PWA für die Kalenderanwendung "HTWKalender"

Elmar Kresse, Christoph Walther

Beschreibung

Die HTWKalender-PWA ist eine Progressive Web App, die es ermöglicht, den Stundenplan der Hochschule für Technik, Wirtschaft und Kultur Leipzig anzuzeigen. Die App wurde im Rahmen des Moduls "Progressive Web Apps" im Studiengang Informationstechnik an der HTWK Leipzig entwickelt. Sie ist als Fork vom Projekt "HTWKalender" entstanden. Dabei sind spezielle Funktionen für die Offline-Nutzung und die Installation auf mobilen Geräten hinzugekommen.

Technologien

Die HTWKalender-PWA wurde mit folgenden Technologien entwickelt:

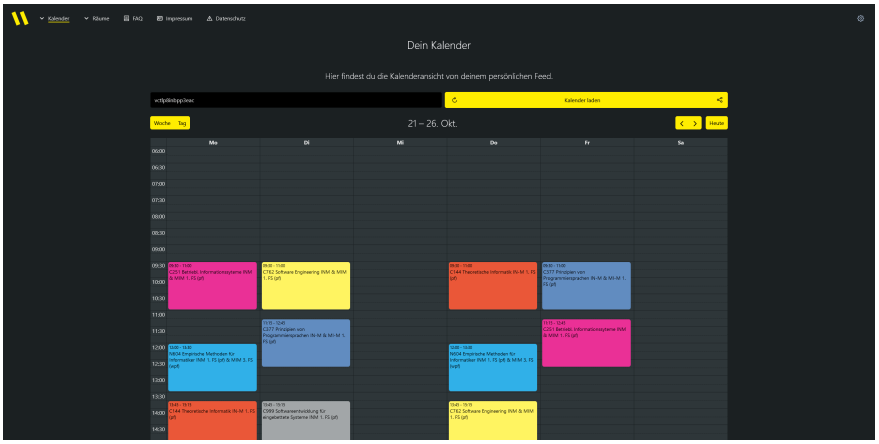
- Vue.js - Web Framework
- PrimeVue - Component Library
- Vite - Build Tool
- Vite-Plugin-PWA - Plugin für PWA-Funktionalitäten
- TypeScript

Features

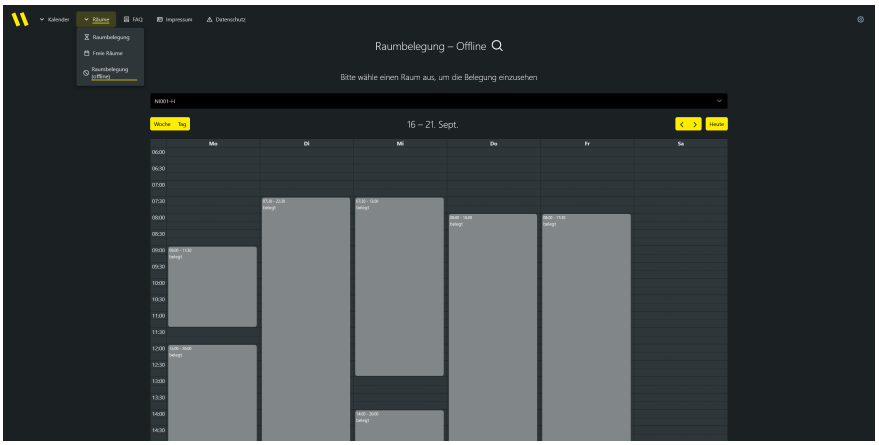
Die HTWKalender-PWA bietet folgende Features:

- Service Worker aus dem Vite-Plugin-PWA mit Workbox zum Caching von Seiten und API-Anfragen (Kalender-Feeds und Raumbelegung)
- ~~Push-Benachrichtigungen für kommende Events~~ (siehe unten)
- Offline-Raumbelegungsprüfung mit feingranularer Anzeige pro Raum
- Offline Kalenderanzeige inklusive Detailinformationen in einem Popup für jedes Event
- ~~Offline-Kalenderbearbeitung~~
- Frei wählbare Startseite bei Öffnen als PWA
- Teilen von Kalender-Links über die Web Share API
- Entwicklungsumgebung mit Docker-Compose und lokalem HTTPs zur PWA-Tauglichkeit

