VERTRETUNGSPLAN TOUCHSCREEN
„LEHRER ZUM ANFASSEN“

EIN PROJEKT VON

THORBEN AUER
(SCHNITTSTELLE, DATENBANK)
UND
DOMINIK ZIEGENHAGEL
(SERVERSEITIGES, NETZWERKSTRUKTUR)

Inhaltsverzeichnis

Kurzfassung

VPlan Touch – Lehrer zum Anfassen

VPlan Touch vollendet die Vorstellung von Vertretungsplänen in der Zukunft: Welche Klasse hat in welchem Raum Unterricht? Welcher Lehrer ist anwesend? Welcher Raum ist frei? – Mit zwei, drei Fingertipps auf dem Touchscreen verfügt der Benutzer über jegliche Information in Bezug auf Unterricht und kann problemlos zwischen Lehrern, Klassen und Räumen hin- und her navigieren – optional ist die Navigation und Authentifizierung auch durch QR-Codes möglich.

Schulen, die mit dem Schulverwaltungsprogramm Untis arbeiten, können alle benötigten Programme in nur einigen Minuten installieren – auch per Fernzugriff. Jedoch ist die Schnittstelle nicht an Untis gebunden. Das sichere Betriebssystem Linux Mint macht das Administrieren der Netzwerke durch Passwortabfrage extrem einfach, und ist dennoch kaum zu knacken. Durch eine „gehärtete“ Version von Mozilla Firefox und wartungsfreie Skripte ist das System zusätzlich vor Schülerdummheiten, aber auch Netzwerkzwischenfällen geschützt. Durch die simple Lizenz „Public Domain“ ist die Software komplett kostenfrei und kann – ohne namentliche Nennung - benutzt und verändert werden.

Das System läuft an den Gewerblichen und Hauswirtschaftlich-Sozialpflegerischen Schulen Emmendingen (GHSE) bereits seit einem Jahr und ersetzt auf diese Art und Weise sechs komplizierte, alte und wartungsaufwendige Einheiten durch ein neues wartungsfreies System.

Das Problem

Seit einiger Zeit arbeitet unsere Schule mit dem Stundenplansystem der Firma Untis. Zur Anzeige des Stundenplans verwendeten wir bisher Windows XP - Computer mit Röhrenbildschirmen, an die nur eine Maus angeschlossen war.

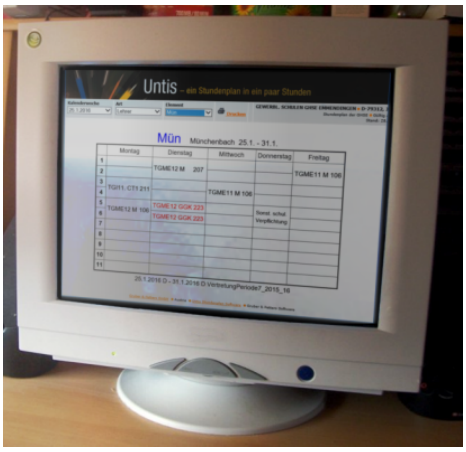


Abbildung 1: Alter Stundenplan

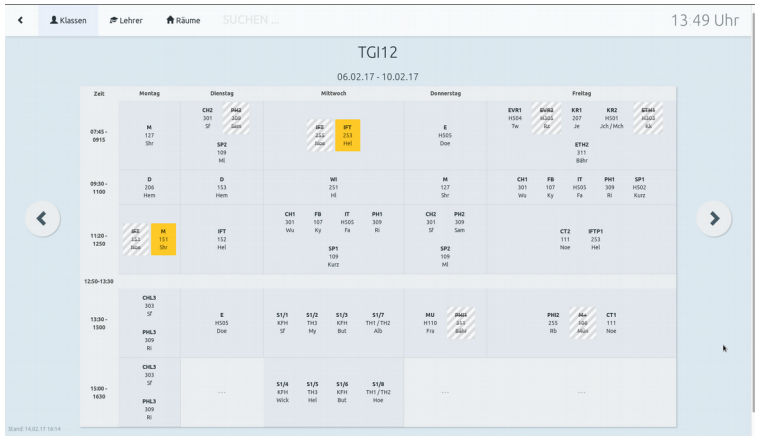


Abbildung 2: Das neue Programm (Mit Vertretungen und Ausfällen)

Diese PCs bezogen die kompletten Stundenpläne von statisch generierten HTML-Dateien, die im Intranet lagen. Um die Vertretungen anzuzeigen, gab es weitere Bildschirme, die, zumindest zu 80% der Zeit, weitere HTML-Dateien und die Vertretung in HTML-Tabellen anzeigten (bis zu 20 Tabellen, teilweise ineinander verschachtelt). Der Export zu diesem, vom Gremium zur Standardisierung der Techniken im World Wide Web (W3C) vor Jahren als veraltet beschriebenen Format dauerte oft Stunden. Zudem fielen die Bildschirme manchmal aus oder zeigten nur den Desktop an. Auch wurde der Browser für die Stundenplan-Anzeigen von Schülern ab und zu geschlossen oder Einstellungen geändert. Zusätzlich gab es einen Bildschirm vor dem Lehrerzimmer, der nicht im Intranet war. Zu diesem musste ein Lehrer hinlaufen und manuell bei jeder Änderung die Daten von einem USB–Stick laden. Diesen Rechner konnte jeder nach Belieben herunterfahren.

