XAMPP/xamppfiles/htdocs/Projekt

2. XAMPP Control Panel öffnen
   - Apache starten
   - MySQL wird NICHT benötigt

3. Composer Dependencies (optional, vendor/ bereits vorhanden)
   cd Projekt
   composer install

9.3 Datenbank-Setup (optional, falls man es selber machen moechte)
----
1. Supabase-Account erstellen
   https://supabase.com

2. Neues Projekt erstellen
   - Projektname wählen
   - Datenbank-Passwort notieren

3. SQL-Schema importieren
   - Supabase Dashboard → SQL Editor
   - Inhalt von database/schema.sql kopieren und ausführen

4. Testdaten importieren (optional)
   - Inhalt von database/testdaten.sql ausführen

5. API-Keys konfigurieren
   - api/config/supabase.php öffnen
   - PROJECT_URL und ANON_KEY eintragen

9.4 Frontend-Installation
----
1. Terminal/CMD öffnen

2. In Frontend-Verzeichnis wechseln
   cd Projekt/frontend

3. Dependencies installieren
   npm install

4. Development-Server starten

npm start

5. Browser öffnet automatisch
http://localhost:3000

9.5 Produktion (optional)
---
1. Frontend build erstellen
npm run build

2. Build-Ordner auf Webserver deployen

3. API-URLs in Produktion anpassen

==========
10. TESTING
==========

10.1 Manuelle Tests

Alle Funktionen wurden manuell getestet:

Test 1: Login/Logout
- Login mit allen drei Rollen erfolgreich
- Logout funktioniert korrekt
- Session-Persistenz über Seitenwechsel

Test 2: Auftrag erfassen
- Formular-Validierung funktioniert
- Pflichtfelder werden geprüft
- Automatische Auftragsnummer wird generiert

Test 3: Disponieren
- Mitarbeiter-Auswahl funktioniert
- Geplantes Datum wird gespeichert
- Interne Notizen werden korrekt gespeichert

Test 4: Rapport erstellen
- Nur zugewiesene MA können Rapport erstellen
- Zeiterfassung (von-bis oder Stunden) funktioniert
- Material-Erfassung funktioniert
- Bemerkungen-Validierung (min. 20 Zeichen)

Test 5: Freigeben
- Nur BL/ADMIN können freigeben
- Status-Wechsel funktioniert korrekt
- Rapport-Daten bleiben erhalten

Test 6: Verrechnen
- Nur ADMIN kann verrechnen
- Rechnungsnummer wird validiert
- Betrag wird korrekt gespeichert

Test 7: PDF-Export
- PDF wird korrekt generiert
- Alle Daten werden angezeigt
- Layout ist professionell

Test 8: Berechtigungen
- MA sieht nur eigene Aufträge
- BL sieht alle Aufträge
- ADMIN hat volle Rechte

10.2 Browser-Kompatibilität

Getestet in:
- Google Chrome 120+ ✓
- Mozilla Firefox 121+ ✓
- Safari 17+ ✓
- Microsoft Edge 120+ ✓

10.3 Responsive Design

Getestet auf:
- Desktop (1920x1080) ✓
- Laptop (1366x768) ✓
- Tablet (768x1024) ✓
- Mobile (375x667) ✓

==========
11. BEKANNTE EINSCHRÄNKUNGEN
==========

- Keine E-Mail-Benachrichtigungen implementiert
- Keine Datei-Uploads (z.B. Fotos vom Auftrag)
- Keine Kalender-Ansicht für Termine
- Keine Statistik-Diagramme (Charts)
- Keine Export-Funktion für Excel/CSV
- Keine Multi-Sprach-Unterstützung
- Keine Offline-Funktionalität
- Keine Mobile App (nur Web-App)

==========
12. MÖGLICHE ERWEITERUNGEN
==========

12.1 Kurzfristig umsetzbar

- E-Mail-Benachrichtigungen bei Status-Änderungen
- Kalender-Ansicht für geplante Aufträge
- Foto-Upload für Aufträge/Rapporte
- Excel-Export der Auftragsliste

- Erweiterte Filter (Zeitraum, Mitarbeiter, etc.)
- Dashboard-Charts (Chart.js)

12.2 Mittelfristig

- Mobile App (React Native)
- Barcode/QR-Code für Aufträge
- Digitale Unterschrift
- Material-Lagerverwaltung
- Automatische Rechnungserstellung (PDF)
- Kundenverwaltung mit Historie

