Technisches Brainstorming InvoiceFlow

InvoiceFlow - Technisches Brainstorming

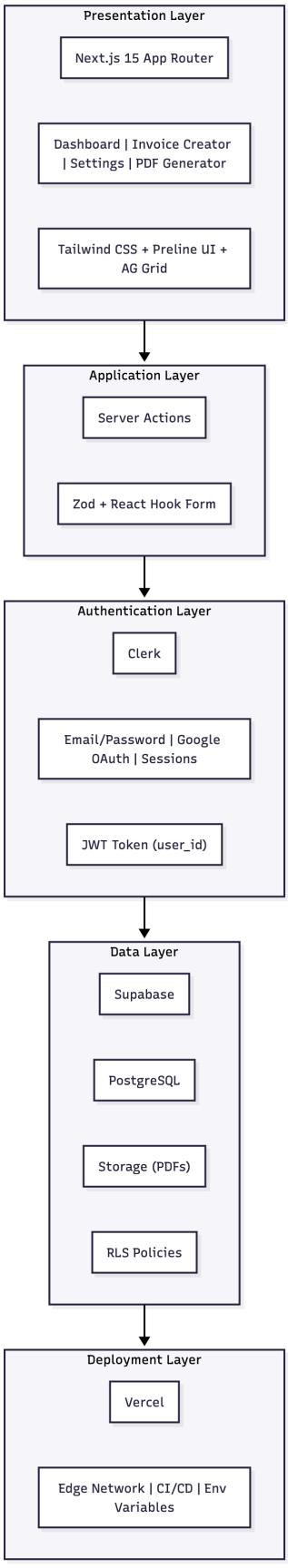
E-Business and Entrepreneurship - Technische Dokumentation

Student: Yejay Demirkan (927077)

1. Softwarearchitektur

1.1 Architektur-Übersicht

InvoiceFlow folgt einer modernen **Jamstack-Architektur** mit serverless Backend-Komponenten. Die Architektur ist auf Einfachheit, Sicherheit und schnelle Entwicklung optimiert.