Der Weg zum Touchscreen

Während einer CT-Unterrichtsstunde kam unser Lehrer Carsten Münchenbach auf uns zu und gab uns folgende Aufgabe: „Erstellen Sie eine grafische HTML5-Oberfläche für den aktuellen Stundenplan“. Anfangs nahmen wir nur die besagten HTML-Tabellen, das statisch erzeugte Array mit allen Lehrer- und Klassennamen in PHP (einen Hypertext Preprocessor) einfügt und dann ein ähnliches User-Interface wie zuvor benutzt, um die HTML-Dateien einzubinden. Diese Version stammte übrigens noch immer aus dem Unterricht. Diese ließen wir dann am darauf folgenden Wochenende auf einem Raspberry PI in einem Fullscreen Iceweasel anzeigen. Wir installierten und setzen erst alle Programme auf und überspielten dann per

SSH (Secure Shell) die Daten auf den Raspberry. Nachdem das nach vielen Problemen endlich einigermaßen funktionierte, erfuhren wir, dass die Bildschirme mitsamt Raspberrys an einer Zeitschaltuhr hängen, die diesen einfach den Strom wegnehmen, um sie quasi in Intervallen zu resetten. Das hieß, der Browser müsste sich direkt nach einem automatischen Login im Vollbildmodus öffnen. Der automatische Login war schnell eingerichtet, Iceweasel mit dem System starten zu lassen funktionierte auch. Doch nachdem in einer Testphase der Browser in Tagesabständen abzustürzen begann, mussten wir etwas Neues überlegen. Wir schrieben ein Bash-Script, das in einer Endlosschleife einen einzigen Thread mit dem Browser hostete. Dieses Script musste nur gestartet werden und zwar unabhängig vom User.
Wir probierten am Ende den GUI–Launchers zu starten, was darin endete, dass Iceweasel manchmal gestartet wurde, bevor das GUI geladen werden konnte.
Wir probierten zahlreiche andere Scripts aus, bis uns die Crontab ins Auge fiel. Wir ließen das Script per @restart starten. Danach wollten wir den Touchscreen anschließen; reibungslos und ohne Treiberinstallation funktionierte der Touch, jedoch versetzt um ca. 50 dp, also ca. 1 cm. Das Problem war eine Konfigurationsdatei im Bootloader, die aus unerklärlichen Gründen direkt nach dem BIOS den Touchscreen kleiner werden ließ. Wir schrieben die Konfigurationsdatei um, so dass dieses Problem gelöst war und nachdem das Debian noch einen kleinen Grub–Eingriff und eine Kernel-Kompilierung wegen geändertem Bootlogo hinter sich gebracht hatte, durften wir vor dem Lehrerzimmer tatsächlich die erste Version des Touchscreens aufstellen. Diese stand dort aber nicht lange, da sie durch gelangweilte „Tipper“ immer weiter nach hinten geschoben wurde, denn der Raspian war einfach viel zu langsam, die 250MB Ram reichten nicht im geringsten. Auch eine 500-MB-Variante war nicht im Stande, mehrere Instanzen des Browsers innerhalb eines vertretbaren Zeitraums zu öffnen. Nach einigen Tagen bekamen wir einen alten PC zur Verfügung gestellt. Wir richteten auch dort wieder ein Betriebssystem ein. Viele Probleme des Raspberry PI tauchten nicht mehr auf.

Während dieser Zeit hatten wir damit angefangen, eine Schnittstelle zu Untis zu programmieren, die die Daten in eine lokale Datenbank parsen sollte. Diese Schnittstelle lud bei Aufruf die Daten einfach herunter und direkt in die Datenbank, die dann während des Imports für 15 Minuten nicht verwendbar war. Da die Schnittstelle alle zwei Stunden in das Anzeige-Script eingebunden wurde, kam es während dieser Zeit immer zu einem kompletten 15-minütigen Ausfall. Auch wenn wir diese Ausfallzeiten immer auf die Unterrichtszeiten gelegt hatten, war es keine Dauerlösung. Leider scheiterte der Versuch, nach einem Update über TeamViewer auf den Touchscreen zuzugreifen. Als wir am nächsten Morgen nach dem Problem suchten, fanden wir einen MemTest vor. Etwas muss beim Update schiefgelaufen sein. Wir installierten daher ein neues Betriebssystem auf eine separate Festplatte und setzten alles erneut auf.

Kurz darauf begannen wir, die beiden Programme mehr und mehr zu separieren; mittlerweile schrieb das Update-Programm in eine separate Datenbank und wurde über einen systemeigenen Cronjob per Kommandozeile aufgerufen. Nach dem Import wurden die Datenbanken umbenannt. So gab es eine fast lückenlose Verfügbarkeit der Daten (< 1 s pro

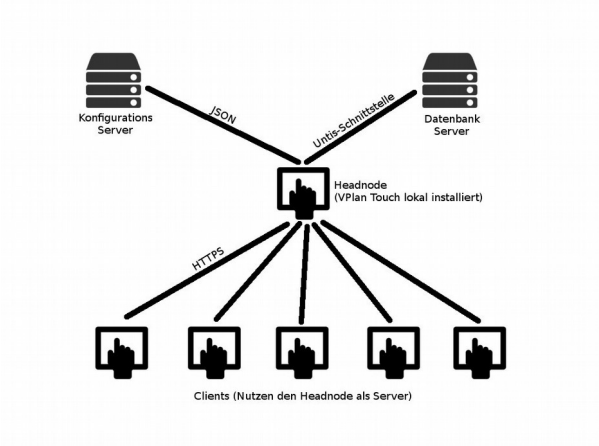
Import). Mittlerweile wurden im Lehrerzimmer die Tische ausgetauscht und auch unsere Internetverbindung gekappt, was prompt dazu führte, dass leere Daten in die Datenbank geschrieben wurden: Wir behoben dies temporär durch manuelles Befüllen der Datenbank und ließen das Update-Script auf eine Internetverbindung überprüfen.

