HTL - Perg
Höhere Abteilung für Informatik

Diplomarbeit

Projektteam: Patrick Kern
Lukas Schleindlhuber
David Schmalzer

Projektbetreuer: Prof. Dipl.-Ing. Helmut Otto

In Zusammenarbeit mit VEDV GmbH

Betreuer Herr Reiner Voglhofer

Bearbeitungszeitraum: 01.10.2016 – 05.04.2017

HTL Perg | Machlandstr. 48 | A-4320 Perg | Tel.: +43 (0) 07262 53926

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer, als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, _____ Unterschrift _____

(David Schmalzer)

Perg, _____ Unterschrift _____

(Lukas Schleindlhuber)

Perg, _____ Unterschrift _____

(Patrick Kern)

Danksagung

Von einer Diplomarbeit spricht man von dem Ergebnis und dem Team, welches dahintersteht. Allerdings sind daran auch viele weitere Personen beteiligt, bei denen wir uns hiermit bedanken möchten.

Wir bedanken uns bei allen Personen die zu dieser Arbeit beigetragen haben. Im Besonderen danken wir unseren Projektbetreuer Herrn Dipl.-Ing. Helmut Otto der uns für Fragen und Rat immer zur Verfügung stand. Mit seinem umfassenden Wissen im Bereich Wirtschaft und speziell im Fachgebiet Informatik, konnte er uns oftmals weiterhelfen.

Weiters bedanken wir uns für die gute Zusammenarbeit mit der Firma VEDV GmbH und bei unserem Auftraggeber Herrn Reiner Voglhofer. Er stellte uns alle benötigten Ressourcen in geraumer Zeit zur Verfügung und wir konnten Ihn bei Unklarheiten jederzeit kontaktieren.

Herzliches Dankeschön!

Inhaltsangabe

Über das Projekt

Kurzfassung

Bei KIH handelt es sich um ein Verwaltungssystem, bei dem über eine Administrator Website auf einem zentralen Server, Kurse und Werbebilder eingefügt und gelöscht werden können. Die Kurse als auch die Werbung können auch editiert werden.

Außerdem sind alle Raspberry Pies, die konfiguriert und mit dem lokalen Netzwerk verbunden sind, auf der Administrator Website aufgelistet. Dabei ist es möglich zu bestimmen, was die Raspberry Pies auf den Monitoren anzeigen sollen.

Von der Administrator Website aus kann man nun in zehner Blöcken definieren, welche Kurse, wann, auf welchen Bildschirm angezeigt werden sollen. Übergibt man einen Raspberry Pi keine Kurse, so zeigt dieser automatisch eine Diashow von den Werbebildern an.

Zusätzlich unterstürzt der Server eine Heizungsfunktion, bei der die Heizung, falls gewünscht, vor einem Kurs, zu einem gewählten Zeitpunkt automatisch zum Heizen beginnt.

Abstract

KIH stands for “Kursverwaltungssystem mit integrierter Heizungssteuerung” and basically means course management system with an integrated heating regulation.

With this application, an administrator can work with courses, rooms, the heating and the monitors on the backend through a graphical user interface which is an ASP.NET web page. That means he can create, read, update and delete the different entities.

Moreover, all the Raspberry Pies, which are configured and connected to the local network are listed on the administrator’s webpage including their monitors to which they are coupled to.

Furthermore, you can see all the rooms and the ports on the relay card. Each of these ports are responsible for heating a specific room.

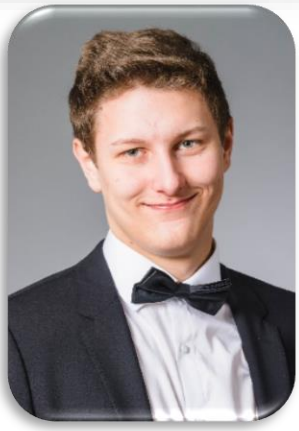
On the Backend, you can determine which course should be shown on which monitor. If a monitor has no courses to show it automatically shows a slideshow of ads.

Additionally, a heating service runs on the server, which communicates with the heating of a room and tells it when it should be switched on or off. This always happens at a certain time before and after a course.

Das Team

Das Team besteht aus drei Klassenkameraden, welche sich seit vier Jahren kennen. Dabei wurde bei der Wahl des Teams besondern darauf geachtet, dass sich die Kenntnisse von den einzelnen Personen gut ergänzen.

Patrick Kern



Geburtsdatum:	17.01.1998
Wohnort:	4293 Gutau
Telefon Nr.:	0660 / 35 61 927
E-Mail:	kern.patrick17@yahoo.de
Spezialgebiet:	HTML/JS

David Schmalzer

Geburtsdatum:	29.03.1998
Wohnort:	4283 Bad Zell
Telefon Nr.:	0664 / 417 21 59
E-Mail:	d.schmalzer@gmx.at
Spezialgebiet:	C# und Serverkonfiguration

Lukas Schleindlhuber

Geburtsdatum:	18.04.1997
Wohnort:	3350 Haag
Telefon Nr.:	0650 / 3350 979
E-Mail:	l.schleindlhuber@gmail.com
Spezialgebiet:	Datenbankdesign und C#

Betreuung

Die Betreuung der Diplomarbeit erfolgte hauptsächlich durch den Betreuungslehrer, welcher vom Diplomarbeits-Team ausgesucht wurde. In diesem Fall wurde Herr Dipl.-Ing. Helmut Otto gewählt, da dieser, einerseits über ein breites, sowie tiefes Wissen im Bereich C# besitzt, andererseits auch sehr gute Kenntnisse im Bereich Projektrealisierung hat.

Herr Dipl.-Ing. Helmut Otto

Beruf:	Professor und Geschäftsführer
E-Mail:	h.otto@htl-perg.ac.at

Auftraggeber

Herr Voglhofer ist seit 1993 in der EDV-Branche tätig und war von 1996 bis zur Firmengründung im Jahr 2010 als EDV-Systemtechniker in einem Systemhaus beschäftigt. Dort hat er den Grundstein für seine berufliche Entwicklung gelegt und war dort mit der Full-Service Betreuung von KMUs befasst. Diesen Aufgabenschwerpunkt führt er nun im eigenen Unternehmen fort und ist Ihr Ansprechpartner bei allen Fragen rund um Kommunikationsinfrastruktur.[1]

Herr Reiner Voglhofer

Beruf:	Geschäftsführer
E-Mail:	office@vedv.at

[1] Vgl.: VEDV Homepage

Entstehung und Planung

Unter diesem Punkt wird erklärt, was der Sinn und das Ziel dieser Diplomarbeit ist und warum sie in Auftrag gegeben wurde. Weiters wird noch auf die Planung der Diplomarbeit eingegangen.

Ist Zustand und Projektidee

Die Idee hinter der Diplomarbeit ist ein Kursverwaltungssystem zu erstellen, welches zusätzlich noch eine Heizungssteuerung bereitstellt. Es gibt zwar bereits ähnliche Systeme, allerdings entsprechen diese nicht der gewünschten Funktionalität und Technologie. Da auch der Source Code von ähnlichen Systemen Lizenzgeschützt ist, begann die Diplomarbeit ohne bestehende Ressourcen.

Projektziel

Das Ziel ist ein lauffähiges Programm, mit dem man Kurse und die Heizung manuell auf einem zentralen System verwalten und anschließend die Kurse auf mehreren Monitoren anzeigen kann.

Programmziele

Das Programm bietet die Möglichkeit, verschiedene Kurse auf mehreren Monitoren darzustellen. Um das Arbeiten mit diesen zu erleichtern, werden die Kurse zentral auf dem Backend gesteuert und anschließend zum Frontend gesendet, welche diese auf einem Bildschirm optisch ansprechend darstellen.

Falls zu einem Zeitpunkt kein Kurs auf einem Monitor angezeigt wird, wird die leere Fläche auf dem Bildschirm automatisch mit Werbebildern gefüllt, welche vom Backend geladen werden. Weiters persistiert das Frontend die Daten, welche vom Backend geschickt werden.

Falls zu einem Zeitpunkt kein Kurs auf einem Monitor angezeigt wird, werden die leeren Flächen auf den Bildschirmen automatisch mit Werbebildern gefüllt, welche vom Backend zuvor geladen werden. Weiters persistiert das Frontend die Kurse und Bilder.

Der Administrator kann die Kurse mit einer .TXT Datei in das System importieren, oder die Kurse händisch eintragen.

Als zusätzliches Feature gibt es im Backend eine Funktion, mit der man die Heizung eines Raumes aktivieren und deaktivieren kann.

Projektabgabe

Abgegeben wird das lauffähige Programm, in Form von Source Code, sowie das dazugehörige Installationshandbuch in schriftlicher Form.

Kommunikation

Zwischen den Diplomanten wurde hauptsächlich persönlich, aber auch viel über Messenger Systeme kommuniziert. Dabei wurde WhatsApp für kurze Informationen und Vereinbarung von Terminen verwendet. Für Problemlösungen, bzw. Ausarbeitungen, welche zusammen umgesetzt wurden, sind persönliche Treffen und Skype Sitzungen abgehalten worden. Das Programm Skype wurde deshalb verwendet, da man hier einfach Daten versenden kann und zusätzlich gibt es die Möglichkeit das der Bildschirm der Konferenzteilnehmer live übertragen wird. Mit dem Auftraggeber wurde telefoniert oder über E-Mails korrespondiert. Dem Betreuungslehrer wurden, soweit es möglich war, Fragen unter dem Schulalltag gestellt. Sonstige Fragen wurden über E-Mails geklärt.

Teamabwicklung

Wenn die Diplomarbeit als ein Ganzes abgegeben wird, so ist das Team bemüht die Arbeit zu unterteilen, um möglichst effizient zu arbeiten. Um dies zu verwirklichen bedarf es allerdings bestimmten Regeln.

Dabei wurden einige Rollen von Scrum übernommen.
So gibt es einen Scrum Master, welcher darauf achtet, ob Termine eingehalten werden. Dies wurde mithilfe eines Online-Scrumboards vereinfacht, indem jeder seine aktuelle Aufgabe einträgt, an der er gerade arbeitet. Wie für Scrumboards üblich, stehen hier auch die Aufgaben, welche noch ausstehend sind, bzw. welche schon fertig gestellt sind.
Eine weitere Rolle ist der Product Owner. Er ist die Schnittstelle zum Auftraggeber und kommuniziert hauptsächlich mit ihm.
Diese Rollen übernahmen Teammitglieder als zusätzliche Aufgabe, neben der Arbeit an der Diplomarbeit.

Weiters gab es verschiedene Meetings, bei denen definiert wurde, wer eine bestimmte Aufgabe bis wann erledigt, um gemeinsam im Team weiter an größeren Problemen arbeiten zu können.

Zeitplanung

In der nachfolgenden Tabelle sieht man die Meilensteine, welche am Beginn dieser Arbeit im Pflichtenheft festgelegt wurden. Die Tabelle wurde zu Beginn der Diplomarbeit festgelegt und die Abwicklung an sich wurde zur Gänze an diesem Plan orientiert. Die genauere Planung vor einem Meilenstein wurde im Abschnitt “Teamabwicklung“ angesprochen.

31.10.2016	Pflichtenheft und Systemspezifikationen
01.12.2016	Datenmodell der Datenbank
13.12.2016	Prototypen
31.01.2017	Integrations- und Gesamt- Tests
20.02.2017	Grundriss der schriftlichen Diplomarbeit
05.04.2017	Diplomarbeit Abgabe

Die Termine wurden, bis zum Punkt „Integrations- und Gesamt- Tests“ eingehalten. Durch die Testung wurden allerdings noch verschiedene Verbesserungen am Quellcode und Daten Design vorgenommen, wodurch sich die nachfolgenden Meilensteine nach hinten verschoben haben.

Budget und Ressourcen

Ein wichtiger Punkt einer Diplomarbeit sind die Finanzen, denn ohne Geld können oft benötigte Ressourcen und somit auch die Arbeit nicht abgeschlossen werden. In unserem Fall konnten die meisten Ressourcen geliehen werden, oder sie befanden sich schon in unserem Besitz, wie die nachfolgende Tabelle zeigt.

Die gesamten Kosten, welche durch die Diplomarbeit entstanden sind, wurden vom Auftraggeber übernommen.

Software	
Visual Studio 2015	gratis Schullizenz
Microsoft Server	gratis Schullizenz

Der Auftraggeber ist dazu verpflichtet eine eigene Visual Studio, sowie Microsoft Server Lizenz zu erwerben und den Source Code mit diesen Lizenzen zu kompilieren, um das Produkt kommerziell nutzen zu dürfen.

Hardware	
Relaiskarte	Leihgabe des Auftraggebers
Raspberry Pi	Leihgabe durch Patrick Kern
Server	von der Schule zur Verfügung gestellt
Relaiskartentester	Leihgabe des Auftraggebers

Fahrtkosten	**km**	**Kosten**
Meeting	42,5 km	17,85 €
Abholen der Relaiskarte	45,6 km	19,15 €

Die Kosten wurden nach dem Kilometergeld-Satz des ÖAMTCs mit 0,42 (Stand 21.03.2017) verrechnet.

Risikomanagement

In diesem Kapitel geht es darum, wie mit Risiken, die im Laufe der Diplomarbeit aufgetreten sind, umgegangen worden ist. Dabei geht es um die Erläuterung, Beurteilung, Analyse und Priorisierung der aufgetretenen Risiken.

Risiken

Zeitmanagement

Ein mögliches Risiko wäre, dass die Zeit zum Arbeiten nicht richtig eingeteilt wird und dadurch die Diplomarbeit nicht fertig abgeschlossen werden kann. Falls man einen Meilenstein unterschätzt, kann es leicht sein, dass dadurch der Zeitplan gefährdet ist.

Motivation

Es besteht immer eine Möglichkeit, dass eine bestimmte Motivation für die Ausarbeitung fehlt. Dadurch kann der Projektabschluss verschoben werden oder gar nicht stattfinden. Möglichkeiten für den Motivationsnachlass wären zum Beispiel, schulischer und privater Stress.

Knowhow

Natürlich ist es ebenfalls ein Risiko, falls ein bestimmtes Knowhow fehlt. Speziell mit Technologien mit denen man nur wenig bis gar nicht gearbeitet hat.

Struktur

Ein weiterer Grund wäre die Struktur des Programms. Ist diese unüberlegt, so kann sie schnell unübersichtlich werden und die Arbeit erschweren, bis zu einem Punkt, wo manche Funktionen nicht mehr realisierbar sind.

Ressourcen

Unter den Ressourcen versteht man die Geräte und Gegenstände, die zur Verfügung gestellt werden, aber auch die Zeit in der die Arbeit abgeschlossen werden muss. Da an vielen schulischen Projekten gleichzeitig gearbeitet wird, könnte es sein, dass ein Projekt zu kurz kommt. Auch die Gegenstände, wie ein Raspberry Pi oder eine Relaiskarte sind erfolgsrelevant für die Diplomarbeit.

Kommunikation

Die Kommunikation innerhalb des Projektteams, aber auch außerhalb mit dem Auftraggeber und Betreuungslehre sind wichtige Faktoren. Denn umso mehr Missverstände entstehen, umso höher wird das Risiko für einen Misserfolg.

Risikobewertung

Die vorher genannten Risiken werden in diesem Kapitel passend zu dieser Risikobewertungsmatrix zugeteilt. Wobei in der Matrix unten eine Wahrscheinlichkeit von unvorstellbar mit einem geringen Schweregrad des Schadens angibt und oben rechts eine Wahrscheinlichkeit von häufig mit eine Schweregrad des Schadens von katastrophal angibt.

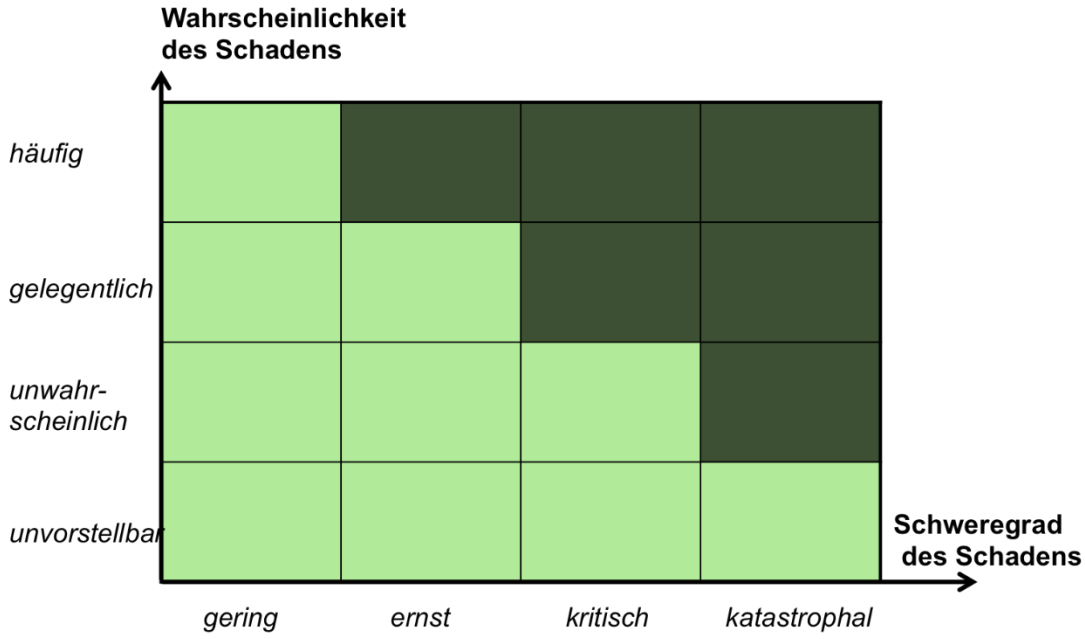


Abbildung 1 Risikobewertungsmatrix

Risiko	Beurteilung	Begründung
Zeitmanagement	-gelegentlich -ernst	Da in der Schullaufbahn noch kein derartiges Projekt durchgeführt wurde, kann es leicht zu Fehleinschätzungen kommen.
Motivation	-unvorstellbar -ernst	Da das Team aus 3 Personen besteht, ist dies kein großes Risiko, da untereinander geholfen und motiviert wird.
Knowhow	-unwahrscheinlich -gering	Mit einigen der Technologien wurde schon gearbeitet, deswegen ist es ein geringes Risiko, welches mit einer Wahrscheinlichkeit von *unvorstellbar* auftritt.
Struktur	-unwahrscheinlich -kritisch	Da es zu großen Problemen kommen kann, wen dieses Risiko eintritt, wird die Eintrittswahrscheinlichkeit meistens durch dementsprechende Vorausplanung, verhindert.
Ressourcen	-unwahrscheinlich -katastrophal	Falls bestimmte erfolgsrelevante Ressourcen nicht zur Verfügung stehen, wäre es *katastrophal*, aber das wäre sehr *unwahrscheinlich* da auf dem Auftraggeber Verlass ist und die Zeit gut eingeteilt wird.
Kommunikation	-unvorstellbar -kritisch	Da alle Diplomanten gute Freunde sind und sich auch gut mit dem Betreuungslehrer und dem Auftraggeber verstehen ist es *unvorstellbar*, dass es vorkommt aber es wäre *kritisch*.

Technische Grundlagen

Einleitung

In diesem Kapitel werden die benötigten Hilfsmittel beschrieben, die zur Realisierung der Diplomarbeit benötigt wurden. Zu Beginn werden die verwendeten Entwicklungsumgebungen und danach die zur Implementierung benötigten Tools erklärt. Dazwischen gibt es noch eine kurze Übersicht, wo und warum die verwendeten Tools eingesetzt wurden.

Verwendete Entwicklungsumgebungen

Visual Studio 2015

Visual Studio wurde von Microsoft entwickelt. Diese Entwicklungsumgebung ermöglicht es dem Programmierer Websites, Windows Programme, Windows Services und noch viele andere Anwendungen zu entwickeln. Außerdem unterstützt Visual Studio das, ebenfalls von Microsoft entwickelte, .NET Framework. Im Zuge der Diplomarbeit wurde Visual Studio zur Entwicklung des Server Backends und der Erstellung des Windows Services verwendet.[2]

Abbildung 2 Visual Studio Logo

Internet Information Service

Bei dem Internet Information Service handelt es sich um einen von Microsoft entwickelten Webserver. Auf diesem können unteranderem ASP.NET Anwendungen bereitgestellt werden. Die Kommunikation erfolgt in unserem Fall über das Hypertext Transfer Protocol. Auf diesem Webserver können einige Einstellungen vorgenommen werden. Einige Beispiele dafür wäre mit welcher Methode sich der User Authentifizieren muss, welche Seite als Startseite verwendet wird oder welche Seite bei welcher Art von HTTP-Fehler angezeigt werden soll.[3]

Notepad++

Notepad++ ist, wie Notepad, ein Editor. Das ++ soll in diesem Fall für den erweiterten Funktionalitäts-Bereich von Notepad stehen. Eine Funktion von Notepad++ ist zum Beispiel das farbliche hervorheben von der Syntax einer Programmiersprache.

C, C++, C# über Java, JavaScript, HTML, sowie Pascal, Perl und Python ist ein kleiner Auszug der unterstützten Sprachen.

