ShiftPLAN

Plan your HR smarter

Betreuer
Dr. Fridtjof Toenniessen

Team
Andreas Gantner [39724]
Anne Naumann [38991]
Maximilian Tellmann [39627]
Sascha Wirtz [39740]

Inhaltsverzeichnis

Einleitung

Mit Excel Listen rumschlagen, kostenlose Software, die nicht vernetzt werden kann, nutzen, E-Mails versenden oder kostspielige Lösungen kaufen. Alles Probleme, die gerade in kleineren Unternehmen bei der Einsatzplanung auf einen zukommen. Diese Beobachtung durften wir auch im privaten Umfeld machen. Lösungen fürs kostengünstige selbst Hosten sind selten, wenn nicht sogar gar nicht vorhanden. Meist bittet man den Kunden zur Kasse und greift tief in sein Portemonnaie.

Motiviert durch die Fehlschläge bei der Suche nach einer Lösung, begannen wir mit dem Vorhaben eine OpenSource Lösung für diese Problemstellung zu entwickeln. Eine Software, mit der Aufgaben definiert werden können, diesen Aufgaben Personalvoraussetzungen zu verschiedenen Uhrzeiten und Tagen zugewiesen werden können, Mitarbeitern die Schichten zugeteilt werden können und das Ganze mit einem durchdachten Berechtigungs- und API-System. Außerdem wollten wir noch kritische Daten verschlüsseln, damit man nicht von kostspieligen Zertifikaten abhängig ist und kostenlose Shared Hoster ohne https-Verbindung nutzen kann, ohne beispielsweise Angst haben zu müssen, dass Passwörter in die falschen Hände gelangen.

Kommerzialisieren könnte man dieses Projekt indem man gerade kleinere Unternehmen, ohne explizite Administratoren, beim Hosting und bei der Wartung der Infrastruktur unterstützt. Auch individuelle Entwicklungen wären eine Möglichkeit aus dem Projekt einen kommerziellen Erfolg zu machen.

Technisch orientierte Anleitung

Voraussetzungen

Folgende Voraussetzungen werden für die Installation und Konfiguration von ShiftPLAN benötigt. Stellen Sie sicher, dass die Software installiert und konfiguriert ist.

Apache2 Webserver (aktuelle Version)
PHP Interpreter für Apache2 (PHP 7.4)
MariaDB bzw MySQL (aktuelle Version)
FileZilla (aktuelle Version)

Die Installation des LAMP-Stacks (Apache2, PHP Interpreter und MySQL) betrachten wir in dieser Anleitung nicht. Wir zeigen die Installation mit einem gängigen Shared Hosting Provider auf, da dies mit Sicherheit der Weg der Wahl für die meisten Nutzer unserer Zielgruppe ist. Allerdings gehen wir auch hierbei nicht auf die Registrierung und die genaue Einrichtung ein, da sich diese Punkte von Shared Hoster zu Shared Hoster unterscheiden. Beim Webserver ist zu beachten, dass die Nutzung von .htaccess Dateien erlaubt ist. Dies kann man in der Webserver Konfiguration einstellen. Bei Shared Hosting Anbieter ist diese Einstellung in der Regel gesetzt und muss nicht extra aktiviert werden. Jedoch ist es sinnvoll dies im Vorfeld zu klären.

Wir haben während der Implementationsphase eine MySQL Datenbank genutzt und die Funktionsweise mit dieser getestet. Allerdings gehen wir von voller Funktionalität auch mit einer MariaDB Datenbank aus. Unsere Testumgebung auf bplaced.net nutzt voraussichtlich den OpenSource Fork MariaDB und bisher sind uns keine Probleme mit der ShiftPLAN Anwendung bekannt.

Auch die Installation von FileZilla betrachten wir nicht, da dies in der Regel eine einfache Standard Installationsroutine unter dem jeweiligen Betriebssystem ist. Natürlich ist die Nutzung jedes alternativen FTP-Clients ebenfalls möglich. Je nach Shared Hosting Anbieter könnte der Upload auf den Webserver ggf. auch über eine Weboberfläche oder andere Protokolle ablaufen. Dies sollte normalerweise keine Probleme mit der Übertragung unserer Software verursachen. Da unsere API durch die genutzten Frameworks viele kleine Dateien enthält, kann der Upload jedoch etwas verlangsamt werden.

Aufbau

ShiftPLAN besteht aus einer API (geschrieben in PHP) und einer WebApp (geschrieben in HTML, CSS, JS mithilfe von Angular). Die WebApp kommuniziert hierbei dauerhaft mit der API und stellt die Daten, die sie erhält, grafisch dar. Die API wiederum ist zuständig für das Persistieren der Daten und das Lesen bereits persistenter Daten. Sie wird mit REST Abfragen abgefragt und Daten werden im JSON Format ausgetauscht.

Sowohl die WebApp als auch die API können und sollten im besten Fall auf demselben Webserver laufen. Dies macht einen entscheidenden Sicherheitsfaktor aus, da dann die API und die WebApp der Same Origin Policy folgen. Allerdings kann in der API Konfigurationsdatei die Same Origin Policy auch abgeschaltet werden und die WebApp auf einem anderen Webserver gehostet werden. Dies empfehlen wir jedoch nicht, da dies die Anwendung angreifbarer macht. In dieser Dokumentation gehen wir auf die Konfiguration mit der Same Origin Policy ein.

