Ein Bild, das Grafiken, Schrift, Text, Logo enthält.

Automatisch generierte Beschreibung

Diplomarbeit in höherer Abteilung für Informatik

Green Crypto Mining

**Überschussbasierte Crypto-Mining Software für eine Photovoltaik-Anlage und Speicher mit InfluxDB und MongoDB**

Diplomarbeitsnummer:

**5AHINF-2023/24-DA06**

Autoren:

Alessandro Davare

Sarah Hagenhofer

Betreuer:

Dipl.-Ing. Raimund Eigner

Prof. Mag. Robert Gröbl

Ausgeführt im Jahr 2023/24

Vorwort

Die Aufgabe dieses Projektes besteht darin, eine Software für unseren Auftraggeber Raimund Eigner zu entwickeln, welche Daten einer geplanten PV-Anlage mit dazugehörigem Stromspeicher erfasst und anhand der ausgelesenen Daten eine automatische Steuerung der für uns bereitgestellten Mining Asics der Marke Whatsminer und Antminer bietet. Ebenfalls soll es eine Möglichkeit geben, die Rechner manuell zu konfigurieren und die Hashrate von jeden einzelnen sowie von allen Geräten gleichzeitig verändern zu können

…

Abstracts

The goal of this project is the development of a software for our partner Raimund Eigner, which records data from a planned PV system with associated electricity storage and based on the data read, offers automatic control of the mining asics from Whatsminer and Antminer provided to us. here should also be an option to configure the miners manually and to be able to change the hashrate of each individual and all devices at the same time…

**Kurzfassung/Aufgabenstellung**

**5AHINF – Reife und Diplomprüfung 2023/24**

|  |  |  |
| --- | --- | --- |
| **Thema** | **Entwicklung einer Software zum überschussbasierten Crypto-Mining für eine Photovoltaik-Anlage mit Speicher** | |
| **Aufgabenstellung** **(Kurzfassung)** | Ziel ist die Planung und Programmierung einer Software, welche auf einem handelsüblichen RasPi zur Steuerung von Crypto-Mining-Hardware umgesetzt werden soll. Das System soll Daten aus einer Photovoltaik-Anlage zur optimier­ten Steuerung nutzen. Einerseits die Momentan Produktionsleistung der An­lage sowie die Momentan Kapazität des Speichers.  Die intelligente und adaptive Steuerung der Hashrate der Mining-Hardware ba­siert auf momentan produzierter und gespeicherter Energie, der aktuellen Ta­ges- und Jahreszeit sowie die momentan betrieblich benötigte Leistung (saiso­nal starke Schwankungen). Dadurch soll lediglich Überschussenergie sowie nicht benötigte Speicherkapazität des Systems (über Nacht) für Mining zur Ver­fügung gestellt werden. | |
| **Kandidaten / Kandidatinnen** | | **Betreuer / Betreuerin** |
| Sarah Hagenhofer | | Dipl.-Ing. Raimund Eigner  Prof. Mag. Robert Gröbl |
| Alessandro Davare | | Dipl.-Ing. Raimund Eigner  Prof. Mag. Robert Gröbl |

|  |
| --- |
| **Externe Kooperationspartner** |
| Firma / Institution: - |
| Betreuer / Kontaktperson: Raimund Eigner |
| Schriftliche Kooperationsvereinbarung liegt vor: / |
| **Budget:** 10.000 € |
| Bedeckung durch: Dipl.-Ing. Raimund Eigner |
| **Geplante Verwertung der Ergebnisse:**  Inbetriebnahme der Software |

Erklärung

Die unterfertigten Kandidaten haben gemäß § 34 (3) SchUG in Verbindung mit § 22

(1) Zi. 3 lit. b der Verordnung Uber die abschließenden Prüfungen in den

berufsbildenden mittleren und höheren Schulen, BGBl. II Nr. 70 vom 24.02.2000

(Prüfungsordnung BMHS), die Ausarbeitung einer Diplomarbeit mit der umseitig

angeführten Aufgabenstellung gewählt.

**Die Kandidaten / Kandidatinnen nehmen zur Kenntnis, dass die Dip­lomarbeit in eigenständiger Weise und außerhalb des Unterrichtes zu bearbeiten und anzufertigen ist, wobei Ergebnisse des Unterrich­tes mit einbezogen werden können.**

**Die Abgabe der vollständigen Diplomarbeit hat bis spätestens**

**05.04.2024**

**beim zuständigen Betreuer zu erfolgen.**

**Die Kandidaten nehmen weiters zur Kenntnis, dass gemäß § 9 (6) der Prüfungsordnung BMHS nur der Schulleiter bis spätestens Ende des vorletzten Semesters den Abbruch einer Diplomarbeit an­ordnen kann, wenn diese aus nicht beim Prüfungskandidaten (bei den Prüfungskandidaten) gelegenen Gründen nicht fertiggestellt werden kann.**

|  |  |
| --- | --- |
| Kanditaten / Kandidatinnen | Unterschrift |
| Alessandro Davare |  |
| Sarah Hagenhofer |  |

**Inhaltsverzeichnis**

[1. Planung 12](#_Toc160182969)

[1.1. Aufgabenstellung 12](#_Toc160182970)

[1.1.1. Projektfindung 12](#_Toc160182971)

[1.1.2. Ausgangslage 12](#_Toc160182972)

[1.1.3. Ziele 13](#_Toc160182973)

[1.1.4. Funktionale Anforderungen 13](#_Toc160182974)

[1.1.5. Nichtfunktionale Anforderungen 14](#_Toc160182975)

[1.2. Projektorganisation 15](#_Toc160182976)

[1.2.1. Rollenverteilung 15](#_Toc160182977)

[1.2.2. Aufgabenverteilung 15](#_Toc160182978)

[Alessandro Davare 15](#_Toc160182979)

[Sarah Hagenhofer 15](#_Toc160182980)

[1.2.3. Zeitplan 16](#_Toc160182981)

[2. Projektrecherche 17](#_Toc160182982)

[2.1. Betriebssysteme (Davare) 17](#_Toc160182983)

[2.1.1. Windows 10/11 (Hagenhofer) 17](#_Toc160182984)

[2.1.2. Linux (Davare) 17](#_Toc160182985)

[2.1.2.1. EndeavourOS „Galileo“ (Davare) 19](#_Toc160182986)

[2.1.3. Android (Hagenhofer) 19](#_Toc160182987)

[2.2. Anwendungen (Davare) 19](#_Toc160182988)

[2.2.1. Programmierumgebungen (Davare) 20](#_Toc160182989)

[2.2.1.1. Visual Studio (Davare) 20](#_Toc160182990)

[2.2.1.2. Visual Studio Code (Davare) 21](#_Toc160182991)

[2.2.1.3. Jetbrains PyCharm (Davare) 21](#_Toc160182992)

[2.2.2. Microsoft Office Produkte 22](#_Toc160182993)

[2.2.2.1. Microsoft Word (Hagenhofer) 22](#_Toc160182994)

[2.2.2.2. Microsoft Powerpoint (Hagenhofer) 22](#_Toc160182995)

[2.2.2.3. Microsoft Excel (Hagenhofer) 22](#_Toc160182996)

[2.2.2.4. Microsoft OneDrive 22](#_Toc160182997)

[2.2.2.5. Microsoft Teams 23](#_Toc160182998)

[2.2.3. Weitere Anwendungen 23](#_Toc160182999)

[2.2.3.1. Figma (Hagenhofer) 23](#_Toc160183000)

[2.2.3.2. Draw.io (Hagenhofer) 23](#_Toc160183001)

[2.2.3.3. Adobe Photoshop (Hagenhofer) 24](#_Toc160183002)

[2.2.3.4. MongoDB Compass (Hagenhofer) 24](#_Toc160183003)

[2.2.3.5. Postman (Hagenhofer) 24](#_Toc160183004)

[2.2.3.6. Discord (Hagenhofer) 24](#_Toc160183005)

[2.2.3.7. Google Chrome (Hagenhofer) 25](#_Toc160183006)

[2.2.3.8. Microsoft Edge (Davare) 25](#_Toc160183007)

[2.2.3.9. OpenVPN (Davare) 25](#_Toc160183008)

[2.2.3.10. Oracle VM Virtual Box (Davare) 26](#_Toc160183009)

[2.2.4. Projektverwaltung (Hagenhofer) 27](#_Toc160183010)

[2.2.4.1. Git (Hagenhofer) 27](#_Toc160183011)

[2.2.4.2. GitHub (Hagenhofer & Davare) 27](#_Toc160183012)

[2.3. Programmiersprachen (Davare) 28](#_Toc160183013)

[2.3.1. Python (Davare) 28](#_Toc160183014)

[2.3.2. C# (Davare) 29](#_Toc160183015)

[2.3.3. JavaScript (Davare) 29](#_Toc160183016)

[2.4. Datenbanken (Davare) 30](#_Toc160183017)

[2.4.1. MongoDB (Davare) 31](#_Toc160183018)

[2.4.2. InfluxDB (Davare) 32](#_Toc160183019)

[2.5. Frameworks (Davare) 32](#_Toc160183020)

[2.5.1. .NET (Davare) 32](#_Toc160183021)

[2.5.2. Blazor (Davare) 33](#_Toc160183022)

[2.5.3. MudBlazor (Davare) 33](#_Toc160183023)

[2.5.4. Swagger (Davare) 34](#_Toc160183024)

[2.6. Nuget’s (Davare) 35](#_Toc160183025)

[2.6.1. Newtonsoft.Json (Davare) 35](#_Toc160183026)

[2.6.2. MongoDB Driver (Davare) 35](#_Toc160183027)

[2.6.3. Asp.Net.Cors (Davare) 35](#_Toc160183028)

[2.6.4. InfluxDB Client (Davare) 35](#_Toc160183029)

[2.7. Systemkomponenten 36](#_Toc160183030)

[2.7.1. Stromkomponenten 36](#_Toc160183031)

[2.7.1.1. Huawei PV-Anlagen-Steuerung 36](#_Toc160183032)

[2.7.1.2. BYD-Stromspeicher 36](#_Toc160183033)

[2.7.2. Miner 36](#_Toc160183034)

[2.7.2.1. Whatsminer 36](#_Toc160183035)

[2.7.2.2. Antminer 36](#_Toc160183036)

[2.7.3. Weitere Komponenten 36](#_Toc160183037)

[2.7.3.1. OpenVPN Verbindung 36](#_Toc160183038)

[2.7.4. Revolution Pi 36](#_Toc160183039)

[3. Umsetzung 37](#_Toc160183040)