Weiters wird bis zu einen gewissen Grad die IntelliSense Funktion unterstützt. Durch Einbindung von bestimmten Plugins zum Verbessern der oben genannten Funktionen, sowie Plugins zum Kompilieren von gewünschten Programmiersprachen, kann man Notepad++ auch als IDE ansehen.

Darum reicht im Falle dieser Diplomarbeit auch ein Editor aus, um das komplette Frontend zu programmieren.

[2] Vgl.: Visual Studio Einleitung
[3] Vgl.: Internet Information Service Einleitung

MySQL Workbench

Die MySQL Workbench ist eine grafische Benutzer Schnittstelle (GUI) des MySQL Servers. Das bedeutet man kann einfacher durch eine grafische Oberfläche mit der Datenbank arbeiten, da es einfacher ist Daten auszulesen, einzufügen, bearbeiten und zu löschen. Die Struktur der Tabelle wird hierbei visualisiert.

Übersicht

Da nun die verwendeten Entwicklungsumgebungen aufgelistet und beschrieben wurden, wird in diesem Kapitel kurz erklärt, wo die von uns eingesetzten Tools verwendet wurden.

Backend

Zu Beginn wurde ASP.NET MVC als Grundlage für das Backend ausgewählt, weil uns dieses System bereits von anderen Projekten bekannt war. Ein weiterer Grund für ASP.NET MVC ist, dass sehr gute Dokumentationen und Anleitungen im Internet gibt die von Microsoft aktuell gehalten werden. Darüber hinaus ist ASP.NET MVC ein Teil des .NET Frameworks. Dadurch können andere Technologien von Microsoft leicht eingebaut werden und es gibt auch dafür wieder genügend Erklärungen und Beispiele.

Entity Framework wurde ausgewählt, weil es in fast jedem Tutorial von Microsoft vorkommt und dadurch ist es „State of the Art“ dieses Framework in Kombination mit ASP.NET zu verwenden. Ein weiterer Vorteil des Frameworks ist es, dass sich der Programmierer nicht mehr im Detail um die Verwaltung der Daten kümmern muss. Durch das Entity Framework ist es egal welche Datenbank zugrunde liegt, weil der Zugriff im Code immer derselbe ist. In Java wäre das passende Gegenstück dazu die „Java Persistence API“.

Die Windows Communication Foundation wurde uns bereits im Unterricht beigebracht. Bei WCF hat der Programmierer den Vorteil, dass er sich nicht mehr im Detail darum kümmern muss, wie die Daten beim Client ankommen. Auch hier wurde diese Technologie verwendet, da sie aus dem Unterricht bekannt war und ebenfalls weitläufig verbreitet ist.

Um dem Frontend und dem Windows Service mitzuteilen, wann die Daten von dem WCF Service neuzuladen sind, wurde SignalR verwendet. SignalR ist ebenfalls Bestandteil des .NET Frameworks und ist daher perfekt für unseren Anwendungsfall geeignet.

Für die Zusammenarbeit mit der Relaiskarte hat man sich für einen Windows Service entschieden, weil der jederzeit im Hintergrund läuft und eine einfache Implementierung von Timern zur Verfügung stellt. Der Windows Service verbindet sich mittels Entity Framework zur MySQL Datenbank und stellt komplexe Abfragen

mit der Abfragesprache LINQ. LINQ wurde verwendet, da im Zuge des Schulunterrichts schon ausgiebig damit gearbeitet wurde und deshalb der Wissensstand über diese Technologie dementsprechend hoch ist. Außerdem unterstützte LINQ alle nötigen Funktionen und ist einfach zu verstehen. Mithilfe der UniDAQ Klasse ist es möglich die Relaiskarte zu steuern. Diese Klasse wurde verwendet, weil es eine einfache Implementierung in C# zulässt und speziell für die verwendeten Relaiskarten erstellt wurde.

Vom Windows Service wurde eine Setup Datei anhand von InstallShield erstellt, welches eine einfache Installierung ermöglicht. Es wurde entschieden InstallShield zu nehmen, weil es eine spezielle Anpassung für die Verwendung mit Microsoft Visual Studio mitliefert.

Frontend

Am Frontend wurde entschieden einen Raspberry Pi mit dem Betriebssystem Rasbian einzusetzen, da der Raspberry Pi jegliche Erweiterungsfreiheit zulässt. Dazu kommt, dass auf sehr vielen Raspberry Pies Raspbian als Betriebssystem installiert ist. Daher gibt es zahlreiche Dokumentationen und Walkthroughs.

Nach dem Start des Betriebssystems wird automatisch im Chromium Browser eine Webseite gestartet, welche die Informationen anzeigt. Chromium bietet dabei Vorteile, wie die implementierte WebSQL Datenbank und das einfache verwenden des Kiosk Modus (Vollbildmodus) und im Gegenzug zu Chrome, bleibt man mit Chromium im Stil von Open Source. Der Weg einer Web Seite wurde gewählt, da er einfach und übersichtlich ist. Mithilfe des AngularJS Frameworks und des JQuery Framework inklusive SignalR Plugin, bietet die Web Lösung so ein gut Dokumentiertes System, welches Plattformunabhängig verwenden werden kann.

Zusätzliche wurde ein Script erstellt welches bei Systemstart die Mac-Adresse des Raspberry Pies in ein Text File schreibt. Das Script wurde in Python geschrieben, da Python einen knappen Programmierstils fördert.

Verwendete Tools

In diesem Kapitel werden die bereits in der Übersicht angeführten Technologien genau beschrieben und erklärt.

.NET Framework 4.5.2

Das .NET Framework ist Teil von .NET und hilft dem Programmierer viele verschiedene Arten von Programmen zu entwickeln. Bei diesem Framework handelt es sich um eine Zusammenfassung von Klassenbibliotheken, Programmierschnittstellen und Services. Dadurch muss der Programmierer nicht alles selbst immer wieder neuschreiben sondern es können die zur Verfügung gestellten Funktionen verwendet werden. Ein Beispiel dafür wäre, wenn eine Anwendung durch einen ping überprüfen muss, ob ein Rechner im Netzwerk ist, oder nicht. Anstelle den ping selbst zu programmieren, wird die bereitgestellte Methode verwendet. Dadurch muss man nicht immer wieder, wenn man die ping Funktion benötigt, diese in jedem Programm implementieren. Außerdem sind diese bereitgestellten Methoden ausgiebig getestet. Indem diese Methoden verwendet werden, ist der Programmierer in der Lage, sich auf die individuellen Funktionen der Anwendung zu konzentrieren. Ein weiterer Vorteil solcher Frameworks ist es, dass ein Außenstehender den Code leicht verstehen kann, da die selben Methoden verwendet werden und der Benutzer kennt vielleicht bereits einige Funktionen aus einem anderen Framework die gleich funktionieren. Beispielsweise „Speichern Unter“ oder „Öffnen“. Unteranderem ist ASP.NET ein Teil des .NET Frameworks. Dadurch können auch Webanwendungen mit dem .NET Framework programmiert werden.[4]

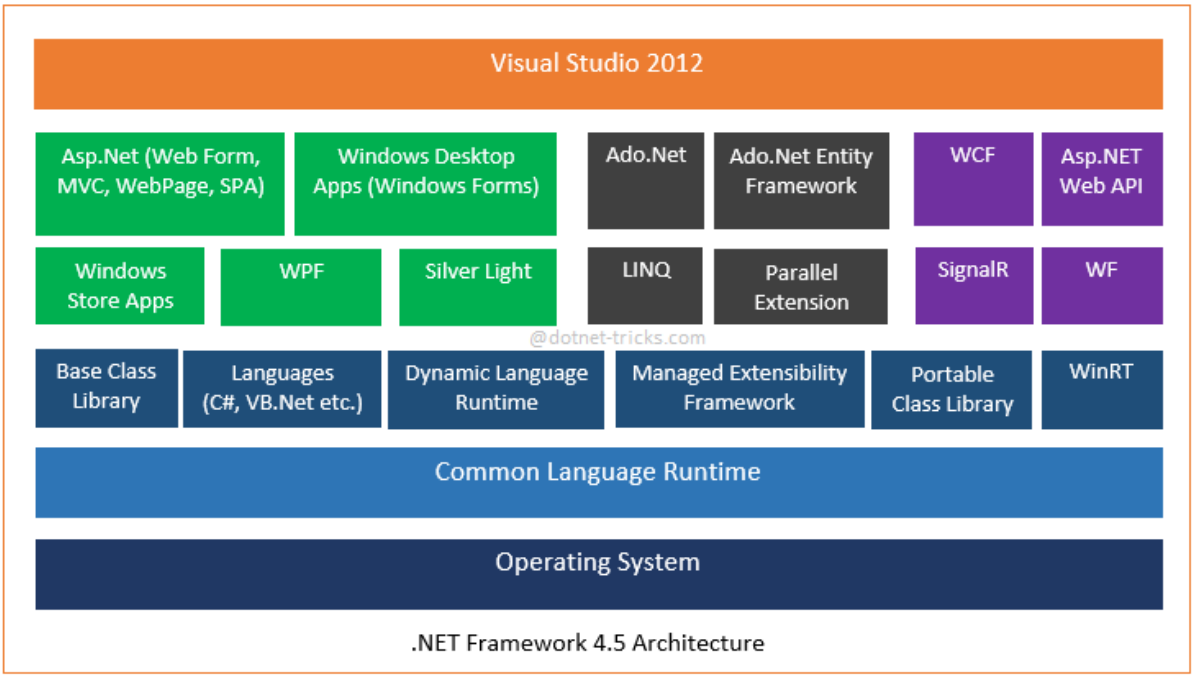


Abbildung 3 .NET Framework 4.5

[4] Vgl.: .NET Framework Erklärung

Wie man in Abb 4. bereits erkennt sind fast alle Komponenten die in dieser Arbeit im Backend vorkommen im .NET Framework vorhanden. Es werden folgende Teile des .NET Frameworks verwendet:

- ASP.NET MVC
- Entity Framework
- WCF
- SignalR
- Windows Service
- LINQ

Die aufgezählten Punkte werden in den nächsten Abschnitten genauer beschrieben

ASP.NET MVC

ASP.NET steht für Active Server Pages .NET. Es gibt 3 Frameworks für ASP.NET die zur Verfügung stehen und von diesen wurde die MVC Variante ausgewählt.

MVC steht für Model-View-Controller und ist ein unter Entwicklern sehr bekanntes Standardentwurfsschema. Das MVC-Framework beinhaltet die folgenden Komponenten:

Modelle. Modellobjekte sind die Teile der Anwendung, die die Logik für die Datendomäne der Anwendung implementieren. Häufig speichern und rufen Modellobjekte den Modellzustand aus einer Datenbank ab. Ein Product-Objekt kann z. B. Informationen aus einer Datenbank abrufen, diese verarbeiten und anschließend aktualisierte Informationen zurück in eine Produkttabelle in einer SQL Server-Datenbank schreiben. Auf das Erstellen der Modellobjekte wird im Abschnitt Entity Framework genauer eingegangen.

Ansichten. Ansichten sind die Komponenten, die die Benutzeroberfläche der Anwendung anzeigen. In der Regel wird diese Benutzeroberfläche aus den Modelldaten erstellt. Ein Beispiel wäre eine Bearbeitungsansicht einer Produkttabelle, in der Textfelder, Dropdownlisten und Kontrollkästchen auf Grundlage des aktuellen Zustands eines Produkt-Objekts angezeigt werden.

Controller. Die Controller-Komponenten behandeln Benutzerinteraktionen, arbeiten mit dem Modell und wählen im letzten Schritt die Ansicht aus, mit der die Benutzeroberfläche gerendert wird. In einer MVC-Anwendung zeigt die Ansicht nur Informationen an. Der Controller übernimmt das Behandeln und das Reagieren auf Benutzereingaben und Interaktion. Der Controller behandelt z. B. Abfragezeichenfolgenwerte und leitet diese Werte an das Modell weiter, dass diese Werte dann verwenden kann, um die Datenbank abzufragen.[5]

[5] Vgl.: ASP.NET Erklärung

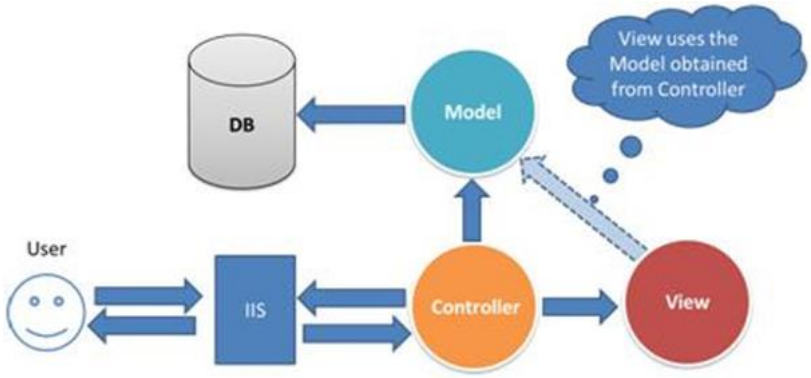


Abbildung 4 ASP.NET MVC

In unserem Fall sieht die ASP.NET Anwendung wie in der obigen Abbildung aus. Der User benutzt die Funktionen des Controllers nicht direkt, sondern er kommuniziert lediglich mit dem IIS der die Aktionen für den User ausführt. Der IIS liefert die Ansichten und die Ergebnisse der ausgeführten Aktionen an den Benutzer zurück. Das Model interagiert mit der Datenbank, um die Daten des Anwenders zu verwalten.

Entity Framework

Das Entity Framework ist ein sogenanntes Object Relational Mapper Framework. Es wurde entwickelt, damit man sich nicht mehr um den Zugriff auf die Daten kümmern muss. Das Entity Framework kann man in 3 verschiedenen Szenarien verwenden.

- Es gibt bereits eine Datenbank und man lässt sich die Klassen aus den Tabellen in der Datenbank erstellen (Database First)
- Man erstellt die Klassen und erstellt daraus die Tabellen und Relationen in der Datenbank (Code First)[6]
- Die Datenbank und die Klassen werden durch ein Datenbankmodell erstellt (Model First)[7]

[6] Vgl.: Code First Erklärung
[7] Vgl.: Entity Framework Erklärung

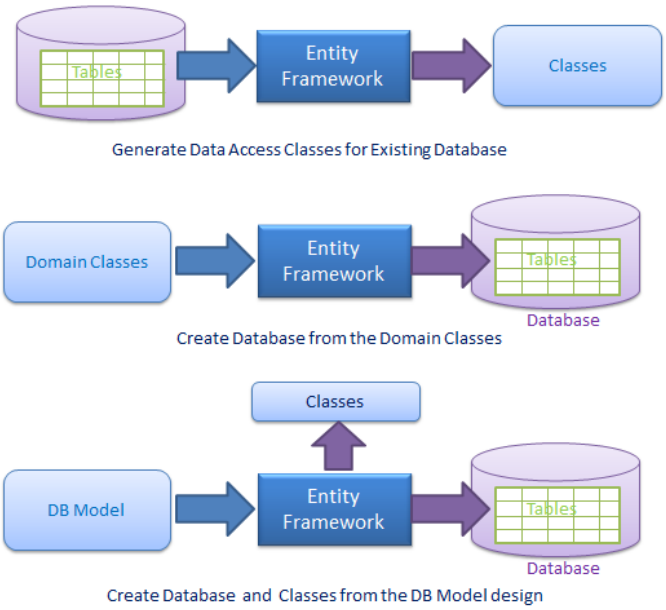


Abbildung 5 Entity Framework Übersicht

In dieser Arbeit ist der Code First Ansatz zum Einsatz gekommen, weil es mit Database First ein Problem gab, welches mit Code First erst gar nicht aufgetreten ist. Code First bietet den Vorteil, dass man schon bevor man das Datenmodell fertig hat programmieren kann und das Datenmodell im Nachhinein leicht geändert werden kann. Dies hat sich angeboten da noch einige Veränderungen am Datenmodell vorgenommen wurden. Bei Database First war das Problem, dass, sobald sich unser Datenmodell änderte auch alle Models im Programmcode neugenerierten. Dies führte dazu, dass man alle DataAnnotations immer wieder neu schreiben musste.

WCF Data Services

WCF bedeutet Windows Communication Foundation und erleichtert die Übertragung von Daten in Netzwerken, da die Funktionen die zur Übertragung benötigt werden, einheitlich verwendet werden können. Den WCF Data Services liegt das Open Data Protocol (OData) zu Grunde. Durch OData ist es möglich Datendienste bereitzustellen. OData stellt die Daten zur Verfügung die man über URIs abrufen kann. Da HTTP ein zustandsloses Protokoll ist, wird es ermöglicht, dass man über die REST-Semantik auf die, zur Verfügung gestellten Daten zugreift. Dies hat den Vorteil, dass jeder Client, der über http eine GET Anfrage machen kann, Zugriff auf die Daten eines WCF Services hat.

Damit OData weiß, wie es die Daten zu Verfügung stellen muss, benötigt man sogenannte Entity Data Models. Ein Entity Data Model beschreibt die Entitäten und Beziehungen unabhängig von einem Speicherschema. Dadurch werden die gespeicherten Daten unabhängig von Anwendungsentwurf und der Entwicklung zur Verfügung gestellt.[8]

[8] Entity Data Model Erklärung

Eine URI um Daten von einem WCF Service abzurufen, würde wie folgt aussehen:

http://localhost:3287/VedvService.svc/course

In diesem Fall lädt der WCF Service alle Kurse die gerade in der Datenbank sind und stellt sie in Form von ODATA für den Client bereit.

OData kann die Daten auch in anderen Formaten zur Verfügung stellen. In dieser Arbeit wird JavaScript Object Notation (JSON) verwendet. Um die Daten in diesem spezifischen Format zu erhalten, muss man eine kleine Änderung in der URI vornehmen.[9]

http://localhost:3287/VedvService.svc/course?$format=json

In der Nachfolgenden Abbildung ist zu sehen, wie die Antwort des WCF Services aussieht. In diesem Fall erkennt man, dass alle Kurse, in diesem Fall zwei, aus der Datenbank gelesen wurden, und die Daten korrekt im JSON Format zurückgegeben wurden.

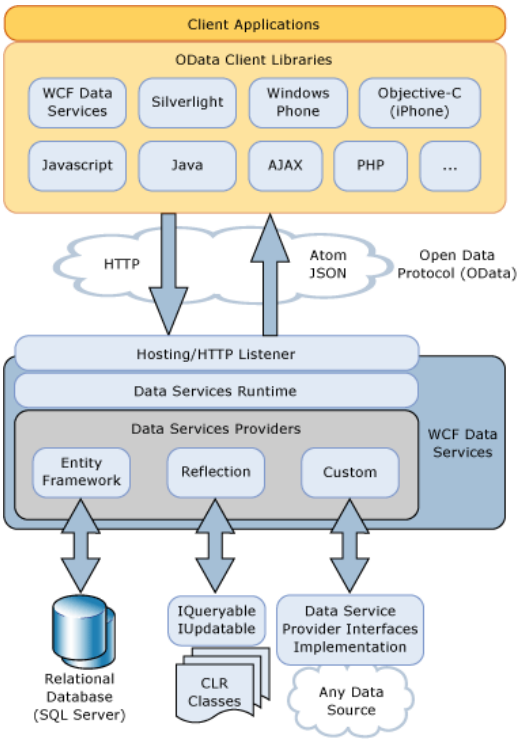
Abbildung 6 WCF JSON Format

In unserem Fall sendet der JavaScript Client eine HTTP-GET Anfrage an den Listener des WCF Service. Dieser greift über das Entity Framework auf die MySQL Datenbank zu welche die Daten zurückgibt. Diese werden dann über ODATA zurück

[9] Vgl.: WCF Data Services Erklärung

an den JavaScript Client gesendet. Dieser konvertiert die Daten von ODATA zu JSON.

Allerdings wird beim Datenaustausch noch SignalR implementiert. Dadurch kann der Server die Clients informieren, sobald neue Daten zur Verfügung stehen, die sie vom WCF Service abrufen können.



Hier erkennt man wie der WCF Service zwischen Clients und Datenquellen, die Daten auf verschiedene Arten übertragen kann.

Abbildung 7 WCF Data Services

SignalR

ASP.NET SignalR ist eine Bibliothek für ASP.NET, welche es ermöglicht, Daten vom Server zu allen verbunden Clients zu senden. Werden zum Beispiel Daten am Server geändert, so werden die Clients sofort über diese Änderungen informiert. Das hat den Vorteil, dass die Clients keine periodischen Anfragen zum Server senden müssen, da der Server bekanntgibt, dass neue Daten zur Verfügung stehen.

Durch SignalR ist es möglich sogenannte Remote Procedure Calls (RPC) zu machen. Ein RPC ruft eine JavaScript Funktion im Client vom Server seitigen Code aus. Der Client ist dadurch auch in der Lage Funktionen vom Server aufzurufen.