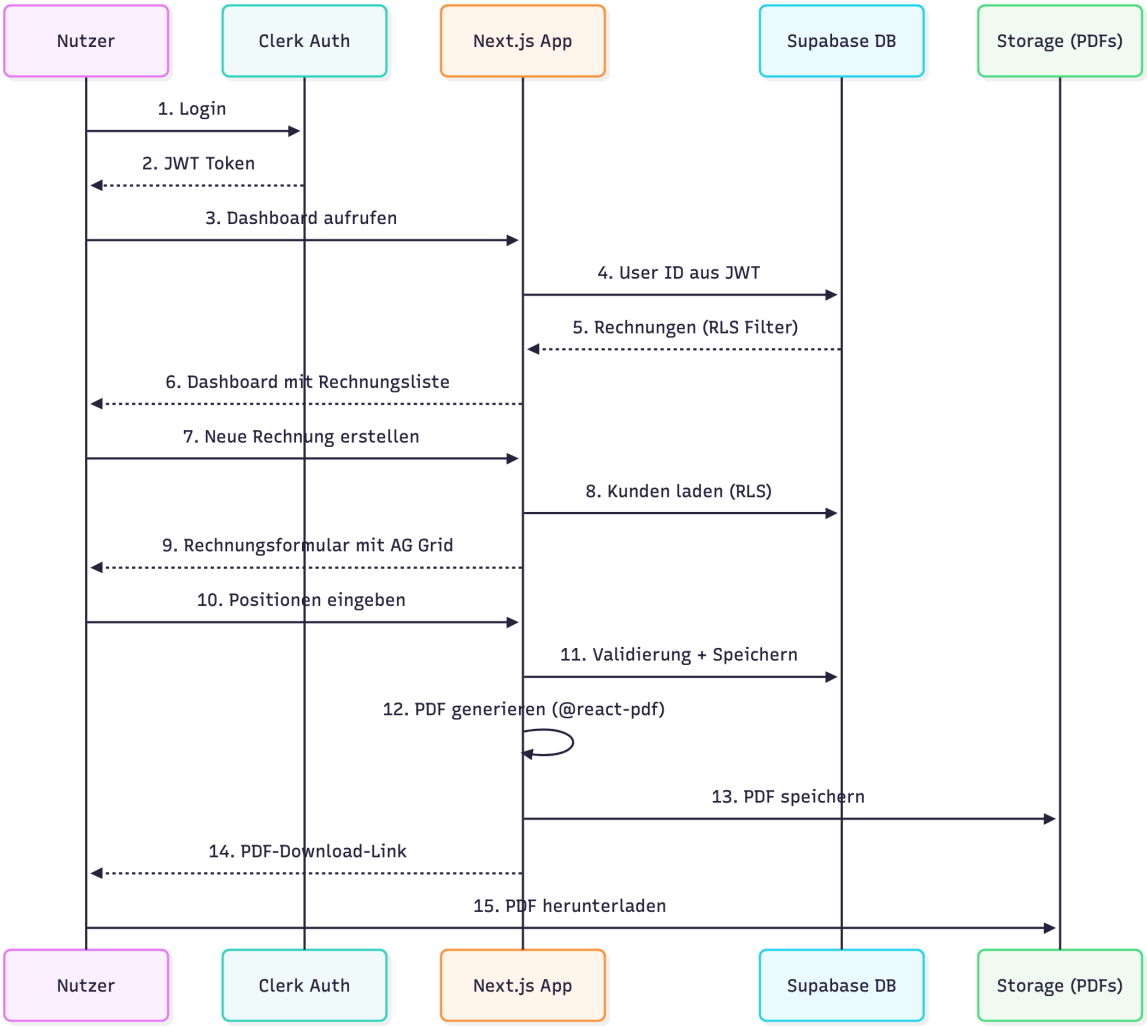

Presentation Layer
Next.js 15 App Router
Dashboard | Invoice Creator | Settings | PDF Generator
Tailwind CSS + Preline UI + AG Grid
Application Layer
Server Actions
Zod + React Hook Form
Authentication Layer
Clerk
Email/Password | Google OAuth | Sessions
JWT Token (user_id)
Data Layer
Supabase
PostgreSQL
Storage (PDFs)
RLS Policies
Deployment Layer
Vercel
Edge Network | CI/CD | Env Variables

1.2 Schichtenbeschreibung

Schicht	Komponenten	Verantwortlichkeit
Presentation	Next.js Pages, React Components, AG Grid	Benutzeroberfläche, Formulare, Datenvisualisierung
Application	Server Actions, Zod Validation	Geschäftslogik, Datenvalidierung, PDF-Generierung
Authentication	Clerk	Benutzerauthentifizierung, Session-Management, JWT-Token
Data	Supabase PostgreSQL, Storage	Datenpersistenz, Dateispeicherung, Zugriffskontrolle (RLS)
Deployment	Vercel	Hosting, CDN, automatische Deployments