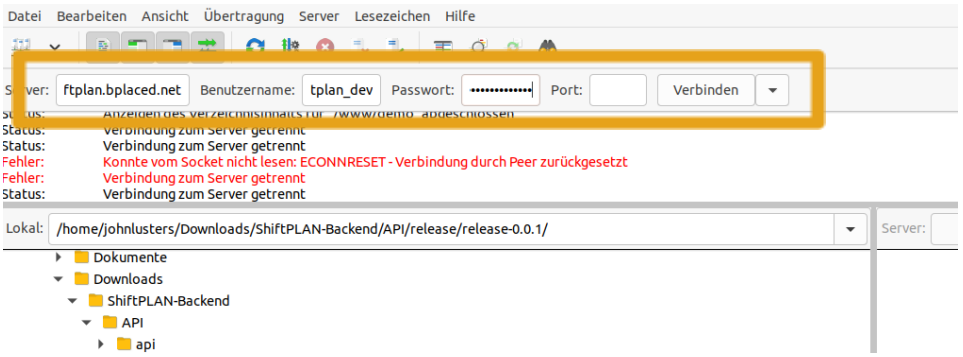
Schritt für Schritt Anleitung

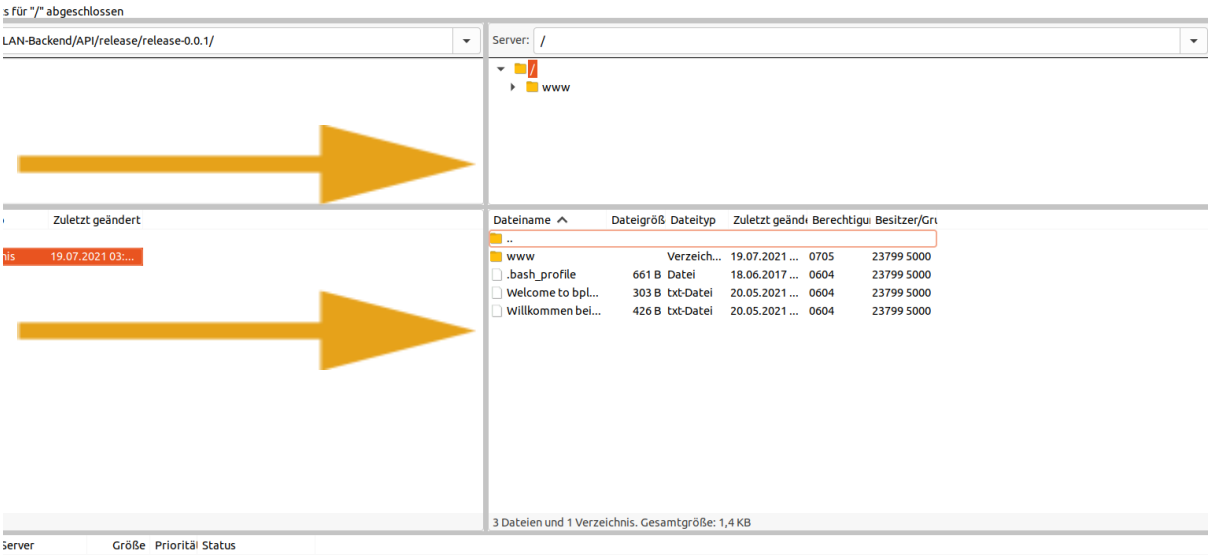
Die Einrichtung unserer ShiftPLAN Anwendung ist sehr intuitiv und erfordert nur geringe IT-Kenntnisse. In der Regel ist die Einrichtung unserer Software schon in wenigen Minuten erledigt und kann gleich genutzt werden.

Zu Beginn sucht man sich auf https://github.com/andyruto/ShiftPLAN/releases den aktuellen Release heraus und lädt die backend-<version>.zip sowie die frontend-<version>.zip herunter. Diese beiden entpackt man nun mit einem Archivmanager seiner Wahl. Auf allen gängigen Betriebssystemen kann eine Zip Datei eigentlich ohne weiterer Software entpackt werden. Falls es doch zu Problemen kommen sollte kann man für diese Aufgabe eine Software wie z.B. 7-Zip nutzen. Diese Software steht kostenlos zur Verfügung. Nach dem Entpacken sollte bei der backend-<version>.zip ein api Ordner zu finden sein. Bei der frontend-<version>.zip sollte man mehrere html und js Dateien sowie einen assets Ordner finden.

Diese Ordner und Dateien werden nun mithilfe von FileZilla in das Root Verzeichnis des Webservers hochgeladen. Wie bereits erwähnt kann sich je nach Hosting Anbieter das Vorgehen leicht unterscheiden. Gleich bleibt jedoch die Tatsache, dass wir alle Dateien in das Root Verzeichnis des Webservers kopieren müssen. Ob dies nun per FTP mit FileZilla oder einer anderen Software oder ggf. sogar einem anderen Protokoll geschieht ist hierbei irrelevant. Da Shared Hosting Anbieter häufig FTP für die Übertragung von Dateien an den Server nutzen und wir dies bei unserem Hosting Anbieter ebenfalls so angeboten bekommen haben, zeigen wir Ihnen diesen Weg auf.

Zu Beginn müssen die FTP Zugangsdaten ermittelt werden. Bei vielen Shared Hosting Anbietern kann man mehrere FTP User einrichten. Bei anderen ist ein FTP Benutzer vorhanden und man kann keinen weiteren FTP Benutzer einrichten. Informieren Sie sich am besten bei dem Hosting Anbieter ihrer Wahl, wie Sie die FTP Nutzerdaten erhalten. Außerdem muss ermittelt werden, unter welcher Adresse sie ihren FTP Server erreichen können. Dies finden Sie ebenfalls bei Ihrem Shared Hosting Anbieter heraus. Daraufhin öffnen Sie FileZilla und geben die ermittelten Daten (Adresse des Servers, Benutzername, Passwort, ggf. Port wenn nicht Standard) in das entsprechende Formular in FileZilla ein. Daraufhin sollte nach einem Klick auf Verbinden die Verbindung aufgebaut werden. Im rechten Bereich des Fensters sollte nun die Ordnerstruktur des Webservers sichtbar sein.

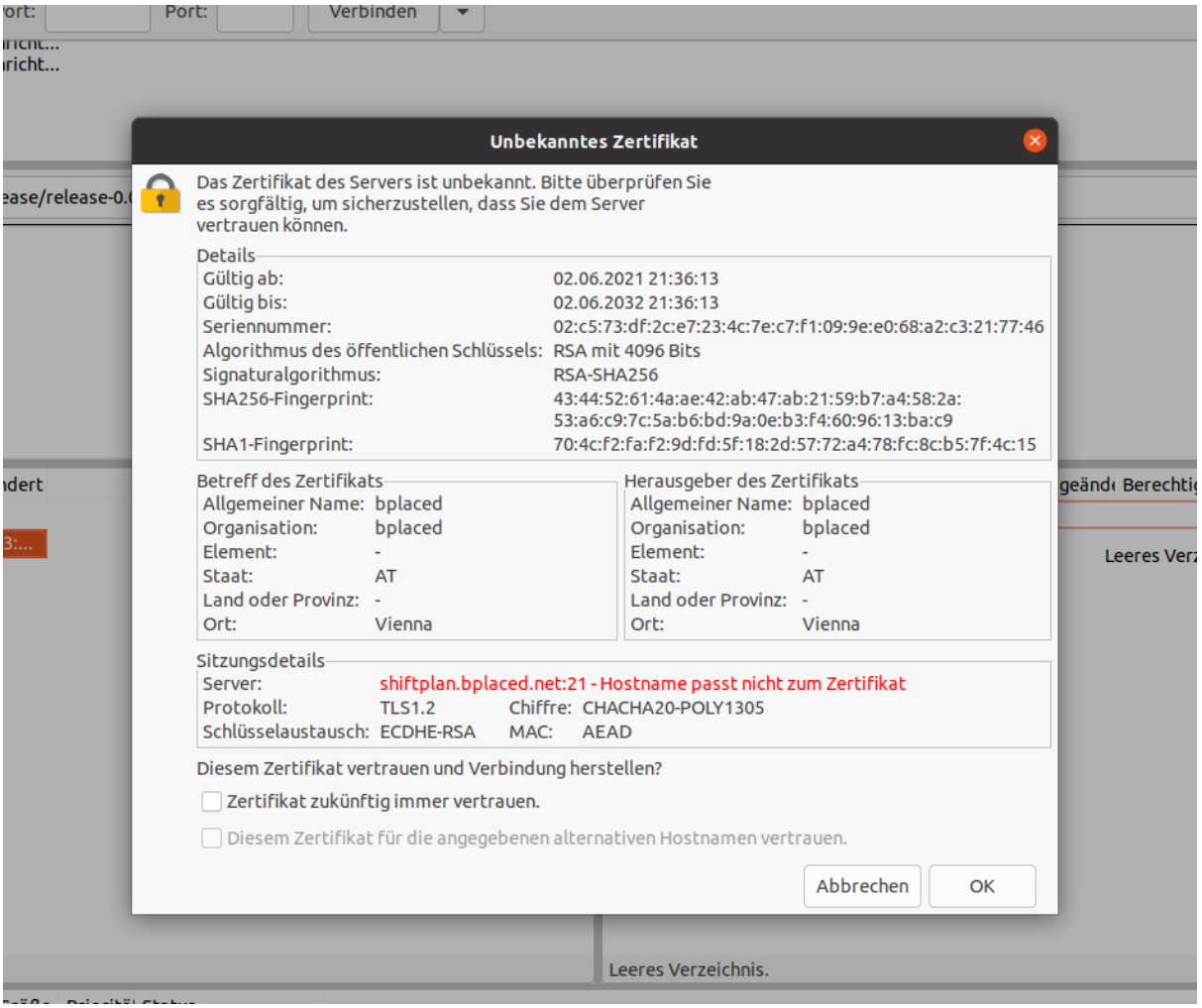




Je nach Shared Hosting Anbieter befinden Sie sich direkt in Ihrem Root Verzeichnis oder wie in den Abbildungen eine Ebene höher. In diesem Beispiel muss das Verzeichnis www geöffnet werden. Dieses ist das Root Verzeichnis des Webservers. Die Information, wo sich der Root Ordner Ihres Webservers befindet, erhalten Sie ebenfalls bei Ihrem Shared Hosting Anbieter.