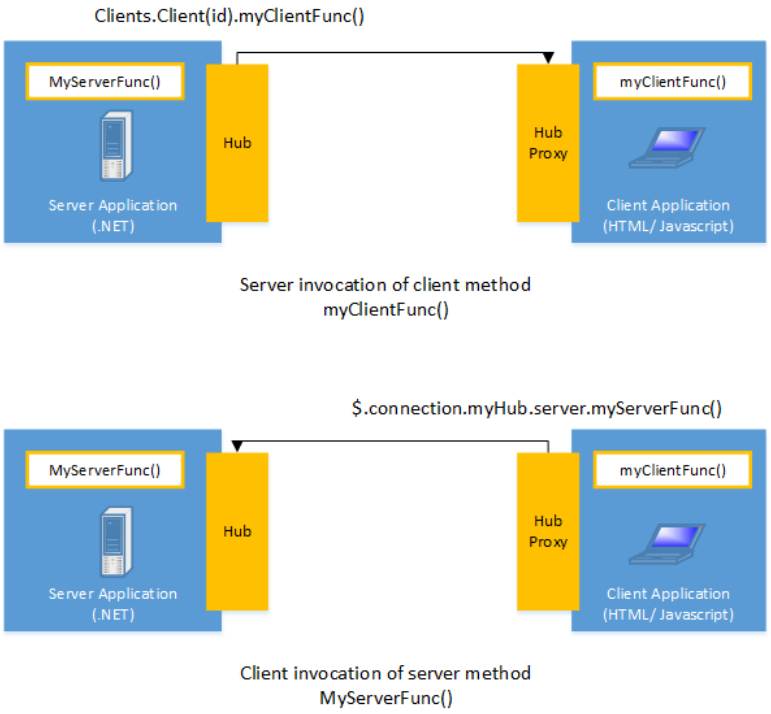


Abbildung 8 SignalR Übersicht

Durch diese Funktion ist es möglich an alle Clients Nachrichten zu senden oder auch an spezifische Clients. Ein großer Unterschied zu normalen HTTP Verbindungen ist es das die Verbindung zwischen Client und Server persistent ist. Dadurch kann man auch Events auf der Client Seite setzen, wenn der Server die Verbindung schließt.

SignalR verwendet zum Datenaustausch sogenannte WebSockets. Sind diese am Client nicht verfügbar verwendet SignalR ältere Methoden zum Informationsaustausch.[10]

Windows Service

Ein Windows Service ist ein Programm das im Hintergrund von Windows läuft. Dieser Service erledigt Aufgaben im Hintergrund von denen ein Benutzer nichts merken soll. Der Anwender kommuniziert nicht direkt mit einem Dienst, weil es keine Benutzerschnittstelle dafür gibt. Mit dem Service Control Manager kann man solche Services starten. Solche Programme müssen so geschrieben sein, sodass sie *start*, *pause*, *continue* und *stop* Befehle vom Service Control Managers entgegennehmen können. Der Dienst kann beim Systemstart *automatisch* -oder *manuell* durch ein anderes Programm gestartet werden.[11]

[10] Vgl.: SignalR Erklärung
[11] Vgl.: Windows Service Erklärung

Solche Windows Service werden in der Desktop App „Dienste“ aufgelistet. Dort kann man außerdem den Service beenden und wieder neu startet. Weiters besteht hier die Möglichkeit den Start Typ und die Anmeldung zu ändern.

InstallShield

InstallShield ist eine Software zur Kreierung von Installern oder Softwarepaketen. Hauptsächlich wird es benutzt zum Installieren von Software auf Microsoft Windows Desktop und Server Plattformen, es kann aber auch zum Verwalten von Software Applikationen und Paketen auf verschiedenen mobilen Geräten genutzt werden. InstallShield generiert eine .msi Datei, die man danach auf einem anderen Computer ausführen kann, um die Software zu installieren. Es ist außerdem möglich Fragen zu stellen, Voraussetzungen zu definieren und Registrierungseinstellungen festzulegen, welche der Benutzer während der Installation auswählen kann.[12]

LINQ

LINQ steht für Language Integrated Query und ist eine Abfragesprache von Microsoft, die es ermöglicht, Abfragen von Daten aus einer Menge von verschieden Quellen in .NET zu erstellen. Es gibt zum Beispiel die Möglichkeiten, aus einer relationalen Datenbank, aus einer XML Datei, oder einfach aus einer Liste Daten abzufragen. LINQ Query Operationen bestehen aus 3 Operationen.[13]

- Die Datenquelle erhalten
- eine Abfrage erzeugen
- die Abfrage ausführen

Simples Beispiel

```
var course = from c in context.courses
             select c;
```

Beispiel 1 LINQ - simple

Man kann in Beispiel 1 ein simples Beispiel von der Erzeugung einer LINQ Abfrage sehen. In diesem Fall würde man eine Liste von Kursen aus der Datenbank zurückbekommen.

course…	Name der Ergebnisvariable
c…	Name der Variable der innerhalb der Abfrage verwendet wird
context.courses…	Datenquelle (Entität aus der Datenbank, Liste, usw.)

[12] Vgl.: InstallShield Erklärung
[13] Vgl.: LINQ Erklärung

Beschreibung einzelner Operationen von LINQ

SELECT

```
var firstTimeBeginQuery = from course in context.courses
                          select course.course_start_time;
```

Beispiel 2 LINQ - SELECT

Mit dem Schlüsselwort SELECT wird bestimmt, welche Attribute von der Abfrage zurückkommen sollen. Im Beispiel 2 wird nur das Attribut ‚course_start_time' vom Kurs ausgewählt. Somit bekommt man alle Startzeiten der Kurse.

WHERE Klausel

```
var firstTimeBeginQuery = from course in context.courses
                          where course.course_start_time > DateTime.Now
                          select course.course_start_time;
```

Beispiel 3 LINQ - WHERE

Mit dem Schlüsselwort WHERE kann man die Rückgabewerte filtern. In Beispiel 3 würden alle Startzeiten die in der Zukunft liegen als Ergebnis zurückkommen.

ORDERBY Klausel

```
var firstTimeBeginQuery = from course in context.courses
                          orderby course.course_start_time
                          select course.course_start_time;
```

Beispiel 4 LINQ - ORDERBY

Mit dem Schlüsselwort ORDERBY sortiert LINQ den Rückgabewert je nachdem, ob man descending oder ascending auswählt, absteigend oder aufsteigend. Standartmäßig ist ascending ausgewählt.

JOIN

```
var room_desc = from room in context.rooms
                join course in context.courses on room.room_id equals course.room_id
                where course.course_start_time > DateTime.Now
                select room.room_desc;
```

Beispiel 5 LINQ - JOIN

Mit dem Schlüsselwort JOIN kann man mehrere verschiedene Tabellen über ihre gemeinsamen Werte verbinden. Im Beispiel 5 wird ein Raum mit einem Kurs durch die Raum ID verbunden, die in beiden Tabellen vorhanden ist.

ANY

```
if (firstTimeEndQuery.Any())
```

Beispiel 6 LINQ - ANY

Mit der Funktion Any kann festgestellt werden, ob eine Sequenz leer ist, oder ob es etwas Bestimmtes aufweist.

DISTINCT

Mit Distinct bekommt man keine doppelten Werte zurück, sondern nur eindeutige.[14]

UniDAQ Klasse

In dieser Klasse befinden sich alle benötigten Methoden die zur Bedienung der Relaiskarte nötig sind. Diese Methoden importieren Funktionen aus der UniDAQ.dll. Das bedeutet man braucht immer die UniDAQ.cs und die UniDAQ.dll damit man mit der Relaiskarte arbeiten kann. Die angewandten und somit wichtigsten Methoden davon lauten:

- **Ixud_DriverInit**
  Sie fordert das System die Ressourcen zuzuordnen, danach sucht sie alle Relaiskarten und initialisiert jede einzelne davon. Als Rückgabewert bekommt man die Anzahl der erkannten Karten. Dies muss immer vor dem Benutzen der Relaiskarte ausgeführt werden.
- **Ixud_WriteDO**
  Diese Funktion schreibt einen digitalen Output an einen bestimmten digitalen I/O Port. Das bedeutet sie schreibt zum Beispiel eine 1 für das Einschalten eines Ports und eine 0 zum Ausschalten. Falls es keine Fehler gibt bekommt man als Rückgabewert eine 0 zurück und sonst eine andere Zahl wobei jede Zahl für einen bestimmten Fehler steht. In der nachfolgenden Grafik ist ein kleiner Ausschnitt davon.

```
//Return Code
public const UInt16 Ixud_NoErr                 = 0;//Correct
public const UInt16 Ixud_OpenDriverErr          = 1;//Open driver error
public const UInt16 Ixud_PnPDriverErr           = 2;//Plug & Play error
public const UInt16 Ixud_DriverNoOpen           = 3;//The driver was not open
public const UInt16 Ixud_GetDriverVersionErr    = 4;//Recieve driver version error
public const UInt16 Ixud_ExceedBoardNumber      = 5;//Board number error
public const UInt16 Ixud_FindBoardErr           = 6;//No board found
public const UInt16 Ixud_BoardMappingErr        = 7;//Board Mapping error
public const UInt16 Ixud_DIOModesErr            = 8;//Digital input/output mode setting error
```

Abbildung 9 Return Code

[14] Vgl.: LINQ Erklärung

- **Ixud_DriverClose**
  Diese Funktion gibt die Ressource dem System wieder frei. Sie unterbricht den Treiber. Diese Methode soll jedes Mal, nach einem Aufruf einer anderen Funktion, auf die Relaiskarte ausgeführt werden.[15]

Raspbian

Raspbian ist ein Debian basiertes Betriebssystem. Es ist für den Raspberry Pi entwickelt worden und wird auch von Raspberry Pi Foundation als primäres Betriebssystem angesehen.

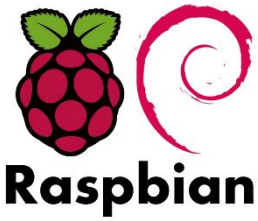


Abbildung 10 Raspbian – Raspberry Pi Logo und Debian Logo

In der Raspbian Version mit PIXEL sind standartmäßig Programme wie LibreOffice, Chromium, Wolfram, Python 3 (IDLE), Scratch und Minecraft Pi installiert. PIXEL, LXDE, XFCE, MATE und Openbox, sind dabei nur eine der Möglichkeiten welches Graphical User Interface (GUI) man verwenden kann.

Weiters gibt es eine Lite Version von Rasbian, in der nur die wichtigsten Elemente des Betriebssystems vorhanden sind. Diese Version besitzt keine GUI. Man kann allerdings diese, sowie viele andere Erweiterungen installieren und so eine Plattform zu erstellen, welche nur das beinhaltet, was später auch wirklich benötigt wird.[16]

WebSQL - SQLite

SQLite ist eine Web-API, welche es Browsern ermöglicht, Daten in einer Datenbank mittels SQL-Querys zu erstellen und verwalten. Die Speicherung und Datenmanipulation wird dabei im JavaScript durch fast alle SQL üblichen Sprachbefehlen ermöglicht.

Abbildung 11 SQLite Logo

WebSQL wird von Browsern wie Google Chrome, Opera, Safari und Android unterstützt. Von WebSQL spricht man dann, wenn SQLite direkt vom Browser Implementiert wird und ein Client einen direkten Zugriff auf SQLite hat, ohne dazu ein Plug-In zu installieren.

Auch wenn es nicht notwendig ist, kann man eine SQLite Plug-In dennoch „händisch“ implementieren. Diese wird dann vom Browsern wie Chromium automatisch als WebSQL erkannt und Chromium gewährt im Entwicklungsfenster direkten Einblick in die SQLite Datenbank. Durch die händische Implementierung mithilfe eines SQLite-API File kann SQLite in JavaScript Code unabhängig von Browser ausgeführt werden, einzig und allein die Manager Umgebung mit Einblick in die Datenbank über den Browser wird nicht überall unterstützt. [17]

[15] Vgl.: UniDAQ Klasse Erklärung
[16] Vgl.: Raspbian Erklärung
[17] Vgl.: WebSQL Erklärung

Chromium

Chromium ist ein Opensource Browser, welche durch die BSD-Lizenz frei für jeden zugänglich ist. Dabei basiert es auf Teilen von Quelltext des Chrome Browsers, welche von Google Inc. (jetzt Alphabet Inc.) als Open Source Projekt zur Verfügung gestellt wurde.

Abbildung 12 Chromium Logo

Bei Chromium gibt es, wie auch bei Google Chrome, den Kiosk-Mode, welcher die Website im Vollbildmodus anzeigt und die Titel- und Suchleiste ausblendet.

Allerdings unterstützt Google Chrome einige Features welche Chromium nur teils durch separate Pakete und teils gar nicht unterstützt:

- Google Logo sowie Namen
- eine automatische Updatefunktion
- kMedia Codecs wie MP3, AAC und H.264
- PPAPI Version des Adobe Flash Players
- eine opt-in Option, welche Google ihre Statistiken und Crash Reports sendet
- RLZ identifier[18]

Python

Python ist eine Programmiersprache, welche sowohl objektorientierte, aspektorientierte und funktionale Programmierung unterstützt.

Abbildung 13 Python Logo

Sie wird oft als Skriptsprache genutzt und will einen gut lesbaren und knappen Programmierstil fördern und gilt als einfach zu erlernen. Dies wird unteranderem durch relativ wenig Schlüsselwörter und eine Syntax, welche auf die Übersichtlichkeit optimiert ist, gefördert. Dadurch lassen sich Python-basierte Skripte deutlich knapper formulieren als in anderen Sprachen.[19]

HTML – CSS /JavaScript

HTML (Hypertext Markup Language) ist das Grundformat des World Wide Web und wird bei Webseiten in Kombination mit CSS und JS eingesetzt. Wie der Name schon sagt, handelt es sich hierbei um eine Auszeichnungssprache. Sie wird mithilfe eines Web-Browsers angezeigt. Weiterentwickelt wird die Sprache vom W3C (World Wide Web Consortium) und der WHATWG (Web Hypertext Application Technology Working Group).

Abbildung 14 HTML - CSS – JS Logos

[18] Vgl.: Chromium Erklärung
[19] Vgl.: Python Erklärung

HTML ist aktuell in der Version 5.1 (1. November 2016) und wird oftmals mit den Erweiterungen HTML5 und XHTML verwendet. Bei einer Website gibt HTML die Grundstruktur der Elemente an, sagt allerdings nichts über ihr späteres, visuelles Aussehen aus.
Dies übernimmt CSS (Cascading Style Sheets). Mit dieser Stilsprache werden die Elemente des HTML grafisch beschrieben. Damit wird z.B.: Farbe, Höhe, Breite und Hintergrund eines Objektes definiert. Mit ihr können sogar Effekte erzeugt und Elemente ausgeblendet werden.
JavaScript übernimmt schlussendlich die Funktionalität des Systems. Es ist eine Skriptsprache, für die es bereits zahlreiche Frameworks gibt.[20]

JS Frameworks

Bei JavaScript gibt es, wie für alle Programmiersprachen, viele Wege ans Ziel. Damit diese allerdings kompakt und nachvollziehbar sind, bedarf es an verschiedenste Framework. Im konkreten Fall wurde bei dieser Diplomarbeit am Fronten die Frameworks AngularJS und jQuery verwendet.

jQuery & SignalR

jQuery ist das meistverwendete JS Framework auf der Welt. Dabei stellt es Funktionen zur Verbesserung und Vereinfachung des Codes zur Verfügung.

Abbildung 15 jQuery Logo

Die Funktionen sind:

- Elementselektion im Document Object Model über die Sizzle Selector Engine, die wetgehend den CSS-3Sektoren entspricht
- Document-Object-Model-Manipulation
- Erweitertes Event-System
- Hilfsfunktionen wie zum Beispiel die each-Funktion
- Animationen und Effekte
- Ajax-Funktionalitäten
- Erweiterbarkeit durch zahlreiche freie Plug-ins, etwa jQuery UI für die einheitliche Gestaltung von Benutzeroberfläche[21]

Weiters gibt es auf GIT-Hub eine SignalR Bibliothek, welches jQuery als Basis nimmt. Diese Bibliothek ermöglicht es SignalR in einen JS Frontend ohne ASP.NET zu verwenden.

[20] Vgl.: HTML – CSS /JavaScript Erklärung
[21] Vgl.: JQuery Erklärung

AngularJS

AngularJS, oft einfach als Angular bezeichnet, ist ein clientseitiges JavaScript-Webframework zur Erstellung von Single-Page-Webanwendungen nach einem MVW (Model-View-Whatever works for you) Muster. MVW steht dafür, wie der Name schon sagt, für MVC, MVVM oder MVP. Je nachdem, was für die einzelne Applikation geeigneter ist.

Abbildung 16 AngularJS Logo

Sie wurde von Google Inc. als Open-Source-Framework unter der MIT-Lizenz entwickelt und freigegeben. Dabei unterstützt sie die Sprache JavaScript und seit Version Angular 2 auch TypeScript. Dabei ist. Die Version Angular 2 ist dabei nicht rückwärts kompatibel zur Version 1.

In Angular gibt es einige Erneuerungen. Eine davon ist die Syntax Scope, welche sowohl an HTML, als auch an JavaScript gebunden werden kann. Sie stellt damit eine direkte Bindung zwischen den beiden Komponenten dar.[22]

Hardware

Raspberry Pi

Der Raspberry Pi ist ein Einplatinencomputer und ist besonders für seine flexiblen Einsatzmöglichkeiten bekannt. Er wurde von der britischen Wohltätigkeitsorganisation „Raspberry Pi Foundation“ entwickelt, um jungen Schülern und Studenten beim Experimentieren mit Software, sowie Hardware zu fördern.

Das Model Raspberry Pi 3B ist deshalb ab 40,80 € schon erhältlich und besteht ausfolgenden Hardware-Komponenten:

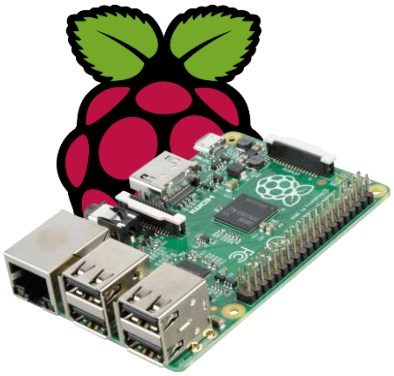


Abbildung 17 Raspberry Pi Logo mit Raspberry Pi Platine

CPU	ARM Cortex-A53 quad-core CPU mit 1.2 GHz
GPU	Brodcom VideoCore IV
RAM	1 GB (mit GPU gemeinsam)
Internet	10/100 Mbit/s Ethernet, Broadcom 2,4 GHz WLAN
Strom	Micro-USB-Anschluss
Sonstige Anschlüsse	4 USB 2.0; microSD-Kartenleser, 3,5-mm-Klinkenbuchse, HDMI-Buchse,40 Pins, CSI(Camera Serial Interface), DSI (Display Serial Interface)[23]

[22] Vgl.: AngularJS Erklärung
[23] Vgl.: Raspberry Pi Erklärung

Relaiskarte

Dazu sollte man zuerst einmal den Begriff Relais beschreiben. Ein Relais ist ein durch elektrischen Strom betriebener, elektromagnetisch wirkender, fernbetätigter Schalter mit, in der Regel, zwei Schaltstellungen. Das Relais wird über einen Steuerstromkreis aktiviert und kann weitere Stromkreise schalten.[24]

Eine Relaiskarte beinhaltet genau solch ein Relais und so eine Karte ist in diesem Fall im Server eingebaut. Weiters kann man mit verschiedenen Programmiersprachen darauf zugreifen. Das bedeutet man kann in C# den Befehl geben, dass ein Strom aus einem bestimmten Port fliesen soll. Diese Funktion ermöglicht es die Heizung simpel zu steuern.

Architektur

Da nun die technischen Grundlagen bekannt sind, wird in diesem Kapitel die Anwendung genau beschrieben. Wie man in der Abbildung 18 bereits entnehmen kann gibt es zwei Aufgaben die der Server zu verrichten hat. Er muss den Raspberries, welche in beliebiger Anzahl vorkommen können, Kurse zur Verfügung stellen, welche die Kurse auf den TV's anzeigen. Außerdem muss er den Windows Service benachrichtigen, sobald neue Daten zur Verfügung stehen.

Dadurch entstehen die drei Kernpunkte dieser Arbeit

- Der Server
- Die Raspberry Pi Clients
- Der Heizungsservice

Diese drei Kernpunkte werden in den nachfolgenden Abschnitten genauer beschrieben.