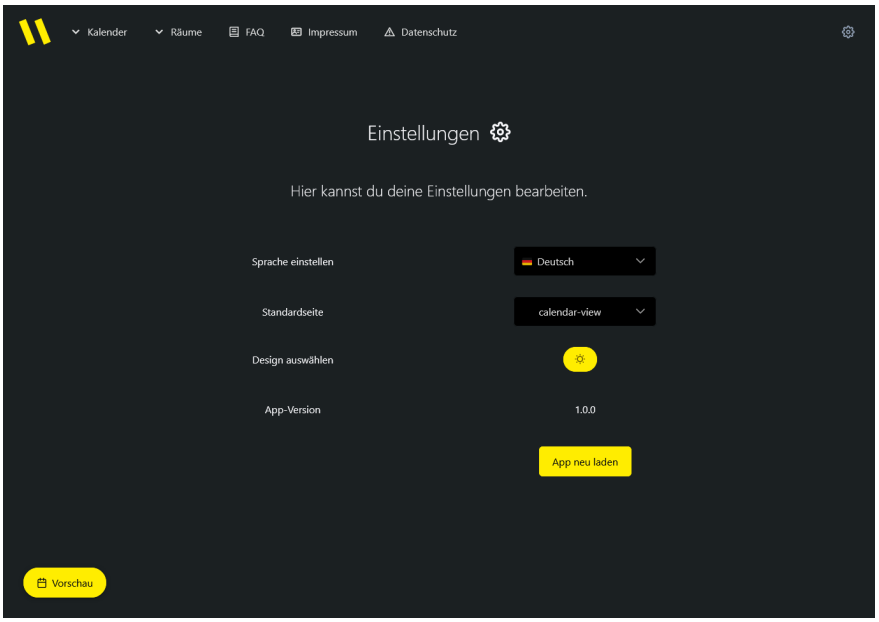
Showcase der wichtigsten Seiten:



Offline Kalender
Der Offline-Kalender ermöglicht es, den persönlichen Kalender-Feed ohne Internetverbindung anzuzeigen. Der Kalender kann manuell aktualisiert werden und über die Share-API mit anderen geteilt werden.



Raumfinder
Der Raumbelegungsplan ermöglicht es, auf die Suche nach freien Lerngruppenräumen zu gehen und deren zukünftige, sowie vergangene Belegung anonym einzusehen.



Einstellungen
In den Einstellungen kann zwischen Dark- und Light-Mode gewechselt und die Startseite beim Öffnen der App festgelegt werden. Zudem wird hier die aktuelle Version der App angezeigt, manuelle Updates sind möglich. Die App ist in drei Sprachen verfügbar, zwischen denen hier gewechselt werden kann: Deutsch, Englisch und Japanisch (automatische Übersetzung).

Alle anderen Features des HTWKalenders, deployed unter cal.htwk-leipzig.de, sind auch in der PWA verfügbar, gehören aber nicht zu den PWA-spezifischen Features, die im Rahmen des Moduls hinzugefügt wurden. Dazu gehören unter anderem auch die englische Sprachvariante und der Dark-Mode.

Raumfinder

Um einen möglichst effizienten Raumbelegungsplan für alle gespeicherten Räume der HTWK Leipzig offline zur Verfügung stellen zu können wurde ein neuer API-Endpunkt `/api/schedule/rooms` entwickelt. Dieser kodiert für jeden Raum ab Semesterbeginn die Belegung in

15 minütigen Schritten, ob ein Event in den jeweiligen Zeitslot fällt binär. Die Daten werden in der PWA lediglich bei Bedarf für einzelne Räume und nur im jeweils dargestellten Zeitbereich dekodiert.

Hierzu wird das Dokument als BSON (Binary JSON) umgewandelt und beim Versenden komprimiert. Dadurch kann der Raumbelegungsplan für alle Räume in einem einzigen Request abgerufen werden und ist problemlos client- wie auch serverseitig cachebar. Eingesetzte Standardkomprimierungsverfahren, wie GZIP oder Brotli, können diese Daten, die aus langen zusammenhängenden Sequenzen von mehreren Nullen und Einsen bestehen, effizient genug komprimieren. Ein Raumbelegungsplan für ein Semester wird somit auf wenige Kilobyte reduziert und kann schnell und platzsparend übertragen und gespeichert werden.