[3.1. Recherche 37](#_Toc160183041)

[3.1.1. PV-Anlage (Hagenhofer) 37](#_Toc160183042)

[3.1.2. Speicher / Hagenhofer 37](#_Toc160183043)

[3.1.3. Miner APIs / Davare 39](#_Toc160183044)

[3.1.4. Bitmain Whatsminer 39](#_Toc160183045)

[3.2. Designfindung 41](#_Toc160183046)

[3.2.1. Mobile-App (Hagenhofer) 41](#_Toc160183047)

[3.2.2. Web-App (Davare) 44](#_Toc160183048)

[3.3. Erstellung Mobile-App (Hagenhofer) 44](#_Toc160183049)

[3.3.1. Grundsätzlicher Aufbau 44](#_Toc160183050)

[3.4. Erstellung Web-App (Davare) 45](#_Toc160183051)

[3.4.1. Zusammenfassung 46](#_Toc160183052)

[3.4.2. Literaturverzeichnis 47](#_Toc160183053)

[3.4.3. Anhänge 49](#_Toc160183054)

[3.4.4. [ Selbstständigkeitserklärung ] 50](#_Toc160183055)

**NOTES AN UNS WÄHREND WIR SCHREIBEN:**

* **Umbrüche, Schriften etc. vor Abgabe noch einmal korrigieren**
* **Überschriften 1. Ordnung Arial 16, fett, Abstand** 🡪 **12 Pt. links (0,42 cm) und 5 Pt. rechts (0,18 cm)**
* **Überschriften 2. Ordnung Arial 14 fett, Abstand 🡪 12 Pt. links und 5 Pt. rechts**
* **Standard-Schrift: Arial 12, Blocksatz, Zeilenabstand 1.5,**

[ Abbildungsverzeichnis ]

# Planung

# Aufgabenstellung

# Projektfindung

Das Interesse an Kryptowährungen jeglicher Art als Geldanlage oder generell als sichere sowie verschlüsselte digitale Währung ist heute auf einem All-Time-High und immer mehr und mehr Menschen wollen sich damit auseinandersetzen, genauso wie unser Auftraggeber Raimund Eigner.

Die Idee hinter dieser Diplomarbeit ist die alternative Verwertung von PV-produzierten Überstrom. Nachdem die Einspeisetarife wieder gefallen sind und die Crypto-Adoption im Vormarsch ist, kann diese alternative Verwertung von Überstrom durchaus als ein interessantes Pilotprojekt gesehen werden.

# Ausgangslage

Der Auftraggeber plant derzeit eine PV-Anlage für einen bestehenden Ferienhauskomplex. Der Strombedarf des Betriebes schwankt saisonal sehr stark, wobei die PV-Anlage den Großteil des nötigen Strombedarfs decken soll. Die Anlage liefert eine Engpassleistung von ca. 70 kWp und wird über einen Stromspeicher von ca. 60 Ah verfügen.

In den Wintermonaten zwischen Dezember und März wird ein zusätzlicher Stromzukauf aufgrund der wenigen Sonnenstunden und des niedrigen Sonnenstandes notwendig sein. In diesen Monaten beträgt der Verbrauch im Schnitt 140 kWh. In den Zwischensaisonen im Frühjahr und im Herbst sink der tägliche Verbrauch auf ca. 11 kWh, was zu einer Überkapazität von geschätzten 340 kWh pro Tag führt. Der Verbrauch in der Sommersaison beträgt 130 kWh. Mit den Überkapazitäten soll in erster Instanz der Stromspeicher gefüllt werden und folglich die Restkapazität für Mining-ASICs zur Verfügung stehen.

# Ziele

Das Projektziel ist es, über einen Raspberry Pi 4 den Controller umzusetzen, welcher folglich im Headless-Mode (ohne Maus, Tastatur, Bildschirm) im 24/7 Betrieb läuft und über entsprechende APIs der Hersteller (PV-Steuermodul sowie Batterie-Speicher-Steuermodul) die nötigen Daten bzgl. Verfügbarer Leistung – sowie über die APIs der Mining-ASICs die nötigen Steuerbefehle bzgl. Anpassung der Hashrate – auswertet bzw. absetzt

Der Zugriff soll über eine Mobile-App sowie über den Browser auf eine Maintenance Page zur Einstellung und zum Monitoring des Systems umgesetzt werden. Es sollen auch Status-Meldungen von den Minern als auch der PV-Anlage sowie des Stromspeichers angezeigt werden

# Funktionale Anforderungen

**Muss-Ziele**

Diese Ziele müssen nach Fertigstellung des Projektes erfüllt sein, damit es als gelungen betrachtet werden kann.

* **Recherche: Hersteller APIs der einzelnen Komponenten**

Es muss eine umfangreiche Recherche über die verfügbaren APIs der Hersteller durchgeführt und dokumentiert werden.

* **Kommunikationsfähigkeit: RasPi-GCMC-Controller / Mining-Hardware / PV-Controller / Stromspeicher-Controller**

Es muss der Softwarecontroller mit den nötigen Interfaces zu den Systemkomponenten voll funktionsfähig auf dem RasPi umgesetzt werden.

* **GCMC Maintenance-App mobile und Browser Webapp**

Es muss eine App mit responsivem Design für mobile Endgeräte und Browser umgesetzt werden, über welche diverse Einstellungen vorgenommen- und das Monitoring des Systems dargestellt werden kann (Verbrauchskurve, geminte Crypto Tokens, Produktionsleistung, Ladestand Stromspeicher, etc.)

**Soll-Ziele**

Diese Ziele sind optional, an ihnen wird nur gearbeitet, wenn nach der Bearbeitung der Muss-Ziele noch Zeit zur Verfügung steht.

* **Adaptive Logik für prädikative Stromproduktion und Verbrauch auf Basis historischer Daten**

Es soll eine Logik integriert werden, welche auf Basis von Jahreszeit (Produktionsleistung aufgrund des Sonnenstandes und der Sonnenstunden) und saisonaler Auslastung (Verbrauchsdaten im saisonalen Betrieb) einen Stellwert für eine optimale Wahl der Hashrates definiert. Es werden daher historische Daten verwendet um eine Vorhersage (predictive algorithm) über Produktion und Verbrauch treffen zu können.

* **Einbindung von ROI (return of investment) Daten in die App auf Basis von Crypto-Kursen und Strompreisen**

Es soll eine Rentabilitätsdarstellung in der App umgesetzt werden.

# Nichtfunktionale Anforderungen

**Muss-Ziele**

Die folgenden nichtfunktionalen Ziele müssen erledigt sein, bevor das Projekt als gelungen betrachtet werden kann.

* Dauerbetrieb

Die Software soll im 24/7 Betrieb Headless (Ohne Maus, Tastatur, Bildschirm) laufen und es soll jederzeit möglich sein, die Miner anzusprechen.

* Zeitreihen Datenbank

Für die regelmäßige Speicherung der zu aufzeichnenden Daten von sowohl der PV-Anlage bzw. dem PV-Speicher als auch der Miner wird eine Zeitreihen Datenbank verwendet.

* Design der Apps

Sowohl die Maintenance Mobile- als auch die Webapp soll einen modernen UI/UX Ansatz verfolgen.

# Projektorganisation

# Rollenverteilung

Auftraggeber Dipl.-Ing. Raimund Eigner

Hauptbetreuer Dipl.-Ing. Raimund Eigner

Nebenbetreuer Prof. Mag. Robert Gröbl

Projektleiter Alessandro Davare

Projektteam Alessandro Davare, Sarah Hagenhofer

# Aufgabenverteilung

## Alessandro Davare

* Recherche und Dokumentation: Mining Asics bzw. deren APIs
* Aufsetzung und Konfiguration Raspberry Pi
* Miner Controller
* Webapp für Monitoring sowie Maintenance Page
* Möglichkeit mehrere Asics hinzuzufügen
* Einstellungsseite User
* Return of Investment (Soll Ziel)

## Sarah Hagenhofer

* Recherche und Dokumentation: PV-Anlage und Speicher
* PV Controller
* Speicher Controller
* Mobile App für Monitoring sowie Maintenance Page
* Vergangene Wetterdaten in Datenbank (Soll Ziel)
* Vorhersage für „beste“ Einstellungen anhand Wetterdaten (Soll Ziel)

# Zeitplan (Davare)

**Recherche**

Die ausführliche Recherche sowie die Dokumentation zu den Schnittstellen der einzelnen Komponenten dieser Arbeit ist der erste und grundlegende Schritt dieser Arbeit.

**Vorarbeit**

Als Vorarbeit wird das Design und das Implementieren der Datenbankmodelle sowie das Installieren von benötigten Anwendungen bezeichnet. Ebenso zählt hier das Einrichten des Raspberry PI’s.

**Implementierung Apps**

Der erste große Schritt ist das Planen, das Designen und Implementieren der Mobile sowie der Webapp. Dieser Schritt gilt als erfolgreich, wenn alle Funktionen mit der angebundenen Datenbank funktionieren.

**Implementierung Controller**

Der zweite große Schritt ist das Implementieren der Lokalen Software zur Steuerung der Miner. Hier sollen die Hersteller APIs benutzt werden, um Logs der Miner auszulesen und die Miner mit gespeicherten Daten der Datenbank zu steuern. Ebenso sollen Stromdaten der PV-Anlage sowie dem Stromspeicher ausgelesen werden und in die Datenbank gespeichert werden. Wenn die Geräte erfolgreich angesteuert werden können, gilt dieser Punkt als erledigt.

**Deployment**

Der letzte Schritt dieser Arbeit wird das Aufsetzen des fertigen Projektes auf einem Industrieklassen Raspberry PI. Sollte unsere Software völlig funktional auf dem PI laufen ist das Projekt inklusiver aller Muss Ziele abgeschlossen.

**(Soll-Ziel) Return-of-Investment (ROI) Rechner**

Es soll durch eine ROI-Berechnung mit aktuellen Crypto Kursen sowie einem gegebenen Strompreis erfolgen. Hier soll der Nutzer sehen, wie lange er Minen müsste bis er Profit erreicht. Dies ist jedoch eines der Soll-Ziele und wird erst umgesetzt, wenn das Projekt abgeschlossen ist und genug Zeit verbleibt.

**(Soll-Ziel) Predictive Algorithm**

Es sollen Wetterdaten vom vergangenen Jahr ausgelesen und dazu benutzt werden, eine vorrausichtlich Ideale Einstellung für die Miner einzustellen. Dies ist jedoch das zweite Soll-Ziele und wird erst umgesetzt, wenn das Projekt abgeschlossen ist und genug Zeit verbleibt.