[24] Relaiskarte Erklärung

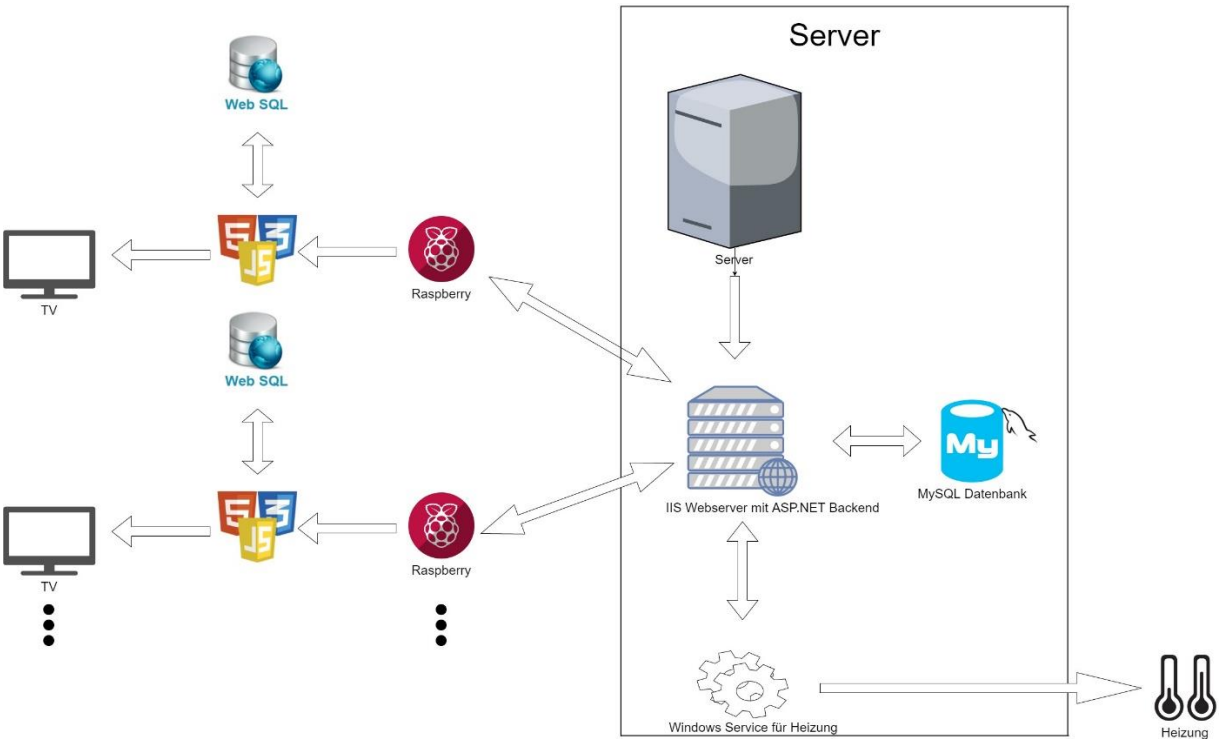


Abbildung 18 Übersicht

Backend

Wird die Anwendung das erste Mal verwendet so werden die Kurse, die von einem anderen System in Form einer Textdatei bereitgestellt werden, im Backend eingelesen. Es werden alle Kurse angelegt und wenn ein Kurs in einem Raum ist, der noch nicht angelegt ist, wird dieser Raum automatisch erstellt. Dies sieht dann so aus wie in Abbildung 19 zu sehen ist.

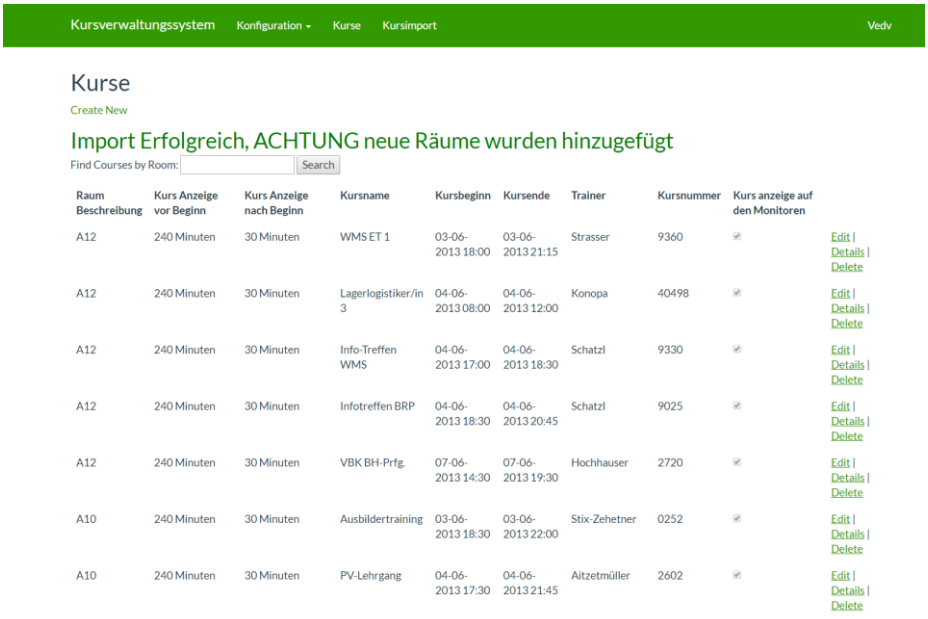


Abbildung 19 BackendImport.png

Jedes Mal, wenn eine Textdatei bestehend aus Kursen hochgeladen wird, werden diese eingetragen und alle alten Kurse aus der Datenbank und somit auch aus dem Backend entfernt.

Den Räumen muss man allen eine Zuweisung auf die Relais geben. Die Relais geben dabei Auskunft darüber, wann der Raum geheizt werden soll. In Abbildung 19 ist zu erkennen, dass der Raum A12, der in Abbildung 20 durch das Einfügen der Kurse erstellt worden ist, der Relais Port fünf zugewiesen wurde. Außerdem weiß man auch, auf welcher Karte sich der Port befindet. In diesem Fall ist er auf der Relaiskarte mit 16 Slots zu finden.

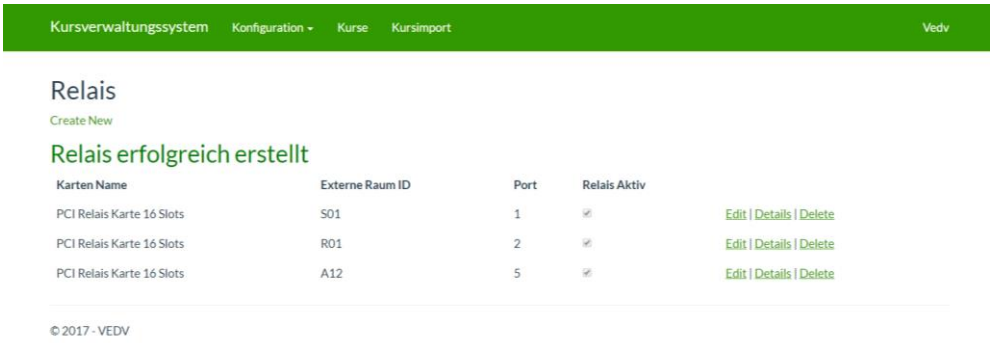


Abbildung 20 Relaiserstellen.png

Meldet sich ein Raspberry Pi beim Server das erste Mal über SignalR, so wird er in die Raspberry Tabelle eingetragen. Dadurch ist es möglich dem Raspberry Pi eine Werbung und einen Bildschirm zuzuordnen.

Abbildung 21 Raspberries.png

In Abbildung 21 erkennt man das es keine Möglichkeit gibt die Raspberries über das Backend einzutragen, da sie sich selbst beim Backend melden und dadurch eingetragen werden, falls sie noch nicht existieren. Außerdem sieht man in diesem Screenshot sehr schön, dass mehrere Raspberries dieselbe Monitor Nummer haben können. Das heißt, dass zwei Raspberries dieselben Kurse anzeigen können. Außerdem ist es möglich für jeden Raspberry Pi die Werbung zu spezifizieren die er anzeigen soll.

Die Werbung kann man über das Backend hochladen. Die Werbung selbst besteht aus JPEG Dateien. Nachdem die Bilddateien hochgeladen wurden, werden alle alten Dateien aus dem Verzeichnis gelöscht. Außerdem werden alle Raspberries über die Änderungen informiert, damit sie, falls sie von der Änderung betroffen sind, die Werbung aktualisieren.

Über die Konfiguration im Backend kann man Werte einsehen oder auch speziell anpassen.

Heizungsservice

Der Heizungsservice ist ein Windows Service bei dem es darum geht, dass bestimmte Ports, die jeweils für einen Raum im Gebäude stehen, auf einem Relais zu einer bestimmten Zeit eingeschalten bzw. ausgeschalten werden, sodass es automatisch bei Kursen in den jeweiligen Räumen heizt, um somit Strom und dadurch Geld zu sparen.

Zuerst baut der Windows Service eine Verbindung zur Datenbank mittels Entity Framework auf. Mit LINQ werden die jeweils benötigten Daten abgefragt. Nun kann der Heizungsservice kontrollieren welcher Raum bzw. Räume zuerst geheizt werden sollen. Dafür werden in einer Schleife alle Kurse, die noch nicht angefangen haben durchgelaufen und mit einem *JOIN* die „Room“ Tabelle mit der „Heat_Room“ Tabelle verbunden. Hierbei werden einfach alle Anfangszeiten und Endzeiten untereinander verglichen und jeweils die nächste Zeit gespeichert. Außerdem werden alle Ports inklusive der relevanten Relaiskartenummer, die zu diesen Zeiten eingeschalten bzw. ausgeschalten werden sollen, in Listen gespeichert.

Es werden zwei Timer erstellt und zu den vorher erwähnten Zeiten gestellt. Diese aktivieren oder deaktivieren die gespeicherten Ports zu den gespeicherten Zeiten, durch die UniDAQ Klasse.

Bei Änderung auf der Datenbank durch das Administrator Interface, gibt die ASP.NET Applikation den Windows Service über SignalR Bescheid, dass sich die Daten auf der Datenbank geändert wurden. Der Service reagiert darauf sofort und wiederholt den Vorgang wieder, wo er die frühesten Zeiten mit den relevanten Ports und Relaiskartenummern speichert.

Entity-Relationship-Modell

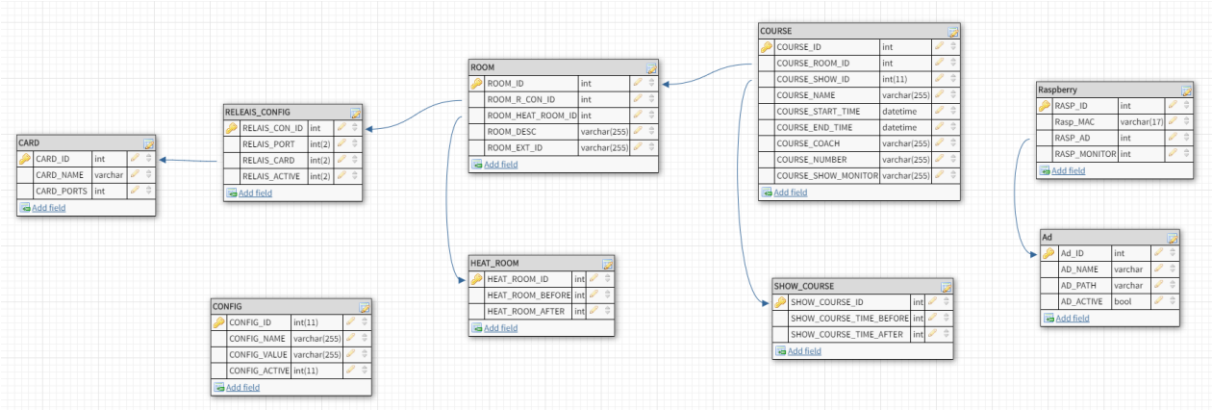


Abbildung 22 ERD

Course (Kurs)

In Kurs wird festgehalten, welcher Trainer welchen Kurs zu welcher Zeit hält. Außerdem wird hier auch entschieden in welchem Raum sich ein Kurs befindet und über eine eigene Tabelle wird beschrieben wann die Kurse an den Bildschirmen angezeigt werden sollen.

Show_Course

Hier werden die Zeiten in Minuten eingetragen. Dies gibt Auskunft um wie viel Minuten vor und nach einem Kurs, dieser auf den Monitoren angezeigt wird. Dafür wurde eine eigene Tabelle erstellt, um Redundanzen zu vermeiden.

Room (Raum)

Ein Raum besteht aus einer externen Raum ID und einer Beschreibung von den jeweiligen Raum. Unter der externen Raum ID versteht man die ID, die der Raum in dem jeweiligen externen System hat. Im Beispiel 7 ist der Zusammenhang zu erkennen.

Raum Beschreibung	Externe Raum ID
Saal	S01
Raum	R01

Beispiel 7 ERD - Room

Heat_Room

Es muss festgehalten werden um wie viele Minuten vor– und nach einem Kurs ein Raum schon zu heizen beginnen soll. Das wird in dieser Tabelle angelegt. Es steht deshalb in einer eigenen Tabelle, weil viele Räume um die gleiche Anzahl von Minuten vor bzw. nach einem Kurs heizen müssen. Mit dieser Art der Speicherung wird Redundanz verhindert.

Relaise_Config (Relaise Konfiguration)

Diese Tabelle bezieht sich auf die Tabelle Room und Card. Sie bestimmt welcher Raum welchen Port auf welcher Relaiskarte bekommt und ob dieser aktiv sein soll.

Card (Karte)

Hier werden alle Relaiskarten gespeichert. Dazu gehört immer der Name der Relaiskarte mit der dazugehörenden Anzahl von Ports.

Raspberry

In Raspberry wird die MAC-Adresse des jeweiligen Raspberry Pies gespeichert. Zusätzlich wird auch der Monitor Nummer gespeichert, die festlegt, welche Kurse angezeigt werden sollen. Es wird auch festgelegt, welche Werbung es anzeigen soll, falls keine Kurse mehr am Bildschirm angezeigt werden. Dies wird aber genauer in der Tabelle Ad beschrieben.

Ad (Werbung)

In dieser Tabelle wird der Pfad von der zu anzeigenden Werbung gespeichert. Zusätzlich soll man es durch ein Attribut „Ad_Active“ markieren können, ob eine Werbung auch eingeschaltet werden soll. Eine Werbung kann man einen oder mehreren Raspberry Pies zuteilen.

Config (Konfiguration)

Es ist eine Tabelle die alle wichtigen Konfigurationsdateien umfassen. Sie beinhaltet die Archiv Ordner zum Importieren der Kurse, der Werbung und die Default Anzeigezeit vor und nach einem Kurs.

MySQL Server

Dieses ERD wurde mittels Entity Framework Code-First in einer MySQL Datenbank realisiert. Der Auftraggeber verlangte, dass die MySQL Datenbank verwendet wird. Das Diplomarbeitsteam arbeitete davor nur selten bis kaum damit, aber die Syntax davon war gleichartig als andere SQL Sprachen, mit denen schon im Unterricht gearbeitet wurde.

Frontend

Das Frontend wird mithilfe der nachfolgenden Grafik erklärt:

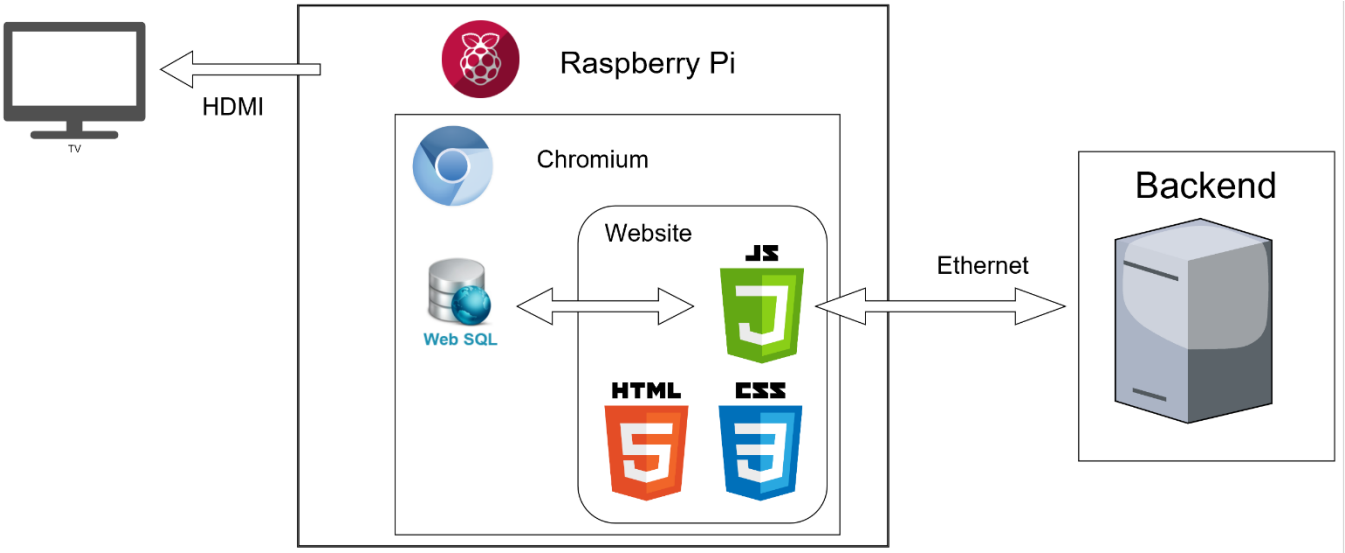


Abbildung 23 Frontend

Das Frontend besteht aus zwei Hardware Komponenten:

- einem Raspberry Pi und
- einem beliebigen Full HD Bildschirm.

Der Raspberry Pi dient als Schnittstelle zwischen dem Server und einem Bildschirm.

Sobald er eingeschaltet wird, startet Rasbian inkl. grafischer Benutzeroberfläche, wobei diese wiederum in der Autostart einen Eintrag hat, wonach eine lokale Website automatisch im Chromium gestartet wird. Mittels der Kiosk Funktion des Chromium, erscheint nun eine Website im Vollbildmodus, welche live auf den Bildschirm über die HDMI Schnittstelle ausgegeben wird.

Web Seite

Die Web Seite bildet das Kernstück des Frontends. Diese besteht, wie man in Abbildung 24 sieht, aus einer Index.html, den dazugehörigen Bibliotheken (lib), den ausprogrammierten JavaScript Files (js), dem Stylsheet (css) und Konfigurations Dateien (config).

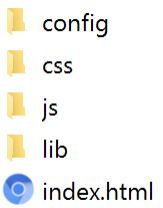


Abbildung 24 Web Seite - Verzeichnis

Nun wird die Bedeutung der Ordner und Dateien hinsichtlich der Model-View-Controller (MVC) Architektur noch genauer erläutert.

Modell
Das Model bildet die zwei Dateien des config Verzeichnis, config.txt und mac.txt, sowie die WebSQL Datenbank.

View
Die index.html gibt Auskunft über die Struktur der Darstellung. Die raspy.css im Ordner css, formatiert die Struktur und bildet somit mit der index.html die View.

Controller
Der js Ordner besteht aus drei Files:

- model.js
- controller.js
- services.js

Diese drei Files repräsentieren den Controller bei der MVC Architektur.
Im model.js wird dabei zuerst die Angular Applikation initialisiert, danach die Werte aus dem config-Files initialisiert.

Die services.js bietet Funktionen für die Kommunikation zum Server und die controller.js übernimmt die restliche Funktionalität.

WebSQL Datenbank
Die Datenbank beim Frontend besteht aus drei Tabellen:

- Course
- Image
- sqlite_sequence

Course

Die Course Tabelle speichert alle Informationen zu den Kursen (siehe Abbildung 25). Dieses sind Großteiles gleich zu den Attributen der Course Tabelle auf Seiten der Server DB, allerdings wird die lokale DB redundant hinsichtlich der Uhrzeit gespeichert.

Course		
id	integer	
course_id	integer	
room_id	string	
show_course_id	integer	
course_name	integer	
course_name	string	
course_start_time	date	
course_end_time	date	
course_beginn	integer	
course_end	integer	
course_coach	string	
course_number	integer	
course_show_monitor	boolean	

Abbildung 25 WebSQL - Course

Image

Die Tabelle Image besteht (siehe Abbildung 26) aus einer id und einem data Eintrag und den dazugehörigen Data. Die Data bestehen dabei aus einen base64 String des Bildes.

Image		
id	integer	
data	string	

Abbildung 26 WebSQL - Image

Endgültiges Ergebnis

Die Diplomarbeit umfasst alle Funktionen, die zu Beginn vorgegeben waren.

Es ist möglich die Kurse aus einem Textfile zu importieren und die Räume, die es noch nicht gibt, werden automatisch angelegt. Um die Räume einem Relais Port für die Heizung zuzuweisen, kann man beliebig viele Relaiskarten mit beliebiger Anzahl an Ports für das System anlegen. Der Benutzer ist dadurch in der Lage, die Räume den Ports auf der Relaiskarte zuzuweisen. Außerdem kann man für jeden Raum die Heizzeiten vor bzw. nach den Kursen festlegen.

Im Backend ist es ebenfalls möglich den angemeldeten Raspberries Werbung zuzuweisen. Der Benutzer kann beliebig viele Variationen von Werbung, bestehend aus einem oder mehreren Bildern, anlegen und eine dieser Variationen einem Raspberry Pi zuordnen. Dieser zeigt die Werbung dann automatisch in Form einer Diashow an, wenn ihm keine Kurse zugewiesen sind. Es ist auch möglich das zwei oder mehrere Raspberry Pies dieselbe Werbung anzeigen oder auch dieselben Kurse.

Durch die Konfigurationswerte im Backend kann der Administrator die Standardwerte für die Heizung vor und nach Beginn einstellen, als auch die standardmäßige Anzeigezeit der Kurse vor und nach Beginn. Dies hat den Vorteil, dass sowohl beim Importieren diese Werte benutzt werden sowie beim händischen Erstellen eines Kurses bzw. eines Raumes.

Durch den Einsatz von SignalR werden sowohl die Raspberries, als auch der Heizungsservice über Änderungen in den Kursen innerhalb von Sekunden informiert. Dies funktioniert über die eindeutige MAC-Adresse der Raspberries.

Die Raspberry Pies persistieren alle Daten lokal, sodass sie auch mit einer unterbrochenen Internetverbindung noch angezeigt werden.

