BitFlow

Statistiken über Zeitaufwand

Stunden pro Person

Name	Zeitaufwand	Hauptbeitrag
Florian Blum	18h 15m	Demo-Vorbereitung, Wochenaufgaben
Moritz Czekalski	33h 15m	Kabel-Programmierung
Simon Just	22h 45m	Dokumentation, Blogbeiträge & Kommentare
Mohid Syed	27h 25m	Gatter-Programmierung

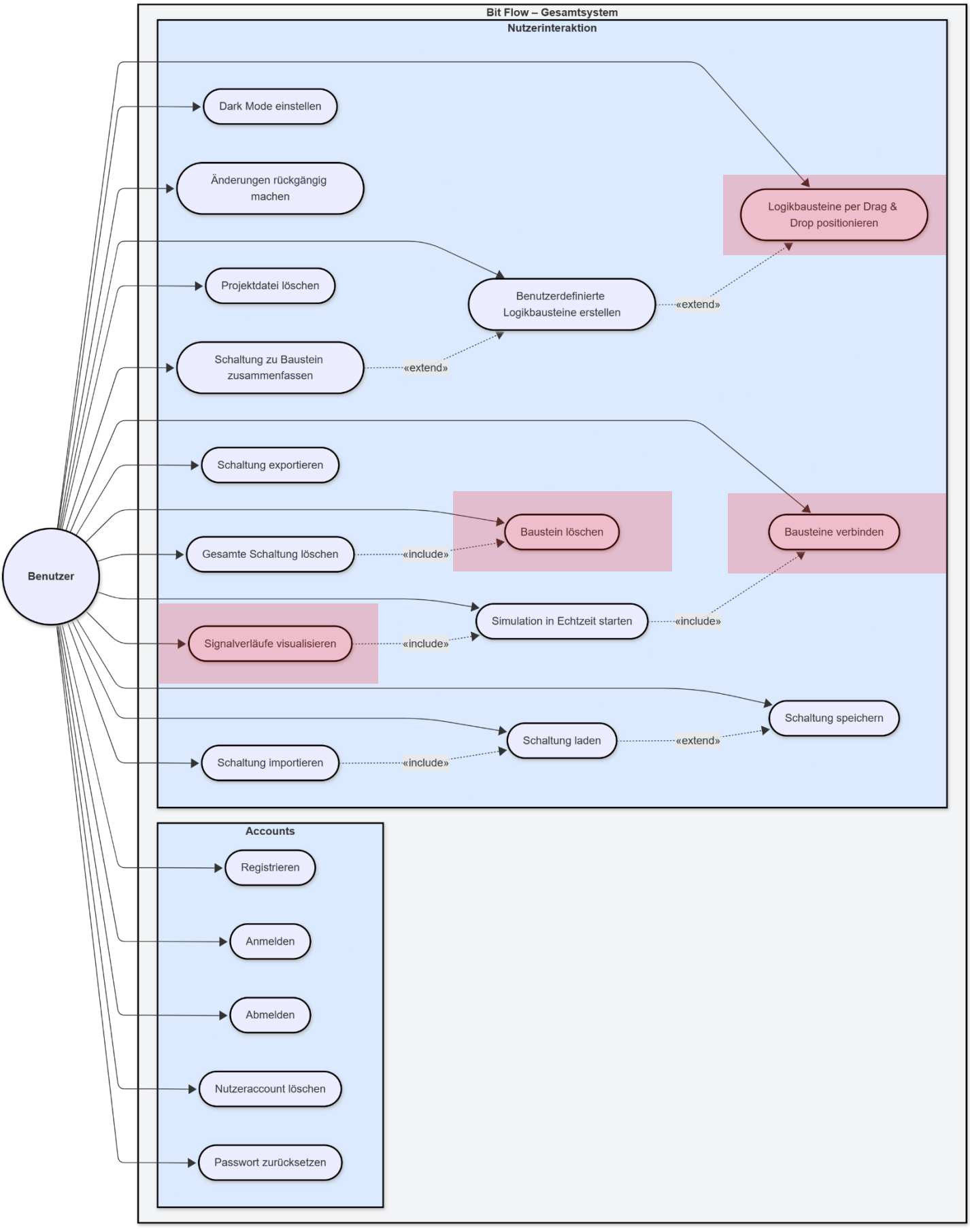
Stunden pro Workflow

Workflow	Zeitaufwand	Beispiele
Requirement Analyse	14h 45m	SRS, ASR, Use-Case-Beschreibungen, User-Stories
Projektmanagement	10h 50m	Aufgabenplanung, Teamkoordination, Projektstruktur
Implementierung	50h 10m	Programmierung, Simulation, Diagramme, technische Umsetzung
Dokumentation	15h 45m	Blogeinträge, Kommentare, Feedback
Präsentation	10h 10m	Demos, Präsentationsvorbereitung, Handout

Stunden pro Phase

Phase	Zeitaufwand
Projekt Erstellung	26h 10m
Frontend aufsetzen	8h 30m
Erste Implementierung	37h 15m
Demo vorbereiten	29h 45m

Overall Use-Case-Diagramm



Architekturstile-/entscheidungen

Architekturentscheidung	Beschreibung / Stil	Hauptargumente / Motivation
Client-Server + Single Page Application (SPA)	Das Frontend läuft als React SPA im Browser, das Backend als REST-API	• Plattformunabhängigkeit, läuft auf nahezu jedem Gerät und OS • Kein Installationsaufwand: Nutzer öffnet einfach Browser • Klare Trennung zwischen UI und Logik / Persistenz
Monolithisches Backend (vs. Microservices)	Ein einzelner ASP.NET-Server übernimmt Authentifizierung, Benutzerverwaltung, Projekt- & Schaltungs-Persistenz, ggf. Export/Import	• Einfachere Entwicklung und Wartung bei kleinem Team • Ausreichend für das erwartete Nutzungsszenario