# Projektrecherche

# Betriebssysteme (Davare)

Ein Betriebssystem bringt die grundlegenden Funktionalitäten in die unterschiedlichsten Geräte, sodass die Hardware überhaupt laufen kann und wir Menschen weitere Funktionalitäten in die Maschinen bringen können. Diese Funktionalitäten gehen von Ressourcenverwaltung, also der Nutzung eingebauter Hardware wie Speicherplatz, RAM-Speicher, CPU-Cache-Speicher, etc. zu einer Benutzeroberfläche bzw. einem einfachen Terminal, um mit der Maschine interagieren zu können, sowie einer Dateiverwaltung um Dateien geordnet zu speichern.

[https://www.ionos.at/digitalguide/server/knowhow/was-ist-ein-betriebssystem/]

Bei unserem Projekt ist es notwendig mehrere Betriebssysteme zu nutzen, da unter Windows 10 bzw. Windows 11 programmiert, auf einer Linux Distribution für Raspberry PI (Raspbian) deployed und durch Apps auf mobilen Android Geräten angesehen wird.

# Windows 10/11 (Hagenhofer)

Windows 10 und Windows 11 stammen vom US-Unternehmen Microsoft und sind die neusten Mitglieder der Windows-Reihe, letztere ist die aktuell neuste Version der Reihe. Von insgesamt drei Geräten, auf denen wir hauptsächlich am Projekt arbeiten, laufen zwei auf Windows 10, während eines bereits auf Windows 11 upgegradet wurde.

…

# Linux (Davare)

Im Gegensatz zu Windows spricht man, wenn man über Linux redet ein Betriebssystem an, sondern Betriebssysteme, die auf dem Linux „Kernel“ basieren. Der Kernel ist in diesem Fall das Herzstück einer jeden Linux Distribution und bietet die Hauptfunktionen des Systems. Linux ist durch die die Open Source Entwicklung des Linux „Kernels“ mit einer riesigen Anzahl von freiwilligen Programmierern bekannt, da es dieses Betriebssystem sehr sicher sowie Personalisierbar macht. Theoretisch ist es so möglich, sich seine optimale Distribution je nach Nutzen, Design, und Oberfläche zu bauen.

[https://www.redhat.com/de/topics/linux/what-is-linux]

Eine Linux Distribution (abgekürzt Distro.) ist ein richtiges Betriebssystem, dass von einem Menschen benutzt werden kann. Jede Distro wurde rund um den Linux Kernel gebaut und je nach Zielgruppe, deren Bedürfnisse und Zweck entwickelt. Grundlegend benutzt Linux ein Command Line Interface (Kurz. CLI), um Befehle entgegenzunehmen, wobei es auch Distros gibt, welche ein „Graphical User Interface“ (Kurz. GUI) anbieten.

[https://www.computerweekly.com/de/definition/Linux-Distribution]

Das GUI ist an diesem Punkt aber auch nicht einheitlich und wird durch sogenannte Desktop Environments geregelt. Das sind zusammengebundene Pakete von einfachen UI-Komponenten wie Icons, Tool Bars und auch meistens Window-Manager. Diese können wiederum auch wieder je nach Bedarf konfiguriert und installiert werden, wobei dies für diese Arbeit nicht relevant ist.

[https://wiki.archlinux.org/title/desktop\_environment]

Beispiele für bekannte Linux Distributionen sind:

* Debian: Relativ altes, aber noch sehr aktive Distro, welches als sehr stabil gilt. Sie wird für PCs, Server, Embedded Systems, etc. bis heute von Erfahreneren Menschen verwendet.
* Ubuntu: Ein auf Debian basiertes Betriebssystem, dass dafür entwickelt wurde Linux für die breite Masse zugänglicher zu machen. Wird ebenfalls für Personal Computer und Server Systeme benutzt.
* Kali Linux: Das „Hacker-Linux“ basiert wie Ubuntu ebenso auf Debian und wird hauptsächlich von Profis für Penetration Tests und digitale Forensik benutzt.
* Linux Mint: Auf Ubuntu basierend wurde „Mint“ damals als attraktive Alternative zu Microsoft XP entwickelt. Sie wurde extra so entwickelt, dass es Windows Nutzern leichter viel, auf Linux umzusteigen und mehr und mehr Menschen mit Linux vertraut zu machen.
* Arch Linux: An sich ist „Arch“ nur eine minimalistische Linux Distro, welche die nötigsten Tools mit sich bringt, aber es sticht mit seiner Möglichkeit von Individualisierung hervor. Der Nutzer kann sich das Betriebssystem genau so zusammenstellen, wie er es braucht und wie genau er es möchte.

[https://www.heise.de/tipps-tricks/Linux-Betriebssysteme-eine-Uebersicht-4119937.html

# EndeavourOS „Galileo“ (Davare)

EndeavourOS ist eine relativ neue Linux Distribution welche regelmäßige Updates bekommt. Sie basiert auf Arch Linux und erleichtert es dem Benutzer Arch zu installieren. Mit Galileo bekommt man ein bereits fertiges System geliefert, welches nach der Installation direkt benutzt werden kann, im Gegensatz zu Arch Linux.

[https://endeavouros.com/]

Dieses Betriebssystem wurde nach einer gescheiterten Arch Linux Installation ausgewählt, da es ein ansprechendes Design bzw. eine angenehme User Experience bietet, ohne viele Konfigurationen tätigen zu müssen.

# Android (Hagenhofer)

Android ist eines der gängigsten Betriebssysteme für mobile Geräte, es wird vom Unternehmen Google vertrieben. Zum Stand 2024 trägt Android unter Mobile-Betriebssystemen einen Marktanteil von über 70% [Demandsage, Zugriff: 14.02.2024]. Technisch basiert Android auf einem Linux-Kernel, die verwendete Laufzeitumgebung ist Android Runtime, kurz ART. Die meisten Anwendungen werden in den Sprachen Java, C oder auch C++ geschrieben, auch bietet Android selbst Ressourcen an, die bei der App-Entwicklung für Android-Geräte helfen [Android Developer, Zugriff: 14.02.2024; Wikipedia Android, Zugriff: 14.02.2024].

# Anwendungen (Davare)

Für die Umsetzung dieser Arbeit werden einige externe Softwares benutzt, um die genannten Ziele zu erreichen. Die folgenden Programme werden hauptsächlich für das Programmieren unserer Software, Planung bzw. Design der Software sowie Datenaustausch und Projektumsetzung benutzt.

# Programmierumgebungen (Davare)

Da die Hauptaufgabe dieses Projekts darin besteht, eine Anwendung zur Steuerung von sogenannten Mining-Asics zu programmieren, sind mehrere Anwendungen erforderlich, die für die Programmierung genutzt werden können. Integrated Development Environments (IDEs) bieten Entwicklern eine Vielzahl von Tools, die ihnen direkt oder indirekt bei der Programmierung helfen. Beispiele für solche Tools sind Texteditoren mit Code-Autovervollständigung und Syntaxhervorhebung, Compiler für die Code-Übersetzung, Projekt- und Dateiverwaltung sowie viele weitere nützliche Funktionen. Diese Tools erleichtern die Entwicklung und Optimierung des Programmcodes, was zu einer effizienteren Entwicklung und einem besseren Endprodukt führt.

[https://www.pi-informatik.berlin/pi-lexikon/softwareentwicklung/was-ist-eine-entwicklungsumgebung/]

In diesem Projekt werden IDEs wie Visual Studio, Visual Studio Code oder JetBrains PyCharm verwendet, um die benötigten Anwendungen zu entwickeln. Diese IDEs bieten eine umfassende Entwicklungsumgebung, die alle erforderlichen Tools und Funktionen bereitstellt, um die Entwicklungsziele zu erreichen.

# Visual Studio (Davare)

Die von Microsoft entwickelte Integrated Development Environment (IDE) ist die erste Anlaufstelle für die Umsetzung von C++, C#, J#, Visual Basic und .NET-Projekten. Im Gegensatz zu anderen bereits vorhandenen Programmierumgebungen bietet Visual Studio nicht nur einen Compiler, sondern auch Codeabschluss-Tools, grafische Designer und viele weitere Quality-of-Life (QoL)-Funktionen, die Entwickler dabei unterstützen, die Projektentwicklung so einfach wie möglich zu gestalten.

[https://visualstudio.microsoft.com/de/vs/]

Durch die Möglichkeit, zusätzliche Pakete (sog. Workloads) zu Visual Studio hinzuzufügen, eröffnen sich viele weitere Möglichkeiten, die mit der Standardversion nicht verfügbar sind. Dies umfasst unter anderem die Entwicklung für Azure, Python und Node.js, Desktop- oder Mobile-Entwicklung mit C++, sowie Spieleentwicklung mit Unity oder anderen ähnlichen Engines.

[https://learn.microsoft.com/en-us/visualstudio/install/modify-visual-studio?view=vs-2022]

In diesem Projekt wird die Software für die Umsetzung der Webanwendung in .NET MudBlazor verwendet.

# Visual Studio Code (Davare)

Visual Studio Code (kurz. VSC) ist im Vergleich zu seinem Bruder VS keine wirkliche IDE sondern ein Text-Editor mit einer großen Community im Hintergrund, welche bis heute neue Plugins auf den Community Marketplace veröffentlicht. Auch wenn es „nur“ ein Texteditor ist, hat er trotzdem viele Tools, die es einem ermöglichen, in VSC normal zu programmieren. Die offiziell unterstützten Sprachen sind unter anderem Java- bzw. TypeScript, HTML und CSS. Weitere Sprachen werden indirekt durch den Marketplace unterstützt, indem man sich die jeweilige Extension installiert. Neben Sprachen kann man sich auch Themes und weitere Funktionen aus dem Marketplace holen.

VSC wird als hauptsächlich für die Entwicklung der Mobile App mit HTML, JS, und CSS verwendet und als allgemeiner Texteditor für beispielsweise Konfigurationsdateien.

# Jetbrains PyCharm (Davare)

PyCharm ist die von Jetbrains veröffentlichte IDE welche explizit für die Python Programmierung entwickelt wurde. Durch die Python Spezialisierung, bringt sie eine große Menge an Funktionalitäten, welche dem Entwickler entgegen kommen wie beispielsweise die automatische Python typische Formatierung anstatt von Semi-Colons und Klammern, einer intelligenten Codevervollständigung und noch viele Weitere. Es werden außerdem noch viele Frameworks wie beispielsweise Flask und Django unterstützt, die uns aber in diesem Projekt nicht weiter interessieren.