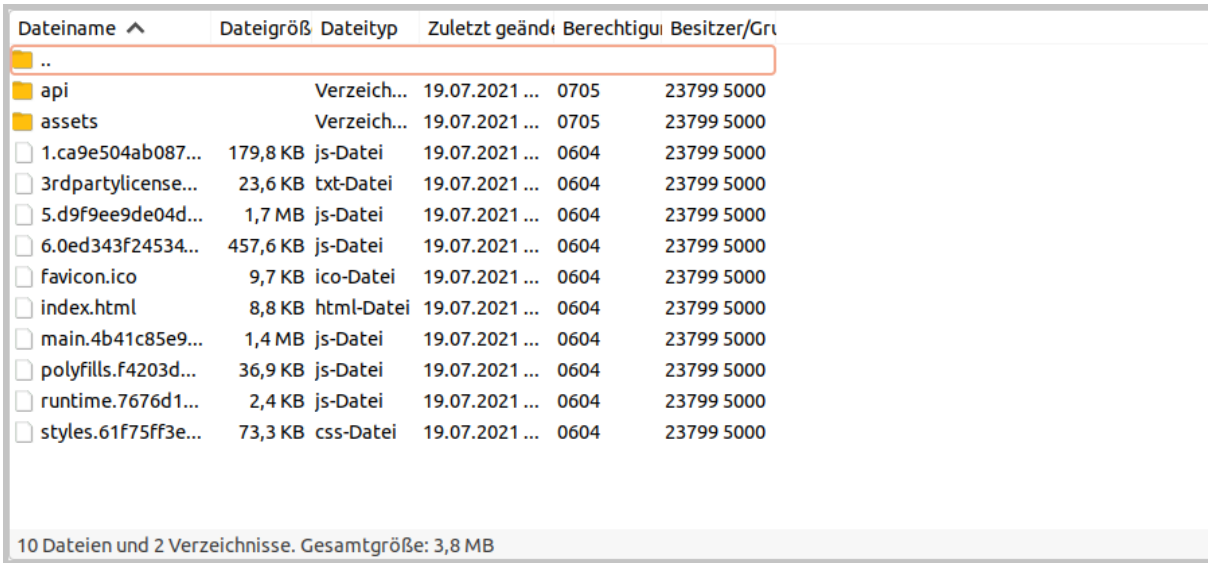
Auf der linken Seite des Fensters findet man die lokale Ordnerstruktur des PCs, an dem man sich im Moment befindet. Hier navigiert man zu den entpackten Dateien aus dem ersten Schritt. Diese markieren wir und laden sie auf den Webserver hoch indem wir die markierten Dateien in den rechten Bereich des Fensters verschieben.

Sollten während des Vorgangs Meldungen zu Zertifikaten kommen, dann können diese einfach ignoriert werden. Man kann dem Zertifikat in der Regel vertrauen und ggf. falls gewünscht auch diese Auswahl in FileZilla vermerken, damit die Meldungen nicht nochmals angezeigt werden für dieses Zertifikat.



Nach erfolgreicher Übertragung muss man nun die Konfigurationsdatei des Backends anlegen. Hierzu dupliziert man die sample_settings.conf Datei im lokalen Verzeichnis und benennt die neue Datei in settings.conf um. Anschließend konfiguriert man die settings.conf Datei und lädt sie über FileZilla auf den Server hoch. Daraufhin muss sie im Hauptverzeichnis der API liegen. Im Kapitel Konfigurationsdatei gehen wir näher auf die settings.conf Datei ein und erklären die einzelnen Konfigurationsvariablen. Zusätzlich sind auch in der settings.conf Kommentare, die die einzelnen Konfigurationsvariablen erklären.

Die fertige Anwendungsstruktur sieht dann wie folgt aus. Wir haben im Root Verzeichnis des Webservers einen Ordner api und einen Ordner assets sowie diverse html und js Dateien. Im api Ordner haben wir wiederum diverse Ordner und ganz wichtig eine settings.conf, die wir an unsere Bedürfnisse angepasst haben. Alles weitere an Konfiguration wird beim ersten Start der Anwendung durchgeführt.



Dateiname	Dateigröß	Dateityp	Zuletzt geänd	Berechtigu	Besitzer/Gru
..					
api		Verzeich...	19.07.2021 ...	0705	23799 5000
assets		Verzeich...	19.07.2021 ...	0705	23799 5000
1.ca9e504ab087...	179,8 KB	js-Datei	19.07.2021 ...	0604	23799 5000
3rdpartylicense...	23,6 KB	txt-Datei	19.07.2021 ...	0604	23799 5000
5.d9f9ee9de04d...	1,7 MB	js-Datei	19.07.2021 ...	0604	23799 5000
6.0ed343f24534...	457,6 KB	js-Datei	19.07.2021 ...	0604	23799 5000
favicon.ico	9,7 KB	ico-Datei	19.07.2021 ...	0604	23799 5000
index.html	8,8 KB	html-Datei	19.07.2021 ...	0604	23799 5000
main.4b41c85e9...	1,4 MB	js-Datei	19.07.2021 ...	0604	23799 5000
polyfills.f4203d...	36,9 KB	js-Datei	19.07.2021 ...	0604	23799 5000
runtime.7676d1...	2,4 KB	js-Datei	19.07.2021 ...	0604	23799 5000
styles.61f75ff3e...	73,3 KB	css-Datei	19.07.2021 ...	0604	23799 5000

10 Dateien und 2 Verzeichnisse. Gesamtgröße: 3,8 MB

Dateiname	Dateigröß	Dateityp	Zuletzt geänd	Berechtigu	Besitzer/Gru
key		Verzeich...	19.07.2021 ...	0705	23799 5000
login		Verzeich...	19.07.2021 ...	0705	23799 5000
logout		Verzeich...	19.07.2021 ...	0705	23799 5000
logs		Verzeich...	19.07.2021 ...	0755	23799 5000
shifts		Verzeich...	19.07.2021 ...	0705	23799 5000
src		Verzeich...	19.07.2021 ...	0705	23799 5000
tasks		Verzeich...	19.07.2021 ...	0705	23799 5000
timespans		Verzeich...	19.07.2021 ...	0705	23799 5000
users		Verzeich...	19.07.2021 ...	0705	23799 5000
vendor		Verzeich...	19.07.2021 ...	0705	23799 5000
.htaccess	487 B	Datei	19.07.2021 ...	0604	23799 5000
index.php	3,9 KB	php-Datei	19.07.2021 ...	0604	23799 5000
prepareExec.php	5,1 KB	php-Datei	19.07.2021 ...	0604	23799 5000
sample_setting...	993 B	conf-Datei	19.07.2021 ...	0604	23799 5000
settings.conf	1,1 KB	conf-Datei	19.07.2021 ...	0604	23799 5000

5 Dateien und 12 Verzeichnisse. Gesamtgröße: 11,4 KB

Konfigurationsdatei

Die Konfigurationsdatei ist das Herzstück des Backends, mit ihrer Hilfe kann man das Verhalten der API an seine Bedürfnisse anpassen. Die Datei ist nach INI Strukturregeln aufgebaut und kann mit einem beliebigen Texteditor angepasst werden. Die sample_settings.conf bietet eine Beispielkonfiguration, anhand der man seine persönliche Konfiguration erstellen kann. Im Folgenden gehen wir auf die einzelnen Variablen der Konfigurationsdatei ein und erklären kurz, was man mit diesen anpassen kann.