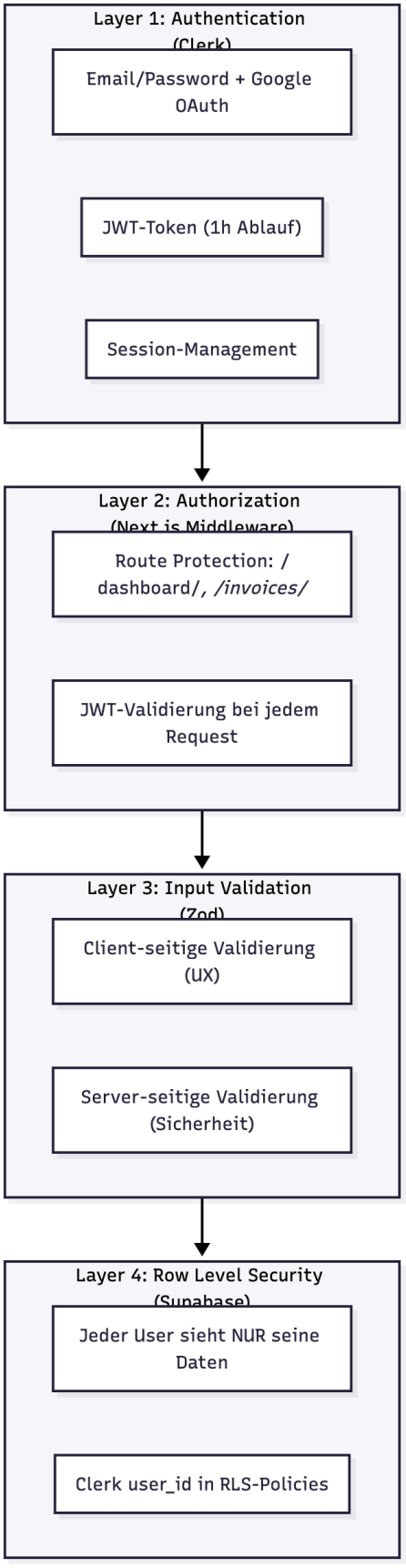
1.3 Datenfluss: Rechnungserstellung

Der folgende Ablauf zeigt den kritischen Pfad von der Anmeldung bis zum PDF-Download:



1.4 Sicherheitsarchitektur

Die Sicherheit basiert auf dem **Zero-Trust-Prinzip** mit mehreren Verteidigungsschichten:


Layer 1: Authentication (Clerk)
Email/Password + Google OAuth
JWT-Token (1h Ablauf)
Session-Management
Layer 2: Authorization (Next.js Middleware)
Route Protection: /dashboard/, /invoices/
JWT-Validierung bei jedem Request
Layer 3: Input Validation (Zod)
Client-seitige Validierung (UX)
Server-seitige Validierung (Sicherheit)
Layer 4: Row Level Security (Supabase)
Jeder User sieht NUR seine Daten
Clerk user_id in RLS-Policies

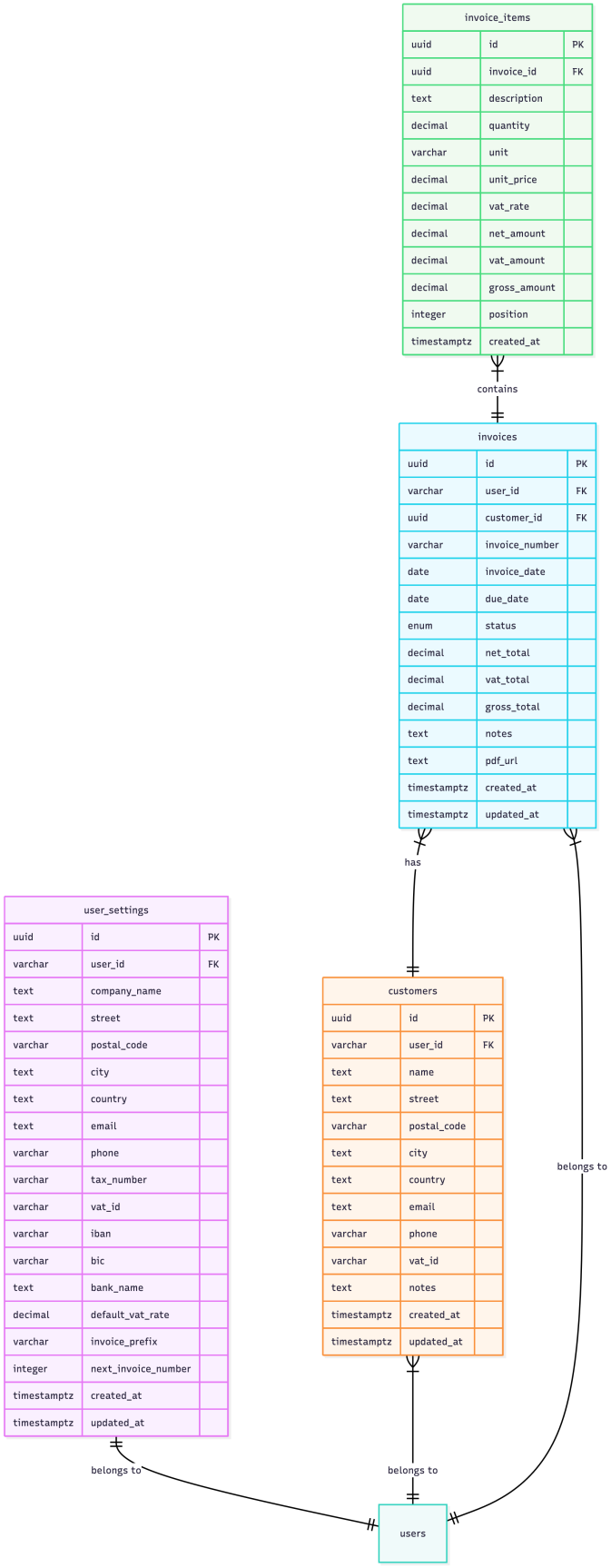
RLS-Policy Beispiel:

```
-- Nutzer sehen nur ihre eigenen Rechnungen
CREATE POLICY "Users can only access their own invoices"
ON invoices FOR ALL
USING (user_id = auth.jwt() ->> 'sub');
```