[https://www.jetbrains.com/de-de/pycharm/]  
Daher, dass der Auftraggeber Python als Sprache vorgegeben hat brauchten wir eine Python IDE, und haben und nach kurzer Recherche direkt für PyCharm entschieden.

# Microsoft Office Produkte

# Microsoft Word (Hagenhofer)

Microsoft Word ist ein Textverarbeitungsprogramm und Teil des „Microsoft Office“-Pakets. Es bietet zahlreiche hilfreiche Funktionen und erlaubt einfaches Arbeiten mit Texten und Berichten. Auch für die gänzliche Dokumentation unserer Diplomarbeitsdokumentation haben wir Microsoft Word gewählt, hauptsächlich weil wir damit bereits jahrelang gearbeitet haben, aber auch aufgrund der anpassbaren und wiederverwendbaren Formatvorlagen und Möglichkeiten der Absatzgliederung und -nummerierung. Für das eigene Mitschreiben und Dokumentieren von Notizen und Ideen haben wir genauso das Programm verwendet.

# Microsoft Powerpoint (Hagenhofer)

Microsoft Powerpoint ist das Microsoft-Office-Produkt zur Erstellung von Präsentationen. Es erlaubt das Gestalten von verschiedenen Folien, sowie deren Übergänge, auch gibt es zahlreiche hilfreiche Funktionen und Textbearbeitungsmöglichkeiten. Während unserer Arbeit haben wir das Programm für alle Diplomarbeitspräsentationen verwendet.

# Microsoft Excel (Hagenhofer)

Mit Microsoft Excel lassen sich Tabellen erstellen und mit Daten befüllen. Man kann die Daten auch beispielsweise in Graphen darstellen, Rechenaktionen mit ihnen durchführen und vieles mehr. Grundsätzlich haben wir im Rahmen der Diplomarbeit nur für private Zwecke Microsoft Excel verwendet, beispielsweise für das eigene Projekttagebuch oder die Ideensammlung zu Beginn.

# Microsoft OneDrive

Microsoft OneDrive ist ein Simpler Clouddienst, welcher es ermöglicht, Daten in die „Cloud“ hochzuladen und von verschiedenen Geräten wieder abzurufen. Das Hauptmerkmal von OneDrive ist die automatische Sicherung und darauffolgend der Upload in die Cloud, welcher in regelmäßigen Abständen passiert.

[https://www.microsoft.com/de-at/microsoft-365/onedrive/online-cloud-storage]

Es werden mehrere Geräte für die Entwicklung des Projektes verwendet, sodass eine Cloud Lösung für den Austausch der Daten praktisch ist, sodass keine unfertigen Versionen auf GitHub gepusht werden.

# Microsoft Teams

Teams ist ein von Microsoft entwickelte Plattform, welche verschiedenste Funktionen für die Nutzer bietet, um effektiv im Team zusammenzuarbeiten. Die wichtigsten Tools für die Zusammenarbeit enthalten unter anderem einen Messenger Dienst, Funktionen, um (Video)Konferenzen mit mehreren Personen zu planen bzw. abzuhalten oder auch eine einfache Integration von anderen Office365 Produkten wie unter anderem Word, Powerpoint und Excel.

Teams wird bei uns einerseits zur Kommunikation mit unserem Auftragsgeber verwendet sowie zur Dokumentation von unseren Arbeitsstunden und an was in diesen Stunden gearbeitet wurde.

# Weitere Anwendungen

# Figma (Hagenhofer)

Das Programm Figma ermöglicht das einfache und vollständige Designen von Prototypen für verschiedene Apps. Geboten werden verschiedenste Tools, unter anderem um Bereiche zu füllen oder Steuerelemente einzufügen, mit denen der User später interagieren können soll, auch lassen sich beispielhaft Abläufe definieren. Online gibt es außerdem von Communitymitgliedern Designvorlagen und -ideen, an denen man sich inspirieren und ein Beispiel nehmen kann [Figma, Zugriff: 15.02.2024]. In unserer Planungsphase haben wir für Mobile-App und Web-App unsere ersten Design-Mock-Ups erstellt und damit grundlegende Dinge wie beispielsweise die Farbwahl oder den Aufbau festgelegt.

# Draw.io (Hagenhofer)

Draw.io, auch Diagrams.net genannt, ist ein Tool, mit dem verschiedene Diagramme für das eigene Projekt erstellt werden können [Wikipedia Diagrams.net, Zugriff: 15.02.2024]. Es gibt viele Vorlagen, um Komponenten und deren Verbindungen darzustellen, die verschiedenen Abschnitte können außerdem beispielsweise farblich gestaltet werden, was die Darstellung umso übersichtlicher macht. Für zahlreiche unserer Diplomarbeitspräsentationen haben wir das Programm für Ablauf-Diagramme und Aufbau-Darstellungen verwendet.

# Adobe Photoshop (Hagenhofer)

Das Bildbearbeitungsprogramm auf dem Hause Adobe bietet eine Vielzahl von Funktionen, mit denen man Bilder editieren kann. Beispielsweise lassen sich Filter anwenden, Bilder zuschneiden oder erweitern, Bereiche ausschneiden oder ersetzen, aber auch Bereiche retuschieren [Adobe Photoshop, Zugriff: 15.02.2024]. Für die kreative Gestaltung des Diplomarbeitsposters war das Programm ideal.

# MongoDB Compass (Hagenhofer)

MongoDB Compass ist ein Tool, um die Daten der eigenen MongoDB-Datenbanken grafisch anzuzeigen [MongoDB, Zugriff: 20.02.2024]. Einige Funktionen sind beispielsweise klassische Datenabfragen mit einer „Query“, das Hinzufügen oder Löschen von Daten, aber auch das Erstellen von „Pipelines“. Es ist ein sehr nützliches Tool zum Arbeiten mit MongoDB, das wir vor allem verwenden, um die Funktionalität unserer API’s zu kontrollieren.

# Postman (Hagenhofer)

Postman ist eine der führenden Anwendungen, wenn es um API-Testing geht – insgesamt wird es von 500.000 Unternehmen, darunter viele Top-Unternehmen, genutzt [Postman, Zugriff: 20.02.2024]. Bereits im Laufe des Unterrichts haben wir Postman mehrfach verwendet und Erfahrungen damit gemacht, deshalb binden wir es auch in unsere Diplomarbeit ein. Es spielt eine wichtige Rolle beim Testen der Funktionalität unserer API’s und stellt sicher, dass alle Endpoints ausreichend ausprobiert werden.

# Discord (Hagenhofer)

Discord ist ein Chatprogramm, das wir im Rahmen unserer Diplomarbeit für die private Kommunikation nutzen. In sogenannten „Servern“ können wir mögliche Links für Lösungsansätze, persönliche Dokumentation oder auch Screenshots, Bilder und Code-Ausschnitte miteinander teilen und sicher ablegen, sodass wir von mehreren Geräten freien Zugriff darauf haben. Besonders wichtige Abschnitte können auch angepinnt und somit besonders übersichtlich festgehalten werden.

# Google Chrome (Hagenhofer)

Google Chrome verwenden wir in unserem Projekt als Haupt-Web-Browser. Das Google-Produkt ist schnell, sicher und übersichtlich, zudem gibt es viele hilfreiche Funktionen und auch die Möglichkeit, Add-Ons über den Web Store zu ergänzen. Chrome läuft auf aktuelleren Versionen von Windows, Mac, Android und auch iOS und funktioniert damit auf einem Großteil der Geräte [Chrome, Zugriff: 20.02.2024].

# Microsoft Edge (Davare)

Der vorinstallierte Webbrowser auf jedem Windows 10- oder Windows 11-Gerät stellt einen würdigen Nachfolger des legendären Internet Explorer dar. Da er nun auf dem Open-Source-Chromium-Framework von Google basiert, ist er erheblich schneller und sicherer als sein Vorgänger, der IE. Wie andere Browser bietet er Tools zur einfachen Navigation im Internet. Ein herausragendes Merkmal ist die integrierte KI Microsoft Copilot, die ähnlich wie andere bekannte KIs eine "Chat"-Funktion bietet. Der Unterschied besteht darin, dass sie durch das DALL-E-3-Modell auch in der Lage ist, Bilder zu generieren. Da der Browser von Microsoft entwickelt wurde, ist er perfekt auf Windows und andere Office-Produkte abgestimmt, weshalb eine reibungslose Zusammenarbeit und Interaktion mit den anderen Anwendungen entsteht.

[https://www.microsoft.com/de-de/edge?form=MA13FJ]

# OpenVPN (Davare)

OpenVPN ist eine freie VPN (Virtual Private Network) Software, die in diesem Projekt verwendet wird, um eine sichere Verbindung zwischen einem Zweitnetzwerk, in dem die Hardware des Ferienhauskomplexes eingerichtet ist, und unseren Endgeräten herzustellen. Diese Open-Source-Software zeichnet sich durch ihre plattformübergreifenden Releases aus und läuft auf jedem gängigen Betriebssystem, sodass sie auch problemlos auf Netzwerkgeräten wie Routern installiert werden kann, was bei uns der Fall ist. Um eine Verbindung herzustellen, wird eine Konfigurationsdatei benötigt, die wichtige Einträge wie die Ziel-IP, die Verschlüsselungsmethode, ein Zertifikat und den privaten Schlüssel enthält.

[https://openvpn.net/client/client-connect-vpn-for-windows/]

Der Auftraggeber hat uns die Konfigurationsdatei bereitgestellt und sich für diese Art der Datenübertragung entschieden, da sie einen hohen Sicherheitsstandard bietet und keine permanente Anwesenheit an den Geräten erforderlich ist, da der Netzwerkzugriff aus diesem Netzwerk eingeschränkt ist.

# Oracle VM Virtual Box (Davare)

VirtualBox ist ein von Oracle entwickelter Typ-2-Hypervisor, welcher es einem ermöglicht, dass man mehrere Betriebssysteme auf einer Virtuellen Maschine (Virtual Machine oder kurz. VM) gleichzeitig laufen lassen kann.   
Eine Virtuelle Maschine ist, wie der Name schon sagt, eine Digitale Maschine oder auch Gast genannt auf dem eigentlichen Rechner der Host genannt wird und werden im eigentlichen Sinne für Aktionen benutzt, die auf einem richtigen Gerät zu riskant wären wie z.B. das Arbeiten mit Virenverseuchten Dateien oder generell, wenn das Betriebssystem beschädigt werden kann.

Ein Hypervisor ist theoretisch der Verwalter und Ersteller der VMs und isoliert das digitale System vom richtigen System. Er verwaltet ebenfalls die Ressourcen, welche vom Host zur Verfügung gestellt werden. Es gibt zwei Arten von Hypervisor, Typ1 und Typ2.