Logging	
path	Der Pfad in dem die Log Dateien erstellt werden. Diese Variable ist optional und hat den Standardpfad /logs/. Dabei beginnt / im Root Verzeichnis der API und nicht im Root des Webservers. Log Dateien werden nach außen vom Webserver über .htaccess Dateien versteckt und sind nur über FTP einzusehen. Die Ordnerstruktur wird hierbei von der API automatisiert erstellt und muss nicht manuell angelegt werden.
logLevel	Gibt an welche Arten von Log Einträgen geschrieben werden sollen. Mögliche Werte sind DEBUG, INFO, WARNING, ERROR, CRITICAL. Dabei werden immer Einträge von der Relevanz, die angegeben wurde, und relevantere Einträge in das Log geschrieben. Beispielsweise werden bei Angabe von WARNING alle WARNING, ERROR und CRITICAL Log Einträge in die Logs geschrieben. Alle anderen werden nicht geschrieben. Der Standardwert ist DEBUG und diese Variable ist optional.
logCount	Diese Variable gibt an, wie viele Log Dateien geschrieben werden, bevor die älteste Datei gelöscht wird um eine neue Log Datei zu erstellen. Pro REST Abfrage wird eine Log Datei angelegt. Diese erhält einen eindeutigen Namen durch einen genauen Zeitstempel.

Database	
dbHost	Hier gibt man den Datenbankserver an, der für die Persistierung der Daten genutzt werden soll. Dieser Wert ist nicht optional und muss zwingend angegeben werden. Wie man an diese Information kommt wird später erläutert. Man kann den dbHost als FQDN oder als IP-Adresse angeben.
dbPort	Diese Variable wird genutzt, um eine Anpassung des Datenbank Ports vorzunehmen. Dies kann der Fall sein, wenn man einen Datenbankserver selbst eingerichtet und konfiguriert hat. In den allermeisten Fällen wird der Port jedoch beim Standard belassen. In dem Fall trägt man einfach die Zahl 0 als Wert dieser Variable ein. Auch diese Variable muss zwingend angegeben werden.
dbName	An dieser Stelle gibt man den Namen der zu nutzenden Datenbank auf dem Datenbankserver an. Eine Datenbank muss bereits bestehen, damit die Anwendung ihr ORM-Modell auf die Datenbank abbilden kann. Auch diese Konfiguration ist Pflicht.
dbUser	Bei dieser Variable gibt man den Usernamen an, der für den Login am Datenbankserver genutzt werden soll. Der Benutzer muss mindestens Vollzugriff auf die Datenbank haben, die für diese Applikation vorgesehen ist. Es wird außerdem empfohlen dem Benutzer nicht unnötig mehr Rechte zu vergeben, um Sicherheitsrisiken vorzubeugen. Auch diese Angabe ist zwingend nötig.
dbPassword	Hierbei handelt es sich um das Passwort, welches beim Login mit dem dbUser genutzt wird. Dieses Passwort muss einige Richtlinien erfüllen, die in der settings.conf näher erläutert werden. Diese sollte man beim Konfigurieren des dbUser unbedingt im Hinterkopf behalten. Auch diese Information ist zwingend erforderlich.

Webserver	
sameOrigin	Hiermit kann man die Same Origin Policy der API aktivieren (true) oder deaktivieren (false). Der Standard ist true und es wird auch empfohlen dies so zu belassen, da das Deaktivieren Sicherheitsrisiken zur Folge hat. Sollte man jedoch WebApp und Backend unbedingt auf unterschiedlichen Server laufen lassen wollen und ist sich der Gefahr bewusst, dann kann man mit dieser Variable die Same Origin Policy abschalten. Dieser Wert ist optional.

Datenbank

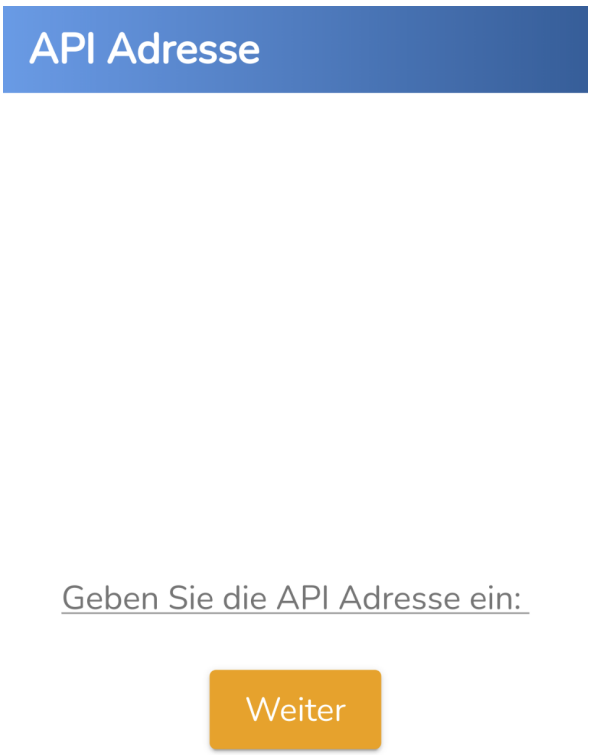
Die Datenbank ist der zentrale Speicher für alle anwendungsbezogenen Daten. Nur das Logging wird direkt auf dem Webserver in Dateien betrieben. Alles Andere findet in der Datenbank statt. Von einer genauen Installations- und Konfigurationsanleitung sehen wir auch hier ab. Bei den Shared Hosting Anbieter unterscheidet sich das Vorgehen immer wieder. Was jedoch immer ähnlich ist, ist dass man erstmal auswählen kann, welche Datenbanksoftware man nutzen will. Hier ist es wichtig MySQL oder MariaDB zu wählen. Andere DBMS unterstützen wir im Moment noch nicht. Da Doctrine jedoch eine sehr intuitive und schnelle Implementierung von weiteren DBMS ermöglicht, ist nicht auszuschließen, dass wir weitere DBMS in Zukunft unterstützen werden.

In diesem Schritt wird auch der Name der Datenbank angegeben und ein Benutzer für diese Datenbank erstellt. Diese Daten werden in der zuvor genannten Konfigurationsdatei benötigt um die API mit den nötigen Informationen zu versorgen, die sie braucht um sich mit der Datenbank verbinden zu können. Auch hier ist es wieder hilfreich sich auf den Hilfeseiten des jeweiligen Shared Hosting Anbieter umzusehen oder ggf. den Support zu kontaktieren.