Die Kurse werden ordnungsgemäß im Zeitraum zwischen definierten Anzeigebeginn und Anzeigeende des jeweiligen Kurses angezeigt, unabhängig ob das Gerät online oder offline ist. Voraussetzung dafür ist einzig und allein das Verbinden mit dem Netzwerk beim Starten des Frontend.

Das Design und die Usability wurden entsprechend den Wünschen, des Kunden angepasst.

In Abbildung 27 ist ein Beispiel dafür, wie das Frontend mit den Kursen aussieht.

		04.04.2017	10:43 Uhr	
Beginn	Kurs Nr.	Veranstaltung	Trainer	Raum
10:45	9360	WMS ET 1	Strasser	A01
10:45	9051	BRP Mathematik	Piwerka	A21
11:00	40498	Lagerlogistiker/in 3	Konopa	A33
11:00	9330	Info-Treffen WMS	Schatzl	B22
11:00	9025	Infotreffen BRP	Huber	B5
11:00	2720	VBK BH-Prfg.	Hochhauser	A12
11:30	0252	Ausbildertraining	Baumgartner	A32
12:00	2602	PV-Lehrgang	Aitzetmüller	A10
13:00	9041	BRP Englisch	Bräuml	B11
13:00	0252	Ausbildertraining	Stix-Zehetner	A01

Abbildung 27 Endgültiges Ergebnis - Frontend

Implementierung

Unter diesem Abschnitt wird anhand von detaillierten Beschreibungen und anhand von Code-Beispielen dargestellt, wie wir unser System umgesetzt haben.

Datenmodell

Vor der Erstellung des Datenmodells wurde mit dem Auftraggeber diskutiert, was das Datenmodell beinhalten soll. Durch das Besprechen der Eigenschaften, ist das erste Datenmodell erstellt worden (siehe Abbildung 28). Dieses war aber zu simpel und hatte noch einiges an Verbesserungspotential. Man konnte zum Beispiel nur Relaiskarten eingeben, die 8 Ports haben und manche Attribute waren nicht sehr effizient.

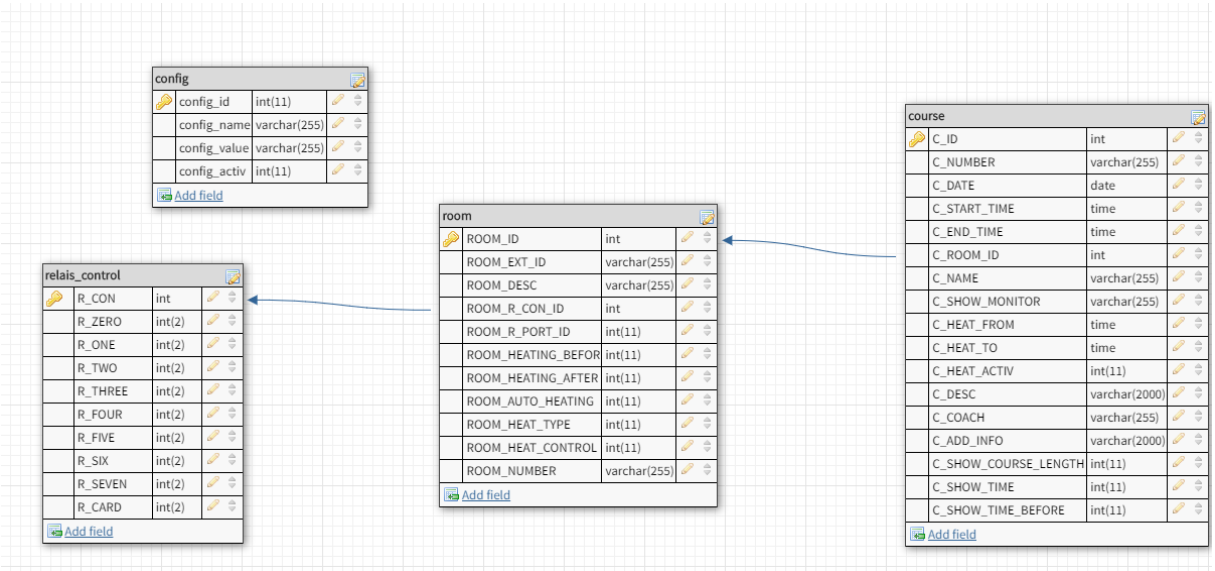


Abbildung 28 Datenmodell v1

Daraus entstand die zweite Version (siehe Abbildung 29). Man konnte zwar Relaiskarten mit 16 Ports eingeben, aber noch keine Karten mit beliebig viele Ports anlegen. Weiters wurden beim Kurs ein Datum, eine Anfangszeit und eine Endzeit verwendet, welche auf zwei Datetimes geändert werden müssen, um einen Platzhalter loszuwerden. Außerdem wurde festgestellt, dass viele Kurse um jeweils dieselbe Zeit vor dem Beginn und nach dem Ende eines Kurses angezeigt werden sollen. Dasselbe gilt auch für das Vor- und Nachheizen der verschiedenen Räume.

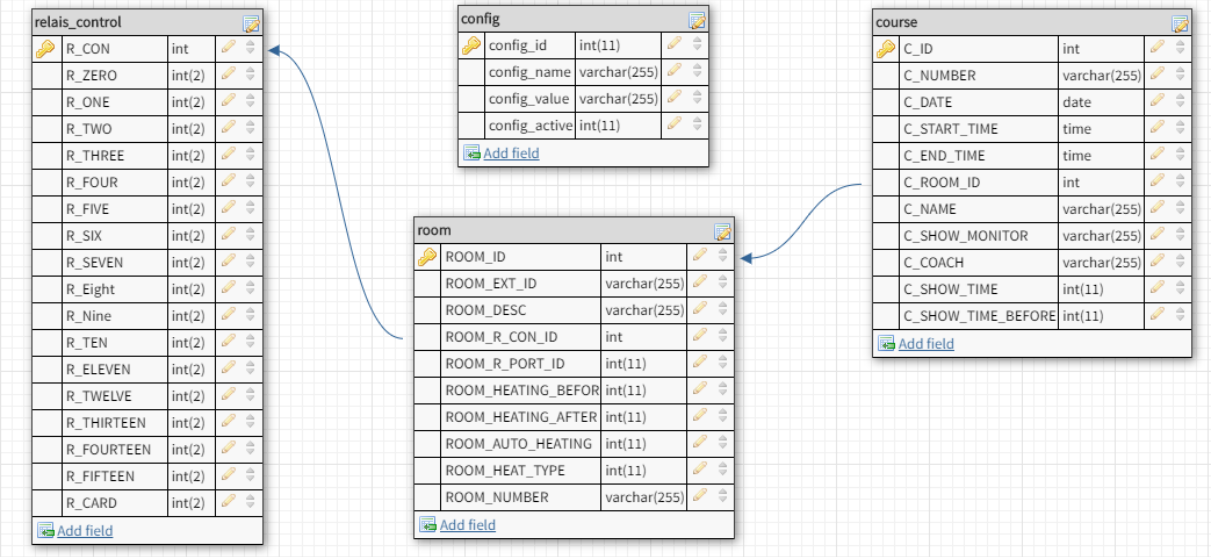


Abbildung 29 Datenmodell v2

Bei der dritten Version (siehe Abbildung 30) wurde das vorhin erwähnte Datum mit der Startzeit und der Endzeit zu zwei Attributen mit den Typen Datetime geändert. Um die redundante Speicherung zu verhindern, wurde das Datenmodell um die Tabellen „HEAT_ROOM“ und „SHOW_COURSE“ erweitert. Darüber hinaus sind die Tabellen „RELAIS_CONFIG“ und „CARD“ hinzugefügt worden, sodass beliebig viele Relaiskarten angelegt werden können, mit jeweils beliebig vielen Ports. Zu diesem Zeitpunkt fiel es auf, dass die Raspberry Pies und die Werbung noch nicht in dem Datenmodell integriert waren.

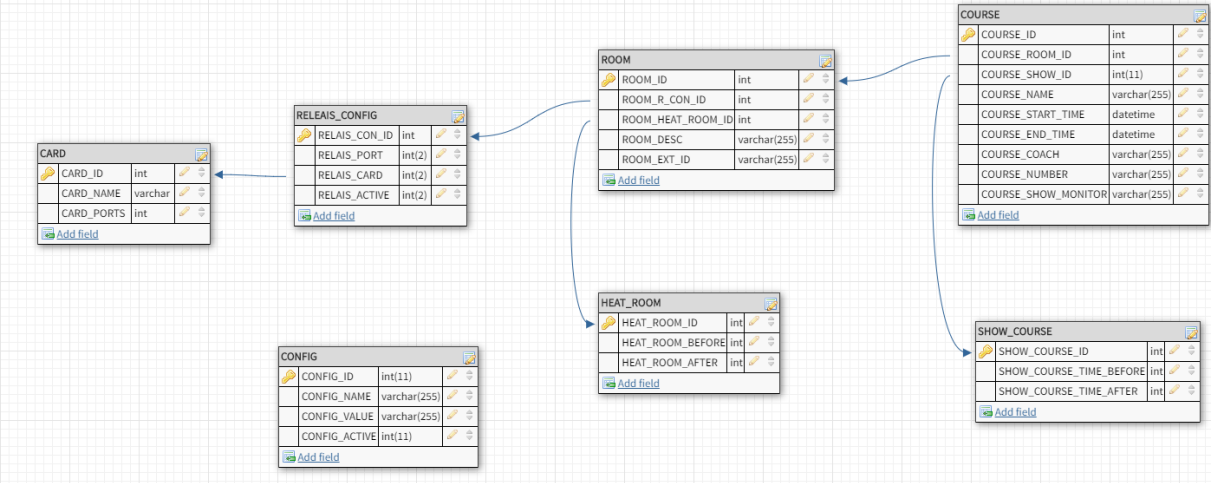


Abbildung 30 Datenmodell v3

Darum wurden bei der Version 4 (siehe Abbildung 31) die zwei Tabellen „RASPBERRY“ und „AD“ hinzugefügt. Jetzt konnte man ohne Problem Raspberry Pies hinzufügen, die durch ihre MAC-Adresse bestimmt werden und einen Monitor zugewiesen bekommen Zusätzlich kann man ihnen Werbung zuteilen, welche ein Attribut „AD_PATH“ haben, indem ein Pfad gespeichert ist, der zu den Werbebilder führt. Alle Bilder in dem angegebenen Ordner werden als Diashow auf dem Monitor angezeigt. Weil mit dieser Version alle Anforderungen erfüllt waren und keine Probleme mehr gefunden wurden, ist dies die finale Version.

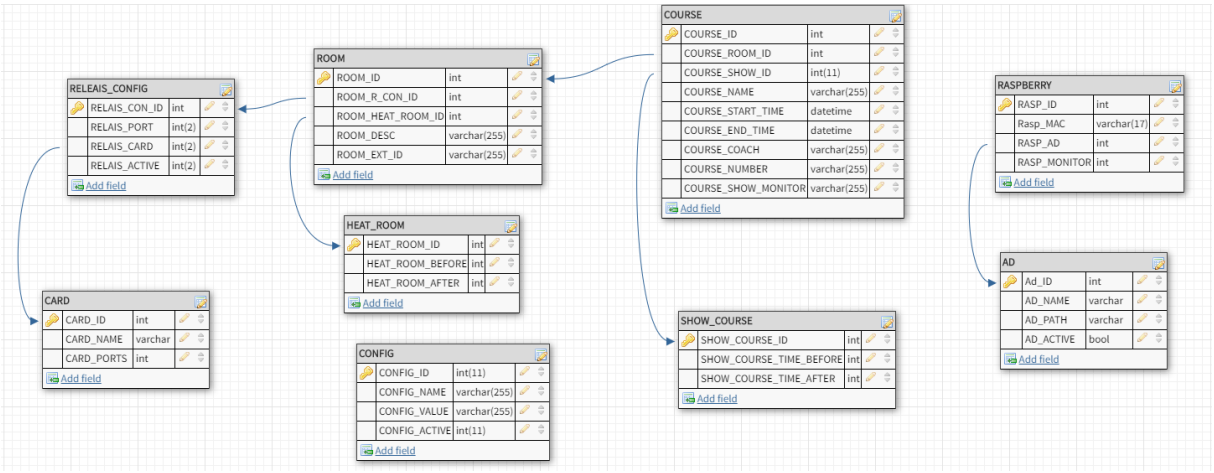


Abbildung 31 Datenmodell v4

Server

Nachdem das Datenmodell fertig war, ist auf der Serverseite entschieden worden, dass der Database First Ansatz verwendet wird. Da das Datenmodell im Nachhinein noch verbessert worden ist, kristallisierte sich bald heraus, dass der Code First Ansatz die bessere Entscheidung war. Einerseits, weil es leichter ist die Models in Visual Studio zu modifizieren, als die Datenbanktabellen ständig zu ändern und andererseits, weil sich die Models nicht neu generieren, wenn man Änderungen vornimmt.

Nachdem einige Fehler, die beim Versuch mit Code First aufgetreten sind, behoben wurden, war nun das Grundgerüst fertig.
Die Tabellen und Relationen in der Datenbank wurden erfolgreich erzeugt. Gleichzeitig sind die Controller und Views für die einzelnen Datensätze wie zum Beispiel Kurse oder Räume generiert worden.

Ein generierter MVC 5 Controller mit Views und Entity Framework beinhaltet vorgefertigte Methoden wie Index, Details, Create, Edit und Delete. Diese Methoden werden über den Browser aufgerufen und verwendet. Sie können entsprechend der gestellten Ansprüche angepasst werden. In der folgenden Abbildung wurde die Methode so umfunktioniert, dass man einen Suchtext übergeben kann, der als Filter zum Suchen von Kursen in bestimmten Räumen verwendet wird.

```
public ActionResult Index(string searchString)
{


    if (TempData["errorMsg"] != null)
    {
        ViewBag.ErrorMsg = TempData["errorMsg"].ToString();
    }

    var courses = db.Courses.Include(c => c.Room).Include(c => c.Show_Course);

    if (!String.IsNullOrEmpty(searchString))
    {
        courses = courses.Where(s => s.Room.room_desc.Contains(searchString));
    }

    return View(courses.ToList());
}
```

Abbildung 32 Kurs Index Methode

Außerdem wurde noch eine Fehler- bzw. Erfolgsmeldung eingebaut. Diese sagt aus, ob das Importieren von Kursen erfolgreich war.

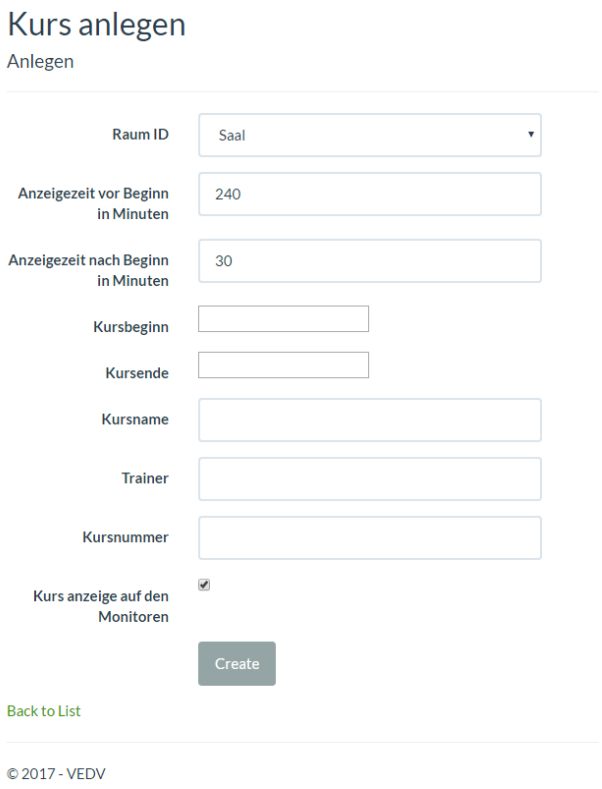


Abbildung 33 Kurse.png

In Abbildung 33 ist zu sehen wie die generierte View aussieht. Einige Verschönerungen der Views sind erst später vorgenommen worden. Dies gilt auch für die weiteren Abbildungen in diesem Kapitel.

Erwähnenswert ist noch, dass hier die View ursprünglich aus dem originalen Model erzeugt wurde, allerdings konnte man dies nicht verwenden, weil anstelle der „show_course_id“ die Werte Anzeigezeit vor und nach Beginn angezeigt werden. Da sich die „show_course_id“ aus den vor und nach Beginn Werten zusammensetzt, wird nicht nach der ID gefragt, die man einfügen möchte, sondern nach den besagten Werten. Dies hat den Vorteil, dass der Benutzer nicht nachsehen muss, welche Werte hinter den IDs sind. Es werden automatisch neue Einträge in der „Show_Course“ Tabelle generiert, wenn der Benutzer eine Wertekombination einträgt die noch nicht vorhanden ist.

Der Nachteil von dieser Methode ist, dass man die Werte für jeden Eintrag mit der Datenbank abgleichen muss und das Viewmodel wird immer zu dem ursprünglichen Model zurück konvertiert, um dies dann erfolgreich mit Entity Framework in die Datenbank abzulegen.

Dasselbe Prinzip wurde auch bei den Heizzeiten der Räume angewandt.

Die nächste Aufgabe war es, den Kursimport zu programmieren. Hier galt es ein Textfile, welches von einem anderen System erzeugt wird, einzulesen. Die Werte werden nach dem Einlesen in das richtige Format gebracht um sie zu persistieren. Falls Räume vorkommen, die noch nicht im System eingetragen sind, werden diese ebenfalls erzeugt und der Benutzer bekommt nach dem Einfügen einen Hinweis, dass neue Räume eingefügt wurden. Werden die Räume auf diese Weise eingefügt, so wird für die Heizzeiten der Standardwert in der Konfigurationstabelle verwendet, falls dieser Wert auf aktiv gesetzt ist.

```
130607;14.30;2720 ;A12 ;VBK BH-Prfg.          ;19.30;;-1;Hochhauser
130606;17.30;2602 ;A10 ;PV-Lehrgang           ;21.45;;-1;Aitzetmüller
130604;17.30;2602 ;A10 ;PV-Lehrgang           ;21.45;;-1;Aitzetmüller
130605;18.00;9041 ;B11 ;BRP Englisch          ;22.00;;-1;Bräuml
130604;18.00;9041 ;A20 ;BRP Englisch          ;22.00;;-1;Gumpetsberger
```

Abbildung 34 Beispiel Text Datei

In Abbildung 34 ist ein kleiner Auszug dieses Textfiles zu sehen. Die Werte die zusehen sind, werden in ein richtiges Format gebracht, um die Daten persistieren zu können.

Jetzt wurden die Verschönerungen im Backend vorgenommen. Das Bootstrap Theme „flatly“ wurde eingebaut und farblich verändert. Außerdem wurden einige Verbesserungen in Bezug auf die Usability vorgenommen. Dafür wurden Dropdowns für einige Werte eingebaut, um den Benutzer vor einer falschen Eingabe zu schützen. Dazu wurde noch eine Validierung der Werte eingebaut. Außerdem wurden Kalender, zum Auswählen des Datums bzw. der Zeit, in die Datumsfelder integriert. Zuletzt sind Erfolgs- und Fehlermeldungen bei verschiedensten Aktionen eingebaut worden und im Zuge dessen auch Redirects, um einen flüssigen Ablauf der Anwendung zu gewährleisten.

Bei den Relais wurde noch eine Auswahl für die Relaiskarte und den Raum eingebaut. Es werden nur noch die Räume angezeigt, die noch keine Zuordnung zu einem Relais haben. Wenn dies geschehen ist, werden die noch verfügbaren Ports in einem Dropdown angezeigt.

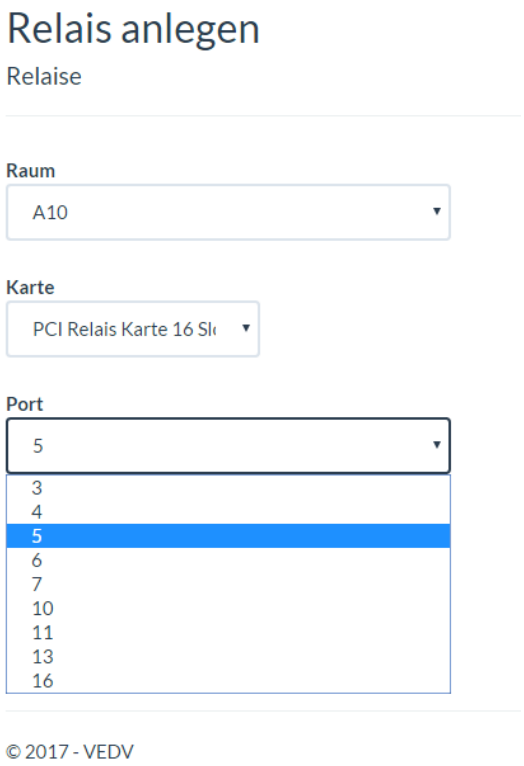


Abbildung 35 Relais Anlegen

In Abbildung 35 sieht man, dass auf der „PCI Relais Karte 16 Slots“ nur noch 9 Ports zur Verfügung stehen, da bereits 7 Räume eine Zuordnung auf dieser Relaiskarte haben.