Typ1 Hypervisor ist die effizientere Version von den beiden, weil er direkt mit der Hardware kommunizieren kann. Typ2 Hypervisor dagegen brauchen das Host Betriebssystem als Zwischenschicht, um mit der Hardware zu interagieren.

[https://www.redhat.com/de/topics/virtualization/what-is-a-hypervisor]

[https://www.oracle.com/de/virtualization/virtualbox/]

[https://aws.amazon.com/de/compare/the-difference-between-type-1-and-type-2-hypervisors]

[https://www.vmware.com/content/vmware/vmware-published-sites/de/topics/glossary/content/virtual-machine.html.html]

# Projektverwaltung (Hagenhofer)

Im Laufe des Projekts ist es wichtig, das Zusammenarbeiten möglichst effizient und sicher zu gestalten. Dazu haben wir uns entschieden, einige Tools in unsere Arbeit mit einzubinden.

# Git (Hagenhofer)

Git ist ein Source-Control-System, das es einem erlaubt, durch verschiedene „Branches“ effektiv mit anderen Personen zusammenzuarbeiten. Neben einem Master-Branch gibt es mehrere lokale Branches, an denen man ganz unabhängig voneinander arbeiten und Änderungen vornehmen kann, dies bietet sichereres Arbeiten, da man sich keine Sorgen darum machen muss, Fortschritt zunichte zu machen oder zu verlieren. Außerdem gibt es die Möglichkeit, jederzeit zu älteren Versionen des Projekts zurückzukehren. [Git, Zugriff: 20.02.2024]

# GitHub (Hagenhofer & Davare)

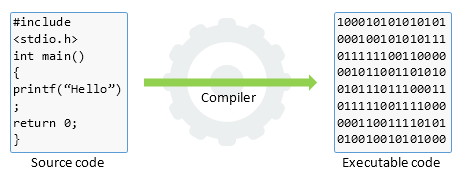
GitHub ermöglicht es, die Softwareentwicklung durch sicheres Ablegen und ein Push- und Pull-System einfacher zu machen. Es wird getrennt voneinander in Lokal-Branches gearbeitet, bevor schließlich über die GitHub Desktop App die eigenen Änderungen und Ergänzungen zum Main-Branch hinzugefügt und für alle anderen zur Verfügung gestellt werden – dieses System ist sicher, effizient und vor allem denkbar einfach. GitHub ist ein sehr großes Netzwerk mit über 420 Millionen Repositories und über 100 Millionen Entwicklern, der Austausch mit anderen ist also dementsprechend einfach und effizient. Weiters gibt es die Möglichkeit, Projekte auf public zu stellen und damit der Öffentlichkeit zur Verfügung zu stellen. [Github, Zugriff: 20.02.2024]

Auf wichtige Begriffe im GitHub-Umfeld möchte ich genauer eingehen:

* Push: Die eigenen Änderungen werden dem Main-Branch zugefügt und dort übernommen.
* Commit: Man erfasst alle Änderungen seit dem letzten Commit, und gibt sie zur Weiterverarbeitung weiter (z.B. Push).
* Pull: Der Lokal-Branch abgeändert, sodass er alle Änderungen aus dem Main-Branch übernimmt.
* Merge: Der Lokal-Branch und der Main-Branch werden zusammengeführt, hierbei kann es möglicherweise auch zu Kollisionen kommen.
* Brach: Man kann sich ein Repo wie einen Baum vorstellen, es gibt den Stamm, welcher der Master bzw. Main Branch ist, und andere Branches sind die Äste des Baumes. Nebenbranches werden in der Regel dazu erzeugt, um neue experimentelle Funktionen zu entwickeln, ohne den funktionierenden Haupt Branch zu zerstören.
* History: Vergangene Pushes und Pulls werden aufgelistet.
* Lokales Repository: Lokal geklonter eigene Version vom Remote Repo.
* Clone: Das Remote Repo wird auf den lokalen Rechner heruntergeladen.
* Fetch: Daten werden vom Repo heruntergeladen, ohne zwingenden Merge.
* Remote Repository: Repository welches in der GitHub Cloud liegt und von dort gecloned werden kann. Wenn es ein öffentliches Repository ist, kann es jeder klonen.

# Programmiersprachen (Davare)

Für die Umsetzung dieses Projektes ist es notwendig verschiedenste Programmiersprachen zu verwenden. Programmiersprachen sind dazu da, um der Hardware mitzuteilen, was sie tun soll. Sie bietet eine genaue Syntax, die es zu befolgen gilt, sodass der Compiler den Quellcode in eine Maschinensprache übersetzt, welche der Computer versteht.



[https://codeforwin.org/fundamentals/compiler-and-its-need

# Python (Davare)

Python ist weltweit durch die einfach zu befolgende Syntax und der Plattformunabhängigkeit die beliebteste Programmiersprache der Welt. Sie wird sowohl von Programmieranfängern sowie von Profis in verschiedenen Branchen wie z.B. in der Datenwissenschaft und dem Machine Learning verwendet. Python hat aber eine gewisse Eigenheit, weshalb es sich von der „Konkurrenz“ abhebt, es setzt auf Einrückungen und Formatierung anstatt von Semi-Colons und Klammern. Außerdem ist es für Python typisch, dass gewisse Quellcode Implementierungen um einiges kürzer als in anderen Programmiersprachen sind, da Python mehr Aufgaben vom Entwickler abnimmt.

[https://aws.amazon.com/de/what-is/python/#:~:text=Python%20ist%20eine%20Programmiersprache%2C%20die,Learning%20(ML)%20verwendet%20wird.]

Python wird in dieser Arbeit für die Umsetzung der Controller verwendet, da es vom Auftraggeber so festgesetzt wurde. Der Anteil an Python Code sollte ca. genau die Hälfte des gesamten Quellcodes betragen.

# C# (Davare)

Die von Microsoft entwickelte Objektorientierte Programmiersprache C# ist ein indirekter Doppelgänger von Java, da die beiden größtenteils ähnlich sind. Auch wenn beide ihre Vor- und Nachteile haben, ist C# generell mehr angesehen als Java. C# wird hauptsächlich dann verwendet, wenn eine Mobile App entwickelt wird oder wenn generell eine Software mit der .Net Plattform gebaut wird.

In dieser Arbeit wird C# für die Implementierung der Webapp in MudBlazor mit .Net, sowie der Umsetzung der Verschiedenen APIs mit Swagger benutzt.

[https://hackr.io/blog/c-sharp-vs-java#:~:text=C%23%20is%20not%20copied%20from,%2Dpurpose%20object%2Doriented%20languages.]

# JavaScript (Davare)

JavaScript ist im üblichen Sinne keine richtige Programmier- sondern eine Skriptsprache. Der Unterschied zu einer traditionellen Programmiersprache, liegt im Zweck sowie der Ausführungsumgebung. Programmiersprachen werden vom Compiler direkt in Computersprache übersetzt (kompiliert) und so nach der vollständigen Übersetzung ausgeführt. im Gegensatz dazu werden Skriptsprachen vom Webbrowser Zeile für Zeile interpretiert und so ausgeführt. Heutzutage wird JavaScript jedoch nicht nur mehr für die Funktionalitäten von Websites benutzt, sondern auch serverseitig durch die Nutzung von Node.js.

[https://www.ionos.at/digitalguide/websites/web-entwicklung/was-sind-skriptsprachen/]

# Datenbanken (Davare)

In unserer digitalen Welt steht der Schutz unserer wertvollen Daten an erster Stelle. Um eine große Menge Daten sowohl sicher als auch effizient zu speichern bzw. sie abzurufen, werden sogenannte Datenbanken benutzt.   
  
Eine „Datenbank“ besteht aus zwei Teilen, einmal einem Datenbank Management System (DBMS), welche die eigentliche Verwaltung Software ist und der eigentlichen Datenbank selbst, was eine organisierte Sammlung von Daten ist  
.

Das DBMS ist die Schnittstelle zwischen der Abgelegten Daten der Datenbank sowie dem Endnutzer bzw. einer anderen Software, welche die Daten im Endeffekt braucht. Sie können sich ans ACID (Atomicity, Cosistency, Isolation, Durability) oder im deutschen AKID (Atomarität, Konsistenz, Isolation, Dauerhaftigkeit) halten, was für Verlässlichkeit und Konsistenz steht. [https://www.ionos.at/digitalguide/hosting/hosting-technik/datenbanken/]

Atomicity oder Atomarität bedeutet, dass eine Aktion in der Datenbank entweder vollständig oder gar nicht ausgeführt wird. Dadurch wird sichergestellt, dass Änderungen vollständig übernommen werden und dadurch keine ungewollten Fehler in die Datenbanken gelangen. Wenn eine Transaktion trotzdem fehlschlägt, werden einfach alle Änderungen ignoriert und die Datenbank in den Stand vor den Änderungen gebracht (Rollback).

Consistency oder Konsistenz bedeutet, dass eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand gelangt. Der Zustand muss konsistent sein, da es sonst zu verschiedenen Ständen von Datenbanken kommen kann, wenn mehrere Clients gleichzeitig zugreifen.

Isolation bedeutet, dass eine Aktion nicht eine andere Aktion beeinflusst und eigenständig ausgeführt werden muss. Diese Sicherstellung wird meistens durch Sperrmechanismen garantiert und wirkt dadurch blockierend.

Durability oder Dauerhaftigkeit beschreibt, dass Transaktionen der Datenbank dauerhaft gespeichert werden und auch bei Technischen Problemen erhalten bleiben.

[https://www.bigdata-insider.de/was-ist-acid-a-776182/]

Es gibt verschiedene Arten von Datenbanken, auf welche in den zu benutzenden Datenbanken eingegangen wird.

# MongoDB (Davare)

Mongo ist eine dokumentenorientierte Datenbank, was bedeutet, dass ganze Dokumente in Form von JSON (JavaScript Object Notation) bzw. BSON (Binary JSON) gespeichert werden, im Gegensatz zu normalen relationalen Datenbanken, die Daten mit Zeilen bzw. Spalten organisieren. Der Entwickler bekommt in der Regel jedoch nichts von der BSON-Speicherung mit, da die jeweiligen Mongo Treiber für die verschiedenen Programmiersprachen automatisch sowohl von als auch zu BSON Übersetzen, sodass mit einem lesbaren Format gearbeitet werden kann. In der folgenden Abbildung sieht man einen direkten Vergleich zwischen Dokumentenorientierten- sowie Relationalen Datenbanken.