Mittlerweile schrieben wir an einem Installer, der sich nicht nur um sämtliche anfänglichen Einrichtungen des Betriebssystems kümmerte, sondern darüber hinaus noch Konfigurationsdaten schrieb, um so das System auch anwendungsfreundlich für andere Schulen zu machen. Dazu kam ein Update-Script, das die Programme und Pakete auf den Touchscreens installiert und aktuell hält. Aufgrund der weiteren Separierung von Schnittstelle und Anzeige bekam die Schnittstelle ihren eigenes Updateprogramm.

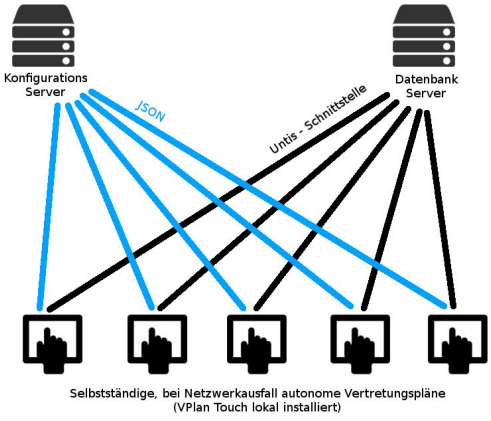
Die Netzwerkschicht

Von Anfang an stand der Plan, mehrere unserer Touchscreen-Rechner miteinander zu vernetzen. Unsere Idee war es, eine Art „Headnode" in ein Netz mit einem späteren Stundenplan-Server zu bringen, momentan noch Untis-Server, von dem wir per Schnittstelle die Daten beziehen und in eine lokale Datenbank speichern. Von dieser Headnode aus sollen später die Touchscreens die Daten beziehen. Wenn möglich sollte kein lokaler Apache installiert sein und der Touchscreen somit als reiner Client fungieren. Dafür planten wir die Konfigurationsdateien auf einen Konfigurationsserver zu legen, da von dort aus eine gemeinsame Konfiguration möglich ist.
Zudem konnten wir dann per SSH Cronjobs auf die einzelnen Rechner legen und gemeinsam von einem Rechner aus steuern. Doch da in letzter Zeit das Intranet der Schule öfters zusammenbrach, hatten wir das System so konfiguriert, dass nur ein Rechner notwendig ist. Dazu legten wir den Konfigurationsordner jeweils als Kopie auf den lokalen Server.

Möglichkeit 1:
(weniger Netzwerkbelastung)



Möglichkeit 2:
(komplett gegen Netzwerkausfall gesichert)



Schnittstelle zu Untis

Geplant war, die Stunden-/Vertretungspläne über die Untis-Schnittstelle auf eine lokale Datenbank, die auf einem Schulserver sein wird, zu speichern. Carsten Noeske, unser jetziger IFT-/CT-Lehrer, hatte sich zur Datenbank-Struktur bereits Gedanken gemacht und gab uns seine bisherige Arbeit für eine MySQL-Datenbank und die Untis-API (Schnittstelle von Untis). Außerdem gab er uns noch zwei PHP-Dateien, mit denen man Arrays per POST-Request an Untis senden kann.
Zuerst informierten wir uns über objektorientierte Programmierung in PHP. Für Testzwecke empfahl uns Herr Noeske, den Demo-Server von Untis zu nutzen, da bei zu vielen falschen Anfragen der Benutzer für die GHSE-Pläne gesperrt werden würde.
Wir sendeten die POST-Request für alle Klassen, alle Lehre und Räume an Untis, zurück bekamen wir JSON-String, ein Format zum Austausch von Daten. Nach Prüfung der Daten schrieben wir diese in die MySQL-Datenbank mit eigener Strukturerstellung. Das war sehr praktisch, da wir so auch mehr über MySQL-Datenbank Programmierung lernen konnte. Nachdem wir uns überlegt hatten, wie viele Tabellen benötigt werden und wie groß der Speicher pro Spalte sein musste, begannen wir, die Daten in die Tabellen zu parsen. Für die Kategorien Lehrer, Klassen und Räume war das auch kein Problem.
Danach begannen wir mit den Stunden- und Vertretungsplänen. Allerdings musste Dominik sich in den ersten sechs Monaten wegen des großen Datenvolumens zuerst mit den Stundenplänen zufriedengeben - Vertretungen und Ausfälle konnten nicht berücksichtigt werden. Schon jetzt war die Request-Zeit - die Zeit, die die Anfrage benötigte - gigantisch. Es dauerte rund fünf Minuten, bis alle Daten gefetcht und diese dann in die Tabellen geparst waren. Auf dem Touch-Monitor sind es auch heute, aufgrund der relativ langsamen Internetverbindung, noch immer gut sieben Minuten Request-Zeit. Die Dauer der Request-Zeit war deshalb so lange, da wir die Stundenpläne für jede Klasse für das ganze Schuljahr bezogen.
Ebenfalls wollten wir die Klausuren von jeder Klasse fetchen, bekamen jedoch eine Fehlermeldung von Untis: „No rights“.