Einrichtungsassistent

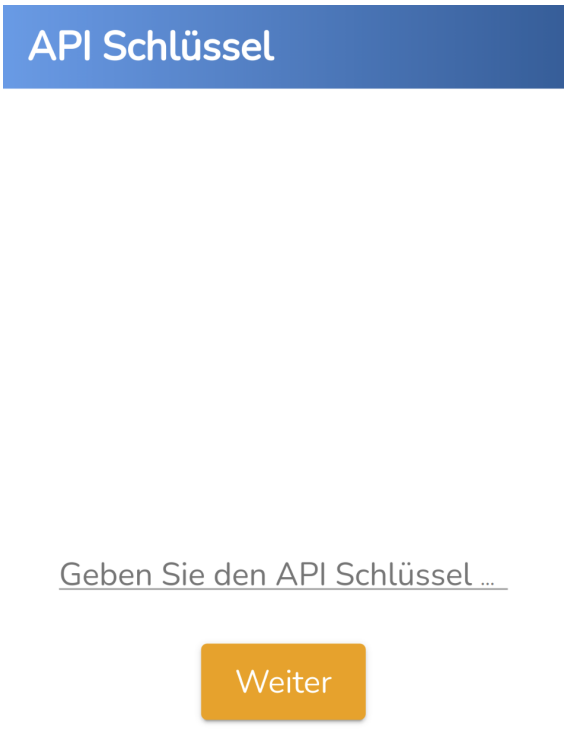
Nachdem nun alle Voraussetzungen erfüllt sind, die Applikation auf dem Server vorliegt und die Konfigurationsdatei angepasst wurde, ist es nun an der Zeit die WebApp das erste Mal zu starten. Dabei richtet sich die API automatisch ein. Sie bereitet die Datenbank vor und legt alle Standarddaten in der Datenbank ab. An dieser Stelle ist die Applikation jedoch noch nicht bereit um zu laufen, denn das Standardpasswort des Standard Admin muss noch geändert werden. Erst dann ist die Applikation von der Backend Seite her fertig und kann genutzt werden. Um diese und auch weitere Konfigurationen einfach zu gestalten, läuft beim Start der Applikation ein Wizard ab, der einen durch alle wichtigen Schritte führt, um die WebApp und ggf. auch das Backend ordnungsgemäß zu konfigurieren.

Der erste Punkt des Einrichtungsassistenten ist der API Address Screen. Hier wird die Standardadresse (Same Origin im Ordner api/) kontaktiert und geguckt, ob die API antwortet. Sollte dies nicht der Fall sein muss die Adresse zur API manuell angegeben werden.



Hierbei wird nun überprüft ob ein API Key als Antwort von der API zurück kommt. Dies ist immer dann der Fall, wenn das Backend noch nicht vollständig konfiguriert wurde, ergo der Admin Benutzer noch sein Standardpasswort hat und dieses noch nicht geändert wurde. Sollte der API Key jedoch nicht zurück kommen, dann muss dieser manuell angegeben werden. Er wird benötigt um zu authentifizieren, dass die Anfragen der WebApp berechtigt ist.

Daraufhin kommt im Falle der Ersteinrichtung der Passwort ändern Bildschirm. Hier wird das Standard Passwort des Admin Benutzers geändert. Dies natürlich aus Sicherheitsgründen.

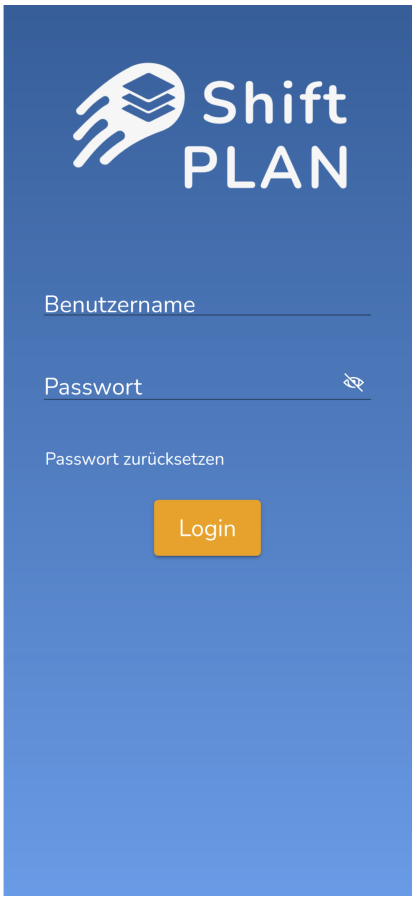


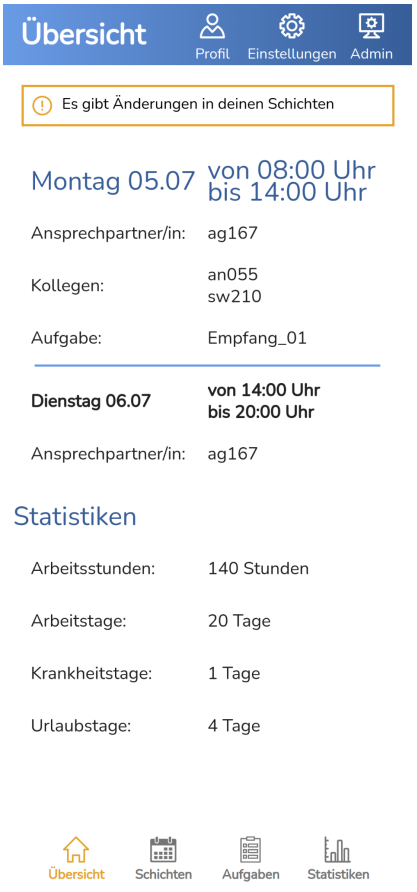
Im nächsten Schritt wird auch schon der Login angezeigt. Im Falle der Ersteinrichtung wird dieser Punkt direkt übersprungen. Man wird direkt eingeloggt und kann die WebApp nutzen.

Die WebApp hat eine relativ flache Navigationsstruktur. Eine Toolbar und eine Bottom Navigation bilden dabei das Grundgerüst. Von hier aus findet man alle wichtigen Funktionen der WebApp. Je nach Nutzertyp hat man hierbei mehr oder weniger Funktionen zur Auswahl.

Auf den Screenshots sieht man

- den Passwort Screen, für das Ändern des Standard Passworts
- den Login Screen, für das Einloggen mit dem Benutzernamen und dem Passwort
- die Übersicht, den ersten Screen der App, mit Demodaten





Infrastruktur

Die Infrastruktur unserer Anwendung ist relativ einfach gehalten. Auch um die Komplexität für den Benutzer zu reduzieren. ShiftPLAN hat nämlich seine einfache und kostengünstige Selbstkonfiguration als Alleinstellungsmerkmal. Aus diesem Grund ist die Anwendung auch einfach als Zip Dateien für Backend und Frontend vorhanden und muss nur auf dem Webserver abgelegt werden. Bei der Beschreibung der Infrastruktur teilen wir aus diesem Grund auch nach Backend und Frontend auf.

Die Kommunikation der beiden Teile der Software findet über http/https statt und nutzt POST Anfragen. Der Request-Body sowie der Response-Body folgen dem JSON-Format. Dies ermöglicht uns eine einfache Nutzung der übertragenen Daten sowohl im Frontend als auch im Backend. PHP sowie TypeScript haben einfache Werkzeuge um in JSON formatierte Daten zu parsen und zu nutzen.