Ein Bild, das Text, Screenshot, Diagramm, Zahl enthält.

Automatisch generierte Beschreibung

# InfluxDB (Davare)

InfluxDB ist eine sogenannte Open-Source Zeitreihen Datenbank welche speziell dafür ausgelegt ist, einen Wert mit einer Einheit und mit einem genauen Zeitpunkt in einem Point zu speichern. Durch die Speicherung mit einem Zeitpunkt ist sie ideal für die Visualisierung von Daten, weshalb wir sie in diesem Projekt für Wetterdaten benutzen.

[https://weencrypt.pro/glossar/zeitreihendatenbanken-eine-umfassende-einfuehrung-und-analyse/]

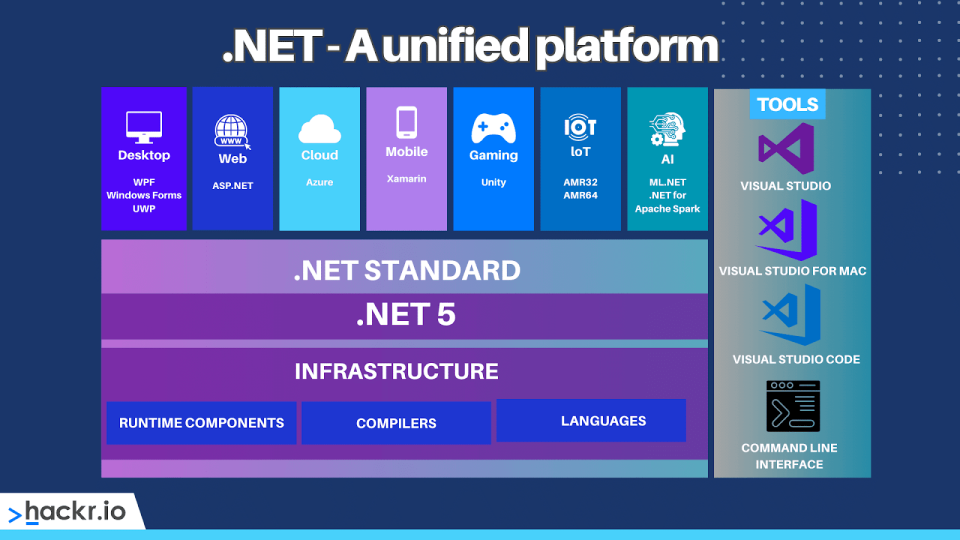
# Frameworks (Davare)

In der Softwareentwicklung gilt es neben den Hauptzielen des Projektes auch einen qualitativ hochwertigen Code zu erzeugen, sodass ihn auch jeder versteht. Ein Framework ist eine Plattform, welche die grundlegende Struktur sowie Architektur einer Anwendung bestimmt und dadurch die Code Qualität drastisch erhöht.

[https://it-talents.de/it-wissen/framework/]

# .NET (Davare)

DotNet (oder auch .Net) ist eine moderne Open Source Plattform von Microsoft, welche verschiedene Programmiersprachen unterstützt und eine Menge an Tools und Libraries für das Entwickeln von Apps bietet. Durch die einfache Entwicklung und der großen Community hinter dem Framework ist es eines der beliebtesten Frameworks für sowohl den Web- sowie dem Desktop- als auch dem Mobile Bereich.   
[https://aws.amazon.com/de/what-is/net/] [https://dotnet.microsoft.com/en-us/learn]



[https://hackr.io/blog/net-core-vs-net-framework] BILD

# Blazor (Davare)

Blazor ist ein von Microsoft entwickeltes Front-End Framework, dass einem helfen soll, eine Webapp schnell und einfach ohne JavaScript zu bauen, was aber nicht heißen muss, dass man es nicht kann. Blazor ist eine sogenanntes Single Page Application (SPA) Framework, was bedeutet, dass der Inhalt der Website dynamisch aktualisiert wird und nicht zu anderen Seiten gewechselt wird.   
[https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps]

[https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor]

Der Entwickler gestaltet seine Seiten durch sogenannte Razor Pages, welche HTML (Hyper-Text Markup Language), CSS (Cascading Style Sheet) und C# Code benutzen. Es ist sowohl möglich, einen Clientseitigen Ansatz zu wählen sowie einem Serverseitigen Ansatz.

# MudBlazor (Davare)

MudBlazor ist eine Komponenten Bibliothek, welche auf Microsofts Blazor basiert. Sie wurde anfangs dafür entwickelt mehr Leute für Blazor zu begeistern, sodass mehr Blazor Seiten erzeugt werden. MudBlazor bietet dem Entwickler einige ansprechende Komponenten, welche ihm helfen sollen, eine Blazor Anwendung zu bauen.

Eine Komponenten Bibliothek ist eine Sammlung an Komponenten, welche verwendet werden können. Eine Komponente ist ein vorgefertigtes wiederverwendbares Element, welches auf einer Website verwendet werden kann. Ein Component kann alles sein, zum Beispiel eine Navigation Bar, ein Textfeld, ein Modal, eine Card, etc.. Bei unserer Webapp könnte es Beispielsweise so aussehen. Auf der linken Seite sieht man wo Components sein könnten und rechts siehr man wie es auf der Website aussieht.Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

[https://developer.mozilla.org/en-US/docs/Web/API/Web\_components]

[https://mudblazor.com/mud/introduction]

# Swagger (Davare)

Swagger ist ein Framework, dass es einem Entwickler vereinfacht, eine RESTful API zu Dokumentieren. Swagger generiert automatisch eine übersichtliche Website, auf der man interaktiv die erstellte API dokumentiert ist. Jeder Endpoint kann ausgeklappt werden, um weitere Details über ihn zu sehen wie z.B. die Parameter, welche übergeben werden und welche Response man erwarten kann.

[https://www.ionos.at/digitalguide/websites/web-entwicklung/was-ist-swagger/]Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

# Bootstrap (Hagenhofer)

Bootstrap ist ein Framework zur erleichterten Erstellung von Web-Anwendungen. Es bietet zahlreiche Funktionen, die responsives, einheitliches Design vereinfachen und somit Zeitersparnis bieten. Bootstrap bietet verschiedenste Gestaltungsvorlagen für alles mögliche, somit ist es für unsere Diplomarbeit vielseitig nützlich. Eingebunden wird es direkt im Header des HTML-File.  
[Bootstrap, Zugriff: 01.03.2024]

# Nuget’s (Davare)

Nuget ist ein von Microsoft entwickelter Package-Installer für .Net, welcher es Entwickler erleichtert, nützlichen Code zu teilen bzw. Code von anderen als Paket zu installieren.

Die Folgenden Pakete wurden für unser Projekt verwendet.

# Newtonsoft.Json (Davare)

Newtonsoft ist ein JSON (De)-Serializer welcher dazu benutzt werden kann, ein Objekt in JSON-Form zu bringen bzw. einen JSON-String wieder in ein Objekt zu verwandeln. Visual Studio hätte einen eigenen Serializer eingebaut, aber wir haben uns für diesen entschieden, da wir mit diesem bereits Erfahrungen gesammelt haben.

# MongoDB Driver (Davare)

Das MongoDB-Driver Nuget ist ein grundlegender Baustein für die Entwicklung mit einer MongoDB Datenbank mit .Net. Es bietet die Grundfunktionen, um mit einer MongoDB Datenbank zu kommunizieren, egal ob Lokal oder in der Cloud mit MongoDB Altas.

# Asp.Net.Cors (Davare)

Eigentlich sollte eine Website keine Daten von einem anderen Server laden. Bei Cors (Cross-Origin Ressource Sharing) handelt es sich um ein Prinzip, dass eine Absprache von mehreren Servern erlaubt.

Dieses Nuget musste hinzugefügt werden, da es beim Benutzen der APIs bei den Apps zu Problemen mit CORS gekommen ist und wir durch dieses Nuget Einstellungen an Cors tätigen können.

# InfluxDB Client (Davare)

Gleich wie der MongoDB Driver ist das InfluxDB Client Nuget der grundlegende Stein um mit Influx Zeitreihendatenbanken zu Arbeiten. Es ermöglicht eine Verbindung zu einer InfluxDB Datenbank herzustellen, Daten hineinzuschreiben und Abfragen auszuführen.

# Systemkomponenten

…

# Stromkomponenten

# Huawei PV-Anlagen-Steuerung

# BYD-Stromspeicher

# Asic Miner (Davare)

Das „Minen“ von Kryptowährungen erfordert durch die komplexen Berechnungen der „Blöcke“ auf der Blockchain eine immense Rechenkraft. Es ist sowohl möglich normale Computer Hardware, sogenannte Grafikkarten (kurz. GPU), als auch eigene Miner, sogenannte „Asics“ die nur für das Minen von Crypto ausgelegt sind und sonst keinen anderen Nutzen haben.

[https://www.bitpanda.com/academy/de/lektionen/was-ist-bitcoin-mining-und-wie-funktioniert-es/#was-ist-bitcoin-mining]

# MicroBT Whatsminer (Davare)

MicroBT ist einer von wenigen großen Asic Herstellern, die in der Mining Community angesehen sind. Mit ihren „Whatsminer“ Modellen produzieren sie Geräte in einer Preisklasse von rund 800$ bis zu 11.000$ und somit Geräte für sowohl Hobby Miner als auch Kommerzielle Mining Farmen.

Whatsminer zeichnet sich sowohl durch die große Auswahl an Modellen sowie der unterschiedlichen Kühlungsmöglichkeiten als auch der Preis-Leistung der Geräte aus.

[https://www.okx.com/de/learn/9-best-asic-miners]

[https://www.whatsminer.com/?lang=en-US]

# Bitmain Antminer (Davare)

Bitmain ist ein anderer großer Hersteller von Mining Asic Rechnern. Ihre Antminer starten bei einem Preis von 1.920$ und gehen mit dem Antspace auf bis zu 100.000$. Der Antspace ist ein ISO-Container, welcher so umfunktioniert wurde, dass er Platz für bis zu 210 Mining Asics hat. Dieser benötigt außerdem eine eigenes Kühlungsmodul, wo der Kunde eine Auswahl zwischen „Dry-Wet Tower“, „Dry Tower“ und „Plate Heat“ hat. Im Container ist außerdem noch ein Zentraler Schaltschrank, welcher zur Steuerung aller Miner im Container dient.

Durch den höheren Startpreis der Geräte richtet sich Bitmain eher an bereits erfahrenere Menschen und Menschen, die eine Absicht verfolgen, Profit mit dem Mining zu machen.