---

2. Datenmodell (ER-Diagramm)

2.1 Entity-Relationship-Diagramm


invoice_items
uuid id PK
uuid invoice_id FK
text description
decimal quantity
varchar unit
decimal unit_price
decimal vat_rate
decimal net_amount
decimal vat_amount
decimal gross_amount
integer position
timestamptz created_at
contains
invoices
uuid id PK
varchar user_id FK
uuid customer_id FK
varchar invoice_number
date invoice_date
date due_date
enum status
decimal net_total
decimal vat_total
decimal gross_total
text notes
text pdf_url
timestamptz created_at
timestamptz updated_at
has
belongs to
user_settings
uuid id PK
varchar user_id FK
text company_name
text street
varchar postal_code
text city
text country
text email
varchar phone
varchar tax_number
varchar vat_id
varchar iban
varchar bic
text bank_name
decimal default_vat_rate
varchar invoice_prefix
integer next_invoice_number
timestamptz created_at
timestamptz updated_at
customers
uuid id PK
varchar user_id FK
text name
text street
varchar postal_code
text city
text country
text email
varchar phone
varchar vat_id
text notes
timestamptz created_at
timestamptz updated_at
belongs to
belongs to
belongs to
users

2.2 Kardinalitäten

Beziehung	Kardinalität	Beschreibung
User → user_settings	1:1	Jeder User hat genau eine Einstellung
User → customers	1:N	Ein User kann viele Kunden haben
User → invoices	1:N	Ein User kann viele Rechnungen haben
Customer → invoices	1:N	Ein Kunde kann viele Rechnungen haben
Invoice → invoice_items	1:N	Eine Rechnung hat mindestens eine Position

2.3 Datenbankschema (DDL)

```
CREATE TYPE invoice_status AS ENUM ('draft', 'open', 'paid', 'cancelled');
CREATE TABLE user_settings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id VARCHAR(255) NOT NULL UNIQUE,
    company_name TEXT NOT NULL,
    street TEXT,
    postal_code VARCHAR(10),
    city TEXT,
    country TEXT DEFAULT 'Deutschland',
    email TEXT,
    phone VARCHAR(50),
    tax_number VARCHAR(50),
    vat_id VARCHAR(50),
    iban VARCHAR(34),
    bic VARCHAR(11),
    bank_name TEXT,
    default_vat_rate DECIMAL(5,2) DEFAULT 19.00,
    invoice_prefix VARCHAR(10) DEFAULT 'INV-',
    next_invoice_number INTEGER DEFAULT 1,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

```
CREATE TABLE customers (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id VARCHAR(255) NOT NULL,
    name TEXT NOT NULL,
    street TEXT,
    postal_code VARCHAR(10),
    city TEXT,
    country TEXT DEFAULT 'Deutschland',
    email TEXT,
    phone VARCHAR(50),
    vat_id VARCHAR(50),
    notes TEXT,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);
CREATE TABLE invoices (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id VARCHAR(255) NOT NULL,
    customer_id UUID NOT NULL REFERENCES customers(id),
    invoice_number VARCHAR(50) NOT NULL,
    invoice_date DATE NOT NULL DEFAULT CURRENT_DATE,
    due_date DATE,
    status invoice_status DEFAULT 'draft',
    net_total DECIMAL(12,2) DEFAULT 0,
    vat_total DECIMAL(12,2) DEFAULT 0,
    gross_total DECIMAL(12,2) DEFAULT 0,
    notes TEXT,
    pdf_url TEXT,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
    UNIQUE(user_id, invoice_number)
);
CREATE TABLE invoice_items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    invoice_id UUID NOT NULL REFERENCES invoices(id) ON DELETE CASCAD
E,
```