Um die richtige Port Anzahl passend zu der ausgewählten Karte anzuzeigen, war es nötig JavaScript zu verwenden, weil es die einfachste Lösung ist herauszufinden, welche Karte im ersten Dropdown ausgewählt worden ist.

```
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>
<script language="javascript" type="text/javascript">
    function GetPort(_cardid) {
        var procemessage = "<option value='0'> Please wait...</option>";
        $("#port").html(procemessage).show();
        var url = "/Relaises/getPortIdByCard/";

        $.ajax({
            url: url,
            data: { cardid: _cardid },
            cache: false,
            type: "POST",
            success: function (data) {
                var markup = "";
                for (var x = 0; x < data.length; x++) {
                      markup += "<option value=" + data[x].Value + ">" + data[x].Text + "</option>";
                }
                $("#port").html(markup).show();
            },
        });

    }
</script>
```

Abbildung 36 JavaScript Ports für Karte

In Abbildung 36 sieht man wie der JavaScript Code aussieht, um für eine ausgewählte Karte das Dropdown für die Ports zu befüllen.

Nun wurden die Werbung und die Raspberries in Form von Models hinzugefügt und durch Code First erstellt. Es ist erwähnenswert, dass es nicht sofort möglich war, das Model für den WCF Service zu aktualisieren. Es trat wieder dieser Fehler von MySQL auf „*MySQL - Entity : The value for column 'IsPrimaryKey' in table 'TableDetails' is DBNull*“. Dieser ist schon zu Beginn aufgetreten, als das Model das erste Mal generiert hätte werden sollen. Über die Behebung dieses Fehlers gibt es in dem Kapitel “Aufgetretene Probleme“ eine detaillierte Information.

Nachdem der Fehler behoben war, wurde der Upload für die einzelnen Bilddateien eingebaut. Dabei ist zu beachten, dass immer alle Bilddateien auf einmal hochgeladen werden müssen, da alle im Zielordner vorhanden Bilder aus dem Verzeichnis gelöscht werden.

Da jeder Raspberry Pi eine Werbung anzeigen soll, wenn keine Kurse mehr zum Anzeigen da sind, kann man im Backend einstellen, welche Werbung auf welchem Raspberry Pi angezeigt werden soll. In Abbildung 37 sieht man, dass man beliebig viel Werbung anlegen kann.

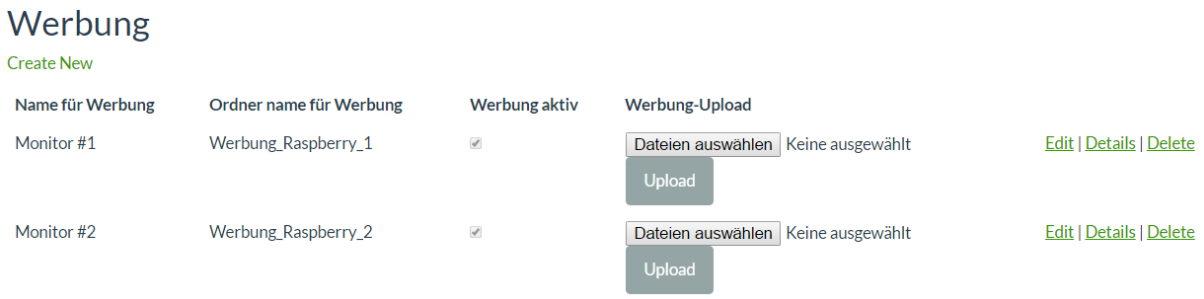


Abbildung 37 Werbung.png

Die Werbung hat einen Namen, um die Zuordnung für den Benutzer leichter zu machen, wie man in Abbildung 38 sehen wird.

Abbildung 38 Raspberries.png

Aus der Abbildung 38 zieht man die Erkenntnis, dass der erste und der dritte Raspberry Pi dieselbe Werbung zugewiesen haben und diese auch anzeigen können.

Jetzt stellt sich nur noch die Frage wie man die Bilddaten, also die Werbung, so überträgt, dass der Client damit arbeiten kann. Die Lösung für dieses Problem war ein Base64 kodierter String, der die Bilddatei abbildet. Der Base64 String wird dadurch generiert, dass man die Bilddatei bzw. die Bilddateien in ein Bytearray konvertiert und dieses dann in einen Base64 String konvertiert. Diese Strings werden dann in einer Liste gespeichert.

```
foreach (FileInfo file in Files)
{
    Image image = Image.FromFile(path + "\\" + file.Name);          // Jedes Einzelne PNG bild aus dem Verzeichnis auslesen
    MemoryStream ms = new MemoryStream();                           // Memory Stream erzeugen
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);        // Image in dem Stream speichern
    byte[] ImageAsByteArray = ms.ToArray();                         // den Stream mit dem Image zu einem byte Array Konvertieren
    ImageAsString = Convert.ToBase64String(ImageAsByteArray);       // Das byte Array in einen base64 encodierten String umwandeln
    ListOfImagesAsBase64String.Add(ImageAsString);                  // Den String in die Liste speichern
}
```

Abbildung 39 Bilder konvertieren

Die Liste, in die die umgewandelten Strings gespeichert werden, wird über JSON serialisiert und an den Client im JSON Format übergeben. Der Client konvertiert die Strings in Bilder zurück und kann diese verwenden. Diese Übertragung findet nicht über den WCF Service statt, sondern über eine Controller Methode, die der Client aufrufen kann.

Da nun alle Funktionen auf dem Server funktionierten, wurde SignalR in die Arbeit eingebaut. Es war sehr leicht SignalR auf dem Server zu integrieren. Allerdings war es schwierig zu testen, da es entweder funktioniert oder nicht und wenn es nicht funktioniert ist es schwierig zu sagen, ob der Fehler auf der Serverseite oder auf der Clientseite ist. Mehr dazu im Unterpunkt „Testen“.

Frontend

Beim Systemstart des Raspberry Pies öffnet sich automatisch Chromium im Kiosk Modus (Vollbildmodus) mit einer index.html.

Bevor allerdings die index.html aufgerufen wird, wird beim Systemstart noch ein kleines Python Programm gestartet, welches die Aufgabe hat, die Mac-Adresse des Gerätes in das mac.txt File zu schreiben, da die Web Seite die Mac-Adresse nicht selbst beziehen kann.

Ablauf

Nun wird die index.html gestartet, welche alle Komponenten deklariert und initialisiert. Die einzelnen Komponenten und ihre gegenseitige Beeinflussung werden nun beschrieben.

model.js

Nachdem die Bibliotheken geladen wurden, wird das model.js File aufgerufen. Dieses deklariert als erstes die Angular Applikation und benennt sie „*VEDVApp*“. Dies ist später in der index.html anzugeben (siehe index.html). Sobald die Applikation nun von der index.html initialisiert wird, werden globale Variablen *($rootScope)* mit den Werten von der mac.txt und der config.txt geladen, für genauere Details zu den beiden Dateien gibt es im Kapitel “Architektur“ genauere Informationen.

```
$rootScope.macAddress;
$rootScope.werbungID;                //ID des BilderPacket am Server
$rootScope.kurseID;                  //Interval der anzuzeigenden Kurse
$rootScope.currentContentShowed;     //Anzeige Kurse oder Werbung
$rootScope.signalRServer;            //Andresse des Servers
```

Beispiel 8 model.js globale Variable

service.js

Die service.js bietet, wie der Name schon vermuten lässt, Services in Form von Angular Factories an. Im konkreten Fall sind es zwei Factory Methoden:

- Factory *HTTPRequest*
- Factory *signalRHubProxy*

Während *HTTPRequest* aus einem simplen http GET Request besteht, wickelt *signalRHubProxy* die Verbindung mit SignalR ab. Dabei wird zuerst eine sogenannte *hubConnection* eingerichtet, welche die Adresse zum Server enthält. Danach wird ein

Proxy (Vertreter) definiert mit den genauen Namen des hubs. Bevor nun die *hubConnection* gestartet wird, wird noch eine Variable *stat* erstellt, welche zuerst *false* und nach den erfolgreichen Verbindungsaufbau *true* wird.
Da nun die Verbindung aufgebaut ist, gibt es einige Methoden, welche man aufrufen kann:

- *on* - Die Proxy Funktion on wird vom Server verwendet, um Methoden des Frontends aufrufen zu können.
- *invoke* – Dient dem Client um Methoden des Backends aufrufen zu können
- *invokeParam* – funktioniert gleich wie *invoke*, nur gibt es einen Übergabe Parameter

Weiters gibt es noch eine *get* und eine *set* Methode von *stat* und eine start und *stopMethode* für den Verbindungsaufbau und Verbindungsabbau.

index.html

Die index.html lädt zuerst die Bibliotheken Angular, jQuery und die jQuery SignalR Komponente. Danach lädt sie die JavaScript Files model.js, services.js und controller.js, in genau dieser Reihenfolge. Zum Abschluss wird noch die raspy.css geladen

Die index.html lädt zuerst alle Dateien ein und initialisiert sie dann mit *ng-app="VEDVApp"*.

Auf weitere Inhalte der index.html, wird im Controller noch eingegangen.

controller.js

Die Controller.js definiert einen Angualr *Controller,* welcher wie die Angular Applikation in der index.html initialisiert wird. Dies geschieht mithilfe von *ng-controller="VEDVController",* wobei *VEDVController* der Name des Controllers ist.
Der Controller enthält verschiedene Übergabeparameter, welche vom Angular Framework und dem services.js stammen.
Konkret werden folgende Funktionen implementiert:

- $scope – sind lokale Variablen, welche auch in der index.html aufgerufen werden können.
- $rootScope – sind globale Variablen, welche von jeder Angular Methode aufgerufen werden können.
- $interval – ist ein Intervall, eine Schleife, welche periodisch sich selbst periodisch aufruft.
- signalRHubProxy
- HTTPRequest

Zu Beginn werden im Controller Variablen festgelegt. Erwähnenswert sind dabei imageHeader, courseHeader, courseShowTimeHeader und db. Die verschiedenen Header definieren den Pfad und eventuelle Parameter der Services, welche später zu einer vollständigen URL zusammengeführt werden. Unter der Variable db wird die WebSQL Verbindung initialisiert.

Nachdem die index.html fertig initialisieren ist, werden im Controllers die aktuellen Kurse und Bilder von der lokalen WebSQL-DB geladen und die Intervalle starten.

In Beispiel 10 sehen wir eine Funktion, welche automatisch die *initDatabase(3),* die *getTableCourse()* und die *getTableImage() Funktion* aufruft, nachdem die index.html vollständig geladen ist.

```
angular.element(document).ready(function () {
    initDatabase(3);
    getTableCourse();
    getTableImage();
});
```

Beispiel 9 controller.js

initDatabase(mode)

Die initDatabase(mode) erstellt die WebSQL Datenbank, wen diese noch nicht existiert.

getTableCourse()

Ist eine Funktion welche mithilfe einer Query, alle Course Daten aus der Datenbank bekommt und diese an getCourse(result) weiterleitet.

```
"SELECT * FROM Course "+
 "WHERE course_show_monitor = 'true' and course_beginn < "   +actualDate.getTime() +
                                  " and course_end >"        +actualDate.getTime() +
 "ORDER BY course_start_time ASC";
```

Beispiel 10 controller.js query

Für die Query (sieht Beispiel 11) wird dabei überprüft, ob die aktuelle Zeit im Anzeigeintervall des Kurses liegt. Dabei liefert *getTime()* die Anzahl der Millisekunden zurück, welche seit 1.1.1970 vergangen sind. Dementsprechend wurde auch der Zeitpunkt von *course_beginn* und *course_ende* in Millisekunden ab 1970 gespeichert.

getCourse(sqlResultSet)

Diese Funktion ladet zuerst aus dem sqlResultSet die Programmpunkte in ein temporäres Array, welches anschließend mithilfe des *slice* Operators an *$scope.programpoints* übergeben wird. Der *slice* Operator dient dabei um nur eine bestimmte Anzahl von Programmpunkte zurückzugeben. Im Falle der Kurse ist dafür *$rootScope.kurseID*1 0-10* bis *$rootScope.kurseID*10* definiert. Dies liefert die Kurse in zehner Blöcken in Abhängigkeit zur *$rootScope.kurseID* zurück.

Die Bindung von $scope.programpoints in der index.html.

Unter der Variable *$scope.programpoints* werden jene Programmpunkte gespeichert, welche auch auf den Bildschirm angezeigt werden.

```
<table class="raspy-table" ng-show="showContent('Kurse')">
    <thead class="raspy-table-head">
        <tr>
            <td id ="td1"></td>
            <td id ="td1"></td>
            <td id ="td3">{{ currentServerTime | serverTimeFormat : 1 | date: 'dd.MM.yyyy' }}</td>
            <td id ="td2">{{ currentServerTime | serverTimeFormat : 2 | date: 'HH:mm' : BST}} Uhr</td>
            <td id ="td1"></td>
        </tr>
        <tr >
            <td>Beginn</td>
            <td>Kurs Nr.</td>
            <td>Veranstaltung</td>
            <td>Trainer</td>
            <td>Raum</td>
        </tr>
    </thead>
    <tbody class="raspy-table-body">
        <tr  ng-repeat="programpoint in programpoints">
            <td>{{programpoint.course_start_time | date: 'HH:mm' }}</td>
            <td>{{programpoint.course_number }}</td>
            <td>{{programpoint.course_name}}</td>
            <td>{{programpoint.course_coach}}</td>
            <td>{{programpoint.room_id}}</td>
        </tr>
    </tbody>
</table>
```

Beispiel 11 index.html Übersicht

In Beispiel 12 sieht man die Definition der Struktur von Kursen, in der index.html.

Class ist wie die *id*, für die Formatierung, welche in der raspy.css definiert ist, wichtig.

ng-show(‘Kurse‘) wird im nachfolgenden Punkt „showContent(string)“ erklärt.

{{ }} – Mithilfe der Geschwungenen Klammern wird eine JS Variable, oder der Rückgabeparameter einer JS Funktion gebunden. Dabei können mithilfe des | Parameter zusätzlich ein Filter bzw. ein Format dazugegeben werden. (Der Filter wird unter der Überschrift „Uhrzeit“ unter Punkt „Controller.js“ näher beschrieben.)

ng-repeat:

ng-repeat hat die gleiche Funktion wie eine ForEach-Schleife. Dies heißt, in diesem Fall wird für jeden Eintrag in der programmpoints Liste ein Object programmpoint erstellt. Programmpoints ist die Variable in dem controller.js File, welche mithilfe von ng-repeat gebunden wird. Anschließend wird danach von jeden programpoint die Inhalte wie *course_start_time* und *course_number* angezeigt.

showContent(string)

ShowContent(string) ist eine Funktion, welche den Übergabeparameter mit der Variable *currenContentShowed* vergleicht und dementsprechend true oder false zurück liefert.

```
<div ng-show="showContent('Bilder')">
    <div id="image">
        <img ng-src="{{ prictureSource }}"></img>
    </div>
</div>
```

```
$scope.showContent = function(Content){
    if(Content == $rootScope.currentContentShowed)
        return true;
    return false;
}
```

Beispiel 12 index.html und controller.js showContent Funktion

In der Beispiel 13 sieht man die Verwendung der *showContent* Funktion in der index.html.
Ng-show bindet dabei das *showContent* auf das div Tag. In diesem Fall wird das div Tag nur dann angezeigt, wenn *currentContentShowed* den String ‘Bilder‘ entspricht.

getTableImage()

Diese Funktion erzeugt ein Variable *imageArray*, in der alle Bilder der Datenbank gespeichert werden. Dabei ist zu beachten, dass jeder Eintrag von *imageArray* ein String ist, welcher aus *""* und den, in der Datenbank gespeicherten, base64 String. Somit kann man den Eintrag direkt in einen img Tag, im Html, unter src gesetzt werden.

Intervalle

Der Controller definiert drei Intervalle:

- manageImage – tauscht die Bilder alle 4 Sekunden aus, sodass eine Diashow entsteht
- signalRConnectionInterval – ruft alle 2 Sekunden die isReady() Funktion auf
- checkCourseInterval – ruft jede Minute *getTableCourse()* auf, damit die Kurse immer nur in ihrer Anzeigezeit, angezeigt werden

isReady()

Sie überprüft ob die aktuelle SignalR Verbindung aufgebaut ist und wenn nicht, wird versucht die Verbindung neu aufzubauen und weiters die Uhrzeit des Betriebssystems angezeigt.

Testen

In diesem Kapitel wird die Vorgehensweise für die Problemfindung der einzelnen beim Testen aufgetretenen Probleme beschrieben.

Server-Client Kommunikation

Bei dem Testen von der Kommunikation zwischen Server und Client sind einige Probleme aufgetreten, die auf den ersten Blick nicht sehr offensichtlich sind.

Für dieses konkrete Problem ist es essentiell zu wissen, was Cross-Origin Requests (CORS) sind. Prinzipiell haben Browser eine Sicherheit eingebaut, die absichert, dass Websites keine AJAX Anfragen an eine andere Domain macht. Diese Absicherung nennt man „same-origin policy“. Dadurch wird verhindert, dass eine bösartige Website sensitive Daten von einer anderen Website lesen kann.

Um die Kommunikation über den WCF Service und SignalR zwischen dem Webserver und den Raspberries herzustellen, muss man das sogenannte Cross-Origin-Request-Sharing (CORS) in der web.config Datei erlauben.

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="Access-Control-Allow-Origin" value="*" />
    </customHeaders>
  </httpProtocol>
```

Abbildung 40 Web.config access control allow origin

In Abbildung 40 sieht man, dass durch die “Access-Control-Allow-Origin” und die Wildcard “*” das CORS auf der Website für alle Domains erlaubt werden. Dadurch wird nun erreicht, dass alle Raspberry Pies Zugriff auf den WCF Service und die SignalR Hubs haben.[25]

```
//map.UseCors(CorsOptions.AllowAll);
```

Abbildung 41 SignalR CORS Einstellung

Erlaubt man allerdings CORS in der Konfiguration für SignalR ein weiteres Mal, so entsteht ein Fehler auf der Clientseite, dass keine Multiplen CORS Header erlaubt sind. Deswegen ist der Code in Abbildung 41 auskommentiert, um diesen Fehler zu vermeiden.

Windows Service

In diesem Abschnitt wird beschrieben, wie bestimmte Teilbereiche vom Windows Service getestet worden sind und welche Folgen es hatte.

Services

Mit der Desktop Applikation Dienste (Services) kann man alle Dienste sehen die auf dem PC installiert sind. Deswegen kann man hier nach dem Installieren des HeizungsServices überprüfen ob der Service läuft. In der Abbildung 42 kann man erkennen, dass der HeizungsService erfolgreich installiert worden ist und ausgeführt wird.

[25] Vgl.: CORS Erklärung

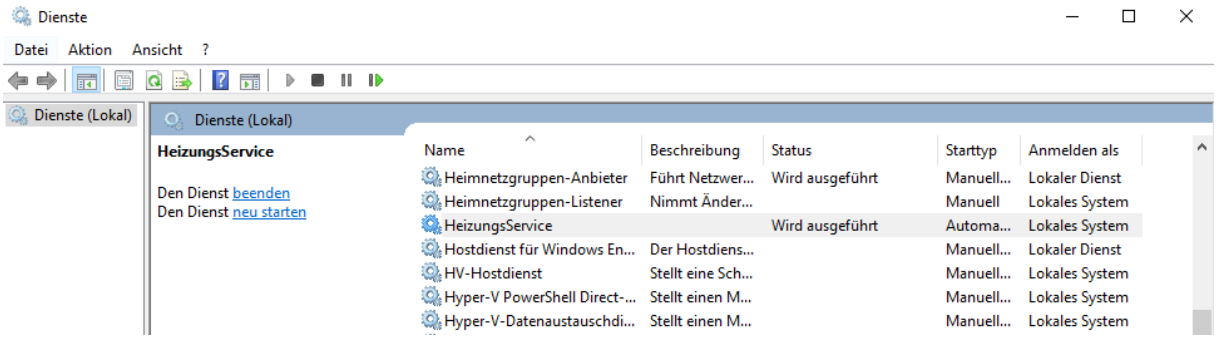


Abbildung 42 Services – HeizungsService

SignalR Client

Mit der Arbeit mit SignalR gab es unter anderem auch Probleme beim Windows Service. Der Service beinhaltete einen SignalR Client, der die Methoden „getValues“ und „setTimers“ immer nach einer Änderung auf der Datenbank durch die ASP.NET Seite aufruft. Bei der Hub Connection wurde die „useDefaultUrl“ auf false gestellt, weil beim Erstellen des SignalR Servers keine spezifische URL angegeben wurde.

```
var hubConnection = new HubConnection("http://localhost:80/", useDefaultUrl:false);
```

Beim Ausprobieren dieser Connection hat es jedes Mal eine Exception gegeben. Nach Recherchen im Internet wurde entdeckt, dass alle Hubs automatisch aufrufbar über eine spezielle URL (/signalr) sind. Dies wurde automatisch auf dem Backend generiert.