[https://www.okx.com/de/learn/9-best-asic-miners]

[https://m.bitmain.com/]

[https://www.youtube.com/watch?v=ofPdztVcHjk]

# Weitere Komponenten (Davare)

Hier werden Komponenten gelistet, die in keine andere Kategorie gepasst haben, aber trotzdem so relevant sind, dass sie genannt werden sollten.

# OpenVPN Verbindung (Davare)

Für die Verbindung in das Netzwerk des Auftraggebers wurde uns eine OpenVPN Konfiguration zur Verfügung gestellt, mit der man sich über den OpenVPN Client verbinden kann. Unter Windows wird die .exe-Datei (Executable oder dt: Ausführbare Datei) von der Website heruntergeladen und installiert. Unter Linux wird diese durch den jeweiligen Paketmanager der Linux Distribution installiert.

In der nachfolgenden Abbildung sieht man einen Ausschnitt der Konfigurationsdatei. Der Befehl „Client“ sagt aus, dass OpenVPN die Konfiguration als Client sehen muss, der sich zu einem Server verbinden will, was in unserem Fall der Router des Auftragsgeber ist. Die Zeile „dev tun“ setzt das imaginäre Rohr, durch die alle Daten geschickt werden, um eine sichere Übertragung zu gewährleisten und „proto udp“ setzt das Übertragungsprotokoll auf UDP (User Datagram Protocol).   
„Float“ sagt aus, dass die IP des Clients jederzeit geändert werden kann, ohne dass die VPN Verbindung abbricht und „nobind“, dass der Client keine bestimmte IP-Adresse bekommen muss. Die Zeile 6 setzt den zu benutzenden Verschlüsselungsalgorithmus auf „AES-128-CBC“ und Zeile 7 lässt die Pakete, welche von sowohl Server als auch Client verschickt werden, je nach Netzwerk komprimieren, sodass je nach Netzwerkbedingungen eine schwächere bzw. stärkere Kompression stattfindet. Mit dem Algorithmus kam es zu einem Verbindungsproblem, auf den später in dieser Arbeit noch eingegangen wird.   
Die nächste Zeile teilt dem Client mit, dass er unendlich lange probieren soll sich mit dem Server zu verbinden und die Zeile direkt danach überprüft bei jeder Verbindung die Zertifizierung des Servers, sodass sichergestellt wird, dass der Client sich mit dem richtigen Server verbunden hat. Zeile 10 stellt sicher, dass der Schlüssel für die VPN-Verbindung persistent (dauerhaft) auf dem Gerät gespeichert wird. Mit der letzten Zeile „remote“ und der darauffolgenden IP-Adresse und Port setzt man fest, auf welcher Adresse der Server liegt und wohin sich der Client verbinden soll. Die Daten wurden aus Sicherheitsgründen zensiert.

[https://openvpn.net/community-resources/reference-manual-for-openvpn-2-4/]

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Unter dem Ausschnitt folgen noch weitere Zeilen mit Zertifikaten und Schlüsseln, auf die aber nicht weiter eingegangen werden muss.

# Revolution Pi (Davare)

Der Revolution Pi ist eine Industrietaugliche Version eines Raspberry Pi’s, welcher nach Europäischer Norm sowie Internationaler Elektrotechnischer Kommission zertifiziert ist. Der Pi ist in einem Hutschienengehäuse untergebracht und unterstützt eine Vielzahl von verschiedenen I/O (Input / Output) Modulen, welche die Möglichkeiten des Industrie-PCs um einiges erweitern.

Der Mini Computer ist ein wichtiges Teil dieses Projektes, da die ganze Software auf diesem 24/7 laufen soll und somit eine Schnittstelle zwischen den Miner und der Stromkomponenten ist. In der nachfolgenden sieht man unseren zur Verfügung gestellten Revolution PI sowie das Digitale IO-Modul und einem Hutschienennetzteil der Firma „Meanwell“.

Ein Bild, das Maschine, Im Haus, Elektronik, Werkzeug enthält.

Automatisch generierte Beschreibung

# Umsetzung

# Recherche

Eines der Mussziele dieser Arbeit ist das sorgfältige Recherchieren sowie Dokumentieren von den offiziellen Schnittstellen der einzelnen Komponenten.

# PV-Anlage (Hagenhofer)

Für eine funktionierende PV-Anlage werden natürlich neben den Paneelen eine PV-Anlagen-Steuerung benötigt. Für unser Projekt war es zunächst unklar, ob hierfür mit der Marke „Fronius“ oder „Huawei“ gearbeitet werden wird.

Fronius ist ein seit 1945 existentes österreichisches Unternehmen, das unter anderem seit 1992 auf Solarenergie spezialisiert ist. Das Unternehmen ist innovativ, hochqualitativ und wurde von Dun & Bradstreet, dem größten Dienstleister für B2B-Wirtschaftsinformation weltweit, mit dem höchsten Sicherheitsfaktor ausgezeichnet [Fronius, Zugriff: 22.01.2024]. Zudem wurde Fronius mit zahlreichen Leistungen wie dem Österreichischen Staatspreis für Umwelt- und Energietechnologie (2007) und dem Intersolar Award (Symo GEN24 Plus, 2020) ausgezeichnet [Fronius, Zugriff: 22.01.2024].

Huawei ist ein ursprünglich chinesisches Unternehmen, das sich seit mehreren Jahren auch auf Solarenergie spezialisiert hat. Inzwischen ist es führend in mehreren verwandten Bereichen, unter anderem auch bei der Innovation von Energie-Speichersystem-Architekturen. Das Unternehmen hat zum aktuellen Stand 997 Mrd. kWh grüne Energie erzeugt und damit 431 Mio. Tonnen CO2 Emissionen reduziert [Huawei, Zugriff: 22.01.2024].

Schlussendlich wurde eine Huawei-Steuerung gewählt, mit der wir arbeiten. Zum aktuellen Stand wurde Anlagen-Steuerung noch nicht abgeschafft, weswegen wir diesen Teil „mocken“ werden müssen.

# Speicher / Hagenhofer

Der Stromspeicher ist wohl das Herz einer hoch-funktionalen PV-Anlage. Wenn der Bedarf an Haushaltsstrom gedeckt ist, wird dieser Stromspeicher von dem überschüssigen PV-Strom aufgeladen wie ein Akku. Sobald der Bedarf des Hauses höher ist als der aktuell produzierte Strom, wird der Speicher wieder entladen.  
Obwohl PV-Anlagen grundsätzlich auch ohne Stromspeicher funktionieren, steigert er den Eigenverbrauch des eigenproduzierten Stroms enorm. Die Energiekosten werden dadurch gesenkt und weniger sonnige Zeiten überbrückt [Bundesverband Photovoltaic Austria, Zugriff: 18.12.2023].

Grundsätzlich sind zum aktuellen Stand zwei Batteriearten auf den Markt verbreitet. Sowohl Lithium-Ionen-Batterien, als auch Natrium-Ionen-Batterien werden zum Großteil verwendet. Beide Arten haben verschiedene Vorteile, die durchaus verglichen werden müssen, um die bessere Option für den eigenen Bedarf aussuchen zu können.

* Lithium-Ionen-Batterien:

Lithium-Ionen-Akkus werden schon seit einigen Jahren in elektronischen Gegenständen wie beispielsweise Smartphones verwendet. Sie haben einen hohen Wirkungsgrad und können sehr oft geladen werden, bevor sie ausgetauscht werden müssen. Außerdem sind sie immer noch voll im Trend der Forschung und werden fortlaufend modernisiert und verbessert. Ein potenzielles Problem stellt eine mögliche Überhitzung da, zu der es potentiell bei Speicherüberladung kommen kann. In modernen Batterien wird mögliche Überhitzung allerdings schon aktiv durch intelligente Ladesysteme vorgebeugt [Verbund AG, Zugriff: 18.12.2023], [Bundesverband Photovoltaic Austria, Zugriff: 18.12.2023].

* Natrium-Ionen-Batterien:  
  Natrium-Ionen-Batterien funktionieren grundsätzlich sehr ähnlich wie Lithium-Ionen-Batterien. Natrium, einer der Grundbestandteile, ist allerdings um einiges leichter anzuschaffen als Lithium, weswegen diese Art in der Anschaffung etwas umweltfreundlicher ist. Die Batterie stellt höhere Lade- und Entladeströme bereit, zusätzlich ist sie gegen Überhitzung und Explosion geschützt und deshalb sehr sicher. Ein Nachteil jedoch ist, dass Natrium-Ionen-Batterien recht groß und schwer sind, weswegen sie einen dementsprechend passenden Platz brauchen, um installiert zu werden [Bundesverband Photovoltaic Austria, Zugriff: 18.12.2023], [Energieheld Schweiz, Zugriff: 18.12.2023].

Im Rahmen unseres Projekts werden wir mit einem Stromspeicher der chinesischen Marke BYD arbeiten. Beim Vergleich des BYD-Speicherangebots fällt auf, dass alle verwendeten Batterien auf Lithium-Ionen-Batterien basieren.

# Miner APIs / Davare

# Bitmain Whatsminer

Die „Whatsminer“ Geräte des weltweit agierenden chinesischen Herstellers Bitmain benutzen eine einheitliche API mit der aktuellen Version 2.0.5 [AWS, Zugriff: 18.12.2023].

Bevor die API vollständig benutzt werden kann, müssen erst ein paar wenige Schritte befolgt werden. Per Default ist nämlich das Schreibrecht deaktiviert und es muss über das Hersteller Tool „WhatsminerTool“ aktiviert werden. Hierfür ist es notwendig das Standardpasswort des Admins zu ändern und die API per Knopfdrück und Bestätigung zu aktivieren.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Abbildung 1: Aktivierung API in Whatminer Tool

Bei Erfolgreicher Aktivierung wird bei dem jeweiligen Miner unter „API“ der Status „ON“ angezeigt.

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

Abbildung 2: API eingeschaltet

Nun sollte auf die API über die IP des Miners sowie dem Port 4028 zugegriffen werden können. Es ist zu beachten, dass wenn eine Verbindung zu dem Port aufgebaut wird und der Miner innerhalb von zehn Sekunden keine Daten erhält, dass die Verbindung automatisch geschlossen wird. Für die Write-Funktionen wird außerdem ein Token benötigt, welcher über den API-Endpoint „get\_token“ angefragt werden kann. Dieser hält für 30 Minuten und muss für weitere Anfragen nach Ablauf erneuert werden. Ein Miner kann gleichzeitig bis zu 16 IP-Clients gleichzeitig haben, wobei jeder Client gleichzeitig bis zu 32 Tokens haben kann. Die Response der API hat immer folgenden Aufbau:

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Als Status wird entweder ‚S‘ für Success oder ‚E‘ für Error zurückgegeben. Die Variable „When“ gibt die Zeit der Anfrage in der Unixzeit zurück, „Code“ zeigt dazu noch einen Code, der den Status genauer erklärt.