```
    description TEXT NOT NULL,
    quantity DECIMAL(10,2) NOT NULL DEFAULT 1,
    unit VARCHAR(20) DEFAULT 'Stk.',
    unit_price DECIMAL(12,2) NOT NULL,
    vat_rate DECIMAL(5,2) NOT NULL DEFAULT 19.00,
    net_amount DECIMAL(12,2) NOT NULL,
    vat_amount DECIMAL(12,2) NOT NULL,
    gross_amount DECIMAL(12,2) NOT NULL,
    position INTEGER NOT NULL DEFAULT 0,
    created_at TIMESTAMPTZ DEFAULT NOW()
);
-- Performance-IndizesCREATE INDEX idx_customers_user_id ON customers
(user_id);
CREATE INDEX idx_invoices_user_id ON invoices(user_id);
CREATE INDEX idx_invoices_customer_id ON invoices(customer_id);
CREATE INDEX idx_invoice_items_invoice_id ON invoice_items(invoice_id);
```

2.4 Row Level Security

```
-- RLS aktivieren
ALTER TABLE user_settings ENABLE ROW LEVEL SECURITY;
ALTER TABLE customers ENABLE ROW LEVEL SECURITY;
ALTER TABLE invoices ENABLE ROW LEVEL SECURITY;
ALTER TABLE invoice_items ENABLE ROW LEVEL SECURITY;

-- Beispiel: Direkte Tabellen (user_settings, customers, invoices)
CREATE POLICY "users_own_data" ON customers FOR ALL
USING (user_id = auth.jwt() ->> 'sub');