Das Verschlüsseln und Entschlüsseln einzelner Daten zwischen dem Frontend und dem Backend wird mit der Sodium Bibliothek gelöst. Dies ist eine Bibliothek, die gängige Verschlüsselungsmethoden einfach und intuitiv für unterschiedlichste Plattformen und Sprachen implementiert. Wenn man nicht gerade Experte im Bereich Verschlüsselung ist, dann kann man bei diesem Thema mehr falsch machen als man richtig macht. Aus diesem Grund ist es sinnvoll bestehende Funktionen der Programmiersprachen zu nutzen. Wenn man jedoch im Frontend und Backend unterschiedliche Programmiersprachen verwendet, kann das durchaus ein bisschen schwieriger sein. An der Stelle setzt Sodium ein. Seine große und aktive Community aus vielen Security Experten und Entwickler implementiert und wartet eine große Menge an Projekten, die alle die Sodium Verschlüsselungsmethoden für unterschiedlichste Sprachen und Plattformen bereitstellen.

Wir nutzen sowohl beim asynchronen als auch beim synchronen Ansatz die Verschlüsselungs und Entschlüsselungs Black Box der Sodium Bibliothek. Diese zeichnet sich dadurch aus, dass sie immer die gängigen und sicheren aber auch gleichzeitig performantesten Verschlüsselungsalgorithmen und Ansätze nutzt. Des Weiteren benötigt man bei diesen wenig bis gar keine Security Vorkenntnisse um sie sicher einzusetzen. Sollte es Sicherheitsbedenken beim aktuellen Black Box Verfahren geben, wird dieser wenn möglich durch neue und sicherere Verschlüsselungsmethoden ausgetauscht, ohne die Nutzung zu ändern.

Wenn es um das Thema Persistieren des Passworts geht, nutzen wir SHA256 für das Hashen des Passworts. Dieses Verfahren gilt laut BSI immer noch als sicher und bietet den Vorteil, dass es noch Kompatibilität zu älteren Systemen und Sprachen besitzt. Diesen Hash übertragen wir aber nur in seltenen Fällen und dann nicht im Klartext. Beim Login stellen wir dem Frontend eine Herausforderung die das Frontend mit dem Passwort Hash symmetrisch verschlüsselt zurück schicken muss. Beim Ändern oder Setzen des Passworts läuft es ähnlich. Sollte ein Admin das Passwort ändern, ohne Kenntnis des ursprünglichen Passworts, wird es asymmetrisch verschlüsselt an das Backend übertragen. Den öffentlichen Schlüssel kann man mit einem separaten API Request anfragen. Er ändert sich automatisch beim Aufruf des Root API Calls falls das alte Schlüsselpaar zu alt sein sollte. Sollte der Benutzer sein eigenes Passwort ändern wird es wieder symmetrisch mit dem alten Passwort verschlüsselt übertragen. Dies sichert den Hash nochmal gegen Sniffing Attacken ab. Angreifer bräuchte also einen Datenbankzugang um die Passwort Hashes herauszufinden.

Backend

Das Backend wurde in PHP 7.4 geschrieben. Es ist einzig und allein für die Backend Logik zuständig und komplett passiv tätig. Alle Skripte laufen erst zum Zeitpunkt des Abrufes des jeweiligen API-Skriptes ab. Jedes API-Skript ist unter einer bestimmten Adresse auf dem entsprechenden Server erreichbar. Folgende Liste zeigt die aktuell verfügbaren API Calls, die man nutzen kann.

http://shiftplan.com/api/	
Request-Body	Response-Body

-	<pre>{
	  "errorCode": 0,
	  "apiKey": "v7RueWjEbDcZxLbkqCna"
	}</pre>

http://shiftplan.com/api/key/check/	
Request-Body	Response-Body

<pre>{
  "apiKey": "v7RueWjEbDcZxLbkqCna"
}</pre>
	<pre>{
	  "errorCode": 0
	}</pre>

http://shiftplan.com/api/key/publickey/	
Request-Body	Response-Body

{
"apiKey": "v7RueWjEbDcZxLbkqCna"
}
	{
	"errorCode": 0,
	"publicKey": "0e3a00c8c438ddf221fba36ab27cec075b8e2112d10e5c8fee6cf3b9ecf30d54"
	}

http://shiftplan.com/api/login/	
Version 1	
Request-Body	Response-Body

{
"apiKey": "v7RueWjEbDcZxLbkqCna",
"userName": "admin"
}
	{
	"errorCode": 0,
	"chlg": "16268126570668292002WSwVBUfncKIA2VDuuJR"
	}

Version 2	
Request-Body	Response-Body

{
"apiKey": "v7RueWjEbDcZxLbkqCna",
"userName": "admin",
"chlgSolved": "7036ea82b2e5f4bfde7d43ee935ee909576ef33ed1c7b8e9d25c9e563182586db0150bd9ba3d0e0ffe63f789d8b3111468a4f8a945c694",
"nonce": "06b95d0cf3bb8b801b3c8c14de01f61911607b7798e91ad8"
}
	{
	"errorCode": "0",
	"session": "7fbed2cfd34f5dea9fa8a619d95cce0bdc3cb423c694d9475e4d37b660437d0b4169df53",
	"nonce": "2b57c0c7911f6aa7efc1d62de05078c50be4452ff09b6286"
	}

Version 3	
Request-Body	Response-Body

{
"apiKey": "v7RueWjEbDcZxLbkqCna",
"session":
"78f60f28746252b930f14179499d4755bf687a242
d03330da1f554980e360374a9a1831c349158a023
3a48c0de59009268533c79688307f98ee8116fbfe
9521ef34d4fd9"
}
	{
	"errorCode": 0,
	"userType": 2
	}

http://shiftplan.com/api/logout/	
Request-Body	Response-Body

{
"apiKey": "v7RueWjEbDcZxLbkqCna",
"session":
"78f60f28746252b930f14179499d4755bf687a242
d03330da1f554980e360374a9a1831c349158a023
3a48c0de59009268533c79688307f98ee8116fbfe
9521ef34d4fd9"
}
	{
	"errorCode": 0
	}

http://shiftplan.com/api/users/get/

Request-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7"
}
```


Response-Body


```
{
  "errorCode": 0,
  "profile": [
    {
      "id": 1,
      "type": 2,
      "name": "admin",
      "hidden": false,
      "overtime": 0,
      "weeklyWorkingMinutes": 0,
      "weeklyWorkingDays": 0,
      "yearVacationDays": 0
    }
  ]
}
```


http://shiftplan.com/api/users/add/

Request-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7",
  "name": "FooBa",
  "pwHash":
"93a9ea2c55c991a982530d114fb2b1cda57aceaf6
5adf432064371d3205f1608a2189f30e885342d97
9cfdee04a6f0531b9845ce1173e8c700e5e0a0c3f
09f7311d194619c7662e5749de4d49f802b0470f1
b3c8cd3e47b0461fdda56b690f0c7d2f645ca7fc6
6e444e9298d07341712",
  "hidden": "false"
}
```


Response-Body


```
{
  "errorCode": 0
}
```

http://shiftplan.com/api/users/modify/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7",
  "user": "2",
  "type": "1",
  "name": "FooBa",
  "password":
"93a9ea2c55c991a982530d114fb2b1cda57aceaf6
5adf432064371d3205f1608a2189f30e885342d97
9cfdee04a6f0531b9845ce1173e8c700e5e0a0c3f
09f7311d194619c7662e5749de4d49f802b0470f1
b3c8cd3e47b0461fdda56b690f0c7d2f645ca7fc6
6e444e9298d07341712",
  "hidden": "true",
  "overtime": "123",
  "weeklyWorkingMinutes": "123",
  "weeklyWorkingDays": "123",
  "yearVacationDays": "123"
}
```