```
var hubConnection = new HubConnection("http://localhost:80/", true);

IHubProxy proxy = hubConnection.CreateHubProxy("ClientPushHub");
proxy.On<String>("getCourseUpdate", (message) =>
{
    refresh.Enabled = true;
    refresh.Interval = 1;

});

hubConnection.Start().Wait();
```

Abbildung 43 Windows Service SignalR Client

Nachdem „useDefaultUrl“ auf true gesetzt wurde, funktionierte schließlich der SignalR Client ohne Probleme.

Relaiskarte

Natürlich musste vor der Implementierung der Funktionalitäten der Relaiskarte im Programm, die Karte selbst einmal getestet werden. Bevor der Tester vom Auftraggeber ankam, hörte man nur das Geräusch, welches die Relaiskarte machte, sobald ein Port eingeschaltet wird. Mit diesem Geräusch konnte man daraus schließen, dass die Relaiskarte etwas fabriziert, aber noch nicht ob die Funktionalität optimal ist. Bei der Ankunft des Testers, war Erleichterung zu spüren, den alles hat wie geplant funktioniert. Auf diesen Grafiken (Abbildung 44 und 45) ist der Tester zu sehen. Auf diesen sind kleine LEDs eingebaut, die leuchten, sobald einer der Ports auf der Relaiskarte eingeschaltet wird.

Abbildung 45 Relaiskarte ohne eingeschalteten Ports

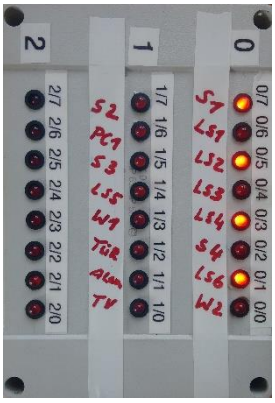


Abbildung 44 Relaiskarte mit eingeschalteten Ports

Logs

Durch die Klasse EventLog kann man verschiedene Einträge in Logfiles schreiben. Zum Beispiel in „Log – onStart“ schreibt den Eintrag, dass der Service gestartet wurde.

```
eventLog.WriteEntry("RelaiseService Erfolgreich gestartet", EventLogEntryType.Information);
```

Beispiel 13 Log - onStart

Im Beispiel 15 wird ein Eintrag eingeschrieben, falls es einen Fehler bei der Connection mit den SignalR Server gibt.

```
catch (Exception ex)
{
    eventLog.WriteEntry(ex.Message + "\n\nError with SignalR hub connection - Maybe its because of the port\n"+
        "Try to fix it and restart the service.", EventLogEntryType.Error);
}
```

Beispiel 14 Log - SignalR

In den Log Dateien werden Statusmeldungen aufgezeichnet, welche man in der Ereignisanzeige einsehen kann. In der Ereignisanzeige wird der Log „LogRelaise“ angelegt.

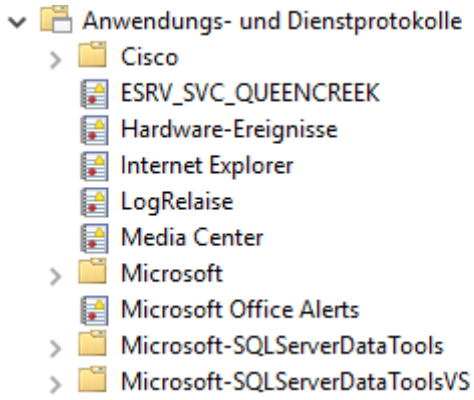


Abbildung 46 Log - Ereignisanzeige

Darin befinden sich alle Logs die irgendwann im Log „LogRelaise“ aufgetreten sind. In der Abbildung 47 kann man einen kleinen Ausschnitt davon sehen.

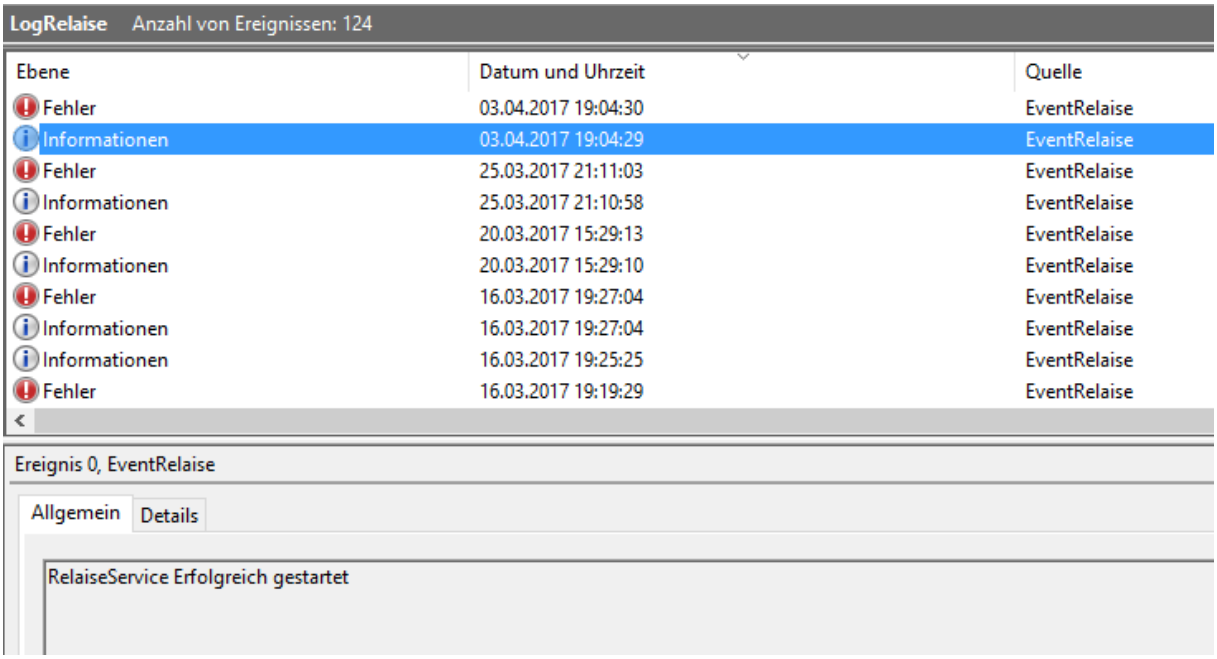


Abbildung 47 Log – LogRelaise

Aufgetretene Probleme und Alternativen

Backend

Aufgrund von Entity Framework 6 kann man nicht direkt die Datenmodelle aus der Datenbank erzeugen. Zuerst muss man folgende Referenzen in unserem Fall in der Version 4.5 in das Projekt importieren: „MySql.Data.dll“, „MySql.Data.Entity.EF6.dll“ und „MySql.Web.dll“. Außerdem muss man in der web.config das Entity Framework Tag, so wie es auf der nachfolgenden Website steht, anpassen.

http://stackoverflow.com/questions/26280396/mysql-connector-with-ef6-in-visual-studio-2013

Um Entity Framework 6 ohne Fehler verwenden zu können, muss man ein weiteres NuGet Package installieren und dadurch den Dataservice im WCF Service zu einem EntityFrameworkDataService machen.

http://stackoverflow.com/questions/25873290/wcf-odata-service-and-ef-6-issue-cant-expose-entities-using-odata-service

Seit 2015 gibt es den Fehler „ MySQL - Entity: The value for column 'IsPrimaryKey' in table 'TableDetails' is DBNull“. Dieser Fehler tritt auf, wenn man versucht das Entity Data Model, mit dem WCF Service arbeitet zu erzeugen.

Die Lösung, die bei uns funktioniert hat, erfolgt durch die Eingabe folgender Befehle in die MySQL Kommandozeile.