Untis verweigerte uns also den Zugriff auf diese Klausur-Daten. Wir probierten eine andere Klasse aus, doch es funktionierte immer noch nicht. Herr Noeske meinte, dass Untis den Zugriff nicht freigeben wolle, da ihnen die Nutzung dieser Informationen von Schülerseite nicht klar sei.
Wir widmeten uns wieder den Vertretungsplänen. Allerdings fanden wir in der API zuerst keinerlei Informationen darüber, wie wir an diese Daten kommen konnten. Dann jedoch stellten wir fest, dass in dem Fetch der Stundenpläne die Vertretungen schon vorhanden waren und wir lediglich diese Daten nicht in die Datenbank bzw. in die Tabellen eingetragen hatten.
Nun richteten wir den ersten Touch-Monitor ein. Die ersten Feedbacks von Lehrern und Schülern ließen nicht lange auf sich warten. Alle waren begeistert von der Idee und meinten, dass es viel praktischer sei als die Standcomputer mit statischen Stundenplänen. Allerdings traten noch einige Fehler auf: Es fehlten Räume und Klassen. Unser Script gab jedoch keinerlei Fehler beim Parsen der Daten an. Also überprüften wir die Fetchs und fanden die

fehlenden Klassen- und Raumdaten. Daraus schlossen wir, dass der Fehler in der Datenbank lag. Wir haben anschließend nach einer Funktion gesucht, die uns MySQL Datenbankfehler ausgeben würde. Nach und nach überprüften wir unser Script mit dieser Funktion und stellten fest, dass wir für die fehlenden Daten den Spalten zu wenig Speicher zugewiesen hatten. Die Erklärung hierfür war wohl, dass in der DEMO API die Datensätze nicht so viel Speicher benötigten wie die original GHSE-Stundenplänen, Klassen- und Lehrerlisten inklusive des Raumplans. Also haben wir die Datenbank-Struktur so bearbeitet, dass die Speichergröße mit den Datensätzen kompatibel war.

Nun wurden alle Daten angezeigt. Unsere Aufgabe war zum größten Teil erledigt. Nach den Sommerferien gab es allerdings ein neues Problem. Die Daten erschienen nicht. Nach kurzer Suche fanden wir den Fehler: Die Rechnung, die das Schuljahr berechnete, ergab aufgrund einer falschen Definition des Post-Request kein Ergebnis. Schnell war der Fehler behoben und es funktioniert alles wieder einwandfrei.

Allerdings war der Vorgang eines Updates noch kompliziert: Das Update der Untis-Schnittstelle musste per Email an den Server-Administrator gesendet werden und dieser musste es dann auf den Server hochladen. Anschließend musste das Update auf dem Touchscreen aufgespielt werden. Deshalb bekam die Untis-Schnittstelle ihren eigenen Update-Manager. Dieser lädt nun die aktuelle Version der Schnittstelle vollautomatisch herunter und installiert sie zeitnah.
Aus Datenschutzgründen durften auf dem privaten Server keinerlei unverschlüsselte Zugangsdaten zu Untis vorhanden sein, so dass die Daten in eine andere Datei outgesourced werden mussten. Aufgrund dieser Auslagerung veränderten sich die Daten bei einem Update nicht, d.h. sie waren nicht in den Update-Vorgang involviert und mussten nicht auf den Server hochgeladen werden.
Damit nicht jedes Mal die Zugangsdaten entfernt werden mussten, programmierten wir einen Versions-Manager in Java, der die ausgelagerten Daten löschte, alles in ein Archiv packte und dieses dann unter einer neuen Versionsnummer hochlud.
Nun musste lediglich bei der Installation eine Version von der Untis-Schnittstelle mit Update-Manager installiert werden.
Somit waren wir nun in der Lage, ein Update komplett eigenständig hochzuladen und zu installieren. Das Ganze war dank des Versions-Managers innerhalb von Minuten erledigt ohne Datenschutzverletzung.

Neuerungen im Laufe der Zeit

Immer wieder bekamen wir weitere Ideen, die wir umsetzen wollten. So haben wir zum Beispiel später eine Funktion erweitert, die die Lehrerübersicht erstellt. Dort werden mittlerweile anwesende Lehrer grün hinterlegt angezeigt, genauso wie freie Räume hervorgehoben werden. Auf der Suche nach einer schnelleren Performance haben wir nach und nach unsere Programme auf Schnelligkeit optimiert und konnten die Seitenladezeiten von 40 Sekunden auf weniger als zwei Sekunden runterbrechen. Nun versuchten wir die Anzahl der benötigten Klicks, um zur gewünschten Information zu gelangen, zu minimieren. Nachdem wir automatische Suchvorschläge hinzugefügt hatten, brachten wir zusätzlich einen QR-Code-Scanner an dem Vertretungsplan an. Unter diesem Scanner konnten generierte QR-Codes gehalten werden, um Klassen, Räume, Lehrer oder andere Navigationsziele ohne einen einzigen Klick direkt anzeigen zu lassen. Zudem können sich Administratoren durch in der Konfiguration definierte QR-Codes authentifizieren und auf das GUI-Konfigurationspanel zugreifen. Dieses erspart einen SSH-Zugriff bei minimalen Änderungen, wie z.B. Code-Resetting oder Pin-Einstellungen.