```
{
  "errorCode": 0
}
```


http://shiftplan.com/api/users/remove/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7",
  "name": "qwe"
}
```



```
{
  "errorCode": 0
}
```

http://shiftplan.com/api/tasks/create/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7",
  "name": "Kundengespraech"
}
```

```
{
  "errorCode": 0
}
```


http://shiftplan.com/api/tasks/get/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7"
}
```

```
{
  "errorCode": 0,
  "tasks": [
    {
      "id": 1,
      "name": "123123",
      "reccuring": false
    },
    {
      "id": 2,
      "name": "Kundengespraech",
      "reccuring": false
    }
  ]
}
```

http://shiftplan.com/api/tasks/modify/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7",
  "id": "2",
  "name": "Kundentreffen"
}
```

```
{
  "errorCode": 0
}
```


http://shiftplan.com/api/tasks/remove/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7",
  "id": "2"
}
```

```
{
  "errorCode": 0
}
```

http://shiftplan.com/api/shifts/create/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7",
  "assignedUser": 2,
  "supervisorUser": 2,
  "connectedTaskId": 2,
  "shiftStart": "2021-10-12 13:13:13",
  "shiftEnd": "2021-10-12 13:13:14",
  "comment": "Überprüfung fällig"
}
```

```
{
  "errorCode": 0
}
```


http://shiftplan.com/api/shifts/get/	
Request-Body	Response-Body


```
{
  "apiKey": "v7RueWjEbDcZxLbkqCna",
  "session":
"30aea74557b987acfdf9fb491c6397135b6d6e481
8b5928632cad44b83af031c254bf5db2eb0d7ef6
35c0e7f87c3e6645951ae7875444c62fdeffdc98c
927981465f1ec7"
}
```

```
{
  "errorCode": 0,
  "shifts": [
    {
      "id": 1,
      "assignedUser": 2,
      "supervisorUser": 2,
      "task": 2,
      "shiftStart": {
        "date": "2021-10-12 13:13:13.000000",
        "timezone_type": 3,
        "timezone": "Europe/Berlin"
      },
      "shiftEnd": {
        "date": "2021-10-12 13:13:14.000000",
        "timezone_type": 3,
        "timezone": "Europe/Berlin"
      },
      "comment": "Überprüfung fällig",
```