-- Beispiel: Verknüpfte Tabelle (invoice_items über invoices)
CREATE POLICY "items_via_invoice" ON invoice_items FOR ALL
USING (
    EXISTS (
        SELECT 1 FROM invoices
        WHERE invoices.id = invoice_items.invoice_id
```

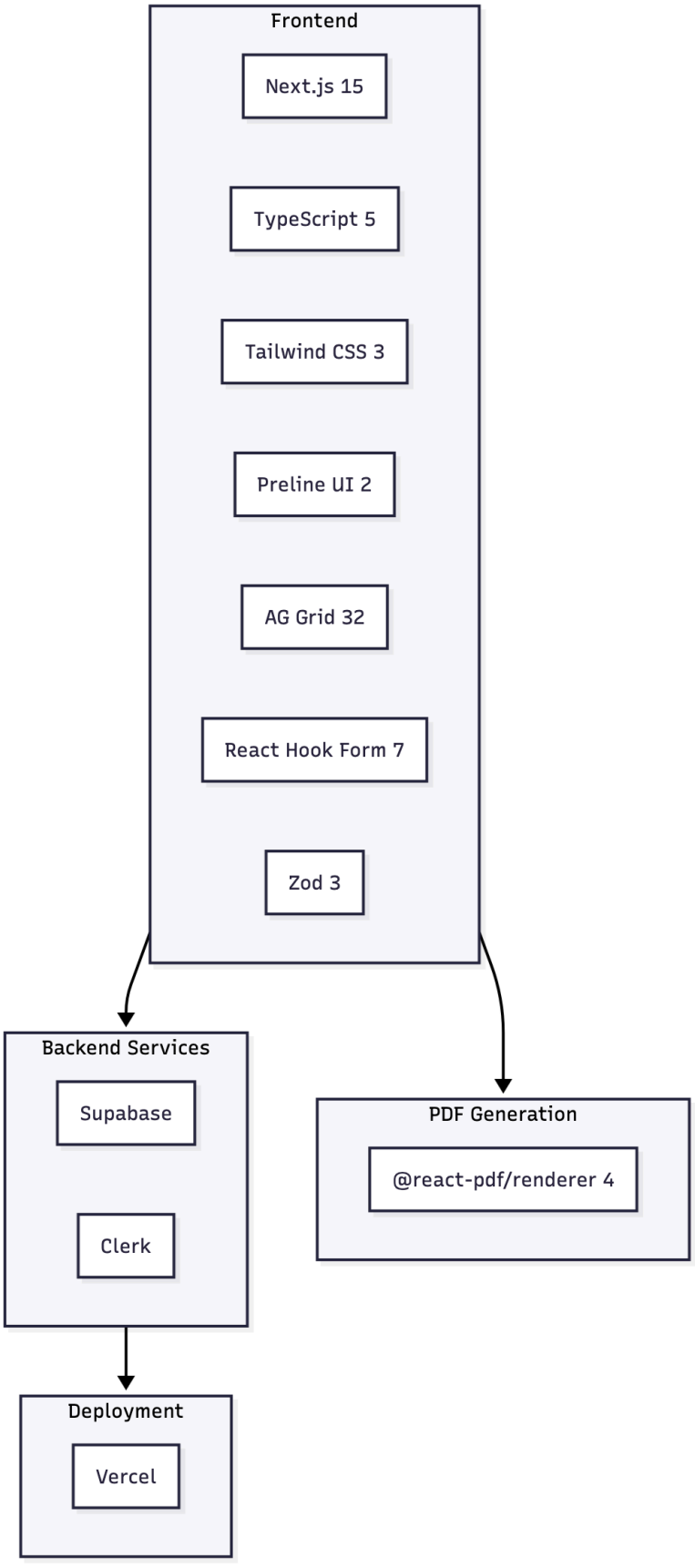
```
        AND invoices.user_id = auth.jwt() ->> 'sub'
    )
);
```

3. Technologien und Frameworks

3.1 Technologie-Stack

Kategorie	Technologie	Version	Begründung
Framework	Next.js	15.x	Server Components, App Router, Server Actions
Sprache	TypeScript	5.x	Type-Safety, bessere Entwicklererfahrung
Styling	Tailwind CSS	3.x	Utility-First, schnelle Entwicklung
UI-Bibliothek	Preline UI	2.x	Tailwind-native Komponenten
Datentabelle	AG Grid	32.x	Enterprise-Grade, Excel-ähnlich
Formulare	React Hook Form	7.x	Performant, uncontrolled components
Validierung	Zod	3.x	TypeScript-native Schema-Validierung
PDF	@react-pdf/renderer	4.x	React-basiert, Server-Side rendering