Abbildung 3: Übersicht der anwesenden Klassen (grüne Klassen sind anwesend)

Abbildung 4: Generierung eines benutzerdefinierten Codes

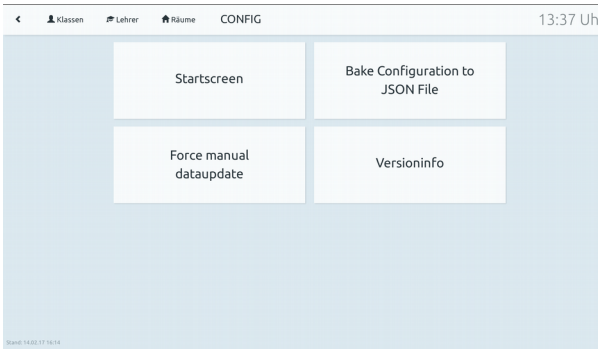


Abbildung 5: Einfache Konfiguration, um eine SSH-Installation zu umgehen

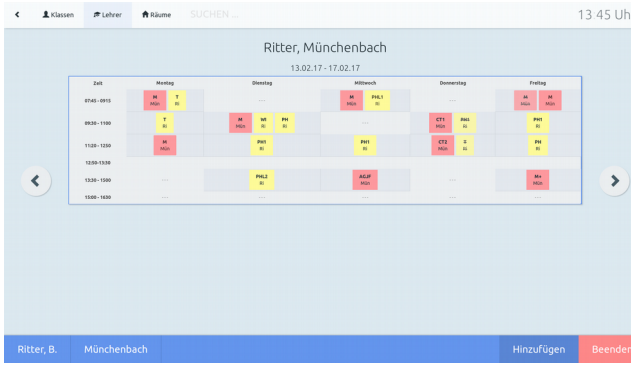


Abbildung 6: Lehrer Aggregation - Verschiedene Lehrerpläne vergleichen

Nach einiger Zeit ist unser Blick auf die Suche gefallen. Diese musste noch verbessert werden, um auch wieder das kleine „Etwas“ zu haben. Dann haben wir Suchvorschläge

hinzugefügt, um noch schneller zu gewünschten Informationen zu kommen. Zudem wird, sobald nur noch ein Ergebnis gefunden wird, automatisch dieser Stundenplan ausgewählt.

Hier eine Übersicht der Möglichkeiten, einen Stundenplan anzuzeigen:

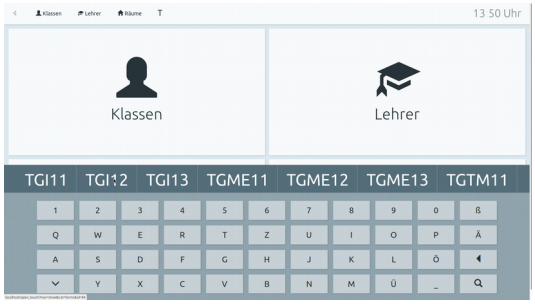


Abbildung 7: Suchvorschläge nach Eingabe "T"

<u>Anzeige der Klasse über das GUI:</u>
Klick auf Klassen > Klick auf TG / SG > Klick auf die TG-Klasse > Anzeige

<u>Anzeige durch Suche:</u>
Klick auf Suche > Eintippen von ‚TGI12' > automatische Anzeige

<u>Anzeige über QR Code:</u>
Scan des QR Codes > Anzeige

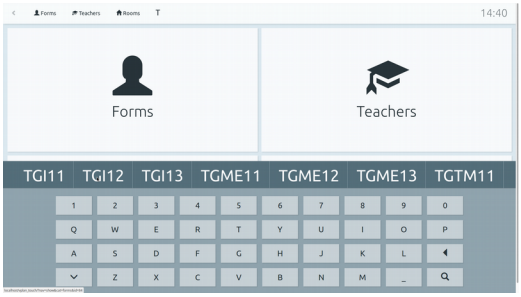


Abbildung 8: Englisch sprachige Version mit englischer Tastatur

Um auch Schulen in anderen Ländern zu ermöglichen, die Software zu benutzen, haben wir vor kurzem die Ausgabetexte in eine externe Datei gelegt und diese bereits auch in die Sprachen Englisch und Französisch übersetzt.

Die Software-Webseite ist ebenso zweisprachig (Deutsch, Englisch).

Kommentare und Variablenbenennungen im Quellcode sowie der Installer sind auf Englisch.

Auf einen Blick:

Vorteile gegenüber dem alten System:

- selbstständige Aktualisierung
- wartungsarm
- selbstständige Sicherung der Daten durch Programm- und Sicherheitsupdates
- über SSH installier- und wartbar
- kein VNC notwendig
- Verwaltung mehrerer Geräte zeitgleich möglich
- nicht virenanfällig
- ansprechende und zeitgemäße Optik
- nicht nur auf eine Software-Firma beschränkt
- keine Lizenzprobleme, da public domain
- problemlos mehrere Lehrerpläne vergleichen
- problemlos unbelegte Räume finden
- Anwesenheitsübersicht der Lehrer (jetzt anwesend/heute anwesend/heute abwesend)
- QR-Code Scanner zur schnelleren Navigation ohne Klicks
- nur eine Schnittstelle notwendig, andere Stundenplansysteme können implementiert werden
- Suche mit Suchvorschlägen und AutoSubmit
- Querverlinkungen: Klickt man z.B. im Stundenplan der Klasse auf den Lehrer, so wird man zu dessen Stundenplan weitergeleitet.