State- und Logiktrennung: Frontend für UI, Backend für Business- und Persistenz-Logik	Die Benutzerinteraktionen, Layout, Drag & Drop etc. laufen im Frontend. Backend liefert User-Daten, Projekt-/Schaltungsdaten, Speichern/Laden.	• UI-Reaktivität & schnelle Interaktionen im Browser • Sicherheit und Datenintegrität (z.B. User-Passwörter, DB-Zugriff) bleiben auf dem Server
Optionale Persistenz lokal oder in Datenbank	Für Gäste oder nicht eingeloggte Nutzer: Speicherung ggf. im Browser-Storage; für registrierte Nutzer: Persistenz in DB	• Komfort für Nutzer ohne Account • Flexibilität und geringer Einstiegshürde • Möglichkeit, später zur vollwertigen DB-Persistenz zu migrieren

Software Tools/Plattformen/Techniken

Bereich	Technologie / Tool	Zweck / Einsatz
Backend	C#, ASP.NET Core Web API	Implementierung der Server-Logik, Authentifizierung, Datenzugriff, API für Frontend
Frontend	React + TypeScript + Vite	Entwicklung der Web-UI, Komponenten, Canvas, Drag-Drop, Interaktion
Styling / UI Design	Tailwind CSS	Schnelles, modernes, responsives Layout; Dark/Light Mode leicht umsetzbar
State Management / Komponentenstruktur	React (komponentenbasiert)	Modularität, Wiederverwendbarkeit, klare Trennung von UI-Elementen
Testing	Jest (Frontend), xUnit (Backend), Postman (API-Tests), manuelle Tests	Qualitätssicherung: Unit-Tests, API-Tests, manuelle UI- und Browser-Tests
Versionsverwaltung & Projektmanagement	GitHub, YouTrack	Quellcodeverwaltung, Issue-Tracking, Team-Kollaboration
Deployment / Hosting	GitHub Pages (für Frontend) oder ASP.NET-kompatibler Webserver	Auslieferung der Webanwendung; einfache Bereitstellung
Persistenz / Speichern	Browser LocalStorage (für Gastmodus), serverseitige DB (für eingeloggte Nutzer)	Speicherung von Projekten/Schaltungen, Nutzerkonten, Export/Import-Funktionalität
UI/UX Features	Dark Mode, Undo/Redo, Drag & Drop, responsive GUI	Benutzerfreundlichkeit, gute Usability und moderne Nutzererfahrung