Datenbank	Supabase (PostgreSQL)	-	Open Source, RLS, Storage
Auth	Clerk	-	Managed Auth, Social Login
Hosting	Vercel	-	Optimiert für Next.js, Edge Network

3.2 Tech Stack Übersicht


Frontend
Next.js 15
TypeScript 5
Tailwind CSS 3
Preline UI 2
AG Grid 32
React Hook Form 7
Zod 3
Backend Services
Supabase
Clerk
PDF Generation
@react-pdf/renderer 4
Deployment
Vercel

3.3 Supabase vs. Alternativen

Kriterium	Supabase	Firebase	PlanetScale
Datenbank	PostgreSQL	NoSQL	MySQL
Row Level Security	Native	Firestore Rules	Nein
Open Source	Ja	Nein	Teilweise
Storage	Inklusive	Separat	Nein
Free Tier	Großzügig	Begrenzt	Begrenzt

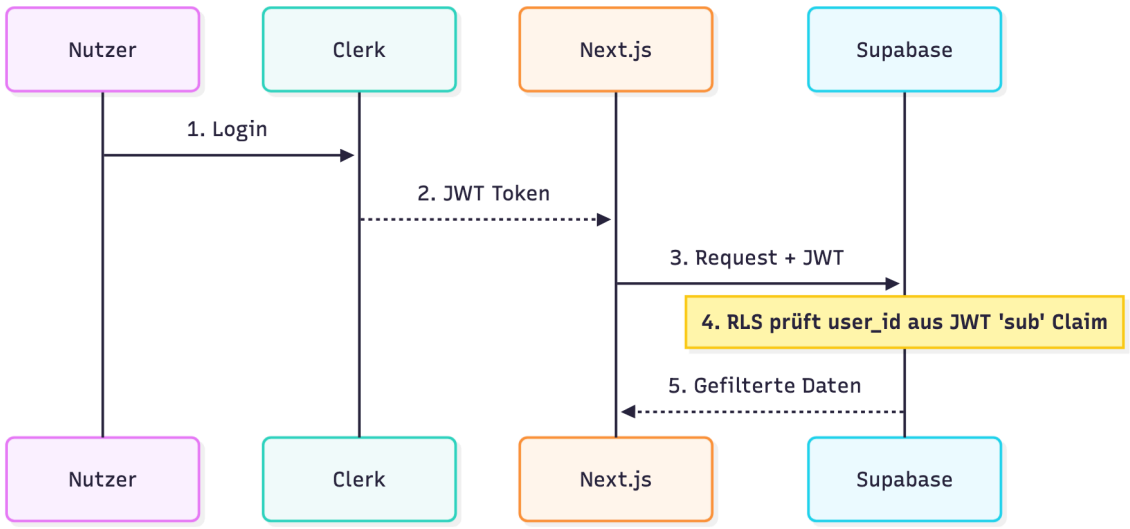
Entscheidung: Supabase bietet PostgreSQL mit nativer RLS, integriertem Storage und großzügigem Free Tier.

---

4. Externe Daten und Services

4.1 Service-Integration

Für den MVP werden bewusst nur zwei externe Services integriert:



4.2 Clerk Integration

Zweck: Vollständiges Authentication-System ohne eigene Implementierung