Abbildung 9: Ersatzstunde wird orange hervorgehoben

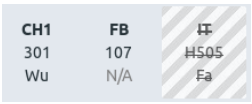


Abbildung 10: Reguläre Stunde, Ein Stunde ohne Lehrer, und eine ausfallende Stunde

Die Datenbank

Momentan benutzen wir zum Administrieren der Datenbanken MySQL. Das User-Interface bezieht die Daten für den Stunden-/Vertretungsplan von „webscheduler“ der Hauptdatenbank.
Bei einem Fetch der Untis-Datensätze wird eine temporäre Datenbank erstellt („webschedulertmp“). Diese temporäre Datenbank wird nun mit den Datensätzen von Untis beschrieben. Nachdem sämtliche Daten von Untis geparst wurden, wird die Hauptdatenbank mit dem Inhalt der temporären Datenbank überschrieben.

Die Struktur der Haupt- und der temporären Datenbank ist identisch. Somit ist das Kopieren der Tabellen recht einfach.

Die Datenbank-Struktur sieht wie folgt aus:

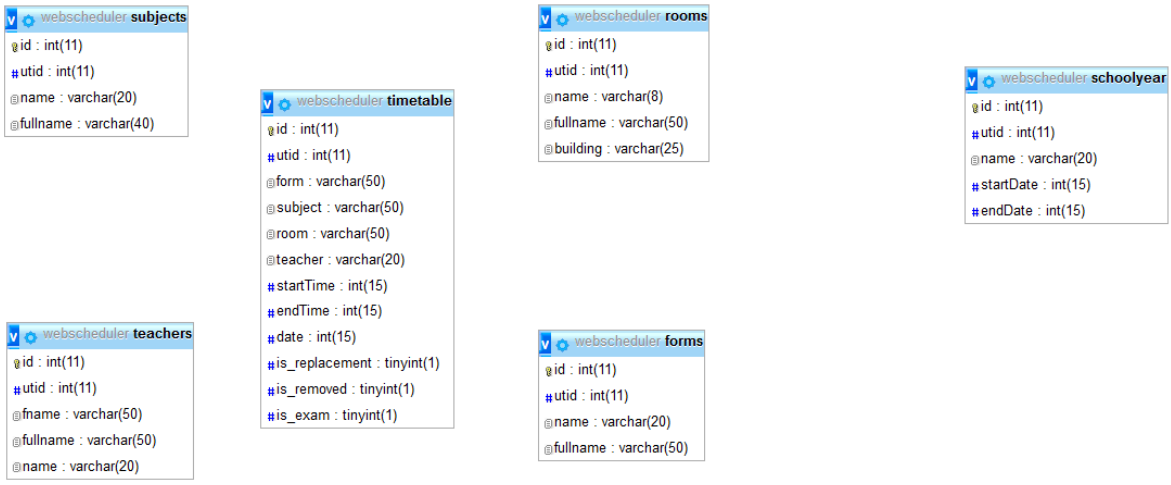


Abbildung 11: Datenbank-Struktur

Es befinden sich insgesamt sechs Tabellen in der Datenbank. Die Tabellen „subjects, teachers, rooms und forms“ beinhalten lediglich die Information, welche Unterrichtsfächer, Lehrer, Räume und Klassen existieren. Jedes Element in diesen Tabellen bekommt eine eigene ID zugewiesen. Desweiteren werden Untis-ID, Kürzel und kompletter Name eingespeichert. In der Tabelle „teachers“ wird noch der Vorname, wenn vorhanden, gespeichert und in der Tabelle „rooms“ wird das Gebäude, in dem sich der Raum befindet, gespeichert.
Die Stunden- und Vertretungspläne werden in der Tabelle „timetable“ gespeichert. Jeder Stundenplan setzt sich aus der Untis-ID von Klassen, Räumen, Lehrern und Fächern zusammen. Es wird jede Unterrichtsstunde einzeln gespeichert.

Um nun die Stunden auszugeben, wird in der Tabelle „timetable“ jeder Eintrag durchgearbeitet und zu jeder dort eingetragenen ID der passende Wert in den verschiedenen

Tabellen gesucht. An der GHSE sind das - alleine in der „timetable“-Tabelle - 134.711 Datensätze.

Da wir im Unterricht bei Herrn Noeske mehr und mehr über SQL-Datenbanken gelernt haben, stellten wir nach dem Regionalwettbewerb fest, dass unsere Struktur in der Datenbank nicht sehr effizient ist. Nach einem Gespräch mit ihm haben wir uns entschlossen, die Struktur grundlegend zu verändern.

Der erste Schritt war, die Datenmenge auf ein Zehntel herunter zu brechen, indem wir das Intervall von einem Jahr auf nur sechs Wochen geändert haben.

Somit sind wir bei einer gesamt Datenmenge von ca. 15.000 Datensätzen gelandet.

Der nächste Schritt war nun, die Datenbank-Struktur zu verbessern. Denn im Moment werden die Array-IDs von Klassen, Fächern, Lehrern und Räumen lediglich als String gespeichert und durch Semikolons getrennt. Doch die Verarbeitung von Strings in SQL dauert um ein Vielfaches länger als mit „Integer“.