```
use vedv;
set global optimizer_switch='derived_merge=off';
set optimizer_switch='derived_merge=off';
commit;
```

Dabei war es wichtig den MySQL Service vor dem Eingeben neu zu starten und die MySQL Eingabeaufforderung als Administrator zu öffnen. Außerdem darf man nicht auf das commit vergessen. Nach der Eingabe war noch ein Neustart von Visual Studio notwendig, um das Auftreten des Fehlers zu verhindern.

http://stackoverflow.com/questions/33575109/mysql-entity-the-value-for-column-isprimarykey-in-table-tabledetails-is

Backend in Bezug auf die Heizungssteuerung

WriteDO

Die Methode Ixud_WriteDO, wie am Anfang schon erwähnt, schreibt eine digitale Ausgabe an die Relaiskarte. Die Parameter sind die Kartennummer, die Portnummer und einen neuen digitalen logischen Zustand. Ich dachte zuerst, dass die Portnummer der entsprechende Port ist, von den man den digitalen logischen Zustand auf 1 oder 0 setzt. Aber durch das Testen und einem schon vorgefertigten

Musterbeispiel, wurde klar, dass die Portnummer immer auf 0 ist und im digitalen logischen Zustand durch die Bit festgestellt wird, welcher Port zum Einschalten ist. Sprich wenn man eine zwei übergibt, schaltet sich der 2. Port ein. Bei einer drei wären es die ersten zwei Ports. Bei einer fünf sind es die Ports 1 und 3.

Dezimal: 2 Binär: 00000010 – 2. Port

Dezimal: 3 Binär: 00000011 – 1. und 2. Port

Dezimal: 5 Binär: 00000101 – 1. und 3. Port

Frontend

Bei dem Frontend gab es sehr viel Entwicklungsfreiraum. Laut Pflichtenheft ist die Definition: „Das Frontend ist mittels Ethernet verbunden und stellt die Kurse und Bilder grafisch (auf einen Bildschirm) dar“. Aufgrund dessen gab es in jedem Schritt mehrere Möglichkeiten, wie die gewünschte Funktion in der Diplomarbeit realisiert wurde, welche nun erklärt werden.

Hardware

Das Frontend selbst besteht grundsätzlich aus zwei Hardwarekomponenten:

- einen beliebigen Bildschirm mit HDMI Anschluss
- einem „Gerät“, welches über Ethernet Daten empfängt und anschließend über die HDMI Schnittstelle am Bildschirm anzeigt

Im Zuge der Diplomarbeit ist es nun eine der Aufgaben ein sinnvolles „Gerät“ zu definieren. Dabei gibt es wiederum zwei sinnvolle Lösungen:

- HDMI-Stick
- Single-Board-Computer (SBC)

Bei dem Vergleich zwischen den beiden Punkten erweist sich allerdings der SBC eindeutig als bessere Lösung. Auch wenn der HDMI-Stick um die Hälfte kleiner und somit auch praktischer auf einen Bildschirm zum Fixieren ist, so punktet der SBC mit seiner Flexibilität. Diese Flexibilität reicht von den Möglichkeiten des Verwendungszweckes durch die Hardware Komponenten bis über die zahlreich existierenden Betriebssysteme.
Ein weiterer wichtiger Vorteil ist auch, dass viele der Einplatinencomputer einen Ethernet Anschluss haben, während man diesen bei HDMI-Sticks nur selten sieht. Denn mit diesen muss man das Gerät nicht extra konfigurieren und auch ein Verbindungsproblem beim Ändern des Wireless LAN Passwortes wird somit umgangen und es macht das System sicherer, da sich Hacker nur über den Anschluss mit dem Netzwerk verbinden können.

Der einzige signifikante Nachteil des Raspberry Pies ist, dass er keine batteriegepufferte Uhr hat. Das heißt, dass der Raspberry Pi beim Starten ohne Internet, die Uhrzeit beim Zeitpunkt des Herunterfahrens, weiter zählt. Durch das Verbinden mit dem Internet, wird allerdings automatisch die aktuelle Uhrzeit bezogen. Dieser Schwachpunkt, ist zum einen nicht problematisch, da man davon ausgehen kann, dass der Raspberry Pi immer Internet hat, zum anderen wurde die Auswirkung bei Internetausfall durch dementsprechende Methoden im JavaScript berücksichtigt (siehe „controller.js“).

Der durchschnittliche Preis eines HDMI-Stick beträgt rund 20-30, --€. Der Minicomputer kostet von 35, --€ aufwärts. Auch wenn der Preis eines SBC höher ist, wird dies Aufgrund seiner Vorteile in Kauf genommen.

Gewählt ist schlussendlich das Raspberry Pi Model 3 worden. Dieser ist einer von den bekanntesten, sowie meist Supporteden SBCs und reicht für den Funktionsumfang des Frontends auf jeden Fall aus.

Zusammen kostet ein vollwertiger Raspberry Pi ~50,0 € pro Stück. Dabei ist ein Raspberry Pi, eine Hülle und eine Micro SD Card.

Darstellung

Nachdem die Hardware festgelegt wird, ergibt sich die Frage: „Welche Software nun?“ In diesem Falle gibt es wiederum mehrere Möglichkeiten. Im konkreten Fall sind während der Diplomarbeit 3 Optionen durchdacht worden:

- OS + Python + GUI
- OS + GUI + Python
- OS + GUI + HTML

Anfangs war die Überlegung ein Operating System (OS, Betriebssystem zu Deutsch) zu wählen, auf denen direkt durch ein Python Programm die benötigten Ressourcen geladen und dargestellt werden. Der Vorteil dieser Methode ist, dass nur Ressourcen geladen werden, welche auch wirklich benötigt werden. Allerdings erschwert es die Leserlichkeit für Entwickler, welche nicht mit dem Produkt vertraut sind und schlussendlich so auch das Warten des Produktes.

Beim Ansatz das OS die GUI (Graphical User Interface) zu laden und automatisch ein Python Programm zu starten, übernimmt das Betriebssystem das Laden des OS und der Entwickler muss nur noch über das Python Programm die gewünschten Elemente anzeigen.

Allerdings bietet sich nun auch die Möglichkeit, die Darstellung der Daten über HTML zu realisieren. Dadurch wird das Programm wiederum vereinfacht, da der Browser Teile der Darstellung abnimmt. Wie eben erwähnt, wird durch die Verwendung von HTML, mit dementsprechenden JavaScript Frameworks, der Sourcecode kurz und einfach gehalten.

Schlussendlich ist der zuletzt erklärte Weg, aufgrund der eben genannten Vorteile,

gewählt worden. Die Web Seite wird anschließend im Chromium Browser angezeigt. Dieser hat zum einen Vorteile wie die Unterstützung von WebSQL, einfaches verwenden des Kiosk Modus und ein integriertes „Developter Interfaces“ zur Verfügung stellt, zum anderen bleibt man mit Chromium, im Gegenzug zu Chrome, im Stil von Open Source. Versuche mit dem Firefox Browser wurden anfangs auch durchgeführt, allerdings unterstütz dieser kein WebSQL und es wurde auf Chromium umgestiegen.

Betriebssystem

Bei der Wahl des Betriebssystems ist besonders auf die Dokumentation geachtet worden. Da Raspbian das bekannteste Betriebssystem ist, wurde es vorläufig zum Testen der HTML-Seite verwendet. Während des Projektes sind einige Probleme aufgetreten. Diese konnten aber durch diverse Foren effizient und präzise gelöst werden. Dies war nur deshalb möglich, da Raspbian sehr verbreitet ist und schon viele Nutzer derartige Probleme hatten und diese auch gelöst haben.
Auflistung der Probleme:

- Uhzeit des Raspberry Pi
- Effizientes Arbeiten mit dem Raspberry Pi
- Bildschirmauflösung anpassen
- Chromium automatisch bei Systemstart starten
- Energie-Warn-Anzeige bei zu wenig Stromzufuhr deaktivieren

Durch die allgemeine Funktionalität, welche Raspbian hat, eignet es sich auch sehr gut für unsere Diplomarbeit. Andere OS haben hingegen nur minimale Vorteile, welche durch schlechteren Support wieder nichtig werden.

Ein Beispiel dafür ist ChromiumOS, welches ebenfalls im Zuge der Diplomarbeit getestet wurde. ChromiumOS hat deswegen seinen Reiz, da es für die Anwendung von Browser optimiert ist und der Browser als einer der wichtigsten Instrumente des Frontends ist. Somit wären auch nur die Ressourcen am Raspberry Pi, welche auch wirklich benötigt werden. Allerdings fangen die Probleme bereits beim Starten von ChromiumOS an, da sich dieses nicht ohne eine ausreichende Stromquelle starten lässt. Dazu kommt dann noch, dass man sich standardmäßig mit einem Google Konto anmelden muss. Im Vergleich zu Raspbian wären es zwar keine außernatürlichen Probleme, jedoch werden sie dies durch mangelnde Dokumentation und keine besonders gute Optimierung für SBC.

Das anfänglich verwendete „Raspbian Jessie with Pixel“ Image hatte allerdings eine Vielzahl an Programmen, welche nicht benötigt werden. Darum wurde „Raspbian Jessie Lite“ verwendet, welches die minimalen Anforderungen von Raspbian enthalten.

Framework

U Um auf der Web Seite den Sourcecode übersichtlicher zu gestalten und effizienter zu Programmieren wurden zwei Framework verwendet. Dabei wurde das AngularJS aufgrund der bereits gesammelten Know-How in der Schule verwendet. Durch das Wissen über das Framework war klar, dass es in dieser Diplomarbeit eine sehr gute Ergänzung ist.

SignalR wurde deshalb verwendet, da es den Vorteil von Push Notifikation hat und somit Server Ressourcen spart.

Internetquellen

VEDV Homepage

https://www.vedv.at/

Visual Studio Einleitung

https://de.wikipedia.org/wiki/Visual_Studio

Internet Information Service Einleitung

https://de.wikipedia.org/wiki/Microsoft_Internet_Information_Services

https://www.iis.net/

.NET Framework Erklärung

http://lifehacker.com/5791578/what-is-the-net-framework-and-why-do-i-need-it

https://www.howtogeek.com/253588/what-is-the-microsoft-net-framework-and-why-is-it-installed-on-my-pc/

ASP.NET Erklärung

https://docs.microsoft.com/en-us/aspnet/overview

https://msdn.microsoft.com/de-de/library/dd381412(v=vs.108).aspx

Code First Erklärung:

http://www.entityframeworktutorial.net/what-is-entityframework.aspx

WCF Services Erklärung:

https://msdn.microsoft.com/de-de/library/cc668794(v=vs.103).aspx

Entity Data Model Erklärung

https://msdn.microsoft.com/de-de/library/ee382825(v=vs.103).aspx

https://www.howtogeek.com/253588/what-is-the-microsoft-net-framework-and-why-is-it-installed-on-my-pc/

SignalR Erklärung

https://docs.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

Windows Service Erklärung

https://msdn.microsoft.com/en-us/library/zt39148a(v=vs.110).aspx

https://en.wikipedia.org/wiki/Windows_service

https://de.wikipedia.org/wiki/Windows-Systemdienst

InstallShield Erklärung

https://de.wikipedia.org/wiki/InstallShield

https://msdn.microsoft.com/en-us/library/dn531020.aspx

https://en.wikipedia.org/wiki/InstallShield

LINQ Erklärung

https://msdn.microsoft.com/en-us/library/bb397906.aspx

https://de.wikipedia.org/wiki/LINQ

UniDAQ Klasse Erklärung

http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/manual/unidaq%20dll%20user%20manual_eng_2.4_0617.pdf

Raspbian Erklärung

https://de.wikipedia.org/wiki/Raspberry_Pi

WebSQL Erklärung

https://en.wikipedia.org/wiki/Web_SQL_Database

Chromium Erklärung

https://de.wikipedia.org/wiki/Chromium_(Browser)

Python Erklärung

https://de.wikipedia.org/wiki/Python_(Programmiersprache)

HTML –CSS/JavaScript Erklärung

https://de.wikipedia.org/wiki/Hypertext_Markup_Language

JQuery Erklärung

https://de.wikipedia.org/wiki/JQuery

AngularJS Erklärung

https://de.wikipedia.org/wiki/AngularJS

Raspberry Pi Erklärung

https://de.wikipedia.org/wiki/Raspberry_Pi

https://en.wikipedia.org/wiki/Raspberry_Pi

Relaiskarte Erklärung

CORS Erklärung

https://docs.microsoft.com/en-us/aspnet/core/security/cors

Relais

https://de.wikipedia.org/wiki/Relais

Access Control Allow Origin (CORS)

Risikobewertungsmatrix

https://www.johner-institut.de/blog/tag/risikobewertung/

Abbildungsverzeichnis

Codebeispielverzeichnis

Schlusswort

Gesammeltes Knowhows

Eine Diplomarbeit ist eine sehr gute Erfahrung, bei der man viel lernt. Dies ist allerdings nicht nur auf die verschiedenen Programmiersprachen bezogen, sondern geht weit darüber hinaus.

Die Diplomarbeit beginnt beim Auftraggeber und auch hier beginnt unsere gesammelte Erfahrung. So bereitet man sich für ein Meeting ganz anders vor, wie zum Beispiel für ein Treffen mit einem Schulkollegen um die Hausübung fertig zu stellen. Mit einem Auftraggeber wird die Arbeit auch ernster genommen, da hier das endgültige Produkt auch verwendet wird. Dadurch steigt der Druck im Team und man betrachtet die Arbeit von einem anderen Blickwinkel.

Weiter geht es mit der Teamarbeit. Diese ist bei einer Diplomarbeit sehr wichtig und wir haben auch gelernt, dass man auch öfters versuchen muss, ein Problem aus der Sicht des anderen zu sehen, um bei einem Problem weiter zu kommen.

Während der Entwicklung ist auch die Entwicklung des produzierten Produkts sehr interessant. So ist es als Mitarbeiter dieses Produkts ein gutes Gefühl zu sehen, wie sich das Produkt weiterentwickelt und es ist schwer zu verkraften, wenn man beim Entwicklungsprozess einen Schritt zurückgehen muss. Dennoch muss man auch die gelernten Programmierkenntnisse erwähnen. Vor allem, wenn es darum geht, wie Ressourcen sinnvoll eingesetzt werden, ist oft mehr als nur eine kurze Überlegung notwendig.

Abschließend kann man sagen, dass die Diplomarbeit nicht nur unsere Programmierkenntnisse erweitert hat, sondern viel mehr unsere sozialen Fähigkeiten.

Lizenzbedingungen

Das Projektteam weist ausdrücklich darauf hin, falls das Produkt kommerziell verwendet werden möchte, dass die folgenden Punkte zu beachten sind.

Das Backend muss mit einer kommerziellen Lizenz von Visual Studio kompiliert werden.

Das „Flatly“ Theme ist unter der MIT Lizenz. Um es verwenden zu dürfen genügt es in den jeweiligen Kopien der Software den Text auf dieser Website zu kopieren und beizulegen.

https://github.com/thomaspark/bootswatch/blob/gh-pages/LICENSE

MySQL kommt ist unter 2 verschiedenen Lizenzen. Ob man dafür bezahlen muss oder nicht hängt davon ab was der Kunde mit der Software macht. Mehr Informationen gibt es hier

https://www.mysql.com/about/legal/licensing/oem/

Die JQuery Bibliothek steht wie auch die AngularJS Bibliothek unter einer MIT-Lizenz.

Die JQuery SignalR steht unter der Apache 2.0 Lizenz.

Zu allen JS Framworks liegen die Lizenzen auf dem Frontend unter dem Verzeichnis \VEDV_Frontend\lib\lincence und weiteres wird in den Bibliotheken auf die jeweiligen Lizenzen referenziert.

Das Projektteam übernimmt außerdem keine Haftung bei Lizenzrechtlichen vergehen des Kunden.

Mögliche Erweiterungen

Backend

Anstelle von ASP.NET 4.5.2 als Framework für das Backend hätte bereits .NET Core verwendet werden können. Diese Technologie ist neuer und auch plattformunabhängig. Da sich aber bereits bei anderen Projekten herausstellte, dass .NET Core noch nicht so gut dokumentiert und eher fehleranfällig ist, wurde zu Beginn der Arbeit entschieden, dass die ältere Version besser ist, um vor allem das Risiko von der technischen Umsetzbarkeit einzuschränken. Eine Verbesserung hinsichtlich der Sicherheit wäre eine Authentifizierung über SSL. Diese Lösung war bereits implementiert, allerdings funktionierte die Authentifizierung über SignalR auf der Clientseite nicht wie gedacht. Daher wurde dieses Feature wieder entfernt.

Frontend

Auf Seiten des Frontend ist es eine Überlegung wert, das ganze Programm anders zu gestalten. Damit ist gemeint, das statt Raspbian, Windows 10 IoT Core verwendet wird und eine ASP.NET Applikation statt der Webseite eingesetzt wird. Somit erreicht man die bestmögliche Verwendung der SignalR Verbindung. Dies wurde während der Diplomarbeit deshalb nicht in Erwägung gezogen, da zu diesem Zeitpunkt Windows 10 IoT Core erst auf den Markt gekommen ist.

Anhang

Installation

Wichtig ist es für den gesamten Installationsvorgang Administratorenrechte auf dem Zielsystem zu haben. Ohne diese Rechte sind manche Installationsvorgänge nicht möglich.

Windows 8.1 oder Windows Server 2012 R2

Als Voraussetzung für das Serverbetriebssystem wurde vom Auftraggeber Windows 8.1 oder Windows Server 2012 R2 vorgegeben. Daher ist es empfohlen eines dieser Betriebssysteme auf dem Server zu installieren, da es mit diesen auch getestet wurde. Allerdings sollte die Arbeit auch auf Windows 7/8/10 bzw. Windows Server 2016 ohne Probleme funktionieren.

.NET Framework 4.6

Als erstes muss .NET Framework 4.6 auf dem Server installiert werden. Auf dieser Website ist das Framework bereitgestellt:

https://www.microsoft.com/de-DE/download/details.aspx?id=48137

Hat man den Installer heruntergeladen, so installiert man ihn wie ein gewöhnliches Programm.

Weiters benötigt man noch die Visual C++ Redistributable für Visual Studio 2015 die man hier downloaden kann.

https://download.microsoft.com/download/3/8/7/387A0F10-C0C1-4C74-82A9-4BB741342366/vcredist_x64.exe

Auch diese Datei wird wieder wie ein normales Programm installiert.
Falls man ein 32 Bit Betriebssystem verwendet muss man sich die 32-bit Version suchen.

Internet Information Service

Bei Windows 8.1 sind folgende Schritte notwendig um die Anwendung ohne Probleme zu starten.

Um den IIS zu installieren drückt man die Windowstaste + R. Es erscheint ein Fenster in das man „optionalfeatures“ schreibt und Enter drückt.

Es sind alle Boxen so anzukreuzen wie in Abbildung 7 zu sehen ist

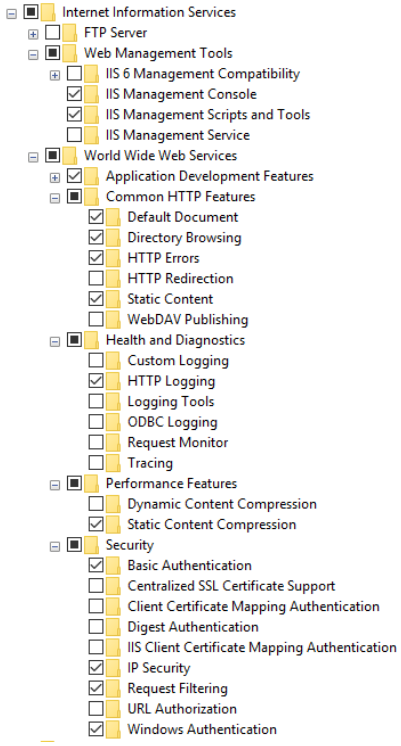


Abbildung 48 IIS Installation

Danach mit OK bestätigen und den IIS installieren lassen.

Verwendet man Windows Server 2012 R2 so öffnet man den Server Manager und drückt links oben auf Manage. Dann wählt man den ersten Eintrag „Add Roles and Features“ aus. Danach 3 mal auf Next drücken und dabei keinen Eintrag verändern. Jetzt wählt man aus der Roles Liste Web Server aus und setzt bei den selben Einträgen die Häkchen wie in Abbildung 7. Drückt man auf Next so kommt man jetzt zu den Features wo man sicherstellen sollte, dass .NET 3.5 und .NET 4.5 wie in Abbildung 8 zu sehen ist, installiert werden.

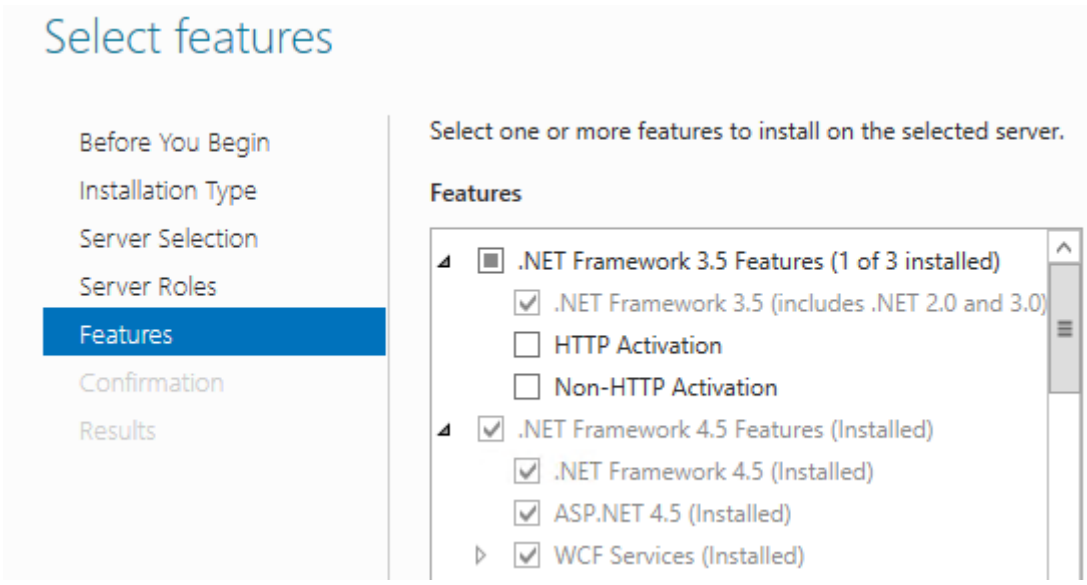


Abbildung 49 .NET Installation auf Windows Server 2012R2

Hat man sichergestellt, dass die Häkchen richtig gesetzt sind, drückt man auf Next, bestätigt die neue Ansicht und startet die Installation.

Ist die Installation abgeschlossen, kann man nun den Internet Information Services Manager starten. Dafür sucht man einfach in der Windows Suche nach „IIS“ und öffnet den Manager.

Es sollte ein Fenster erscheinen, welches so aussieht wie in Abbildung 9.

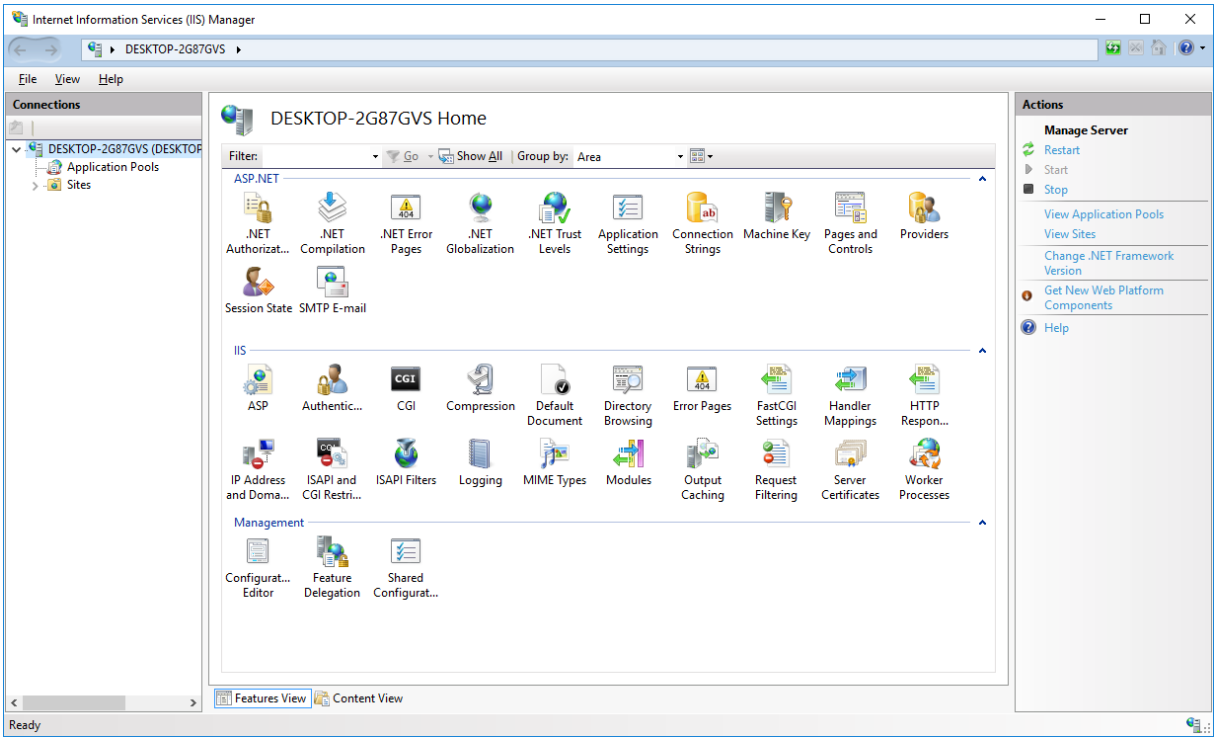
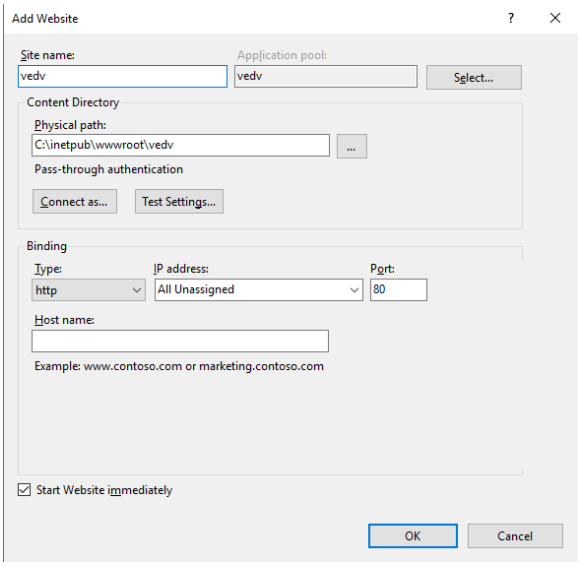


Abbildung 50 IIS Manager nach Installation

Drückt man nun auf Sites sieht man die Default Website die bei der Installation angelegt worden ist. Macht man nun einen Rechtsklick auf Sites und drückt Add Website erstellt man nun eine neue Website.



Hier ist es wichtig, dass der „Site name“ „vedv“ ist. Das Verzeichnis, wo die Website liegt kann frei gewählt werden, wenn bei „Connect as…“ „Specific User“ ein Benutzer angegeben ist, der die benötigten Lese- und Schreibrechte in dem ausgewählten Verzeichnis hat. Man kann dies verifizieren indem man „Test Settings…“ drückt und beide Tests ein grünes Häkchen haben.

Abbildung 51 IIS Seite Anlegen

Wählt man Port 80 als Port aus, so muss zuerst die automatisch erstellte „Default Website“ gestoppt werden, da sonst ein Portkonflikt entsteht.

Jetzt alles mit „OK“ bestätigen.

Unter Sites drückt man jetzt auf vedv. Nun sieht man die vedv Home Ansicht. Hier muss mit einem Doppelklick auf Authentication „ASP.NET Impersonation“ und „Anonymous Authentication“ aktivieren.

Jetzt ist der IIS und die Website für das Backend richtig konfiguriert. Nun kopiert man alle Dateien die sich auf der CD im Ordner vedv befinden und kopiert sie in das Verzeichnis das für die Website ausgewählt wurde. In diesem Fall heißt das Verzeichnis „C:\inetpub\wwwrot\vedv“. War das Kopieren erfolgreich, so kann man nun im Webbrowser unter http://localhost/ das Backend abrufen und damit arbeiten.

Man könnte auch den auf der CD beiliegenden Quelltext für das Backend im Visual Studio öffnen und diesen dann in das Verzeichnis der IIS Seite publishen. Allerdings erfordert dies die Installation von Visual Studio und den Vorgang des publishens, welcher hier nicht beschrieben wird.

MySQL

Zuerst muss man den MySQL Installer am besten auf der offiziellen MySQL Website downloaden (https://dev.mysql.com/downloads/installer/). Bei dieser Version kann man unter anderem der MySQL Server mit Connectors und die Workbench

installieren. Bei dem vorher angegebenen Link stehen zwei Downloads zur Verfügung wobei dieses Datei *mysql-installer-community-5.7.17.0.msi* die richtige ist. Diesen Installer wie gewohnt ausführen. Beim Setup Typ wählt man Custom mit diesen Features auf der benötigten Version aus.

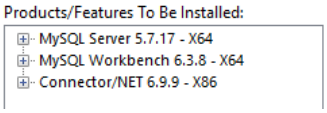


Abbildung 52 MySQL - Features

Danach muss man einen Pfad zum Installieren auswählen. Beim Typ von der Konfiguration ist Server Machine zum Auswählen.

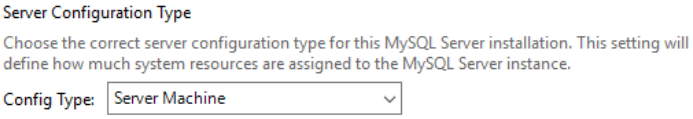


Abbildung 53 MySQL - Config Type

Der Default Port soll nicht geändert werden. Ein Passwort eingeben. Der Rest wird auf Standard gelassen.

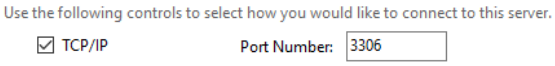


Abbildung 54 MySQL Port Number

Heizung

In diesem Abschnitt wird das Installieren der Komponenten für die Heizung erklärt.

Treiber

Falls man keine CD mit dem Treiber hat, muss man den aktuellen Treiber für die jeweilige Relaiskarte im Internet runterladen. Dieser Treiber soll danach normal installiert werden, somit man auf die Relaiskarte zugreifen kann. Wichtig ist es die Relaiskarte auszuwählen die im System ist.

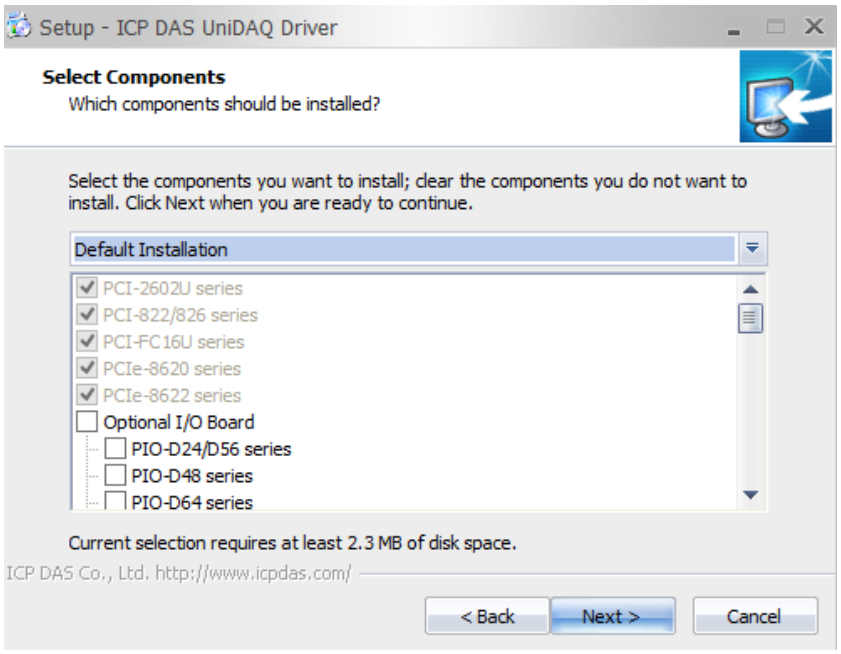


Abbildung 55 UniDAQ Treiber - Komponente

Falls man nicht sicher ist, um welche Karte es sich handelt, kann man im Geräte Manager nachsehen.

Der Treiber hier zum Download bereitgestellt:
http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/unidaq/dll/driver/

Nach der Installation des Treibers sollte der Zielrechner neu gestartet werden.

Service

Bei der Installation des Windows Services ist es egal ob der vor dem Installieren des Relaiskarten Treiber und der MySQL Datenbank erfolgt. Falls man den Relaiskarten Treiber oder die MySQL Datenbank, erst danach installiert, dann schreibt der Service einen Eintrag der die Fehlermeldung enthält in die Log Files. Um den Service zu installieren führt man die HeizungsService.exe aus und installiert diesen. Sobald die Installation fertig ist, befindet sich der Service auf dem Zielrechner und startet automatisch.

Nun muss nur noch die Relaiskarte in den Server eingebaut werden, falls dies noch nicht passiert ist.

Das System ist, sofern alle Komponenten installiert sind, einsatzbereit.

Frontend

Da das Kernstück des Frontends aus einer Webseite besteht, kann dieses mit einen beliebigen Device, welches einen Browser mit WebSQL unterstützt, verwendet werden. Somit kann man das Frontend Plattformunabhängigen kombinieren. Das

einzige auf das man achten muss, ist das definieren der Mac-Adresse in der mac.txt im Verzeichnis „.\VEDV_Frontend\config“. Wird dieses nicht berücksichtigt, werden manche Funktionen des Frontend nicht zu Gänze verfügbar sein.
Weiteres muss man vor starten der Webseite, den Parameter signalRServer, in der config.txt (im Ortner ./VEDV_Frontent/config/confit.txt), auf die aktuelle Serveradresse anstatt auf „serverAdresseEinfügen“.

In unseren Fall wurde als Device der Raspberry Pi Model 3B gewählt. Das automatische ausführen der Website durch Rasbian, wird durch folgenden Walkthrougs erreicht:

https://blogs.wcode.org/2013/09/howto-boot-your-raspberry-pi-into-a-fullscreen-browser-kiosk/

Unter Step 3 muss dabei vor dem Eintrag „chromium –app=http://URL.of.your/choice.html“ der Ordner VEDV_Frontent auf den Raspberry Pi kopiert werden und unter http://URL.of.your/choice.html der Path zur index.html angegeben werden.

Es kann unter Umständen vorkommen, dass während des Walkthrougs, in Abhängigkeit von der verwendeten Rasbian Version, Probleme auftreten können. Lösungen dazu werden in den meisten Fällen unter folgenden Link gefunden.

http://raspberrypi.stackexchange.com/