```
// middleware.ts
import { clerkMiddleware, createRouteMatcher } from '@clerk/nextjs/server';

const isProtectedRoute = createRouteMatcher([
  '/dashboard(.*)',
  '/invoices(.*)',
  '/customers(.*)',
  '/settings(.*)',
]);

export default clerkMiddleware(async (auth, req) => {
  if (isProtectedRoute(req)) {
    await auth.protect();
  }
});
```

4.3 Supabase Integration

Zweck: Datenbank, Storage und Zugriffskontrolle

```
// lib/supabase.ts
import { createClient } from '@supabase/supabase-js';
import { auth } from '@clerk/nextjs/server';

export async function createServerSupabaseClient() {
  const { getToken } = await auth();
  const supabaseAccessToken = await getToken({ template: 'supabase' });

  return createClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
    {
      global: {
        headers: {
```

```
          Authorization: `Bearer ${supabaseAccessToken}`,
        },
      },
    }
  );
}
```

4.4 Nicht integrierte Services (MVP)

Service	Zweck	Begründung für Ausschluss
Email-Service	Rechnungsversand	NICE-TO-HAVE
Payment (Stripe)	Zahlungsabwicklung	Nicht MVP-relevant
Analytics	Nutzerverhalten	Später für Optimierung

---

5. Funktionsumfang (MUST-HAVE vs NICE-TO-HAVE)

5.1 MVP-Philosophie

Der MVP folgt dem **Vertical Slice**-Ansatz: Ein vollständiger Durchstich durch alle Architekturschichten mit minimalem Funktionsumfang.

Kernhypothese: Freelancer bevorzugen eine schnelle, fokussierte Lösung gegenüber Feature-reichen Alternativen.

Validierungskriterium: Time-to-Invoice unter 2 Minuten von Login bis PDF-Download.

5.2 MUST-HAVE Features

1. Authentication (Priorität 1)

- Login via Email/Passwort und Google OAuth
- Automatische Weiterleitung zum Dashboard
- Protected Routes (Middleware)

Akzeptanzkriterien:
- User kann sich registrieren und einloggen

- Nicht-eingeloggte User werden zu /sign-in weitergeleitet

2. Dashboard (Priorität 2)

- Übersicht aller Rechnungen des Users
- Anzeige: Rechnungsnummer, Kunde, Datum, Betrag, Status
- Button “Neue Rechnung erstellen”

Akzeptanzkriterien:
- Rechnungsliste zeigt nur eigene Rechnungen (RLS)
- Empty State wenn keine Rechnungen vorhanden

3. Rechnungserstellung (Priorität 1)

- Kundenauswahl (Dropdown)
- Rechnungsdatum und Fälligkeitsdatum
- AG Grid Tabelle für Positionen (Beschreibung, Menge, Einheit, Preis, MwSt.)
- Automatische Berechnung aller Summen
- Fortlaufende Rechnungsnummer

Akzeptanzkriterien:
- AG Grid erlaubt Hinzufügen/Entfernen von Zeilen
- Berechnungen erfolgen in Echtzeit
- Validierung verhindert ungültige Eingaben

4. PDF-Generierung (Priorität 1)

- Professionelles Rechnungs-Template
- Absender- und Empfänger-Daten
- Positionstabelle mit Summenblock
- One-Click-Download
- PDF-Speicherung in Supabase Storage

Akzeptanzkriterien:
- PDF enthält alle Pflichtangaben einer deutschen Rechnung
- Download funktioniert zuverlässig

5. Benutzereinstellungen (Priorität 2)

- Firmendaten, Kontaktdaten, Steuerdaten
- Bankverbindung (IBAN, BIC)
- Standard-MwSt.-Satz und Rechnungsnummern-Präfix

Akzeptanzkriterien:
- Einstellungen erscheinen korrekt auf PDFs

6. Kundenverwaltung (Priorität 3)

- Demo-Kunden für neue User (vorseeded)
- Kundenauswahl bei Rechnungserstellung

5.3 NICE-TO-HAVE Features

Stufe 1 (Hohe Priorität):
- Kunden CRUD (Anlegen, Bearbeiten, Löschen)
- Rechnungsstatus-Management
- Such- und Filterfunktion

Stufe 2 (Mittlere Priorität):
- Rechnungsvorschau vor PDF-Generierung
- Rechnung duplizieren
- Kleinunternehmer-Regelung
- Dark Mode

Stufe 3 (Post-Launch):
- Email-Versand direkt aus App
- Dashboard Analytics
- Wiederkehrende Rechnungen
- Multi-Währung
- XRechnung/ZUGFeRD Export

---

6. Aktueller Entwicklungsstand

6.1 Ausgangslage

Für das Kursprojekt wird ein **fokussierter MVP** entwickelt, der die Kernhypothese validiert: Freelancer bevorzugen eine schnelle, einfache Lösung gegenüber Feature-reichen Alternativen.

Geplante Technologie-Entscheidungen:
- Next.js 15 mit App Router und Server Actions
- Clerk für Authentication (Email + Google OAuth)
- Supabase für Datenbank und Storage
- AG Grid für Excel-ähnliche Rechnungseingabe
- @react-pdf/renderer für PDF-Generierung

MVP-Scope:
- Neues Supabase-Projekt mit fokussiertem Schema
- Neues Clerk-Projekt für Authentifizierung
- Vercel-Deployment für Hosting
- Reduziertes Feature-Set gemäß MUST-HAVE-Liste

6.2 Kritische Integrationspunkte

Integration	Risiko	Beschreibung
Clerk ↔ Next.js	Mittel	ClerkProvider, Middleware, getAuth() in Server Actions
Supabase RLS + Clerk	Hoch	JWT Template, RLS mit auth.jwt() ->> 'sub'
AG Grid	Mittel	Column Definitions, Editable Cells, Berechnungen
@react-pdf	Niedrig	Server-side Rendering, Storage Upload

6.3 Risiken und Mitigationen

Risiko	Wahrscheinlichkeit	Impact	Mitigation
Clerk-Supabase JWT-Integration	Mittel	Hoch	Frühe Validierung, Fallback zu Supabase Auth
AG Grid Lizenzprobleme	Niedrig	Mittel	Community Edition ausreichend
PDF-Rendering langsam	Niedrig	Niedrig	Server-side Rendering, Caching

Risiko	Wahrscheinlichkeit	Impact	Mitigation
Zeitplan zu ambitioniert	Mittel	Mittel	NICE-TO-HAVEs flexibel priorisieren

---

Anhang: Präsentation 6. Januar 2025

Vorgeschlagene Gliederung (10-12 Minuten)

1. **Architektur-Übersicht** (2 Min) - Schichtenmodell, Security-Architektur
2. **Datenmodell** (2 Min) - ER-Diagramm, RLS-Konzept
3. **Tech Stack** (1 Min) - Kernentscheidungen begründen
4. **Externe Services** (1 Min) - Clerk + Supabase, bewusste Beschränkung
5. **Funktionsumfang** (3 Min) - MUST-HAVE vs NICE-TO-HAVE, MVP-Philosophie
6. **Entwicklungsstand** (2 Min) - Integrationspunkte, Risiken
7. **Fragen & Diskussion** (1-2 Min)

Teile dieses Dokuments wurden mit Hilfe von Generativer KI erstellt, alle Informationen wurden jedoch von Hand überprüft und angepasst.