Wir erstellten also eine Tabelle, die die Referenz zu den Klassen einspeichert. Dieses Prinzip wendeten wir ebenfalls auf die Unterrichtsfächer, die Lehrer und die Räume an. Somit ergaben sich mehr Tabellen, aber kürzere Rechenzeiten und dadurch ein noch schnelleres System mit einer besseren Datenbank-Struktur.

Durch die neu erzeugten Tabellen und Datensätze habe wir nun wieder eine Datenmenge von ca. 100.000 Datensätzen.

Konfiguration

Alle im Verzeichnis „config“ liegenden Dateien werden von Updates nicht beeinträchtigt. Dieses Verzeichnis bestimmt die gesamte Konfiguration, kann also anstatt einer Installation auch auf einen anderen Computer kopiert werden.

Es gibt zwei Möglichkeiten, dem Programm seine Konfiguration zukommen zu lassen: Die eine ist für den Einzelbetrieb gedacht, die andere für mehrere dieser Vertretungspläne innerhalb eines Netzwerkes.

In der Datei „school_config_generate_file.php“ können die Einstellungen angepasst werden. Dabei gibt es sehr viele Möglichkeiten, die jedoch derzeitig alle auf einen Standardwert eingestellt sind. Diese sind in der Datei selbst jeweils beschrieben. Dabei wird innerhalb dieser Datei in drei Blöcke unterschieden: Der erste sind header-Daten (z. B. Sprache), danach Sicherheitseinstellungen (z.B. Pin-Sperren), und zuletzt Konfigurationen zur Anzeige, (z. B. reguläre Ausdrücke, um Kategorisierungen zu definieren, oder um schulspezifische Einstellungen zu regeln).

Um die Konfiguration zu erzeugen, muss das eben veränderte Script lediglich ausgeführt werden, und eine Datei namens „school_config.json“ wird erzeugt. Diese wird zur Konfiguration verwendet und ist notwendig, um VPlan Touch zu starten.

Abbildung 12: Per RegEx Konfigurierbare Kategorisierung

Die zweite Möglichkeit erweitert die erste. Auf einen im selben Netzwerk liegenden Server (dieser kann auch ein Vertretungsplan-Touchscreen sein) wird die Konfigurationsdatei verschoben. Der Pfad dieser Datei wird nun bei den anderen Vertretungsplänen (ebenfalls in der oben genannten Konfigurationsdatei) hinterlegt und nun greifen alle diese Vertretungspläne auf diese Konfiguration zu. Um auch hier eine Netzwerkausfall-Resistenz zu bieten wird automatisch bei Änderung der Daten oder nach Ablauf der „Expire-Einstellung“ eine lokale Kopie hinterlegt.

Datenschutz

Datenschutz ist uns bei diesem Projekt äußerst wichtig, da sämtliche Daten der Lehrerinnen und Lehrer in diesen Datenbanken zu finden sind. Außerdem speichern wir die Zugangsdaten zu den Servern.
Aus diesem Grund unterstützt unsere Schnittstelle SSL-Verbindungen. Lokale Passwörter werden, sofern möglich, mit SHA512 verschlüsselt. Zudem wird bei der Verschlüsselung Salt benutzt, um die Passwörter gegen Rainbow-Tables zu schützen. Die Reverse-programmierte Schnittstelle von uns wird per htpasswd gesichert, zudem ist eine Abfrage nur per POST-Methode möglich.

Installation

Die Installation der Vertretungsplan-Software ist wirklich extrem einfach und schnell. Da mit der Zeit jedoch viele Funktionen und benötigte Libraries hinzukamen, wurde die Installation mit der Zeit immer aufwändiger. Wir mussten teilweise Anpassungen im Betriebssystem vornehmen. Um seitenlange Installationsanweisungen zu vermeiden, schrieben wir nach einiger Zeit ein Installationsskript, das, parallel zum Source-Code, von der Produktseite heruntergeladen werden kann. Dieser Installer kümmert sich um die komplette Installation der Software, sowie optional um die Vornahme von Änderungen am Betriebssystem. Nur ab und zu wird man dazu aufgefordert, Passwörter zu setzen, einzugeben oder Optionen zu wählen.
In unseren Tests dauerte das Setup der Hardware ca. 20 Minuten, die Installation des Betriebssystems ca. 15 Minuten. Die Konfiguration und Installation mit unserem Installer „Vpinst“ dauerte nur wenige Minuten. Nach weiteren 15 Minuten waren bereits die aktuellsten Daten in der Datenbank hinterlegt, die standardmäßig automatisch in Zehn-Minuten-Intervallen auf den Rechnern upgedatet werden.

Unter www.ghse-online.de/vplantouch gibt es eine zweisprachige Installationsanleitung. Zusätzlich kann man zwei deutschsprachige Videos finden, in denen die Installation des Betriebssystems sowie die Installation der Software-Pakete innerhalb weniger Minuten erklärt wird.

Mirror Server

Um eine stärkere Verfügbarkeit garantieren zu können, haben wir ein Script geschrieben, das einen Server zu einem Mirror-Server werden lässt. Somit könnte auch ein Ausfall eines Installations-Servers kompensiert werden. Dieses Script wird von einem Cronjob in Intervallen aufgerufen, holt sich dann Versionsinformation und den Paketpfad der aktuellen Software und stellt diese zum Download bereit.