Für das Caching wurde die Stale-While-Revalidate Strategie gewählt. Hierbei wird der Raumbelegungsplan bei jedem Aufruf der Seite aktualisiert, vorausgesetzt eine Internetverbindung ist vorhanden. Dabei wird allerdings immer die vorhergehende Version des Plans angezeigt, die hinreichend aktuell sein sollte. Möchte der Nutzer dennoch die aktuelle Version sehen, die mittlerweile in den Cache geladen sein sollte, kann er einen manuellen Refresh am Ende der Seite auslösen.

Push-Benachrichtigungen

Die Umsetzung von Push-Benachrichtigungen war angedacht, um die typische Funktion von Kalender-Apps zu simulieren. Bei jeder kommenden Lehrveranstaltung, die im Kalender vom Nutzer mit einer Glocke markiert wurde, sollten z. B. 15 Minuten vor Beginn eine Nachricht auf dem Endgerät erscheinen. Für die Umsetzungen wurden zwei mögliche Lösungen in Betracht gezogen:

Serverseitige Lösung

Standardfall für die Implementierung von Push ist auf Smartphones die Verwendung eines Cloud Messaging Services. Bei Android wäre dies Firebase Cloud Messaging (FCM) und bei iOS Apple Push Notification Service (APNs). Hierbei müsste der HTWKalender-Server eine Nachricht erzeugen und an den Service weiterleiten, der diese an alle abonnierten Geräte sendet. Diese Lösung wurde nicht umgesetzt, da der eingesetzte Server für eine möglichst hohe Stabilität auf statisches Hosting und somit einen Austauschbaren HTTP-Server setzt. Im Gegensatz zu anderen Diensten, wie Nachrichtenseiten oder Wetter-Provider, sind die Kalender hingegen höchst individuell, sodass der Server Nachrichten pro Benutzer generieren müsste. Als offizieller Dienst der HTWK Leipzig im nächsten Semester besorgt uns hierbei die Skalierbarkeit, zumal für diesen Aspekt die HTWK App bereits eine Lösung bieten sollte.

In Zukunft wäre jedoch eine solche Lösung denkbar. Beispielsweise durch Einschränkung der Funktionalität, wie setzen fester Intervalle für Benachrichtigungen (immer 15 Minuten vor Veranstaltungsbeginn) und Verwerfen von eigenen Eventnamen, um alle Benutzer mit der gleichen Nachricht zu Versorgen. Dies müsste dann in einen Extra-Service ausgelagert werden, was gut zur Microservice-Architektur des HTWKalenders passen würde.

Clientseitige Lösung

Im Rahmen der PWA, wo hauptsächlich die Offline-Funktionalität erwünscht ist, wäre eine reine

clientbasierte Lösung zu bevorzugen. Hierbei wird der Service Worker genutzt, um die Nachrichten zu generieren und es ist kein zusätzlicher Server sowie keine Internetverbindung notwendig. Dazu ist es allerdings nötig, trotz Energiesparroutinen und Einschränkungen des Hintergrundbetriebs den Service Worker am Laufen zu Halten. Hierzu haben wir recherchiert und haben zwei Working Drafts gefunden, die sich mit dem Thema beschäftigen:

Working Draft	Beschreibung
Scheduled Task API (https://issues.chromium.org/issues/41417116)	Hiermit ist es möglich, Routinen im Service Worker zu bestimmten Zeitpunkten oder periodisch zu starten.
Notification Triggers API (https://developer.chrome.com/docs/web-platform/notification-triggers)	Hiermit können Benachrichtigungen für die Zukunft geplant werden, die zu einem bestimmten Zeitpunkt erscheinen sollen. Dieses Feature wurde unter anderem explizit für Kalender-Anwendungen konzipiert.