Folgende Codes sind möglich:

* 14: „invalid API command or data”
* 23: „invalid json message“
* 45: „permission denied“
* 131: „command OK“
* 132: „command error“
* 134: „get token message OK“
* 135: „check token error“
* 136: „token over max times“
* 137: „base64 decode error“

„Msg“ gibt die angeforderten Werte zurück und Description bleibt meistens leer.

# Designfindung

# Mobile-App (Hagenhofer)

In der ersten Phase der Designfindung für unsere Mobile-App habe ich mich dafür entschieden, mit Figma ein beispielhaftes Design zu erstellen. Der Fokus hierbei lag auf der Entscheidung, welche Funktionen es in der App geben soll, sowie wo die Funktionen auf welche Weise untergebracht werden soll. Außerdem ging es mir darum, eine erste Farbwahl zu treffen und diese gleich von Beginn an abzuklären, um spätere Missverständnisse zu vermeiden. Im Nachhinein hat sich dieses Design noch etwas verändert, es ist die Möglichkeit zur Bestimmung der „Akzentfarbe“ und des Light-Theme dazugekommen, außerdem ist noch eine zusätzliche Seite dazugekommen – grundsätzlich hat sich der Aufbau allerdings beibehalten und sich dieser Planungsschritt bewährt.

Ein Bild, das Text, Screenshot, Design enthält.

Automatisch generierte Beschreibung

Ein Bild, das Screenshot, Text, Software, Design enthält.

Automatisch generierte Beschreibung

Ein Bild, das Screenshot, Elektronisches Gerät, Computer, Hebel enthält.

Automatisch generierte Beschreibung

Ein Bild, das Screenshot, Text, Rechteck, Quadrat enthält.

Automatisch generierte Beschreibung

# Web-App (Davare)

…

# Erstellung Mobile-App (Hagenhofer)

# Grundsätzlicher Aufbau

Der erste Schritt in meiner Arbeit an der Mobile-App war das Nachdenken darüber, wie die App aufgebaut werden soll. Bereits in der Vorarbeit habe ich ein Mock-Up erstellt, an das ich mich grundsätzlich auch halten wollte (siehe Kapitel 3.2.1). Diese Planung umfasste zunächst folgende Punkte:

* Startseite: Je eine Anzeige für die aktuelle Rate, die Hashrate der Miner und den aktuellen Niederschlag. Das Design sollte schlicht und intuitiv gestaltet sein.
* Settings: Eine Seite für einige Einstellungen, die der User vornehmen kann. Diese umfassen die Auswahl der Währung, den aktuellen Strompreis, die Akzentfarbe (die sich dann in verschiedenen Teilen beider Seiten widerspiegeln soll), und auch die Möglichkeit, die Crypto-Währung auszuwählen, in die investiert werden soll. Auch hier soll das Design einfach gehalten werden, um die Benutzung möglichst einfach zu machen.

Während der Erstellung habe ich dann auch einige Änderungen vorgenommen, weswegen sich die Punkte wie folgt verändert haben:

* Startseite: Die Hauptseite der Mobile-App, auf der die aktuelle Rate in Prozent, die Hashrate der Miner in TH/s und den aktuellen Niederschlag in Prozent dargestellt werden. Bei Klick auf die verschiedenen Anzeigeflächen wird die Ansicht auf eine Grafische gewechselt – im Falle der beiden Raten auf ein Liniendiagramm, im Falle des Niederschlags auf eine Tabelle mit dem aktuellen Wetter. Das Design ist schlicht gestaltet und wird je nach den Usereinstellungen eingefärbt (Akzeptfarbe, Light-/Dark-Theme). In der „Kopfzeile“ kann zu den Folgeseiten weiternavigiert werden.
* Settings: Die Settings-Page umfasst grundsätzliche Einstellungen, die der User vornehmen kann. Neben der Akzentfarbe und dem „Colour-Theme“ kann der Strompreis eingegeben werden. Die anderen geplanten Einstellungen haben wir vorerst auf „To-be-defined“ gesetzt, nachdem wir dies mit dem Auftraggeber abgeklärt haben.
* ASIC-Übersicht: Die Seite bietet in der Mobile-App eine grobe Übersicht über alle verwendeten ASIC’s und listet sie in Anzeigeflächen auf.

# Startseite

Direkt beim Laden der Startseite werden einige wichtige Funktionen aufgerufen, die die richtige Ansicht sicherstellen.



Diese Funktionen umfassen sowohl das Laden der beiden Raten, das Laden des Wetters, sowie das Laden von Akzentfarbe und Theme.

Ein Bild, das Text, Schrift, Screenshot, Zahl enthält.

Automatisch generierte Beschreibung

# LoadRate(), LoadHash(), LoadColour()

Die LoadRate() und LoadHash()-Funktionen funktionieren zu einem großen Teil gleich. Die benötigten Daten werden über die selbsterstellte API (DANN NOCH QUERVERWEIS EINFÜGEN) aus der Datenbank geladen (Zeile 222ff.) und dargestellt (Zeile 231). Wichtig ist es hierbei auch, in entsprechenden Try-Catch-Blöcken zu arbeiten, um mit möglichen Fehler umzugehen und ein Abstürzen der Anwendung zu verhindern

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Auch die LoadColour()-Funktion ist auf diese Weise aufgebaut, eine spezielle Eigenheit ist hierbei jedoch zu beachten. Mit der vorher verwendeten document.getElementById()-Funktion (Zeile 219) lässt sich nämlich nur ein einziges Element auswählen. Dies hat einige Schwierigkeiten verursacht, weil wir anfangs nicht wussten, das sich nicht mehrere Elemente auswählen lassen. Als beste Alternative stellte sich getElementsByClassName (Zeile 293) heraus, mit dem man alle Elemente des gegebenen Namens einfärben kann, indem man einfach durch alle Elemente durchzählt und überall die Farbe anpasst (Zeile 295-297).

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

# LoadWeather()

…

# LoadTheme()

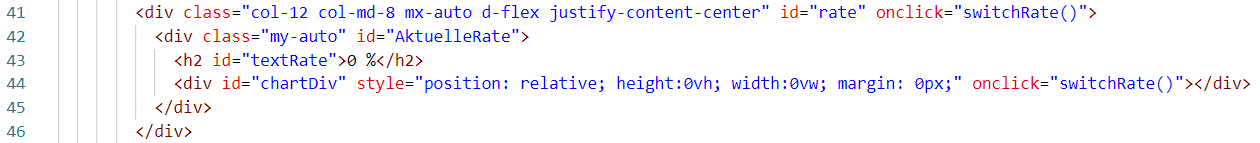
Auch diese Funktion basiert auf dem gleichen API-Prinzip wie die meisten anderen genannten (siehe 3.3.2.1). Einen eigenen Unterpunkt bekommt diese Funktion aufgrund der Besonderheit, dass der GET-Request einen Boolean Wert zurückliefert. Je nachdem, ob true oder false zurückgeliefert wird, wird dann dementsprechend das Light- oder Dark-Theme ausgewählt (Zeile 321) und als aktives Theme ausgewählt (Zeile 329), während das alte Theme entfernt wird (Zeile 327). (Abbildung dann beschriften: nur beispielhaft die Light-Theme-Auswahl)

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

# Wechsel zwischen Zahlen und Graphen

Der Klick auf die Anzeigen erlaubt es, bei Raten-Anzeige und Hash-Rate-Anzeige über eine onclick-Funktion (Zeile 44) zwischen der Zahl und einem Graphen hin- und herzuwechseln.



…

# Erstellung Web-App (Davare)

…

# Zusammenfassung

# Literaturverzeichnis

* [AWS, Zugriff 18.12.2023]:   
  <https://aws-microbt-com-bucket.s3.us-west-2.amazonaws.com/WhatsminerAPI%20V2.0.5.pdf>
* [Bundesverband Photovoltaic Austria, Zugriff: 18.12.2023]: <https://pvaustria.at/pv-speicher/>
* [Verbund AG, Zugriff: 18.12.2023]:  
  <https://www.verbund.com/de-at/privatkunden/themenwelten/photovoltaik/lithium-ionen-akku>
* [Energieheld Schweiz, Zugriff: 18.12.2023]: <https://www.energieheld.ch/solaranlagen/stromspeicher/salzspeicher#:~:text=Salzwasserspeicher%2C%20welche%20auf%20die%20Natrium,Kilowattstunden%20(kWh)%20Kapazit%C3%A4t%20angeboten>
* [Fronius, Zugriff: 22.01.2024]:   
  <https://www.fronius.com/de-at/austria/solarenergie/ueber-uns/warum-fronius>  
  <https://www.fronius.com/de-at/austria/solarenergie/ueber-uns/geschichte>
* [Huawei, Zugriff: 22.01.2024]:  
  <https://solar.huawei.com/at>
* [Android Developer, Zugriff: 14.02.2024]:

<https://developer.android.com/>

* [Demandsage, Zugriff: 14.02.2024]:

<https://www.demandsage.com/android-statistics/#:~:text=There%20are%203.3%20billion%20Android,share%20globally%20as%20of%202024>.

* [Wikipedia Android, Zugriff: 14.02.2024]:

<https://de.wikipedia.org/wiki/Android_(Betriebssystem)#Architektur>

* [Figma, Zugriff: 15.02.2024]:

<https://www.figma.com/community>

* [Wikipedia Diagrams.net, Zugriff: 15.02.2024]:

<https://de.wikipedia.org/wiki/Diagrams.net>

* [Adobe Photoshop, Zugriff: 15.02.2024]:

<https://www.adobe.com/at/products/photoshop/features.html>

* [MongoDB, Zugriff: 20.02.2024]:

<https://www.mongodb.com/docs/compass/current/#:~:text=MongoDB%20Compass%20is%20a%20powerful,View%20installation%20instructions>

* [Postman, Zugriff: 20.02.2024]:

<https://www.postman.com/company/about-postman/>

* [Chrome, Zugriff: 20.02.2024]:  
  <https://www.google.com/intl/de_at/chrome/>
* [Git, Zugriff: 20.02.2024]:  
  <https://git-scm.com/about>
* [Github, Zugriff: 20.02.2024]:  
  <https://github.com/about>
* [Bootstrap, Zugriff: 01.03.2024]:

<https://getbootstrap.com/>

# Anhänge

# [ Selbstständigkeitserklärung ]