12.3 Langfristig

- Integration mit ERP-Systemen
- GPS-Tracking für Mitarbeiter
- Zeiterfassung mit Stempeluhr
- Buchhaltungs-Integration
- Customer-Portal für Kunden
- API für Drittsysteme

==========
13. PROJEKTMANAGEMENT
==========

13.1 Zeitplan

Phase 1: Analyse & Konzept (1 Woche)
- Anforderungsanalyse
- Datenbank-Design
- Mockups erstellen

Phase 2: Backend-Entwicklung (2 Wochen)
- Datenbank-Struktur aufsetzen
- API-Endpunkte implementieren
- Authentifizierung & Autorisierung
- PDF-Generierung

Phase 3: Frontend-Entwicklung (2 Wochen)
- React-Komponenten erstellen
- Routing implementieren
- Formular-Validierung
- API-Integration

Phase 4: Testing & Bugfixing (1 Woche)
- Manuelle Tests durchführen
- Bugs beheben
- Performance optimieren

Phase 5: Dokumentation (1 Woche)
- Technische Dokumentation
- Benutzerhandbuch
- Code-Kommentare

Gesamtdauer: 7 Wochen

13.2 Herausforderungen

- Komplexe Berechtigungslogik mit drei Rollen
- Status-Übergänge korrekt abbilden
- PDF-Generierung mit TCPDF
- Supabase API-Integration
- Responsive Design für alle Geräte

==========
14. FAZIT
==========

14.1 Zielerreichung

Alle Hauptziele wurden erfolgreich erreicht:
✓ Digitale Erfassung von Serviceaufträgen
✓ Rollenbasierte Zugriffskontrolle funktioniert
✓ Kompletter Workflow implementiert
✓ PDF-Export funktioniert einwandfrei
✓ Responsive Design für alle Geräte
✓ Professionelles User Interface

14.2 Gelerntes

- Moderne Webentwicklung mit React
- RESTful API-Design
- Datenbank-Design mit PostgreSQL
- Authentifizierung & Autorisierung
- PDF-Generierung mit PHP
- Responsive Webdesign mit Bootstrap
- Git/GitHub Versionskontrolle

14.3 Persönliche Einschätzung

Das Projekt hat mir sehr viel gebracht. Ich konnte meine Kenntnisse in React und PHP stark vertiefen und habe gelernt, wie man eine komplette Webanwendung von A bis Z entwickelt. Besonders herausfordernd war die Berechtigungslogik mit den drei verschiedenen Rollen und den unterschiedlichen Status-Übergängen.

Die Arbeit mit Supabase als Cloud-Datenbank war eine neue Erfahrung und hat gut funktioniert. Die PDF-Generierung mit TCPDF war anfangs kompliziert, aber am Ende hat alles geklappt.

Insgesamt bin ich sehr zufrieden mit dem Ergebnis und stolz auf das, was ich geschaffen habe.

==========

15. ANHANG
==========

15.1 Verzeichnisstruktur

Projekt/
├── api/
│   ├── auftraege/   # Auftrags-API-Endpunkte
│   ├── auth/   # Authentifizierungs-Endpunkte
│   ├── classes/   # PHP-Klassen (TCPDF)
│   └── config/   # Konfigurationsdateien
├── frontend/
│   ├── public/   # Statische Assets
│   └── src/
│   ├── components/   # React-Komponenten
│   └── services/   # API-Services
└── vendor/   # Composer-Dependencies

15.2 Verwendete Bibliotheken

Frontend:
- react: 19.0.0
- react-router-dom: 7.1.3
- axios: 1.7.9
- bootstrap: 5.3.3
- bootstrap-icons: 1.11.3