Beide APIs stammen aus der Chrome-Entwicklung und wurden noch nicht umgesetzt. Die Notification Triggers API, die exakt die benötigten Funktionen bereitstellen sollte, wurde inzwischen offiziell eingestellt. Somit ist es aktuell nicht möglich, lokale Push-Benachrichtigungen in einer PWA zu realisieren, wie wir sie gerne für dieses Projekt nutzen würden.

Projektstruktur

Für den HTWKalender gibts es als Serverseitige Anwendungen zwei Services die Daten bereitstellen und verarbeiten. Dabei handelt es sich um einen Datenservice für die Raumbelegung und Veranstaltungen der HTWK Leipzig und einen Service nur für die Generierung und Bereitstellung von der iCal-Feeds. Im Rahmen des PWA Projekts steht aber das Frontend im Vordergrund, welches die Daten dieser Services verarbeitet und anzeigt. Dabei wurden bestehende Komponenten aus dem HTWKalender Projekt übernommen und um PWA spezifische Features erweitert. Einen genauen Überblick über die Projektstruktur und die Änderungen sind über den HTWKalender-PWA Fork im DIT-Gitlab einsehbar.

Lokale Entwicklung

Für die lokale Entwicklung der HTWKalender-PWA nutzen wir den Vite-Server. Dieser kann mit dem Befehl `npm run dev` gestartet werden. Aber um Service Worker und PWA Features zu testen, muss die App über HTTPS aufgerufen werden. Zudem werden Daten aus den anderen Services benötigt die auch gestartet werden müssen. Um diese Anforderungen zu erfüllen, haben wir ein Docker-Compose File erstellt, welches die benötigten Services startet und die HTWKalender-PWA über HTTPS bereitstellt. Dazu muss das Docker-Compose File mit dem Befehl `docker-compose up` gestartet werden.

Service Worker

Der Service Worker ist ein zentrales Element einer PWA. Er ermöglicht es, die Anwendung offline verfügbar. Der Service Worker wird beim Start der Anwendung installiert und aktiviert. Er lädt die

benötigten Dateien in den Cache und stellt sie offline zur Verfügung. Der Service Worker wird im `public/sw.js` definiert.

```
import { precacheAndRoute, cleanupOutdatedCaches } from 'workbox-precaching';
import { clientsClaim } from 'workbox-core';

self.skipWaiting();
clientsClaim();

cleanupOutdatedCaches();

// Precache the files from the manifest
precacheAndRoute(self.__WB_MANIFEST);

// Custom precaching logic for the /api/modules endpoint
self.addEventListener('install', (event) => {
  event.waitUntil(
    caches.open('api-modules-cache').then((cache) => {
      return fetch('/api/modules')
        .then((response) => {
          if (response.ok) {
            return cache.put('/api/modules', response);
          }
          throw new Error('Failed to fetch /api/modules');
        })
        .catch((error) => {
          console.error('Precaching /api/modules failed:', error);
        });
    })
  );
});
```

Deployment

Für das Deployment der HTWKalender-PWA haben wir uns für die Nutzung von Gitlab CI/CD entschieden. Dabei wird bei jedem Push in das Gitlab-Repository ein Build- und Deploy-Job ausgeführt. Der Build-Job erstellt ein Docker-Image, welches die HTWKalender-PWA enthält. Der Deploy-Job startet dann ein Docker-Container mit dem erstellten Image. Der Container wird auf einem Server gehostet, der über eine Subdomain erreichbar ist. Die Subdomain wird über einen Reverse-Proxy auf den Docker-Container weitergeleitet. Eine Laufende Instanz der HTWKalender-PWA ist unter link: pwa.htwkalender.de erreichbar. Die Konfiguration für das Deployment ist im Gitlab-Repository unter .gitlab-ci.yml zu finden.

Zukünftige Features

Wie bereits zuvor angesprochen, könnte die Anwendung u. a. um einen Push-Benachrichtigungsdienst erweitert werden. Auch Einbinden von Mensaplänen und eine erweiterte

Suche nach freien Räumen wären denkbar. Hinzu kommen alle Features, die zum HTWKalender zählen, u. a. ist hier bereits eine Schnittstelle zum anderen PWA Projekt "HTWKarte" umgesetzt worden.