```
        "lastModifiedBy": 1,
        "lastModified": {
          "date": "2021-07-20 23:16:00.000000",
          "timezone_type": 3,
          "timezone": "Europe/Berlin"
        }
      },
      {
        "id": 2,
        "assignedUser": 2,
        "supervisorUser": 2,
        "task": 2,
        "shiftStart": {
          "date": "2021-10-12 13:13:13.000000",
          "timezone_type": 3,
          "timezone": "Europe/Berlin"
        },
        "shiftEnd": {
          "date": "2021-10-12 13:13:14.000000",
          "timezone_type": 3,
          "timezone": "Europe/Berlin"
        },
        "comment": "Überprüfung fällig",
        "lastModifiedBy": 1,
        "lastModified": {
          "date": "2021-07-20 23:17:13.000000",
          "timezone_type": 3,
          "timezone": "Europe/Berlin"
        }
      }
    ]
}
```

http://shiftplan.com/api/shifts/modify/	
Request-Body	Response-Body

{
"apiKey": "v7RueWjEbDcZxLbkqCna",
"session":
"30aea74557b987acfdf9fb491c6397135b6d6e4818b5928632cad44b83af031c254bf5db2eb0d7ef635c0e7f87c3e6645951ae7875444c62fdeffdc98c927981465f1ec7",
"id": 1,
"supervisorUser": 2
}
	{
	"errorCode": 0
	}

http://shiftplan.com/api/shifts/remove/	
Request-Body	Response-Body

{
"apiKey": "v7RueWjEbDcZxLbkqCna",
"session":
"30aea74557b987acfdf9fb491c6397135b6d6e4818b5928632cad44b83af031c254bf5db2eb0d7ef635c0e7f87c3e6645951ae7875444c62fdeffdc98c927981465f1ec7",
"id": 1
}
	{
	"errorCode": 0
	}

Hierbei ist es sehr wichtig, dass alle Requests an die URL mit einem abschließenden / geschickt werden. Apache leitet nämlich sonst die Anfrage an den Server um, so dass der Request-Body verloren geht. Das führt zu einem Error bei der Ausführung des jeweiligen API Skriptes.

Für das Logging haben wir im Backend das Logging Framework Monolog genutzt. Dies ermöglicht das einfache Logging in Text Dateien. Wir haben es so implementiert, dass man ein Logger Objekt für das Logging benutzt. Dieses ist nach dem Singleton Pattern aufgebaut. Für den Fall, dass die Konfiguration fehlerhaft ist und somit die Logging Settings nicht geladen werden konnten, wir ein Fatal Logger ausgegeben, der in eine error.log Datei rein schreibt. Außerdem haben wir für das Logging eine Log-Rotation eingebaut. Wir legen pro API Anfrage eine Log Datei an. Das führt zu einer großen Menge an Log Dateien. Dem wollen wir durch diese Maßnahme entgegenwirken. Ein Logeintrag besitzt immer einen Zeitstempel, eine Wichtigkeit, eine Besitzer Klasse und eine

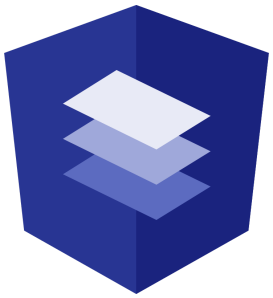
Nachricht. Alles bis auf die Wichtigkeit und die Nachricht wird automatisch ermittelt. Der Zeitstempel über ein Datetime Objekt und die Emitter Klasse durch Reflections. Dadurch sind Log Befehle ein einfacher Einzeller.

Die Datenbank manuell zu erstellen und zu warten ist immer eine sehr aufwendige Sache. Es verschlingt viel Zeit und erhöht den Aufwand beim Programmieren. Um diesen Aufwand zu reduzieren haben wir uns entschieden, dass wir das ORM-Framework Doctrine nutzen. Mithilfe dessen können wir für jede Datenbank Tabelle ein PHP Objekt anlegen und mit diesen Arbeiten. Die Verbindung zu der Datenbank und die Erstellung der SQL Querys übernimmt das Framework. Das hat uns viel Arbeit erspart. Damit wir später auch Änderungen an der Datenbank vornehmen können, ohne komplizierte Bearbeitungen der Datenbank durchführen zu müssen, haben wir eine Migration Logik implementiert, die automatisch sowohl Up-Migrations (bei Updates) und Down-Migrations (bei Downgrades) durchführt. Beim Ausführen des Root Skriptes wird dies automatisch erledigt. Dadurch stellen wir sicher, dass die Applikation immer funktionsfähig bleibt. Im Umkehrschluss ist es dadurch aber umso wichtiger, dass man vor einem Update immer erst eine Sicherung der Datenbank vornimmt.

Frontend

Das Frontend haben wir nach langem Überlegen als WebApp umgesetzt. Dadurch erreichen wir mehr Plattformen mit weniger Entwicklungsaufwand. Wir haben gerade im privaten Beispielsfall gemerkt, dass eine mobile Anwendung am besten wäre. Daher haben wir uns für den Mobile First Ansatz entschieden und zuerst eine Applikation entwickelt, die insbesondere für Handys optimiert ist. Hierbei sprangen uns sofort Angular und Angular Material ins Auge. Zum einen wegen der Fülle an Bibliotheken aber auch aus Gründen der Gruppenkonstellation.

Angular bietet einem die Möglichkeit einfach und schnell Single Page Applikationen für das Web zu entwickeln. Gerade der komponentenbasierte Aufbau macht es einem einfach sehr Modular und organisiert an einer komplexen oder auch simplen WebApp zu arbeiten. Die Lifecycle Hooks mit dem automatisierten DOM Tree Check machen es dank Property Binding und Event Binding sehr einfach den DOM immer auf dem aktuellen und gleichen Stand zu halten. Dabei muss man nur in der Typescript Klasse der jeweiligen Component eine Property anlegen, dass man dann mit einem beliebigen DOM Element verknüpft.



Im mobilen Bereich ist immer noch das Material Design Pattern sehr aktuell. Weil wir eine mobile first Anwendung anstreben haben wir uns entschieden, Angular Material, mit seinen vielen ästhetischen und gängigen Material UI-Elementen, zu verwenden. Damit sieht eine WebApp einer native App zum Verwechseln ähnlich. Zusätzlich ist das Implementieren der UI auch sehr zeitsparend und übersichtlich möglich.

Zu guter letzt hatten wir neben mehreren kleineren Komponenten, wie zum Beispiel der Bottom Navigation, noch ein großes Framework im Einsatz. Das Swiper Framework. Dieses Framework bietet eine einfache Möglichkeit wischbare Pager zu kreieren, die man von vielen Android Apps kennt und liebt. Die Implementation ist sehr einfach und bietet auch die Möglichkeit über Buttons durch die Seiten zu wischen. Ein weiterer Pluspunkt sind die automatischen Animationen, die sowohl bei der Wischgeste als auch bei Nutzung der Buttons abgespielt werden.

Lessons Learned

Dieses Projekt hat uns die große und wichtige Möglichkeit geboten die Arbeit in einem größeren Team mit mehreren Abhängigkeiten kennenzulernen. Gerade die Tatsache, dass wir ein Management benötigt haben zeigt, dass es kein triviales Projekt gewesen ist. Und gerade dies hat einen enormen Lerneffekt zur Folge gehabt.

Im Management kamen die ersten harten Lektionen, die wir lernen mussten. Aufgrund der begrenzten Zeit durch andere Vorlesungen und Nebenjobs, war eigentlich nicht wirklich viel Zeit für eine sehr feingranulare Planung da. Diese hätten wir aber definitiv benötigt, da die Komplexität des Projektes gerade die Grenze überschritten hat, die man mit einer lockeren Planung noch stemmen kann. Gerade in der Design Evaluation mit einem Testbenutzer haben wir rückblickend zu viel Zeit investiert, die uns im späteren Verlauf bei der Implementation gefehlt hat.

Des Weiteren haben wir einzelne Spezialistenprodukte in der Planung nicht so genau beschrieben wie es nötig gewesen wäre. Hier zum einen wegen der beschränkten Zeit aber auch wegen der Tatsache, dass wir ein wenig Kreativität zulassen wollten. Dies war an einigen Stellen eine gute Entscheidung, da dadurch interessante und kreative Lösungsansätze zustande gekommen sind. An den allermeisten Stellen führte es aufgrund Missverständnisse in den Meetings oder fehlender Personen in Meetings jedoch eher zu Problemen bei der Verknüpfung von Backend und Frontend. Am Ende musste im Backend doch oft noch nachgebessert werden.

Was wir insgesamt sehr hilfreich und zielführend fanden war, dass A. Gantner in seiner Tätigkeit als Manager und Springer schon während des Designprozesses die Angular Infrastruktur erkundet und Test Implementationen erstellt hat. Dies hat beim Start der Implementation für das Frontend viel Zeit gespart, da das Frontend Team nach einem kurzen Briefing gleich einsteigen konnte. Zudem konnte A. Gantner jederzeit bei Bugfixes unterstützend mitwirken, da er die Architektur von Angular auch kannte und sich so relativ schnell in Problematiken rein denken konnte.

Bei der Implementation haben wir erkannt, dass man das Frontend in zwei Phasen aufteilen konnte. Eine Designphase, die sich mit dem Layout und dem Style der Screens befasst, und eine Phase zur Verknüpfung mit dem Backend, die dann die Funktionalität bringt. In der Zeitplanung haben wir diese Möglichkeit nicht bedacht und immer basierend auf Funktionen geplant. Das hat zu Beginn einiges an Zeit gekostet, die uns dann hinten raus gefehlt hat um alle Funktionen zu implementieren, die wir uns eigentlich vorgenommen haben. Aus diesem Grund stehen die meisten Funktionen im Moment leider nur mit Demodaten und noch nicht mit einer realistischen Anbindung an die API zur Verfügung.

Auch in der API sind aufgrund dessen einige geplante Funktionen wie z.B. die Krankheitsbescheinigungen und die Urlaubsanträge noch nicht implementiert.

Gerade in der Implementierung ist uns oft aufgefallen, dass eine Aufteilung großer und komplexer Aufgaben in kleinere Aufgabenpakete sinnvoll gewesen wäre. Dies hätte man entweder in der Planung bei der Beschreibung eines Spezialistenproduktes machen können oder jeder Entwickler für sich. Gerade um den Überblick zu behalten, wie man seine Zeit managen kann, wäre ein solcher Ansatz rückblickend sinnvoll gewesen.

Auch die komplexe Umsetzung des Login mithilfe der Sodium Bibliothek war ein Punkt, den wir unterschätzt haben. Hier wäre definitiv mehr Zeit in der Planung notwendig gewesen. Gerade die Tests, die wir zwischen dem Backend und dem Frontend durchführen mussten um sicherzustellen, dass auch wirklich alles funktioniert, waren trotz sehr intuitiver Umsetzung von Sodium anspruchsvoll und haben viel Zeit beansprucht.

Uns ist auch aufgefallen, dass insbesondere das Testen der Software viele Komplikationen beinhalten kann. Man kann nie wirklich davon ausgehen, dass die Software mit allen Versionen einer Abhängigkeit funktionieren. Das Testen der Software unter verschiedenen Versionen und Infrastrukturen ist also unerlässlich.

Ebenfalls hat uns das Updaten von Abhängigkeiten, gerade in Angular aber auch in anderen Bereichen, vor Schwierigkeiten gestellt. Wir haben herausgefunden, dass solche Updates explizit geplant werden müssen, weil viele Faktoren zu beachten sind. Es muss zum Beispiel überprüft werden ob der Code noch in der neuen Version funktionieren würde, oder ob die Abhängigkeiten sich nicht gegenseitig nach einem Update in die Quere kommen. Es ist also mitnichten trivial ein Update auf eine höhere Framework oder Dependency Version durchzuführen und sollte definitiv gut geplant werden.

Auch wenn uns das Projekt vor viele Probleme und Hürden gestellt hat sind wir froh es gemacht zu haben. Wir hatten dadurch die Gelegenheit Einblick in viele verschiedene Aspekte der Projektarbeit zu bekommen, was uns auch im Ausblick auf das Praxissemester viel geholfen hat. Wir hatten die Möglichkeit unser vorhandenes Wissen zu erweitern und konnten es durch die praktische Umsetzung auch sofort einordnen und verknüpfen.