Abbildung 13: Mirror Server stellt Dateien zum Download zur Verfügung

Hardware Anforderungen:

Ram: 1GB (Empfohlen: 2GB Ram)
12 GB HDD (Empfohlen: 32GB SSD)
Linux Mint Mate 18.1 oder neuer
(Manuelle Installation ist getestet unter: Linux Mint Mate, Debian, CentOS, Ubuntu.)
Touchscreen mit mindestens 15“ (empfohlen: FullHD, 21“)
Eine Internetverbindung wird empfohlen

Optional:
QR-Code-Scanner

Testbetrieb an unserer Schule

Nach einem ca. einjährigen Test der Software an unserer Schule können wir sowohl die zuverlässige Funktionalität als auch die Abhärtung gegen Netzwerkausfälle und Schülerdummheiten nur bestätigen. Alle drei alten Computer wurden bereits einen Monat nach unseren ersten Tests entsorgt, weil sie nicht mehr benutzt wurden, da unser neues System nicht nur sicherer und wartungs-unaufwändiger, sondern auch schneller war.

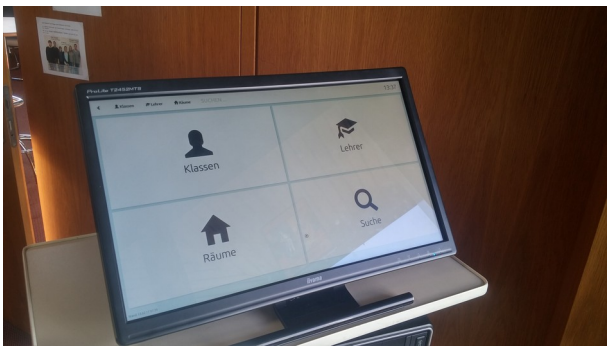


Abbildung 14: Vertretungsplan vor unserem Lehrerzimmer

Obwohl er aufgrund seines Platzes im Hauptgebäude, direkt vor dem Lehrerzimmer am häufigsten frequentiert war, gab es die kürzesten Schlangen davor. Meistens mussten, auch in den großen Pausen, nur ein oder zwei Schüler warten, bis sie an der Reihe waren.

Die Oberfläche haben wir während des Testbetriebs öfters geändert, um sie optimal an die Anforderungen zur schnellen und intuitiven Bedienung anzupassen. Buttons sind so positioniert, dass ähnliche Optionen möglichst nah beieinander sind. Suchanfragen werden mittlerweile unterschiedlich priorisiert, so wird zum Beispiel ein Lehrer, dessen Namen mit „H“ beginnt, vor einem Raum mit dem Bezeichnungsbeginn „H“ vorgeschlagen.
Und nicht nur die Schüler haben sich daran gewöhnt, mit wenigen Klicks ihren Stundenplan ansehen zu können. Oft haben Lehrer bei spontanen Treffen auf die neuen Features unseres Vertretungsplan-Touchscreens zurückgegriffen.
Während des Zeitraumes der Messe von Jugend Forscht Regional bauten wir den Rechner in der Schule ab und in der SICK-Arena wieder auf. Sofort wurde der Rechner an der Schule vermisst, alle hatten sich an diesen Modus der Übersicht gewöhnt. Für die Vertretungsplaner an unserer Schule ist der Vertretungsplan-Touchscreen nicht mehr wegzudenken.

Fazit

Mit dem VPlan-Touch wird das Schulleben sowohl von Schülern als auch von Lehrern stark vereinfacht. Es ersetzt zudem zwei alte Systeme durch ein neues System und ist ebenfalls wartungsarm.

Durch dieses Projekt haben wir gelernt, mit Problemen kreativer umzugehen: wie ein Projekt geplant werden sollte, wie man an Probleme herangeht und auch viel über Projektstrukturierung generell. Zudem haben wir viele neue Kontakte gewonnen.

Wir haben uns vorgenommen, dass wir die Netzwerk-Funktion so schnell wie möglich auch an weiteren Schule in Betrieb nehmen. Des Weiteren wollen wir eine Android-App mit Wear-App programmieren, um uns so schnell wie möglich von Untis distanzieren zu können.

Wir haben zudem gelernt, das die Problemlösung ein Prozess ist. Da wir mittlerweile seit über einem Jahr an unseren Programmen schreiben, haben wir unsere Algorithmen schon oft überdacht und überarbeitet. Meistens haben wir eine weitere Möglichkeit gefunden, diese nochmals zu optimieren. Teilweise sind wir Monate später wieder auf einen Code gestoßen, und haben ihn erneut verbessert.

Danksagung

Wir danken den betreuenden Lehrern Herr Carsten Münchenbach und Herr Carsten Noeske für ihre große Geduld und Unterstützung (auch außerhalb der Schulzeiten). Für die finanzielle Unterstützung, Motivation und Bereitschaft, dieses Abenteuer mitzutragen danken wir ebenfalls unserem Schulleiter Herrn Thomas Kruse.

Quellen:

Untis API
http://php.net/
http://www.w3schools.com/
http://stackoverflow.com/
https://jquery.com/
https://www.wikipedia.org/
https://jqueryui.com/
http://getbootstrap.com/
https://www.linuxmint.com/
https://ubuntuusers.de/
http://httpd.apache.org/docs/2.0/
https://vim-adventures.com/

Bilder von Dominik Ziegenhagel