Backend:
- tecnickcom/tcpdf: 6.7.7

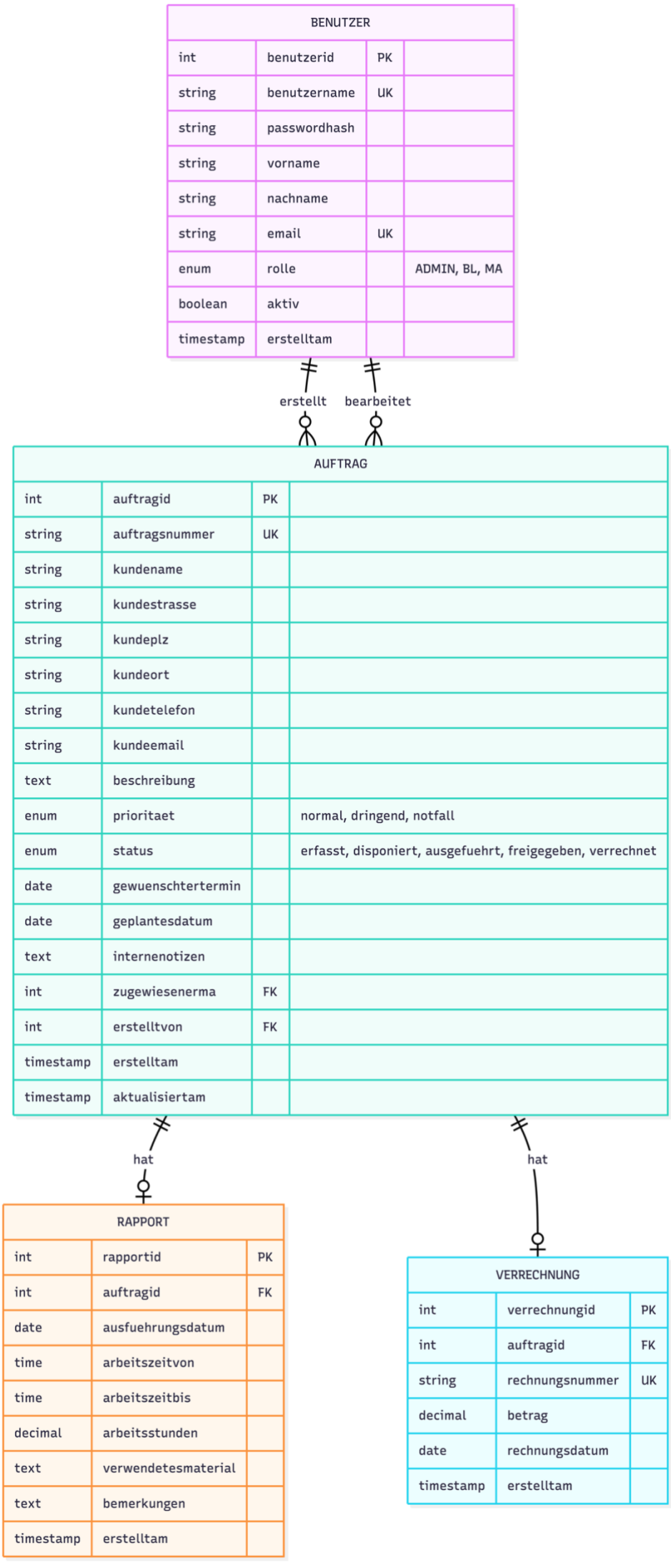
15.3 Konfigurationsdateien

- package.json - Frontend-Dependencies
- composer.json - Backend-Dependencies
-

.gitignore - Git Ignore-Regeln

15.4 Lizenz

Dieses Projekt wurde für Bildungszwecke erstellt.


BENUTZER
int benutzerid PK
string benutzername UK
string passwordhash
string vorname
string nachname
string email UK
enum rolle ADMIN, BL, MA
boolean aktiv
timestamp erstelltam
erstellt
bearbeitet
AUFTRAG
int auftragid PK
string auftragsnummer UK
string kundename
string kundestrasse
string kundeplz
string kundeort
string kundetelefon
string kundeemail
text beschreibung
enum prioritaet normal, dringend, notfall
enum status erfasst, disponiert, ausgefuehrt, freigegeben, verrechnet
date gewuenschtertermin
date geplantesdatum
text internenotizen
int zugewiesenerma FK
int erstelltvon FK
timestamp erstelltam
timestamp aktualisiertam
hat
hat
RAPPORT
int rapportid PK
int auftragid FK
date ausfuehrungsdatum
time arbeitszeitvon
time arbeitszeitbis
decimal arbeitsstunden
text verwendetesmaterial
text bemerkungen
timestamp erstelltam
VERRECHNUNG
int verrechnungid PK
int auftragid FK
string rechnungsnummer UK
decimal betrag
date rechnungsdatum
timestamp erstelltam